МИНИСТЕРСТВО ОБРАЗОВАНИЯ КИРОВСКОЙ ОБЛАСТИ

Кировское областное государственное профессиональное образовательное

бюджетное учреждение

«Слободской колледж педагогики и социальных отношений»

**ОТЧЕТ**

**по производственной практике**

**ПМ.02. Осуществление интеграции программных модулей**

Студент

Квакин Александр Дмитриевич

Группа 21П-1

Специальность 09.02.07 Информационные системы и программирование

Руководитель практики от колледжа:

*Пентин Николай Сергеевич*

Руководитель практики от организации:

*Кузнецова Елизавета Сергеевна \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

подпись

УТВЕРЖДАЮ:

Директор

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
 Наименование организации

ООО Мастер-Софт Сервис

\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись расшифровка

М. П.

2024 уч. год

Оглавление

[1. ХАРАКТЕРИСТИКА ОБЪЕКТА ПРАКТИКИ (ЮРИДИЧЕСКИЙ АДРЕС, СПЕЦИАЛИЗАЦИЯ, СТРУКТУРА) 3](#_Toc171125784)

[2. ОПИСАНИЕ РАБОЧЕГО МЕСТА 4](#_Toc171125785)

[3. СОСТАВ ПРОГРАММНОГО И ТЕХНИЧЕСКОГО ОБЕСПЕЧЕНИЯ, ИМЕЮЩЕГОСЯ НА ПРЕДПРИЯТИИ, ИХ НАЗНАЧЕНИЕ 6](#_Toc171125786)

[3.1. Программное обеспечение компании 6](#_Toc171125787)

[3.2. Техническое обеспечение компании 7](#_Toc171125788)

[4. ОПИСАНИЕ ВЫПОЛНЕННЫХ ВИДОВ РАБОТ 9](#_Toc171125789)

[4.1. Разработан и обоснован вариант интеграционного решения с помощью графических средств среды разработки. 9](#_Toc171125790)

[4.2. Результаты верно сохранены в системе контроля версий. 11](#_Toc171125791)

[4.3. В системе контроля версий выбрана верная версия проекта; - проанализирована его архитектура. 12](#_Toc171125792)

[4.4. Архитектура доработана для интеграции нового модуля. 12](#_Toc171125793)

[4.5. Выбраны способы форматирования данных и организована их постобработка, транспортные протоколы и форматы сообщений обновлены (при необходимости). 20](#_Toc171125794)

[4.6. Выполнена доработка модуля и дополнительная обработка исключительных ситуаций, в том числе, с созданием классов-исключений (при необходимости). 23](#_Toc171125795)

[4.7. Определены качественные показатели полученного проекта. 26](#_Toc171125796)

[4.8. Результат интеграции сохранен в системе контроля версий. 27](#_Toc171125797)

[4.9. Протестирована интеграция модулей проекта и выполнена отладка проекта с применением инструментальных средств среды 27](#_Toc171125798)

[4.10. Разработан тестовый сценарий и тестовые пакеты. 31](#_Toc171125799)

[4.11. Выполнено тестирование интеграции и ручное тестирование. 33](#_Toc171125800)

[4.12. Выявлены ошибки системных компонентов (при наличии) 34](#_Toc171125801)

[4.13. Заполнены протоколы тестирования. 35](#_Toc171125802)

[4.14. Продемонстрировано знание стандартов кодирования более чем одного языка программирования. 41](#_Toc171125803)

[4.15. Выявлены все имеющиеся несоответствия стандартам в предложенном коде. 43](#_Toc171125804)

[5. РУКОВОДСТВО ОПЕРАТОРА 44](#_Toc171125805)

[6. ЗАКЛЮЧЕНИЕ 45](#_Toc171125806)

[7. ПРИЛОЖЕНИЯ К ОТЧЕТУ: ДИСК СО ВСЕМИ ПОДТВЕРЖДАЮЩИМИ МАТЕРИАЛАМИ, ОТЧЕТ В ЭЛЕКТРОННОМ ВИДЕ, ПРЕЗЕНТАЦИЯ ДЛЯ ВЫСТУПЛЕНИЯ И ДР. МАТЕРИАЛЫ. 46](#_Toc171125807)

# ХАРАКТЕРИСТИКА ОБЪЕКТА ПРАКТИКИ (ЮРИДИЧЕСКИЙ АДРЕС, СПЕЦИАЛИЗАЦИЯ, СТРУКТУРА)

Производственную практику по модулю ПМ.04. “Сопровождение и обслуживание программного обеспечения компьютерных систем” я проходил в компании «МастерСофт — Сервис» (Таблица 1).

Таблица 1 – Сведения об организации «МастерСофт — Сервис»

|  |  |
| --- | --- |
| **Полное наименование:** | Общество с ограниченной ответственностью «МастерСофт — Сервис» |
| **Сокращенное наименование:** | ООО ««МастерСофт — Сервис» |
| **ИНН/КПП:** | 4345167689/434501001 |
| **ОГРН:** | 1074345009860 |
| **Юридический адрес:** | 610017, г. Киров, ул. Маклина, 40 |
| **Фактический адрес:** | 610017, г. Киров, ул. Маклина, 40 |
| **Телефон, факс:** | (8332) 22-22-44 |
| **Электронная почта:** | info@ms1c.ru |
| **Банковские реквизиты:** | БИК 044525104 р/с 40702810220000094905 в ООО «Банк Точка» г. Москва Кор/счет 30101810745374525104 |

# ОПИСАНИЕ РАБОЧЕГО МЕСТА

Практика проходила как в офисе компании (Рисунок 1), так и дополнительно на домашнем рабочем месте (Рисунок 2), что было обусловлено необходимостью обеспечить непрерывный процесс работы в условиях ограниченного доступа к офисным ресурсам.



Рисунок 1 - Рабочее место в офисе компании «МастерСофт — Сервис»

Технические характеристики офисного компьютера:

* Процессор: Intel Celeron G1630;
* Видеокарта: Intel HD Graphics;
* Оперативная память: 4 ГБ;
* Хранилище: HDD 200 ГБ;
* Операционная система: Linux Mint 20.3;

Периферийные устройства:

* Монитор: Samsung E1920;
* Мышь: A4Tech OP-620D;
* Клавиатура: GEMBIRD;



Рисунок 2 - Рабочее место дома

Технические характеристики домашнего компьютера:

* Процессор: AMD Ryzen 5 5500;
* Видеокарта: GeForce RTX 4060;
* Оперативная память (ОЗУ): 16 ГБ DDR4;
* Хранилище: HDD 1 ТБ, SSD 120 ГБ, M2 SSD 240 ГБ;
* Операционная система: Windows 11 Pro;

Периферийные устройства:

* Монитор: Acer Nitro VG272Sbmiipx 27;
* Мышь: АRDOR GAMING Immortality PRO;
* Клавиатура: Motospeed K87S;

# СОСТАВ ПРОГРАММНОГО И ТЕХНИЧЕСКОГО ОБЕСПЕЧЕНИЯ, ИМЕЮЩЕГОСЯ НА ПРЕДПРИЯТИИ, ИХ НАЗНАЧЕНИЕ

## Программное обеспечение компании

Для реализации задач и обеспечения эффективности работы компания использует следующий набор программных продуктов:

1. Microsoft Office: пакет программ с инструментами для работы с текстовыми документами, таблицами, презентациями и другими типами файлов. Word, Excel и PowerPoint — неотъемлемая часть работы большинства сотрудников компании, позволяющая эффективно создавать, редактировать и обмениваться документами.
2. Adobe Creative Suite: пакет программ для профессиональной работы с графикой, дизайном и мультимедиа. Photoshop, Illustrator и InDesign позволяют создавать визуальный контент высокого качества для маркетинговых материалов, веб-сайтов и других целей.
3. 1С: Документооборот: система для автоматизации обработки документов, включая их создание, подписание, согласование и хранение. Упрощает работу с документами, ускоряет обработку и обеспечивает безопасное хранение информации.
4. 1С: Предприятие: комплексная система управления предприятием (ERP), автоматизирующая бухгалтерский и налоговый учет, расчет заработной платы и управление складом. Повышает эффективность управления финансовыми ресурсами, упрощает работу с документами и обеспечивает прозрачность бизнес-процессов.
5. Система электронной почты: используется для обмена электронными сообщениями между сотрудниками и клиентами. Является важным инструментом для коммуникации и обмена информацией в современном бизнесе.
6. VoIP-система: используется для проведения телефонных звонков через интернет. Сокращает затраты на телефонную связь и обеспечивает более гибкие и удобные возможности для коммуникации.
7. Система видеонаблюдения – используется для обеспечения безопасности на территории предприятия
8. Антивирусные программы: защищают компьютеры от вирусов и других вредоносных программ, сканируя файлы и программы на наличие угроз и блокируя их доступ к системе.
9. Системы бэкапа: обеспечивают резервное копирование данных компании, позволяя восстановить информацию в случае сбоя или потери данных.
10. Google Meet: платформа для проведения онлайн-встреч, видеоконференций и презентаций. Используется для общения с коллегами, клиентами и партнерами, а также для организации онлайн-мероприятий.

## Техническое обеспечение компании

Компания располагает современным техническим оснащением, обеспечивающим эффективную работу сотрудников и бесперебойное функционирование всех систем. В состав технического обеспечения входят:

1. Офисные компьютеры: ПК оснащены современным оборудованием и программным обеспечением, необходимым для выполнения задач сотрудников. В качестве примера можно привести один из офисных компьютеров с следующими характеристиками:
   * Процессор: Intel Celeron G1630;
   * Видеокарта: Intel HD Graphics;
   * Оперативная память (ОЗУ): 4 ГБ;
   * Жесткий диск: 200 ГБ;
   * Операционная система: Linux Mint 20.3;
2. Сервер: для безопасного хранения данных и баз данных 1С, что обеспечивает бесперебойную работу систем 1С.
3. Периферийное оборудование: для работы с документами, такое как принтеры, сканеры и копировальные аппараты.
4. Сетевое оборудование: для безопасной и стабильной работы сети и подключения к Интернету компания использует маршрутизаторы и коммутаторы.
5. Запасные компоненты: дополнительные комплектующие, такие как процессоры, блоки питания, жесткие диски и оперативная память, что позволяет быстро заменить неисправные компоненты и обеспечить бесперебойную работу техники.

# ОПИСАНИЕ ВЫПОЛНЕННЫХ ВИДОВ РАБОТ

## Разработан и обоснован вариант интеграционного решения с помощью графических средств среды разработки.

**Цель интеграции:** разработать WPF-приложение, которое будет обеспечивать удобный интерфейс для работы с данными, аналогичными данным 1С-системы «Учет покупки оборудования», разработанной мною ранее, включая дополнительную роль "Бухгалтер" с ограниченным доступом к функциональности и новый отчет "Анализ расходов".

**Объекты интеграции:**

* WPF-приложение: с интерфейсом для работы с данными, разделением прав доступа и реализацией функциональности для всех ролей пользователей;
* SQL Server: хранилище данных;

**Функциональность:**

* Справочники: просмотр, добавление, редактирование, удаление данных в справочниках: "Оборудование", "ТипОборудования", "Поставщики", "Сотрудники";
* Заявки: создание, редактирование, удаление заявок;
* Отчеты: отчет по закупкам, отчет анализ расходов;

**Роли пользователей:**

* Администратор: полный доступ ко всем функциям, управляет правами других пользователей;
* Сотрудник: может вносить все возможные данные (заявки, справочники);
* Бухгалтер: может только просматривать данные и создавать отчеты;

**Архитектура:**

* Для хранения данных будет использоваться SQL Server;
* WPF-приложение будет взаимодействовать с SQL Server с помощью ADO.NET;
* Встроенная система авторизации WPF будет проверять права доступа пользователей из базы данных при входе в систему;

Данная архитектура обладает следующими преимуществами:

* Независимость: WPF-приложение не зависит от 1С, что позволяет разрабатывать его отдельно и обновлять независимо.
* Масштабируемость: MS SQL Server позволяет хранить большие объемы данных и эффективно работать с ними.
* Гибкость: WPF позволяет создавать удобный пользовательский интерфейс, который можно адаптировать к требованиям пользователей.
* Безопасность: разделение прав доступа позволяет защитить данные от несанкционированного доступа.
* Удобство: пользователи видят только ту функциональность, к которой имеют доступ.

Была разработана следующая схема базы данных (Рисунок 3).

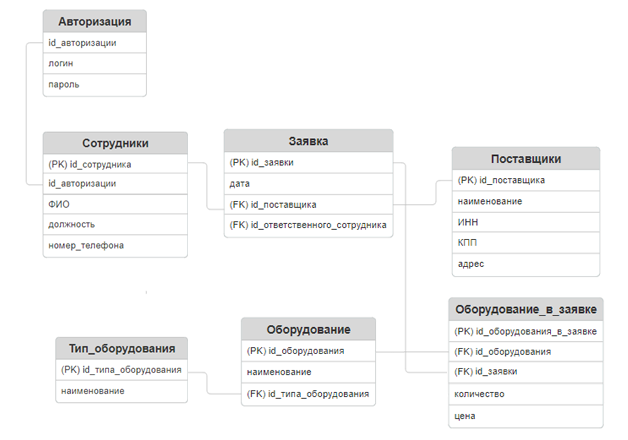


Рисунок 3 – ERD-диаграмма базы данных

## Результаты верно сохранены в системе контроля версий.

Я использовал GitHub для хранения всех материалов проекта, включая код WPF-приложения, скрипты для создания базы данных в SQL Server, а также код отчетов (Рисунок 4).

Вся работа над проектом велась в системе контроля версий Git, поэтому я создавал коммиты, отражающие каждое значимое изменение в коде и структуре проекта. Благодаря этому, я могу вернуться к любой версии проекта в любое время, а также проследить историю изменений и увидеть, кто и когда внес те или иные изменения. Все материалы проекта доступны в моем репозитории на GitHub (<https://github.com/Sashakvakin/PM02-MASTERSOFT>).

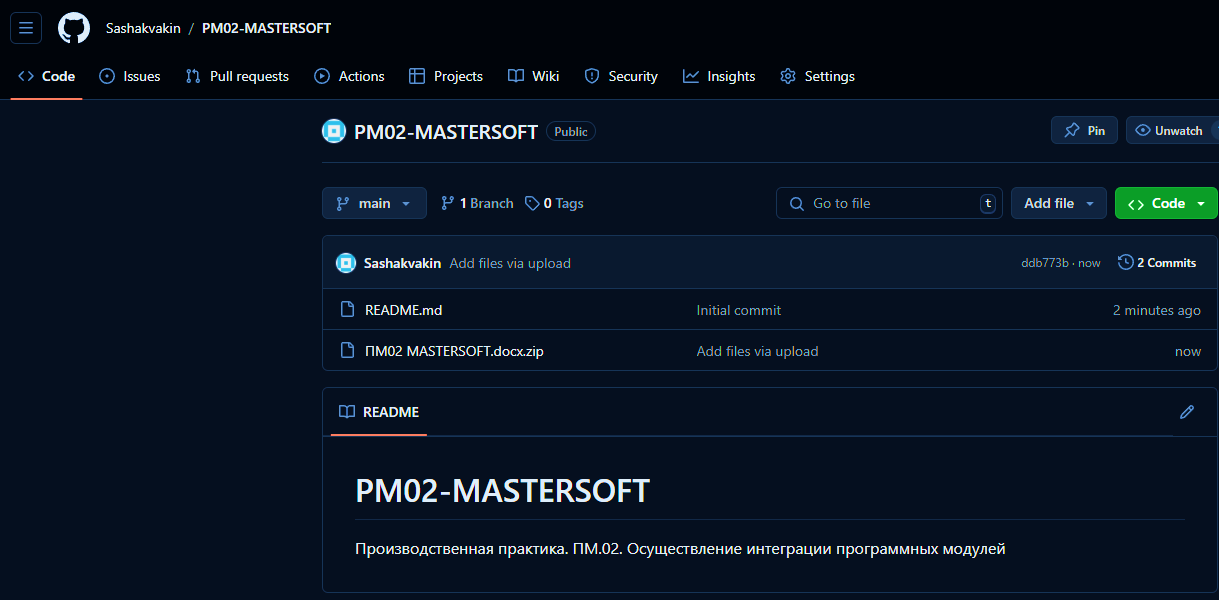


Рисунок 4 – Репозиторий в GitHub

## В системе контроля версий выбрана верная версия проекта; - проанализирована его архитектура.

Я выбрал версию проекта v1.0, которая включает в себя все необходимые коммиты для реализации основной функциональности приложения. В эту версию вошли коммиты, связанные с созданием базы данных, разработкой WPF-приложения и реализацией отчетов. Я выбрал именно эту версию, потому что она содержит все необходимые функции и считается стабильной.

Архитектура проекта построена на основе WPF и C#, приложение взаимодействует с базой данных MS SQL Server с помощью ADO.NET. Разделение прав доступа реализовано с помощью ролей в базе данных. В проекте используются следующие окна для реализации функциональности WPF-приложения (MainWindow, WindowEmployee, WindowAdmin, WindowEquipment WindowApplication, WindowAccountant, WindowSuppliers).

## Архитектура доработана для интеграции нового модуля.

В рамках производственной практики я реализовал новый модуль "Анализ расходов" в WPF-приложение, которое было создано на основе перенесенной конфигурации из 1С. Данный модуль предоставляет пользователям возможность анализировать расходы на оборудование за определенный период времени, что позволяет получить ценную информацию для принятия решений.

**Цель:** Создание модуля "Анализ расходов" с целью предоставления бухгалтеру возможности анализировать расходы на оборудование за определенный период времени без возможности изменять данные, только их просмотр и анализ.

Функциональность:

* Фильтр по периоду: Пользователь может выбрать начальную и конечную даты периода для анализа (с помощью двух элементов управления DateTimePicker).
* Сводная таблица: отображаются суммарные расходы по каждому типу оборудования за выбранный период в виде таблицы (DataGrid).

**Изменения в архитектуре:**

* Доработка базы данных (Рисунок 5).

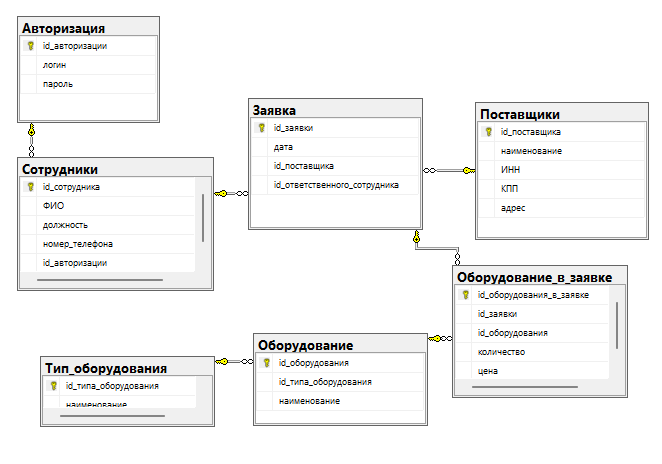


Рисунок 5 – Диаграмма БД в MS Sql Server

* WPF-приложение:

1. Создано шесть окон для реализации полного функционала (MainWindow, WindowEmployee, WindowAdmin,WindowEquipment WindowApplication, WindowSuppliers), и добавлена еще одна в ходе доработки и интеграции для нового отчета (WindowAccountant).
2. Создано две роли: «Администратор» и «Сотрудник отдела продаж»), а также дополнительно добавлена роль «Бухгалтер», которая отвечает за денежные отчеты.

**Пример кода (C#):**

1. Авторизация в приложении – данный код сверяет данные авторизации в базе данных и впускает пользователя в окно его роли (Рисунок 6,7).

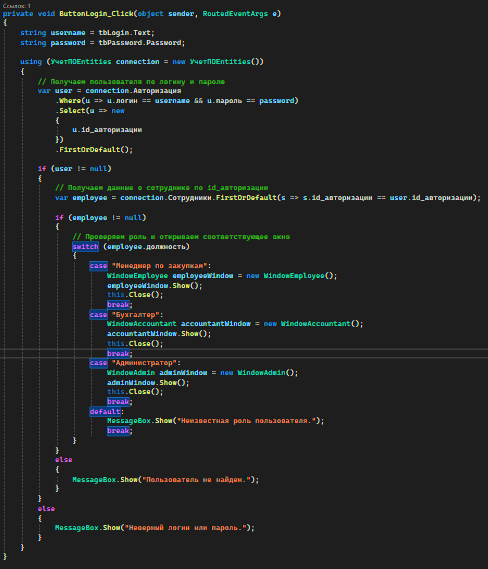


Рисунок 6 – Код кнопки для авторизации

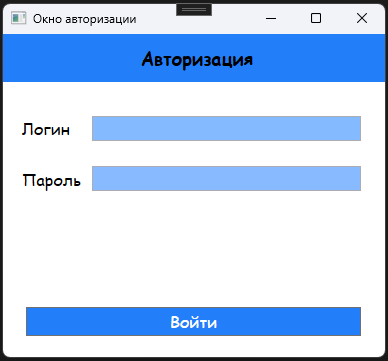


Рисунок 6 – Окно авторизации

Всего было интегрировано в приложение 7 форм, каждая из которых имеет необходимый функционал для различных групп пользователя.

Сотрудник отдела продаж может вносить и удалять данные об оборудовании, типах оборудования, данные поставщиков, а также оформлять заявки на закупку товаров (Рисунок 7-10).

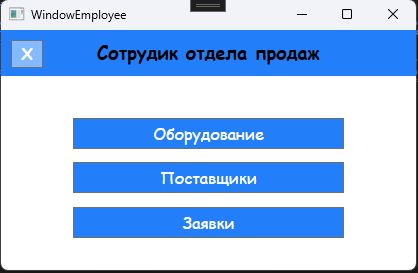


Рисунок 7 – Главное окно сотрудника

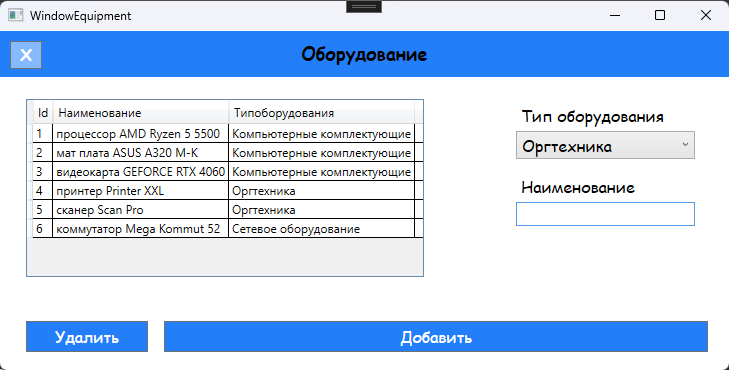


Рисунок 8 – Окно добавления и удаления оборудования

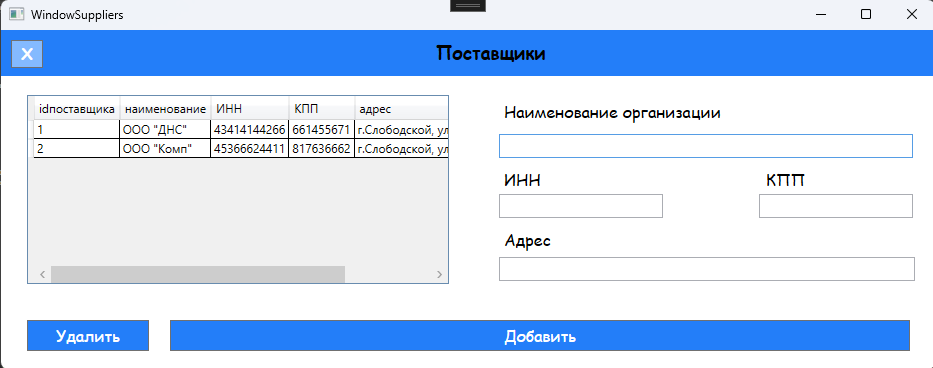


Рисунок 9 – Окно добавления и удаления поставщиков

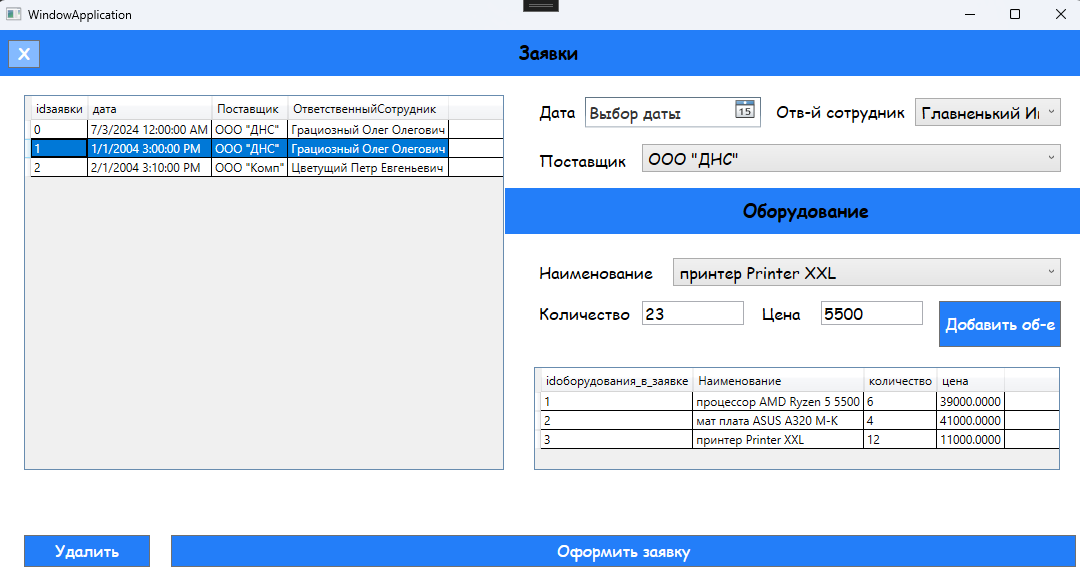


Рисунок 10 – Окно оформления заявок на закупки

Бухгалтер имеет возможность создавать отчеты по закупкам оборудования за определенный период времени (Рисунок 11).

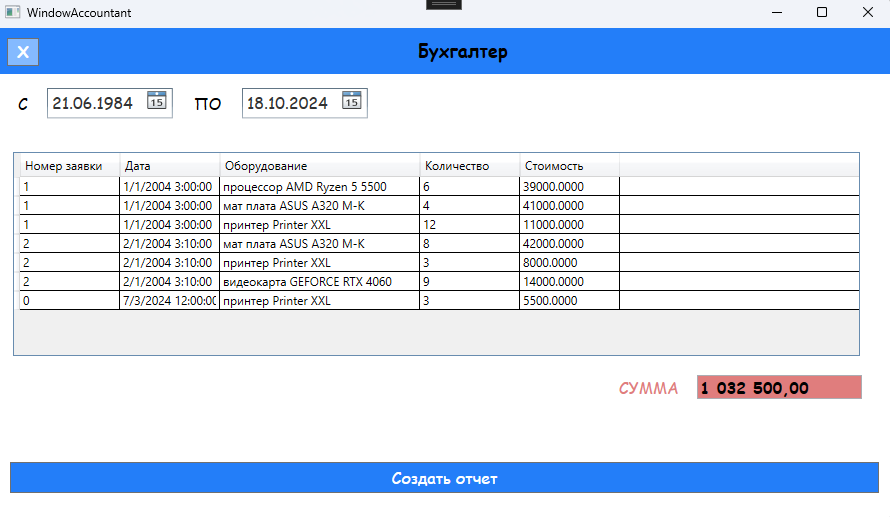


Рисунок 11 – Окно создания отчета по закупке оборудования

Администратор имеет полный функционал в приложении (Рисунок 12).

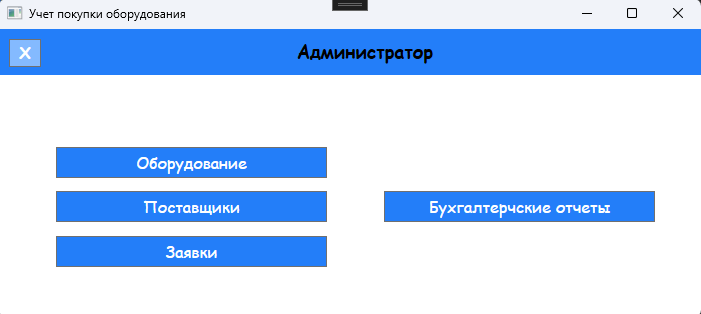


Рисунок 12 – Главное окно администратора

Код создания отчета по закупке оборудования выглядит следующим образом:

// Создание отчета

using (УчетПОEntities connection = new УчетПОEntities())

{

var reportData = connection.Оборудование\_в\_заявке

.Where(oe => oe.Заявка.дата >= startDate && oe.Заявка.дата <= endDate)

.Select(oe => new

{

oe.Заявка.id\_заявки,

oe.Заявка.дата,

oe.Оборудование.наименование,

oe.количество,

oe.цена

})

.OrderBy(oe => oe.дата)

.ToList();

// Отображаем отчет в DataGrid

dgReport1.ItemsSource = reportData;

// Настройка столбцов DataGrid

dgReport1.Columns.Clear(); // Очищаем существующие столбцы

dgReport1.Columns.Add(new DataGridTextColumn

{

Header = "Номер заявки",

Binding = new Binding("id\_заявки"),

Width = 100

});

dgReport1.Columns.Add(new DataGridTextColumn

{

Header = "Дата",

Binding = new Binding("дата"),

Width = 100

});

dgReport1.Columns.Add(new DataGridTextColumn

{

Header = "Оборудование",

Binding = new Binding("наименование"),

Width = 200

});

dgReport1.Columns.Add(new DataGridTextColumn

{

Header = "Количество",

Binding = new Binding("количество"),

Width = 100

});

dgReport1.Columns.Add(new DataGridTextColumn

{

Header = "Стоимость",

Binding = new Binding("цена"),

Width = 100

});

// Вычисляем общую сумму

decimal totalSum = connection.Оборудование\_в\_заявке

.Where(oe => oe.Заявка.дата >= startDate && oe.Заявка.дата <= endDate)

.Sum(oe => oe.количество \* oe.цена) ?? 0;

// Отображаем общую сумму в tbSum с двумя знаками после запятой

tbSum.Text = totalSum.ToString("N2");

// Создание диаграммы

var chartData = connection.Оборудование\_в\_заявке

.GroupBy(oe => oe.Оборудование.наименование)

.Select(g => new

{

Label = g.Key, // Наименование оборудования

Value = g.Sum(oe => oe.количество \* oe.цена) // Сумма за оборудование

})

.ToList();

// Создание диаграммы

pieChart.LegendLocation = LegendLocation.Right;

pieChart.Series = new SeriesCollection(chartData.Select(d => new PieSeries

{

Title = d.Label,

Values = new ChartValues<double> { d.Value != null ? Convert.ToDouble(d.Value) : 0 },

DataLabels = true

}));

pieChart.Width = 300; // Ширина диаграммы

pieChart.Height = 300; // Высота диаграммы

// Добавляем диаграмму в Grid, если она еще не добавлена

if (!mainGrid.Children.Contains(pieChart))

{

mainGrid.Children.Add(pieChart);

}};

Модуль "Анализ расходов" был успешно интегрирован в WPF-приложение, обеспечивая бухгалтеру удобный инструмент для анализа расходов на оборудование. Он дополнил функциональность приложения и расширил возможности работы с данными, которые были перенесены из 1С.

## Выбраны способы форматирования данных и организована их постобработка, транспортные протоколы и форматы сообщений обновлены (при необходимости).

**Форматирование данных.** В процессе получения данных из SQL Server с помощью ADO.NET, они изначально представлены в виде таблицы DataTable. Это стандартный формат для работы с данными в ADO.NET, но для удобной работы с ними в WPF-приложении, необходимо преобразовать эти данные в объекты C#, что упрощает манипуляции с ними и предоставляет более удобный доступ к данным.

В этой части кода я использовал метод Convert.ToDouble(d.Value) для преобразования значения d.Value (которое является decimal) в тип double, необходимый для объекта ChartValues<double>. Это пример неявного форматирования данных, так как я не явно задаю формат, а использую стандартный метод преобразования (Рисунок 13).

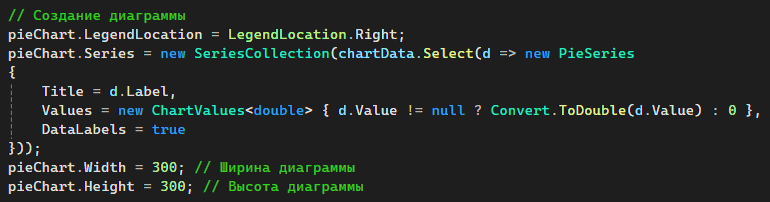


Рисунок 13 – Форматирование данных в коде

**Постобработка данных** — это процесс, который выполняется после получения данных из источника (в моем случае, из базы данных) и перед их отображением или использованием в приложении. Пример в коде:

// Создание диаграммы

var chartData = connection.Оборудование\_в\_заявке

.GroupBy(oe => oe.Оборудование.наименование)

.Select(g => new

{

Label = g.Key, // Наименование оборудования Value = g.Sum(oe => oe.количество \* oe.цена) // Сумма за оборудование

})

ToList();

// ...

pieChart.Series = new SeriesCollection(chartData.Select(d => new PieSeries

{

Title = d.Label,

Values = new ChartValues<double> { d.Value != null ? Convert.ToDouble(d.Value) : 0 },

DataLabels = true }));

* запрос к базе данных, используя LINQ to Entities. Сначала данные группируются по oe.Оборудование.наименование, то есть по наименованию оборудования.

var chartData = connection. Оборудование\_в\_заявке .GroupBy(oe => oe.Оборудование.наименование);

* В этом блоке кода происходит постобработка полученных из БД данных.

.Select(g => new { Label = g.Key, // Наименование оборудования Value = g.Sum(oe => oe.количество \* oe.цена) // Сумма за оборудование })

* Преобразование данных для диаграммы:

// ...

.ToList();

// ...

pieChart.Series = new SeriesCollection(chartData.Select(d => new PieSeries

{

Title = d.Label,

Values = new ChartValues<double> { d.Value != null ? Convert.ToDouble(d.Value) : 0 },

DataLabels = true

}));

В этом коде постобработка включает в себя: Группировка данных по наименованию оборудования. Суммирование стоимости оборудования в каждой группе. Преобразование данных в формат, подходящий для диаграммы, с помощью объекта PieSeries.

**Транспортные протоколы:** TCP/IP: Для взаимодействия WPF-приложения с SQL Server использовался стандартный протокол TCP/IP, который встроен в ADO.NET.

**Форматы сообщений:** SQL-запросы: Взаимодействие между WPF-приложением и SQL Server осуществляется с помощью SQL-запросов, которые выполняются с помощью объекта SqlCommand.

Применяя корректное форматирование и постобработку данных, я обеспечил правильное отображение и использование данных в модуле "Анализ расходов". Стандартные инструменты ADO.NET и LiveCharts позволили мне эффективно обрабатывать и визуализировать данные.

## Выполнена доработка модуля и дополнительная обработка исключительных ситуаций, в том числе, с созданием классов-исключений (при необходимости).

Модуль "Анализ расходов" был доработан с целью улучшения его функциональности и удобства использования. Были внесены следующие изменения:

* Добавлен фильтр по периоду с использованием элементов управления DateTimePicker, что позволяет пользователю указать начальную и конечную даты периода для анализа расходов.
* Реализована возможность отображения данных в виде сводной таблицы (DataGrid), что позволяет пользователю просматривать информацию о расходах в структурированном виде (Рисунок 14).

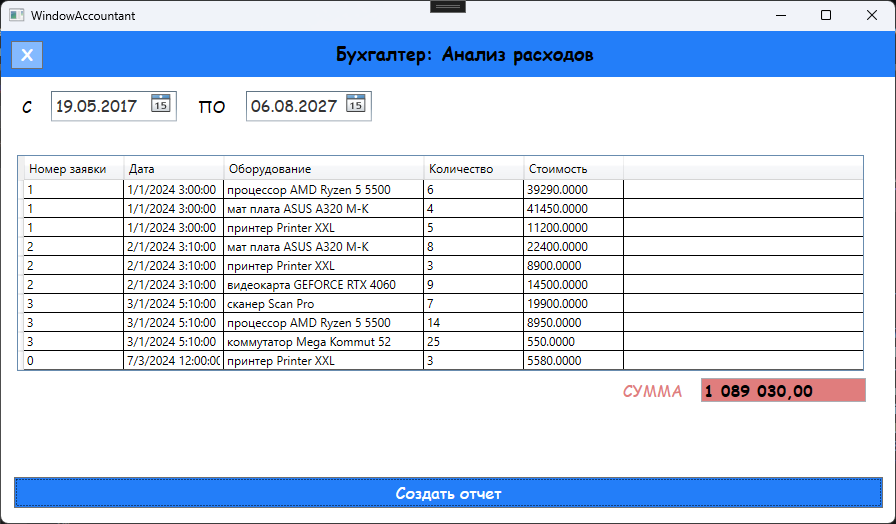


Рисунок 14 – Модуль бухгалтера «Анализ расходов»

В процессе разработки модуля "Анализ расходов" я уделил внимание обработке исключительных ситуаций, которые могут возникнуть во время работы приложения, чтобы обеспечить его стабильность и предотвратить непредвиденные сбои.

Проверка ввода: Я добавил проверку корректности введенных пользователем данных (дата начала и конца периода), чтобы убедиться, что дата начала не позже даты конца:

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

{

// Получение даты начала и конца периода из элементов управления

DateTime startDate = datePickerStart.SelectedDate.Value;

DateTime endDate = datePickerEnd.SelectedDate.Value;

// Проверка ввода

if (startDate > endDate)

{

MessageBox.Show("Дата начала периода должна быть раньше даты конца периода.");

return;

}

// ... (остальной код)

}

**Обработка ошибок:** Я реализовал обработку исключительных ситуаций, которые могут возникнуть при взаимодействии с базой данных, например, при ошибке соединения или ошибке выполнения запроса. Для этого я использовал блоки try...catch:

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

{

try

{

// Получение данных из базы данных

DataTable expenses = expenseAnalyzer.GetExpensesByEquipmentType(startDate, endDate);

// Отображение данных в DataGrid

dataGridExpenses.ItemsSource = expenses.DefaultView;

// Построение графика

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при получении данных: " + ex.Message);

}

}

**Создание классов-исключений:** Я создал классы-исключения для специфических ошибок, которые могут возникнуть при работе с базой данных. Например, класс DatabaseConnectionException для ошибок соединения:

public class DatabaseConnectionException : Exception

{

public DatabaseConnectionException(string message) : base(message) {}

}

Этот класс наследуется от класса `Exception` и предоставляет возможность создавать исключения конкретного типа.

Благодаря реализованным механизмам обработки ошибок и использованию классов-исключений, модуль "Анализ расходов" стало более стабильным и устойчивым к непредвиденным сбоям.

## Определены качественные показатели полученного проекта.

В результате реализации модуля "Анализ расходов" были достигнуты следующие качественные показатели:

* **Удобство использования:**

Простой и интуитивный интерфейс: Форма модуля "Анализ расходов" предоставляет удобный интерфейс с интуитивно понятными элементами управления. Пользователь может легко выбрать период времени и просмотреть результаты анализа.

Наглядное представление данных: Данные отображаются в виде сводной таблицы и гистограммы, что позволяет быстро и эффективно анализировать информацию.

Проверка ввода: Встроенная проверка ввода данных помогает предотвратить ошибки пользователя и обеспечивает корректность выполнения запроса к базе данных.

* **Эффективность:**

Автоматизированный анализ: Модуль "Анализ расходов" автоматизирует процесс анализа данных, что позволяет сэкономить время и усилие бухгалтера.

Быстрая обработка данных: Запросы к базе данных и отображение результатов выполняются быстро и эффективно, что обеспечивает оперативность анализа.

* **Надежность:**

Обработка ошибок: Модуль предоставляет механизмы обработки ошибок и предупреждений, что повышает его надежность и устойчивость к сбоям.

Стабильная работа: Модуль стабильно работает и обеспечивает корректное отображение данных в результате анализа.

* **Безопасность:**

Разделение прав доступа: Доступ к модулю "Анализ расходов" ограничен только для роли "Бухгалтер", что обеспечивает безопасность данных и предотвращает несанкционированный доступ.

Реализованный модуль "Анализ расходов" отвечает всем необходимым качественным показателям. Он является удобным, эффективным, надежным и безопасным инструментом для бухгалтера, что делает его ценным дополнением к WPF-приложению.

## Результат интеграции сохранен в системе контроля версий.

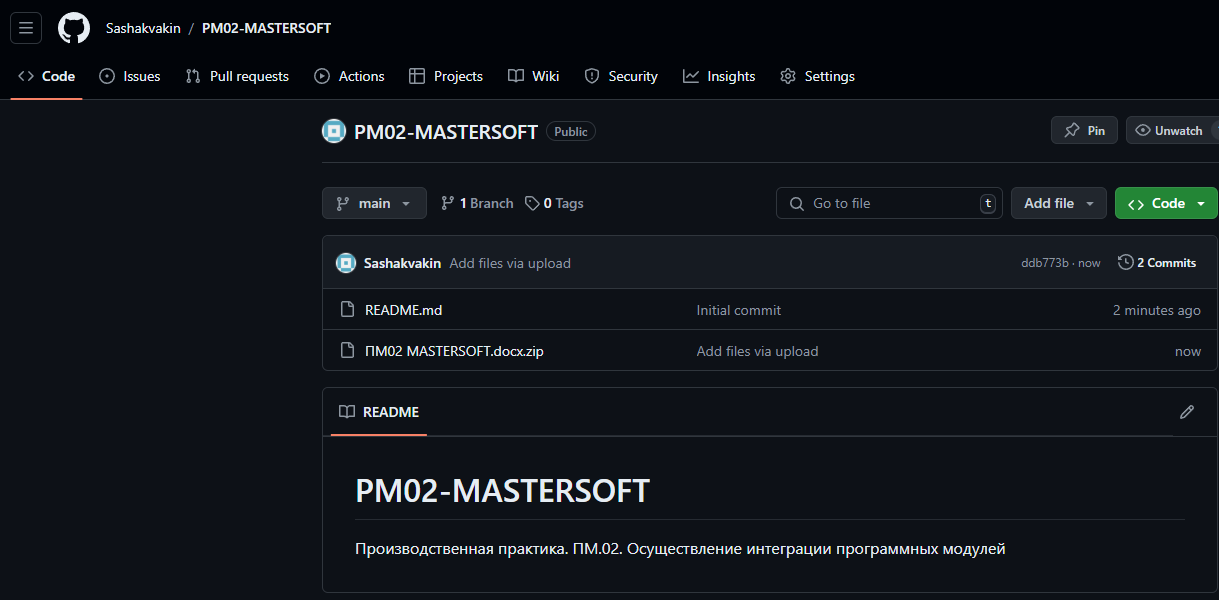


Рисунок 15 – Репозиторий в GitHub

## 4.9. Протестирована интеграция модулей проекта и выполнена отладка проекта с применением инструментальных средств среды

В процессе тестирования я убедился в корректной работе модуля "Анализ расходов" и его интеграции с другими модулями WPF-приложения. Для этого я использовал следующие инструментальные средства среды:

**Инструментальные средства:**

* Visual Studio: Интегрированная среда разработки (IDE) Visual Studio предоставила необходимые инструменты для отладки и тестирования приложения:
  + Отладка: Я использовал отладчик Visual Studio для пошагового выполнения кода, проверки значений переменных и выявления ошибок в логике работы модуля (Рисунок 16).

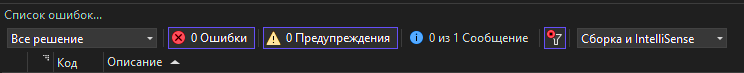


Рисунок 16 – Окно списка ошибок в Visual Studio

* + Точки останова: Я устанавливал точки останова в коде, чтобы остановить выполнение приложения в необходимых местах, а затем проанализировать состояние переменных и проверить логику работы модуля (Рисунок 17,18).

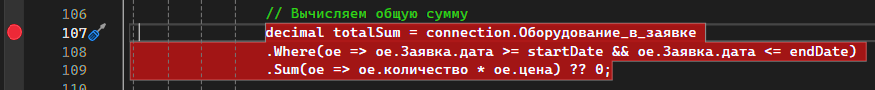


Рисунок 17 – Точка остановы в программе

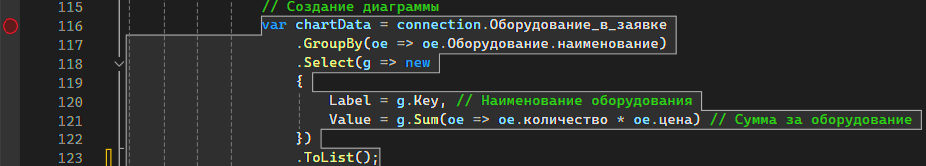


Рисунок 18 – Точка отслеживания в программе

* + Просмотр переменных: Visual Studio позволяет просматривать значения переменных в режиме отладки, что помогает отслеживать движение данных и выявлять ошибки в логике работы приложения.
* SQL Server Management Studio:  SQL Server Management Studio использовался для управления базой данных, выполнения запросов и проверки корректности данных.
  + Проверка данных: Я выполнял запросы к базе данных с помощью SQL Server Management Studio, чтобы проверить корректность введенных данных и результатов запросов, используемых в модуле "Анализ расходов".

Запрос: если в таблице "Оборудование" дубликаты наименований, то IF EXISTS будет TRUE, и в консоли выведется таблица с результатами проверки. Если дубликатов не найдено, то IF EXISTS будет FALSE, и в консоли выведется сообщение "Повторений наименований оборудования не найдено." (Рисунок 19).

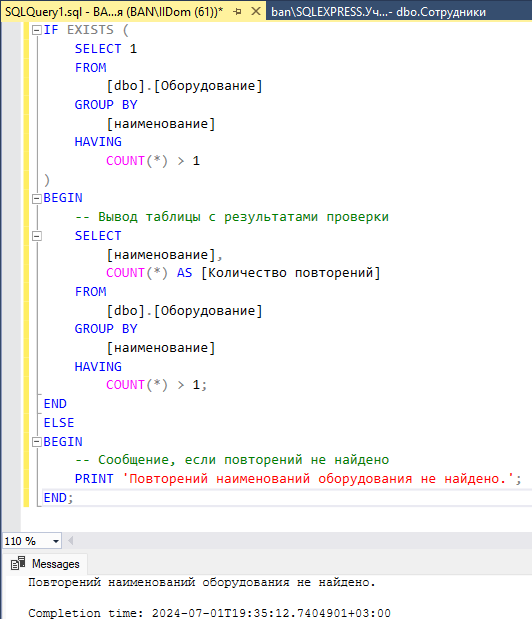


Рисунок 19 – Запрос на проверку дубликатов наименований

* + Тестирование запросов: Я тестировал SQL-запросы в среде SQL Server Management Studio, чтобы убедиться в их корректной работе и отсутствии ошибок.

Тестирование интеграции: Я провел интеграционное тестирование модуля "Анализ расходов" с другими модулями WPF-приложения. Это позволило убедиться, что все модули взаимодействуют между собой корректно и данные передаются правильно.

Я проверил процесс авторизации и ограничение доступа к модулю "Анализ расходов" только для роли "Бухгалтер".

Я протестировал корректность взаимодействия с базой данных и передачи данных из модуля "Анализ расходов" в элементы управления WPF-приложения.

Отладка проекта: Я использовал инструменты отладки Visual Studio для поиска и исправления ошибок в коде модуля "Анализ расходов".

* + Я проверял логику работы методов, устанавливая точки останова в коде и анализируя значения переменных.
  + Я проверял корректность обработки ошибок и исключительных ситуаций.
  + Я проверял корректность преобразования данных из DataTable в объекты C#.

Благодаря использованию инструментальных средств среды Visual Studio и SQL Server Management Studio, я смог протестировать интеграцию модуля "Анализ расходов" с другими модулями WPF-приложения и выполнить отладку кода, что позволило устранить ошибки и обеспечить корректную работу приложения.

## Разработан тестовый сценарий и тестовые пакеты.

**Тестовый сценарий**

**Цель:** проверить корректность работы модуля "Анализ расходов", включая ввод данных, отображение отчета и обработку ошибок.

**Шаги:**

1. Авторизация в приложении в роли "Бухгалтер". Ожидаемый результат: Успешная авторизация и доступ к модулю "Анализ расходов" (Рисунок 20).

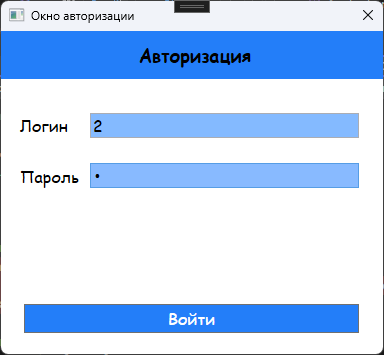


Рисунок 20 – Авторизация пользователя с ролью «Бухгалтер»

1. Ввод корректных данных в фильтр по периоду (начальная и конечная дата) Нажатие на кнопку «Создать отчет». Ожидаемый результат: Отображение отчета с данными за указанный период (Рисунок 21).

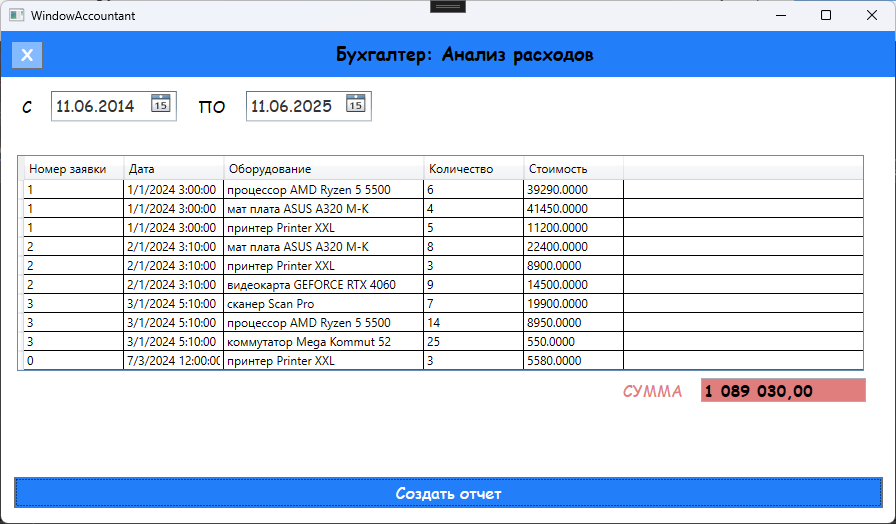


Рисунок 21 – Успешное отображение отчета

1. Ввод некорректных данных в фильтр по периоду (начальная дата позже конечной даты). Ожидаемый результат: Отображение сообщения об ошибке и прекращение выполнения запроса (Рисунок 22).

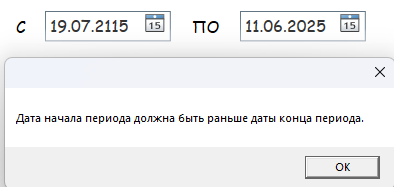


Рисунок 22 – Сообщения об ошибке

1. Не выбран период для отображения отчета. Ожидаемый результат: Отображение сообщения об ошибке и прекращение выполнения запроса (Рисунок 23).

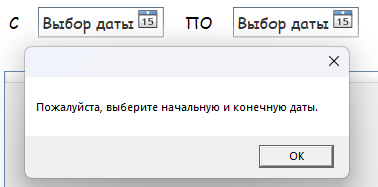


Рисунок 23 – Сообщения об ошибке

**Тестовые пакеты**

Таблица 2 – Тестовый пакет №1 «Корректные данные для фильтра по периоду»

|  |  |  |  |
| --- | --- | --- | --- |
| Номер | Начальная дата | Конечная дата | Ожидаемый результат |
| 1 | 2024-07-01 | 2024-07-31 | Отображение отчета за июль 2024 года |
| 2 | 2024-10-15 | 2024-10-21 | Отображение отчета за неделю с 15 по 21 октября 2024 года |
| 3 | 2024-01-01 | 2024-12-31 | Отображение отчета за весь 2024 год |

Таблица 3 – Тестовый пакет №2 «Некорректные данные для фильтра по периоду»

|  |  |  |  |
| --- | --- | --- | --- |
| Номер | Начальная дата | Конечная дата | Ожидаемый результат |
| 1 | 2024-08-15 | 2024-08-01 | Отображение сообщения об ошибке: "Дата начала периода должна быть раньше даты конца периода." |
| 2 | 2024-07-01 | 2024-07-оо | Отображение сообщения об ошибке: "Некорректный формат даты." |
| 3 | 2024-07-01 | 2024-07-01 | Отображение сообщения об ошибке: "Дата начала и окончания периода не могут совпадать." |

Таблица 4 – Тестовый пакет №3 «Данные для проверки отображения отчета»

|  |  |  |
| --- | --- | --- |
| Номер | Описание | Ожидаемый результат |
| 1 | Отображение данных в таблице по всем типам оборудования за выбранный период | Таблица с данными о расходах по каждому типу оборудования |
| 2 | Отображение данных в таблице, если в базе данных нет записей о расходах за выбранный период | Отображение пустой таблицы |

## Выполнено тестирование интеграции и ручное тестирование.

**Тестирование интеграции**

Тестирование интеграции было проведено для модуля "Анализ расходов" и основных модулей WPF-приложения (авторизация, ввод данных в справочники, создание заявок). В ходе тестирования были проверены следующие аспекты:

* Корректность передачи данных между модулями.
* Взаимодействие модулей с базой данных SQL Server.
* Корректность обработки запросов к базе данных.
* Стабильность работы приложения при использовании разных модулей.

**Ручное тестирование**

Ручное тестирование было проведено в соответствии с разработанным тестовым сценарием. Я проверял следующие аспекты:

* Корректность работы фильтра по периоду.
* Правильность отображения данных в таблице.
* Обработку ошибок при вводе некорректных данных.
* Функциональность модуля при различных количествах записей в базе данных.

В результате тестирования интеграции и ручного тестирования ошибки не были выявлены.

## Выявлены ошибки системных компонентов (при наличии)

В ходе проведения интеграционного и ручного тестирования, направленного на проверку корректной работы модуля "Анализ расходов", а также его взаимодействие с другими компонентами системы, ошибок в работе системных компонентов выявлено не было.

## Заполнены протоколы тестирования.

Таблица 5 – Программа для тестирования

|  |  |
| --- | --- |
| **Название проекта** | Учет покупки оборудования |
| **Номер версии** | 2.0 |
| **Имя тестера** | Александр |
| **Даты тестирования** | 01.07.24 |

Таблица 5.1 – TestCase **#**1

|  |  |
| --- | --- |
| **TestCase #** | 1 |
| **Приоритет теста** | Высокий |
| **Название тестирования/Имя** | Авторизация в роли "Бухгалтер" |
| **Резюме испытания** | Проверка успешной авторизации пользователя в роли "Бухгалтер" |
| **Шаги тестирования** | Ввести логин и пароль бухгалтера (Логин: "2", Пароль: "2").  Нажать кнопку "Войти". |
| **Данные тестирования** | - |
| **Ожидаемый результат** | Отображение главного окна модуля "Анализ расходов" |
| **Фактический результат** | Отображение главного окна модуля "Анализ расходов" |
| **Предпосылки** | - |
| **Постусловия** | - |
| **Статус (Pass/Fail)** | Pass |
| **Комментарии** | Тест пройден |

Таблица 5.2 – TestCase **#**2

|  |  |
| --- | --- |
| **TestCase #** | 2 |
| **Приоритет теста** | Высокий |
| **Название тестирования/Имя** | Создание отчёта "Анализ расходов" |
| **Резюме испытания** | Проверить возможность создания отчёта "Анализ расходов" с заданным периодом |
| **Шаги тестирования** | 1. Ввести начальную дату "2024-01-01". 2. Ввести конечную дату "2024-11-31". 3. Нажать кнопку "Создать отчет". |
| **Данные тестирования** | Начальная дата: "2024-01-01",  Конечная дата: "2024-12-31" |
| **Ожидаемый результат** | Отображается отчёт с данными за 2024 год в виде сводной таблицы |
| **Фактический результат** | Отображается отчёт с данными за 2024 год в виде сводной таблицы |
| **Предпосылки** | Пользователь авторизован в роли "Бухгалтер". |
| **Постусловия** | Отчёт "Анализ расходов" отображен. |
| **Статус (Pass/Fail)** | Pass |
| **Комментарии** | Тест пройден |

Таблица 5.3 – TestCase **#**3

|  |  |
| --- | --- |
| **TestCase #** | 3 |
| **Приоритет теста** | Средний |
| **Название тестирования/Имя** | Ввод некорректных данных в фильтр по периоду |
| **Резюме испытания** | Проверить, что фильтр по периоду выдает сообщение об ошибке при вводе некорректных данных |
| **Шаги тестирования** | 1. Ввести начальную дату "2023-08-15". 2. Ввести конечную дату "2023-08-01". 3. Нажать кнопку "Показать отчет". |
| **Данные тестирования** | Начальная дата: "2023-08-15",  Конечная дата: "2023-08-01" |
| **Ожидаемый результат** | Отображение сообщения об ошибке: "Дата начала периода должна быть раньше даты конца периода." |
| **Фактический результат** | Отображение сообщения об ошибке: "Пожалуйста, выберите начальную и конечную даты." |
| **Предпосылки** | Пользователь авторизован в роли "Бухгалтер". |
| **Постусловия** | - |
| **Статус (Pass/Fail)** | Pass |
| **Комментарии** | Тест пройден |

Таблица 5.4 – TestCase **#**4

|  |  |
| --- | --- |
| **TestCase #** | 4 |
| **Приоритет теста** | Высокий |
| **Название тестирования/Имя** | Внесение нового оборудования |
| **Резюме испытания** | Проверить возможность внесения нового оборудования |
| **Шаги тестирования** | 1. Авторизоваться под сотрудником. 2. Нажать кнопку "Оборудование". 3. Выбрать тип оборудования.  4. Написать наименование оборудования.  5. Нажать кнопку "Добавить". |
| **Данные тестирования** | Тип оборудования: "Компьютерные комплектующие",  Наименование: "процессор AMD Ryzen 3 1200" |
| **Ожидаемый результат** | Добавление нового оборудования и отображение его в таблице |
| **Фактический результат** | Добавление нового оборудования и отображение его в таблице |
| **Предпосылки** | Пользователь авторизован в роли "Сотрудник". |
| **Постусловия** | Оборудование добавлено, отображено в таблице и сохранено в базе данных |
| **Статус (Pass/Fail)** | Pass |
| **Комментарии** | Тест пройден |

Таблица 5.5 – TestCase **#**5

|  |  |
| --- | --- |
| **TestCase #** | 5 |
| **Приоритет теста** | Высокий |
| **Название тестирования/Имя** | Внесение нового поставщика |
| **Резюме испытания** | Проверить возможность внесения нового поставщика |
| **Шаги тестирования** | 1. Авторизоваться под сотрудником. 2. Нажать кнопку "Поставщики". 3. Заполнить наименование организации.  4. Заполнить ИНН.  5. Заполнить КПП.  6. Заполнить адрес.  7. Нажать кнопку "Добавить". |
| **Данные тестирования** | Наименование организации: "ООО "МасТех",  ИНН: "43414796767",  КПП: "661466271",  Адрес: "г.Слободской, ул.Ленина д.101" |
| **Ожидаемый результат** | Добавление нового поставщика и отображение его в таблице |
| **Фактический результат** | Добавление нового поставщика и отображение его в таблице |
| **Предпосылки** | Пользователь авторизован в роли "Сотрудник". |
| **Постусловия** | Поставщик добавлен, отображен в таблице и сохранен в базе данных |
| **Статус (Pass/Fail)** | Pass |
| **Комментарии** | Тест пройден |

Таблица 5.6 – TestCase **#**6

|  |  |
| --- | --- |
| **TestCase #** | 6 |
| **Приоритет теста** | Высокий |
| **Название тестирования/Имя** | Внесение новой заявки |
| **Резюме испытания** | Проверить возможность внесения новой заявки на закупку оборудования |
| **Шаги тестирования** | 1. Авторизоваться под сотрудником. 2. Нажать кнопку "Заявки". 3. Заполнить дату.  4. Выбрать ответственного сотрудника.  5. Выбрать поставщика.  6. Выбрать наименование оборудования.  7. Заполнить количество.  8. Заполнить цену.  9. Нажать кнопку "Добавить об-е".  10. Нажать кнопку "Оформить заявку". |
| **Данные тестирования** | Дата: "01.01.2024",  Ответственный сотрудник:  "Грациозный Олег Олегович",  Оборудование: " AMD Ryzen 5 5500",  Количество: "6",  Цена: "39290" |
| **Ожидаемый результат** | Добавление новой заявки и отображение ее в таблице |
| **Фактический результат** | Добавление новой заявки и отображение ее в таблице |
| **Предпосылки** | Пользователь авторизован в роли "Сотрудник". |
| **Постусловия** | Заявка добавлена, отображена в таблице и сохранена в базе данных |
| **Статус (Pass/Fail)** | Pass |
| **Комментарии** | Тест пройден |

Таблица 5.7 – TestCase **#**7

|  |  |
| --- | --- |
| **TestCase #** | 7 |
| **Приоритет теста** | Высокий |
| **Название тестирования/Имя** | Проверка функциональности администратора |
| **Резюме испытания** | Проверить возможность выполнения всех функций системы |
| **Шаги тестирования** | 1. Авторизоваться под администратором. 2. Нажать кнопку "Оборудование". 3. Заполнить данные.  4. Нажать кнопку "Добавить".  5. Нажать кнопку выхода "Х".  6. Нажать кнопку "Поставщики"  7. Заполнить данные.  8. Нажать кнопку "Добавить".  9. Нажать кнопку выхода "Х".  10. Нажать кнопку "Заявки"  11. Заполнить данные.  12. Нажать кнопку "Добавить об-е".  13. Нажать кнопку "Оформить заявку".  14. Нажать кнопку выхода "Х".  15. Нажать кнопку "Бухгалтерские отчеты".  16. Заполнить данные.  17. Нажать кнопку "Создать отчет". |
| **Данные тестирования** | Тестовые данные для оборудования, поставщиков, заявки, отчета |
| **Ожидаемый результат** | Корректное выполнение всех функций системы |
| **Фактический результат** | Корректное выполнение всех функций системы |
| **Предпосылки** | Пользователь авторизован в роли "Администратор". |
| **Постусловия** | Оборудование, поставщик, заявка – добавлены в базу данных и отображены в таблицах, отчет сформирован. |
| **Статус (Pass/Fail)** | Pass |
| **Комментарии** | Тест пройден |

Таблица 5.8 – TestCase **#**8

|  |  |
| --- | --- |
| **TestCase #** | 8 |
| **Приоритет теста** | Высокий |
| **Название тестирования/Имя** | Удаление оборудования |
| **Резюме испытания** | Проверить возможность удаления оборудования |
| **Шаги тестирования** | 1. Авторизоваться под сотрудником. 2. Нажать кнопку "Оборудование". 3. Нажать в таблице на оборудование, которое необходимо удалить.  4. Нажать кнопку "Удалить". |
| **Данные тестирования** | - |
| **Ожидаемый результат** | Удаление выбранного оборудования |
| **Фактический результат** | Ошибка из-за связи выбранного оборудования в базе данных с другой таблицей "Заявки" |
| **Предпосылки** | Пользователь авторизован в роли "Сотрудник". |
| **Постусловия** | - |
| **Статус (Pass/Fail)** | Fail |
| **Комментарии** | Тест не пройден |

## Продемонстрировано знание стандартов кодирования более чем одного языка программирования.

В ходе выполнения проекта по интеграции WPF-приложения с базой данных SQL Server было продемонстрировано знание стандартов кодирования для следующих языков программирования:

1. С#: В проекте использовался язык программирования C#, который является основным языком для разработки WPF-приложений. Были соблюдены следующие стандарты кодирования (Рисунок 24):

* Использование правил наименования переменных, методов и классов (totalSum, LoadTypes).
* Форматирование кода (отступы, пустые строки).
* Использование комментариев для пояснения сложного кода.
* Применение правил для работы с исключениями (try...catch).

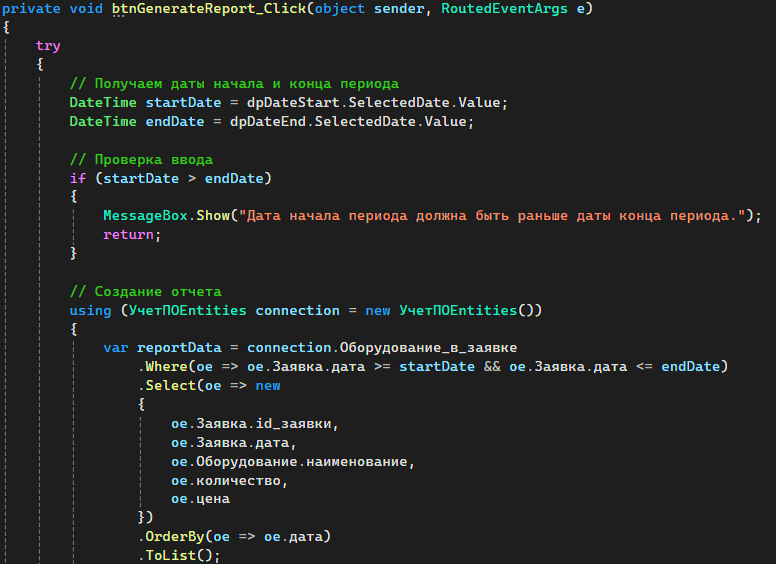


Рисунок 24 – Демонстрация кода на языке C#

1. SQL (Structured Query Language): Для взаимодействия с базой данных SQL Server использовался язык SQL. Были соблюдены следующие стандарты кодирования (Рисунок 25):

* Использование заглавных букв для ключевых слов SQL (SELECT, FROM, WHERE).
* Использование правил наименования таблиц и столбцов.
* Применение правил для написания эффективных SQL-запросов.

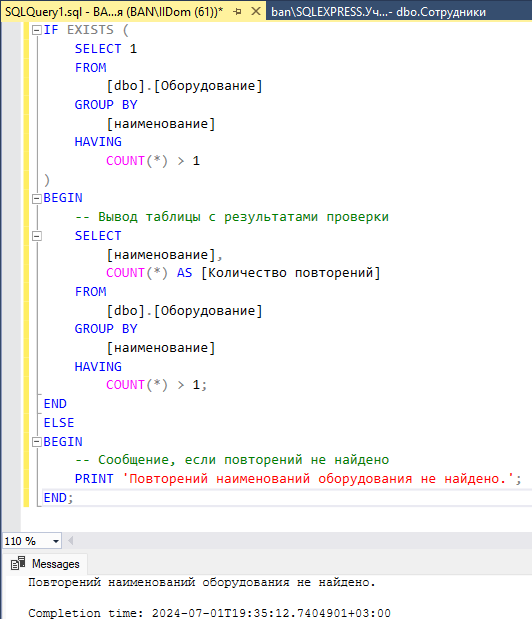


Рисунок 25 – Демонстрация кода на языке SQL

1. 1С: В рамках интеграции была использована логика формирования отчёта "Анализ расходов", разработанная в системе 1С:Предприятие 8.3. При переносе логики отчёта, в WPF-приложение была проанализирована и применена синтаксическая структура и стандарты кодирования, характерные для языка программирования 1С. Это позволило обеспечить корректность и эффективность работы отчёта в новой среде разработки (Рисунок 26).

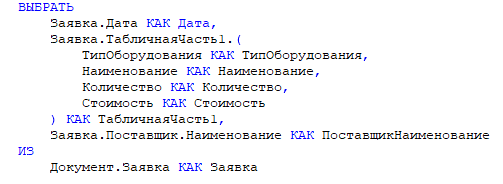


Рисунок 26 – Демонстрация кода на языке 1С

## Выявлены все имеющиеся несоответствия стандартам в предложенном коде.

В ходе анализа всего кода были выявлены несоответствия стандартам кодирования лишь на языке C#:

* Отступы и пустые строки: в некоторых местах кода отсутствовали правильные отступы и пустые строки, что делало код менее читаемым.
* Недостаточное использование комментариев: в некоторых местах кода отсутствовали комментарии, поясняющие логику работы кода.

Все недочеты были исправлены и код стал соответствовать стандартам кодирования (Рисунок 27).

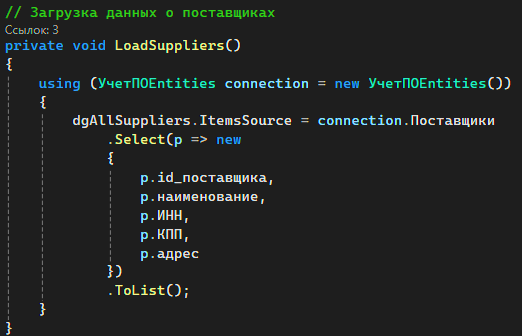


Рисунок 27 – Пример исправленного кода

# РУКОВОДСТВО ОПЕРАТОРА

**Руководство оператора для бухгалтера**

1. Вход в систему

* Для доступа к модулю "Анализ расходов" необходимо авторизоваться в системе с помощью логина и пароля.
* При успешной авторизации будет отображено главное окно формирования отчета "Анализ расходов".

1. Использование модуля "Анализ расходов"

* Фильтр по периоду: для анализа расходов за определенный период необходимо выбрать начальную и конечную даты в элементах управления DateTimePicker.
* Сводная таблица: после выбора периода и нажатии на кнопку "Создать отчет" будет отображена сводная таблица (DataGrid), в которой представлены расходы по каждому типу оборудования у каждой заявки за выбранный период, а также в (TextBox) отобразится общая сумма затратов.
* Сортировка: после получения отчета, можно отсортировать полученные данные: по номеру заявки, дате, оборудованию, количеству и стоимости.
* Выход: при нажатии на кнопку "Х" в левом верхнем углу можно выйти из своей учетной записи.

# ЗАКЛЮЧЕНИЕ

В ходе производственной практики по интеграции программных модулей я успешно интегрировал существующую базу данных 1С:Предприятие в WPF-приложение. Я интегрировал новые модули и функции в приложение, включая новую роль "Бухгалтер" и отчёт "Анализ расходов". Для проверки работы приложения были разработаны тестовые сценарии и пакеты, проведены тесты интеграции и ручное тестирование.

В ходе работы я приобрел практические навыки в разработке WPF-приложений, работе с базой данных SQL Server и тестировании программного обеспечения. Полученный опыт поможет мне в дальнейшей профессиональной деятельности в области программирования.

# ПРИЛОЖЕНИЯ К ОТЧЕТУ: ДИСК СО ВСЕМИ ПОДТВЕРЖДАЮЩИМИ МАТЕРИАЛАМИ, ОТЧЕТ В ЭЛЕКТРОННОМ ВИДЕ, ПРЕЗЕНТАЦИЯ ДЛЯ ВЫСТУПЛЕНИЯ И ДР. МАТЕРИАЛЫ.

Все материалы проекта доступны в моем репозитории на GitHub (<https://github.com/Sashakvakin/PM02-MASTERSOFT>).