МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Факультет компьютерных технологий и прикладной математики
Кафедра анализа данных и искусственного интеллекта

Допустить к защите
заведующий кафедрой
д-р тех. наук, доцент
_____ А.В. Коваленко
_____ 2023 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(БАКАЛАВРСКАЯ РАБОТА)

РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ ДОКУМЕНТООБОРОТА ДЛЯ СЕТИ ЦВЕТОЧНЫХ МАГАЗИНОВ

Работу выполнил(а) _____ М. А. Игнатьева

Направление подготовки 02.03.02 Фундаментальная информатика и информационные технологии

Направленность (профиль) Математическое и программное обеспечение компьютерных технологий

Научный руководитель
канд. физ.-мат. наук, доцент _____ Г.В. Калайдина

Нормоконтролер
канд. физ.-мат. наук, доцент _____ Г.В. Калайдина

Краснодар
2023

РЕФЕРАТ

В выпускной квалификационной работе 55 с., 1 табл., 25 рис., 12 источник.

ИНФОРМАЦИОННАЯ СИСТЕМА, ДОКУМЕНТООБОРОТ, C#, MD5, POSTGRESQL, БАЗЫ ДАННЫХ, ДОКУМЕНТ, ЦВЕТОЧНЫЙ МАГАЗИН

Объектами исследования в данной работе является процесс разработки информационной системы документооборота цветочных магазинов, а также основные методики и принципы, оптимизирующие реализацию документооборота.

Цель выпускной квалификационной работы – разработать информационную систему документооборота для цветочного магазина.

В рамках выпускной квалификационной работы был выполнен анализ доступных систем документооборота и изучены основы построения информационных систем, а также технологии их разработки и программной реализации для цветочного магазина.

Исходя из этого, была разработана информационная система, которая позволит значительно улучшить работу магазина, ускорить и упростить процессы работы сотрудников и сократить время на формирование документов.

СОДЕРЖАНИЕ

Введение 4
1 Общие сведения 6
1.1 Информационная система 6
1.2 Классификация информационных систем 6
1.3 Модели данных 7
1.4 Выбор СУБД 9
2 Анализ существующих систем электронного документооборота 12
2.1 Программа 1С: Розница 8 12
2.2 Программа для автоматизации цветочного магазина СБИС 13
2.3 Программа Florapos 15
3 Описание проектных решений 17
3.1 Обоснование выбора среды разработки 17
3.2 Язык программирования C# 18
3.3 Хеширование паролей MD5 19
3.4 Определение требований к проектируемой информационной системе 21
3.5 Разработка диаграммы классов 22
3.6 Разработка диаграммы прецедентов 24
3.7 Разработка ER-диаграммы 26
4 Разработанная информационная система 30
4.1 Описание разработанной информационной системы 30
4.2 Оценка экономической эффективности используемой системы 40
Заключение 43
Список использованных источников 44
Приложение А 46

ВВЕДЕНИЕ

В настоящее время, автоматизация бизнес-процессов становится необходимой для эффективной работы компаний.

Одной из важнейших задач, которую необходимо решать в этом контексте, является управление документооборотом.

В сфере цветочных магазинов, как и в любой другой отрасли, также требуется использование информационной системы документооборота. Такая система может значительно упростить и оптимизировать рабочие процессы, сократить затраты времени на ручную работу с бумажными документами и уменьшить количество ошибок, связанных с их обработкой. Это, в свою очередь, позволит снизить время обработки заказов и избежать задержек и ошибок при обработке заказов, что может привести к ухудшению репутации магазина и недовольству клиентов. Введение информационной системы документооборота позволит минимизировать риски возникновения этих ситуаций и повысить качество сервиса магазина в целом.

Цель выпускной квалификационной работы заключается в изучении структуры цветочных магазинов и в создании системы документооборота.

Реализация этой цели достигается путем анализа уже существующих информационных систем документооборота для цветочных магазинов, изучение специальной литературы, а также проектирование и разработка информационной системы документооборота для цветочного магазина.

Выпускная квалификационная работа состоит из четырех глав.

В первой главе рассмотрены основополагающие принципы, которые необходимы для построения информационной системы и классифицировали их в соответствии с их основными характеристиками.

Во второй главе приведен реферативный обзор существующих информационных систем и проанализированы их преимущества и недостатки.

Третья глава посвящена проектированию разрабатываемой информационной системы документооборота.

В четвертой главе описана программная реализация разрабатываемой информационной системы документооборота, где представлены подробности разработки и функциональные возможности системы.

1 Общие сведения

1.1 Информационная система

Понятие информационной системы трактуется согласно Федеральному закону РФ от 27 июля 2006 года № 149-ФЗ «Об информации, информационных технологиях и о защите информации» звучит следующим образом «информационная система – совокупность содержащейся в базах данных информации и обеспечивающих ее обработку информационных технологий и технических средств» [1].

Информационная система нужна для сбора, хранения, обработки, передачи и использования информации в рамках определенной организации или процесса.

Она позволяет упрощать и автоматизировать бизнес-процессы, повышать эффективность работы, улучшать качество принимаемых решений, обеспечивать безопасность и конфиденциальность информации. В целом, она является необходимым инструментом для успешного функционирования и развития любой организации.

1.2 Классификация информационных систем

Информационных систем разграничивают по их архитектуре на:

1) локальные информационные системы, внутри которых составные информационные системы (клиентские приложения, БД, СУБД) работающие в пределах одной локации или организации без обязательных подключений к глобальным сетям;

2) клиент-серверные информационные системы, которые работают на основе модели клиент–серверного взаимодействия. Это модель взаимодействия между компьютерными системами, в которой одна система –

клиент запрашивает определенные услуги или ресурсы у другой системы – сервера, которая предоставляет эти услуги или ресурсы;

3) распределенные информационные системы, которые состоят из нескольких автономных компонентов, распределенных по различным узлам сети, которые взаимодействуют друг с другом, обмениваясь информацией и ресурсами.

Классификаций может быть много, и они могут быть использованы для различных целей. Все типы информационных систем взаимодействуют друг с другом и могут быть интегрированы для достижения более высокой эффективности бизнес-процессов и управления ресурсами.

1.3 Модели данных

Модель данных – это представление данных, используемое в базах данных или других системах управления данными, в котором определены сущности и их отношения [2].

В модели данных определяются основные элементы, такие как:

1) Сущности – это объекты или понятия, которые описывают данные, хранящиеся в базе данных. Например, для базы данных о компании, сущностями могут быть клиенты, сотрудники и отделы;

2) Атрибуты – это свойства, описывающие сущности. Например, для сущности «клиент» атрибутами могут быть имя, фамилия, отчество, адрес и номер телефона;

3) Отношения – это связи между сущностями в базе данных. Например, сотрудник может работать в отделе, отдел может иметь руководителя.

На основе модели данных задаются ограничения на хранение и изменение данных, а также определяется структура базы данных и правила доступа к ним. Различные модели данных имеют различные способы описания и типы связей между ними.

Например, существуют такие модели данных, как реляционная, иерархическая, и сетевая.

Реляционная база данных – это тип базы данных, в которой хранятся данные в виде таблиц, называемых реляционными таблицами. Каждая реляционная таблица состоит из строк (записей) и столбцов (атрибутов). Каждый столбец имеет уникальное имя и тип данных, который определяет, какой тип данных может храниться в этом столбце. Каждая запись в таблице имеет уникальный идентификатор.

Реляционная база данных использует язык структурированных запросов SQL (Structured Query Language) для создания, изменения и извлечения данных из БД. SQL является стандартным языком для работы с реляционными базами данных [3].

Иерархическая база данных – это тип базы данных, в которой данные организованы в виде иерархической структуры. Данные хранятся в форме древовидной структуры, состоящей из узлов и ребер, которые связывают узлы.

В иерархической базе данных, данные упорядочиваются в иерархических узлах, каждый из которых может иметь один или несколько подузлов. У каждого узла может быть только один родительский элемент за исключением корневого узла, который является верхним уровнем иерархии. Каждый узел может содержать данные и иметь идентификационный номер. Ребра связывают узлы между собой. Таким образом, каждый узел, кроме корневого, является дочерним элементом родительского узла.

Иерархические базы данных используются для хранения иерархических данных, таких как деревья классификации, документов в формате XML и файловых систем.

Сетевая база данных – это тип базы данных, где данные представлены в виде объединений сетей, состоящих из записей (так называемых «сегментов») и соединений между ними (так называемых «структур»). Каждый сегмент может содержать несколько полей данных в форме «ключ-значение», которые отображают свойства объекта.

Сетевые базы данных представляют данные в виде графа, где у каждой записи может быть несколько ссылок на другие записи. Записи в базе данных являются узлами графа, а связи между записями являются ребрами графа.

Сетевые базы данных были очень популярны в начале и середине 20 века, но сейчас их использование сильно снизилось из-за появления более эффективных и удобных баз данных, таких как реляционные базы данных.

Выбор подходящей модели данных для конкретной задачи важен для обеспечения эффективного хранения и обработки данных в базах данных или других системах управления данными.

1.4 Выбор СУБД

Существует несколько разновидностей СУБД такие, как:

– Oracle. Это система управления реляционными базами данных (СУБД), которая разработана и распространяется корпорацией Oracle. Эта СУБД предоставляет широкий спектр функций для хранения, управления и доступа к данным во многих различных окружениях, включая корпоративные и веб-приложения. Oracle также предоставляет инструменты для разработки, управления и мониторинга баз данных, а также обеспечивает высокую доступность и безопасность данных. Oracle является одной из самых распространенных и популярных СУБД, используемых в масштабных корпоративных окружениях.

– Microsoft SQL Server. Это система управления реляционными базами данных (СУБД), созданная компанией Microsoft. Она позволяет хранить и обрабатывать большие объемы данных в корпоративных и веб-приложениях, имеет богатый набор функций и инструментов для работы с данными, включая поддержку транзакций, аналитических запросов, интеграции с другими программными продуктами Microsoft, а также для разработки и администрирования баз данных. SQL Server также обладает функциями безопасности, автоматического резервного копирования данных,

гибкой масштабируемости и возможностью работы в облаке. SQL Server используется многими организациями во всем мире и является одной из самых популярных СУБД.

– MySQL. Это свободная система управления реляционными базами данных (СУБД), которая была разработана MySQL AB, а теперь поддерживается компанией Oracle. Она отличается от других СУБД тем, что она бесплатная для пользователей и имеет открытый код, что позволяет пользователям модифицировать и распространять ее свободно. MySQL является одной из самых популярных СУБД на рынке, используется в множестве приложений и веб-сайтов во всем мире. MySQL поддерживает широкий диапазон функций, включая язык SQL, транзакции, хранимые процедуры, репликацию данных и многопользовательский доступ, обеспечивая быстрый и надежный доступ к данным. Обычно используется в качестве сервера, к которому обращаются локальные или удаленные клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы.

– PostgreSQL. Это объектно-реляционная система управления базами данных (СУБД) с открытым исходным кодом. Она была разработана с учетом расширяемости, надежности и удобства использования, предоставляя широкий спектр функций для хранения, управления и доступа к данным. PostgreSQL обладает высокой степенью соответствия стандартам SQL и поддерживает многопоточность и многие другие функции. Является свободной альтернативой коммерческим СУБД (таким как Oracle Database, Microsoft SQL Server, Informix и СУБД производства Sybase) вместе с другими свободными СУБД (такими как MySQL и Firebird).

– Sybase ASA (Adaptive Server Anywhere). Это система управления реляционными базами данных (RDBMS), разработанная компанией Sybase Inc. Она предназначена для использования во встроенных системах, мобильных устройствах и в небольших и средних предприятиях. Sybase ASA

обеспечивает функции синхронизации данных, удаленного доступа к данным и поддержку нескольких платформ и языков программирования. Он также поддерживает интерфейсы SQL, ODBC, JDBC и ADO.NET, что делает его совместимым с широким спектром приложений.

– Firebird. Это открытая реляционная база данных, которая предоставляет высокую производительность, надежность и масштабируемость. Он поддерживает множество платформ, включая Windows, Linux и macOS, а также множество языков программирования, включая C++, Delphi, Java и .NET.

– InterBase. Это реляционная база данных, разработанная в 1984 году и приобретенная компанией Borland. Она поддерживает множество платформ и языков программирования, включая C++, Delphi, Java и .NET. Основным достоинством последней версии InterBase являются низкие требования к системе, с одновременной масштабируемостью на несколько процессоров, а также развитая система мониторинга, временные таблицы, встраиваемая аутентификация пользователей. Однако, в отличие от Firebird, InterBase является коммерческим продуктом и требует лицензирования.

2 Анализ существующих систем электронного документооборота

В настоящее время существует ряд программных продуктов, разработанных для автоматизации процесса документооборота цветочного магазина. Некоторые из них обладают своими преимуществами и недостатками. Ниже приведены примеры уже существующих электронных систем документооборота, применяемых в цветочных магазинах.

2.1 Программа 1С: Розница 8

Конфигурация «1С: Розница 8» предназначена для автоматизации бизнес-процессов магазинов, которые могут входить в распределенную розничную сеть торгового предприятия.

Цветочный магазин является специфическим бизнесом, требующим особого подхода. Это связано с особенностями товара, имеющего уникальные маркировки и отличающегося от продуктовых товаров. Из-за того, что цветы скоропортящиеся товары, учет их вручную является достаточно сложным. Кроме того, товар от разных производителей может быть очень похожим, за исключением цены, что усиливает необходимость точного учета.

Программа предназначена для решения таких задач, как:

- взаиморасчеты с покупателями;
- взаиморасчеты с поставщиками;
- оформление заказов покупателей;
- оформление заказов поставщикам;
- розничная торговля;
- ценообразование, прайс-листы.

Программный продукт не имеет существенных недостатков. Кроме того, не удалось найти подходящее решение для автоматизации работы цветочных магазинов на базе программы «1С: Розница».

На рисунке 1 представлен интерфейс программы «1С: Розница Автоматизация магазина».

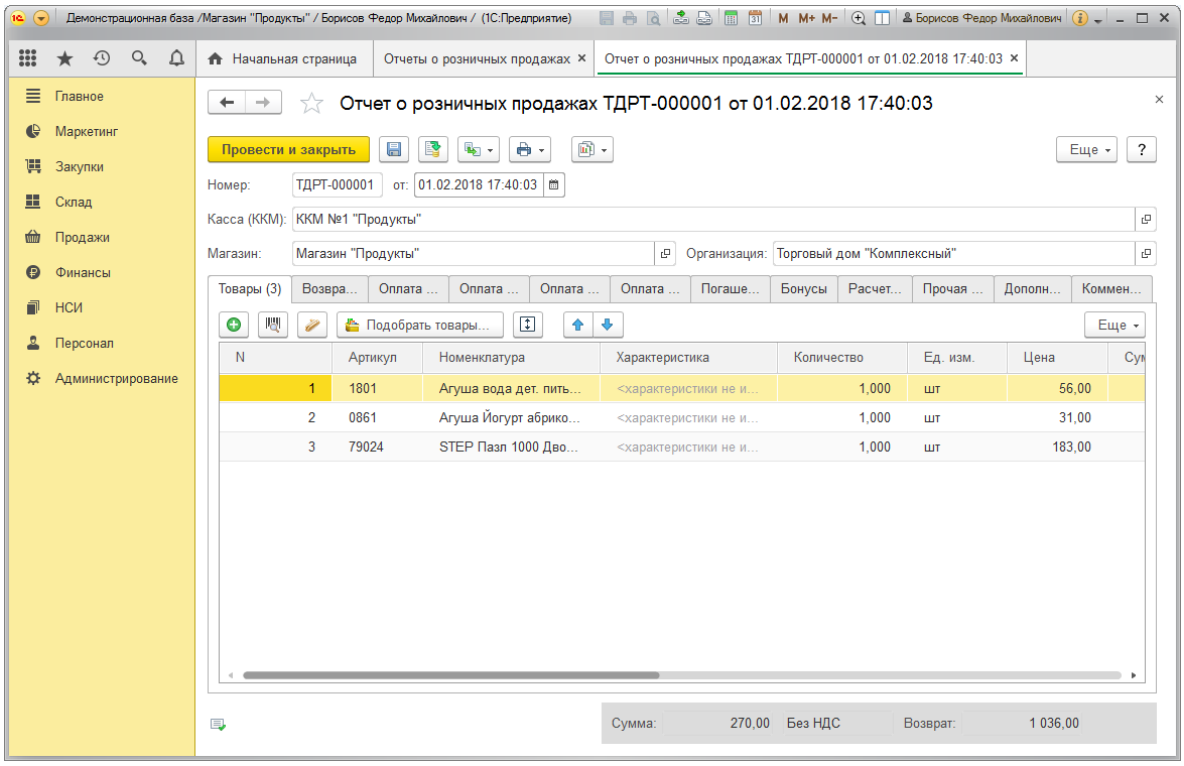

Рисунок 1 – Интерфейс программы «1С: Розница Автоматизация магазина»

Стоимость на «1С: Розница 8.3» фиксирована – 15400 рублей, за исключением базовой версии программы [4].

2.2 Программа для автоматизации цветочного магазина СБИС

СБИС – это сеть деловых коммуникаций, которой пользуется более 1 млн клиентов в России. Она объединяет людей и компании, системы управления и учета, бизнес-процессы и документы в одной системе. СБИС помогает сдавать отчетность в госорганы, обмениваться электронными документами, управлять персоналом, находить интересные тендеры и решать другие задачи бизнеса. Разработчик СБИС – компания Тензор. На рынке IT с 1996 года.[5]

Программа предназначена для решения таких задач, как:

– управление финансами;
– легкая работа с документами;
– кассовая программа;
– банк и взаиморасчеты;
– анализ продаж в рознице;
– телефония и call-центр;
– налоговый учет и ПБУ 18;
– отраслевые решения ОФД;
– CRM для розницы и сферы услуг.

Из недостатков можно выделить множество негативных отзывов в интернете, а именно, невозможно разобраться в запутанном интерфейсе и очень плохо работает техподдержка вместе с этим, жалуются, что компания не выполняет условия договора.

На рисунке 2 показан интерфейс автоматизированной системы для магазина.

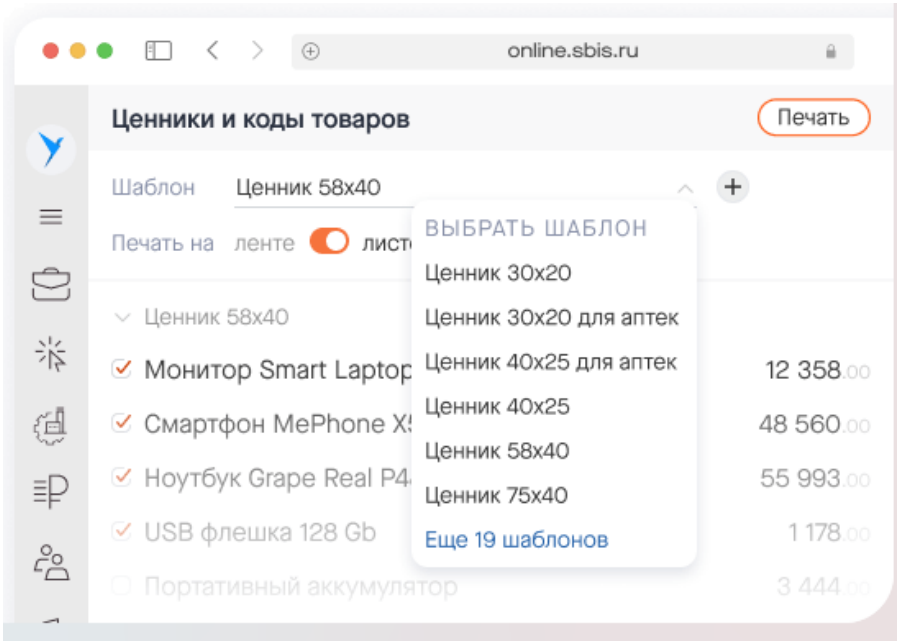

Рисунок 2 – Интерфейс программы «СБИС»

Стоимость такой программы на «СБИС» для автоматизации цветочного магазина составляет около 12500 рублей в год.

2.3 Программа Florapos

Florapos – проект компании «Асгард». Компания занимается разработкой программного обеспечения, и комплексная автоматизация объектов оптовой и розничной торговли – одно из направлений ее деятельности. В этом направлении она обращает отдельное внимание «на изучение тонкостей и специфики ведения флористического бизнеса» [6].

Программа предназначена для решения таких задач, как:

– продажа и возврат товаров;

– составление букетов;

– уценка товара (в том числе и неоднократная);

– инвентаризация (полная и частичная);

– предзаказ и ведение клиентской базы;

– гибкая система лояльности;

– интеграция с интернет-магазинами и CRM.

На рисунках 3 и 4 представлены интерфейсы программ Florapos.

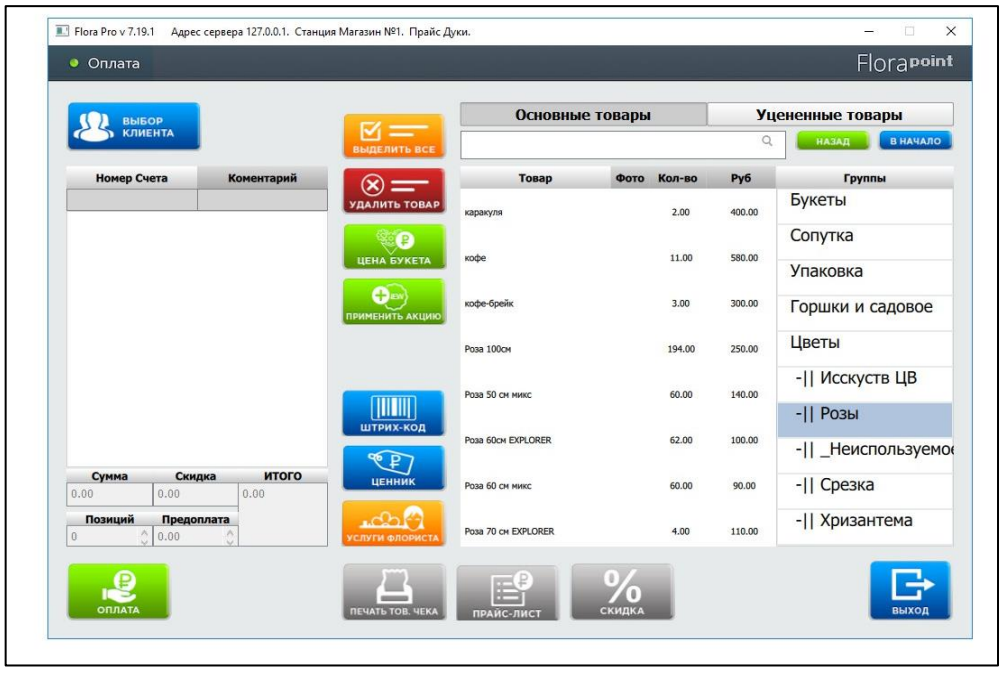

Рисунок 3 – Интерфейс программы Florapos

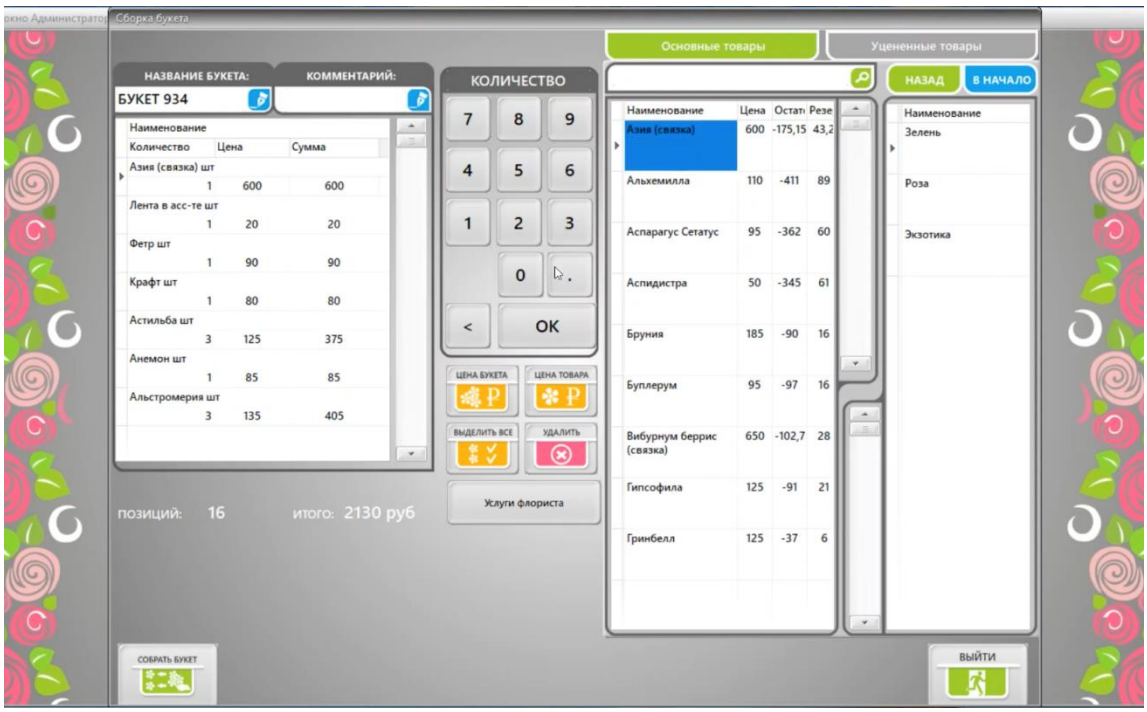

Рисунок 4 – Интерфейс программы Florapos для составления букета

Из недостатков можно выделить также наличии множества негативных отзывов о данной системе.

Стоимость такой программы Florapos для автоматизации цветочного магазина начинается от 17000 рублей в год и зависит от выбора дополнительных функций и инструментария.

Подводя итоги, можно выделить что на сегодняшний день существует множество различных систем электронного документооборота, каждая из которых имеет свои преимущества и недостатки. Для выбора наиболее подходящей системы необходимо провести тщательный анализ требований бизнеса и проанализировать особенности каждой системы. Важно учитывать не только функциональность и удобство использования, но и безопасность, стоимость внедрения и поддержки, а также наличие соответствующих сертификатов и лицензий. Кроме того, следует учитывать возможность интеграции с другими системами, что позволит гладко и эффективно организовать все процессы в рамках компании.

3 Описание проектных решений

3.1 Обоснование выбора среды разработки

Для написания выпускной квалифицированной работы была выбрана среда разработки Visual Studio 2022.

Microsoft Visual Studio – это версия Visual Studio и .NET Framework. Она не только позволяет писать и отлаживать код, но и автоматизирует деятельность всех членов команды, работающих над проектом. Также она поддерживает Microsoft SQL Server, что позволяет создавать и развертывать проекты с применением сервера баз данных.

Visual Studio имеет стиль, схожий с MS Office, и включает в себя Task List, где можно отслеживать ошибки и задачи с возможностью назначения приоритета и отметки о завершении. Свойства проекта можно редактировать встроенным инструментом, сохранять настройки и использовать объектно-ориентированные языки программирования, такие как C# и Visual Basic, а также компоненты сторонних компаний.

Для эффективной и удобной разработки приложений, основанных на принципах объектно-ориентированного программирования, была выбрана платформа .NET.

Она обладает широким инструментарием и библиотеками для создания различных типов приложений, включая веб-приложения, мобильные приложения и создания быстрых приложения с графическим интерфейсом.

Среда разработки Visual Studio, входящая в состав .NET, обеспечивает эффективную и быструю разработку приложений. В рамках платформы .NET был создан язык программирования C#, который является простым, современным, объектно-ориентированным и безопасным. Библиотека Windows Forms позволяет проектировать графический интерфейс для приложений, которые могут быть установлены на машине конечного пользователя.

Система, описанная в данной работе, была разработана с помощью библиотеки Windows Forms.

Язык программирования C# призван реализовать компонентно-ориентированный подход к программированию, который способствует меньшей машинно-архитектурной зависимости результирующего программного кода, большей гибкости, переносимости и легкости повторного использования программ [7].

Так же язык программирования C# является «родным» для создания приложений в среде Microsoft .NET, поскольку наиболее тесно и эффективно интегрирован с ней.

3.2 Язык программирования C#

В качестве основного языка программирования был выбран язык программирования C#. Так как он является объектно-ориентированным и позволяет разрабатывать более сложные системы.

Платформа .NET – это среда, которая предоставляет инфраструктуру для разработки и запуска приложений на языке программирования C#. Так же .NET содержит огромную библиотеку классов, которые можно использовать при программировании на любом языке [8].

C# поддерживается крупными компаниями, такими как Microsoft, что гарантирует долгосрочную поддержку и развитие языка.

C# имеет множество возможностей для разработки, включая поддержку многопоточности, асинхронного программирования, LINQ, атрибутов и многого другого. Язык также имеет широкий спектр инструментов и библиотек, которые упрощают и ускоряют процесс разработки.

Кроме того, C# обладает высокой производительностью и безопасностью приложений благодаря платформе .NET, которая обеспечивает автоматическую сборку мусора, управление памятью и другие функции.

C# также имеет строгую типизацию и проверку ошибок на этапе компиляции, что позволяет уменьшить количество ошибок в коде и повысить надежность приложения.

В целом, C# является мощным и современным языком программирования, который подходит для разработки различных типов приложений, включая информационные системы.

Windows Forms – это библиотека классов .NET, которая позволяет создавать графические пользовательские интерфейсы (GUI) для приложений Windows. C# часто используется вместе с Windows Forms для разработки таких приложений.

Windows Forms предоставляет множество элементов управления, таких как кнопки, текстовые поля, списки и др., которые можно использовать для создания интерфейса приложения. Эти элементы управления могут быть настроены с помощью свойств и методов, что позволяет создавать более сложные интерфейсы.

C# и Windows Forms также обеспечивают возможность обработки событий, таких как щелчок на кнопке или изменение текста в текстовом поле. Это позволяет создавать интерактивные приложения, которые реагируют на действия пользователя.

Кроме того, Windows Forms поддерживает различные стили и темы оформления, что позволяет создавать приложения с уникальным дизайном.

В целом, C# и Windows Forms являются мощным инструментарием для создания приложений Windows с графическим пользовательским интерфейсом.

3.3 Хеширование паролей MD5

Алгоритм MD5 представляет собой метод хеширования на основе 128-битной основы. При хешировании входные данные преобразуются по определенному алгоритму в битовую строку фиксированной длины [9].

Хеш-значение, полученное в результате процедуры, выражается в шестнадцатеричной системе исчисления и может быть также названо хешем, хеш-суммой или хеш-кодом. Хеширование широко применяется в программировании и веб-разработке для создания уникальных значений ассоциативных массивов и идентификаторов.

Область применения хеш-кодов:

– создание электронных подписей;

– хранение паролей в базах данных систем безопасности;

– в рамках современной криптографии для создания уникальных ключей онлайн;

– проверка подлинности и целостности элементов файловой системы ПК.

Алгоритм MD5 был разработан в 1991 году и стал стандартом хеширования для создания уникального хеш-кода от заданного значения с последующей проверкой его подлинности.

Утилита md5sum позволяет хешировать данные заданного файла по алгоритму MD5 и возвращает строку, состоящую из 32-х символов в шестнадцатеричной системе исчисления.

Вот так выглядит возвращающая строка, состоящая из 32-х символов в шестнадцатеричной системе исчисления: 016f8e458c8f89ef75fa7a78265a0025.

Полученный хеш представляет собой 16-байтовую строку, состоящую из 16 шестнадцатеричных чисел. Изменение любого символа в этой строке может привести к значительному изменению каждого бита в строке, что делает хеш-код непригодным для использования в качестве уникального идентификатора безопасности или проверки целостности.

Для безопасного хранения и защиты паролей пользователей в операционных системах UNIX используется алгоритм хеширования на основе MD5. Каждый пользователь имеет свой уникальный пароль, который защищен

этим методом. Также этот метод используется для защиты паролей на некоторых системах на базе Linux.

Этот стандарт кодирования является одним из самых распространенных методов защиты данных не только в прикладном, но и в веб-программировании. Поэтому не будет лишним обезопасить свой md5 hash от намеренного взлома.

Основным способом, гарантирующим безопасность хеша вашего пароля, является использование «соли». Он основан на добавлении к паролю нескольких случайных символов и последующем хешировании результата.

Во многих языках программирования для этого используются специальные классы и функции. Не являются исключением из правил и серверные языки программирования.

3.4 Определение требований к проектируемой информационной системе

В ходе проектирования информационной системы были определены функциональные возможности, предоставляемые пользователям и администратору.

Для информационной системы, предназначенной для автоматизации документооборота, необходимо обеспечить возможностью формирования документов в формате .doc из разных разделов системы.

Автоматизированная информационная система цветочного магазина предназначена для автоматизации документооборота в сети магазина.

Преимущества внедрения автоматизированной информационной системы для цветочного магазина могут быть следующими:

1) увеличение эффективности работы – автоматизация процессов заказа, оплаты и учета товаров позволяет сократить время на выполнение задач и улучшить качество работы;

2) улучшение качества данных – автоматизированная система позволяет уменьшить количество ошибок, связанных с ручным вводом данных о заказах и клиентах, что повышает точность и надежность информации;

3) увеличение скорости обработки заказов – автоматизированная система предоставляет быстрый доступ к необходимой информации о товарах и клиентах, что позволяет быстрее обрабатывать заказы и улучшить сервис для клиентов;

4) сокращение затрат – автоматизированная система позволяет сократить затраты на ручной труд, что может привести к снижению затрат на персонал и улучшению финансовых показателей компании;

5) улучшение клиентского опыта – автоматизированная система позволяет быстрее и эффективнее обрабатывать запросы клиентов, что улучшает их опыт работы с магазином и может привести к повышению уровня удовлетворенности клиентов.

3.5 Разработка диаграммы классов

UML (Unified Modeling Language) – это язык моделирования, используемый для описания и проектирования систем различной сложности. UML представляет собой набор графических символов, которые используются для создания диаграмм, позволяющих визуализировать различные аспекты системы, такие как ее структуру, функциональность, взаимодействие с окружающей средой и т.д. UML является стандартом, поддерживаемым многими инструментами для моделирования, и широко используется в областях информационных технологий [10].

Основная идея UML-диаграммы классов – представление структуры системы на объектно-ориентированном уровне. Такая диаграмма позволяет отобразить классы, их свойства (атрибуты) и методы (операции), а также связи между классами и наследование. Она помогает программистам и разработчикам понять взаимосвязи между классами и характер предметной

области, которую моделирует система, а также визуализировать архитектуру приложения. UML-диаграмма классов является одной из самых фундаментальных и полезных диаграмм в UML-нотации, которая широко используется в процессе проектирования и разработки ПО.

На рисунке 5 представлена диаграмма классов, поясняющая работу всей информационной системы.

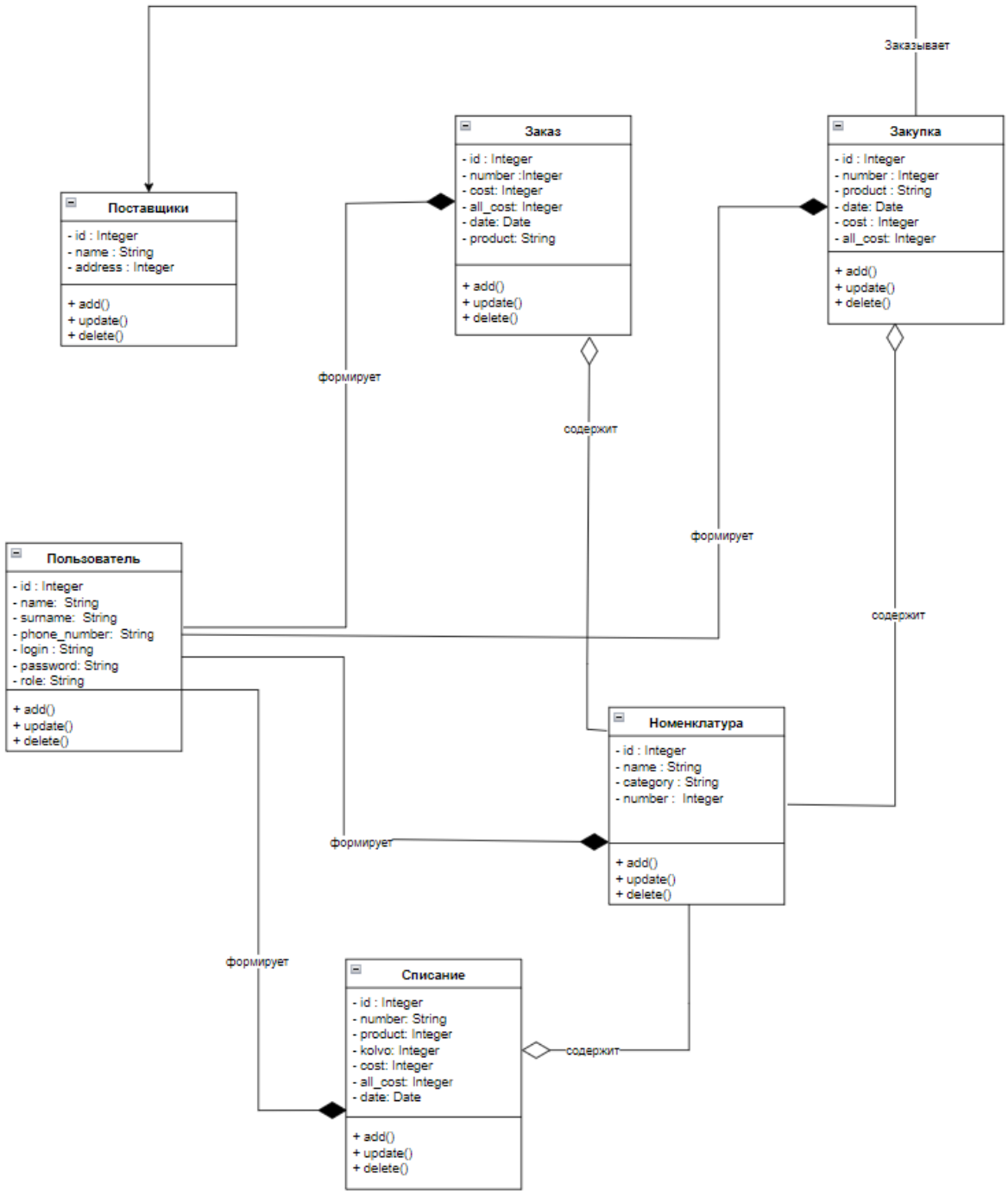

Рисунок 5 – Диаграмма классов информационной системы

Пользователь – это сотрудник магазина, который может осуществлять различную деятельность в разработанной системе.

Списание – это список товаров, которые были списаны в магазине по какой-либо причине.

Номенклатура – это список товаров, которые хранятся в системе.

Поставщики – это список поставщиков для разработанного магазина.

Заказ – это составленный сотрудником заказ на продажу товаров в магазине.

Закупка – это заявка, составленная сотрудником магазина для заказа определенного количества товаров у поставщика.

3.6 Разработка диаграммы прецедентов

Перед построением логической модели, была создана диаграмма прецедентов для рассматриваемого цветочного магазина.

Диаграмма вариантов использования, также известная как диаграмма сценариев поведения или диаграмма прецедентов [11], – это один из первоначальных этапов проектирования и разработки системы. Она состоит из актеров, вариантов использования и связей между ними. При создании этой диаграммы могут использоваться различные элементы нотации, такие как примечания и расширения.

Для данной диаграммы необходимо выбрать актера, который является сотрудником магазина.

Для него предусмотрены следующий поток событий:

- вход в систему (ввод логина и пароля);
- формирование заказа;
- формирование закупки;
- формирование списанных товаров;
- выбор категории продукта;

– добавление товара;

– добавление поставщика;

– добавление пользователя;

– формирование документов.

На рисунке 6 представлена диаграмма прецедентов пользователя системы.

Рисунок 6 – Диаграмма прецедентов пользователя системы

На рисунке 7 представлена диаграмма прецедентов администратора системы.

Рисунок 7 – Диаграмма прецедентов администратора системы

Администратор системы обладает большими полномочиями, такими как добавление новых пользователей в систему, в отличие от обычных пользователей, которые не имеют такой возможности.

3.7 Разработка ER-диаграммы

ER-диаграмма – это графическое представление модели данных, которое используется для описания сущностей и их отношений в системе. ER диаграммы включают в себя объекты, называемые сущностями, которые представляют реальные или абстрактные объекты, такие как люди, места, события, а также связи между этими объектами, которые отображаются в виде линий или стрелок. ER диаграммы широко используются для проектирования баз данных и других приложений, которые требуют хранения и управления данными [12].

Основная идея диаграммы прецедентов заключается в отображении взаимодействий между пользователями и системой на высоком уровне абстракции. Прецеденты описывают, как пользователи (актеры) будут

использовать систему, назначая конкретные задачи и цели, которые нужно выполнить в рамках взаимодействия.

Диаграмма прецедентов помогает понять, как система должна функционировать в рамках пользовательских потребностей и действий, и каким образом она должна обмениваться данными с другими системами или актерами. Она позволяет определить функциональность системы и точно определить требования к ее работе, что является важной частью процесса разработки программного обеспечения.

На рисунке 8 приведена ER-диаграмма разработанной информационной системы.

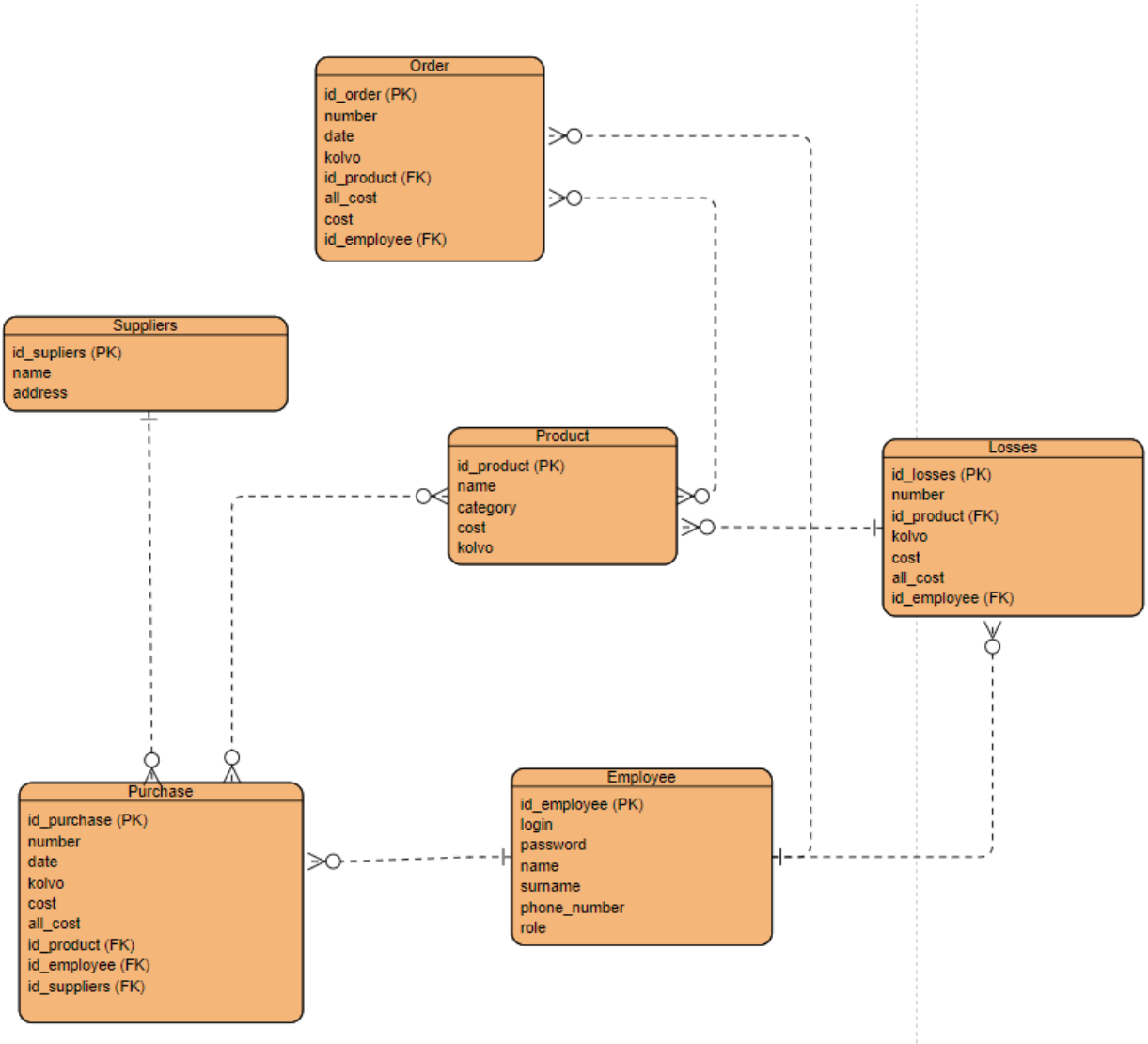

Рисунок 8 – ER-диаграмма информационной системы

В таблице «Order» хранится информация о заказе. Таблица «Order» состоит из следующих полей:

– id_order – тип Integer, идентификатор заказа;
– number – тип Varchar, номер заказа;
– date – тип Date, дата формирования заказа;
– kolvo – тип Integer, количество товара;
– id_product – тип Integer, идентификатор товара;
– cost – тип Integer, цена товара;
– all_cost – тип Integer, полная стоимость товаров;
– id_employee – тип Integer, идентификатор сотрудника.

В таблице «Employee» хранится информация о сотрудниках. Таблица «Employee» состоит из следующих полей:

– id_employee – тип Integer, идентификатор сотрудника;
– login – тип Char, логин сотрудника для входа в систему;
– password – тип Char, пароль сотрудника для входа в систему;
– name – тип Char, имя сотрудника;
– surname – тип Char, фамилия сотрудника;
– phone_number – тип Char, номер телефона сотрудника;
– role – тип Char, переменная, которая показывает роль сотрудника в системе, пользователь или администратор.

В таблице «Purchase» хранится информация о закупках. Таблица «Purchase» состоит из следующих полей:

– id_purchase – тип Integer, идентификатор закупки;
– number – тип Integer, номер закупки;
– date – тип Date, дата формирования закупки;
– kolvo – тип Integer, количество товаров;
– cost – тип Integer, цена товара;
– all_cost – тип Integer, полная стоимость товаров;
– id_employee – тип Integer, идентификатор сотрудника;

– id_product – тип Integer, идентификатор товара;

– id_suppliers – тип Integer, идентификатор поставщика.

В таблице «Product» хранится информация о товарах. Таблица «Product» состоит из следующих полей:

– id_product – тип Integer, идентификатор товара;

– name – тип Integer, название товара;

– cost – тип Integer, цена товара;

– kolvo – тип Integer, количество товара;

– category – тип Char, категория товара.

В таблице «Suppliers» хранится информация о поставщиках. Таблица «Suppliers» состоит из следующих полей:

– id_suppliers – тип Integer, идентификатор поставщика;

– name – тип Char, имя поставщика;

– address – тип Text, адрес поставщика.

В таблице «Losses» хранится информация о списанных товарах. Таблица «Losses» состоит из следующих полей:

– id_losses – тип Integer, идентификатор документа;

– number – тип Integer, номер документа;

– id_product – тип Integer, идентификатор продукта;

– kolvo – тип Integer, количество товаров;

– cost – тип Integer, цена товара;

– all_cost – тип Integer, стоимость товаров;

– date – тип Date, дата формирования документа;

– id_employee – тип Integer, идентификатор сотрудника.

4 Разработанная информационная система

4.1 Описание разработанной информационной системы

В данной информационной системе существует два вида пользователя – это администратор и обычный пользователь. Обычный пользователь может осуществлять формирование новых заказов, закупок, документов, а также изменять список товаров в системе и добавлять поставщиков. Администратор может выполнять все вышеперечисленное, а также добавлять новых пользователей в систему.

Для входа в систему необходимо ввести логин и пароль пользователя. Для начала рассмотрим работу пользователя.

На рисунке 8 показано окно входа в разработанную систему.

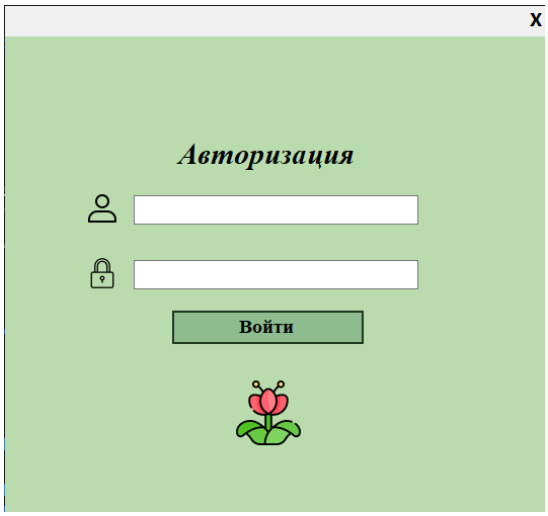

Рисунок 8 – Окно входа в систему

На рисунках 9 и 10 показан вход в систему и что выводится окно при успешном вводе логина и пароля.

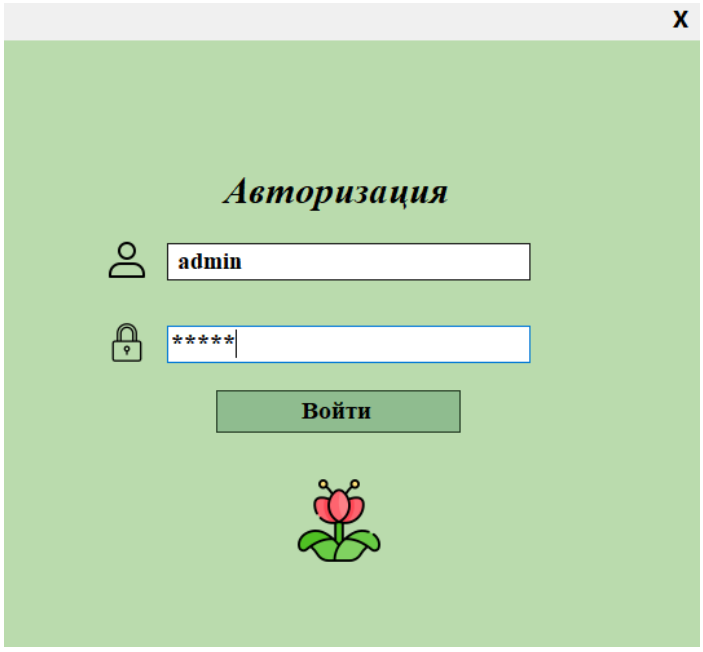

Рисунок 9 – Пример входа в систему

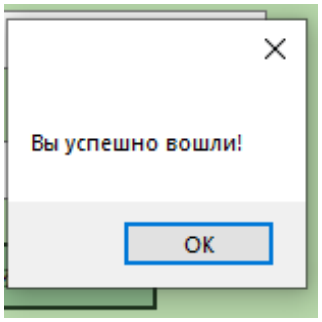

Рисунок 10 – Сообщение об успешной авторизации в системе

После того, как пользователь успешно вошел в систему, он переходит в окно «Главное меню», где он может выбрать интересующий его отдел. На рисунке 11 показано окно «Главное меню».

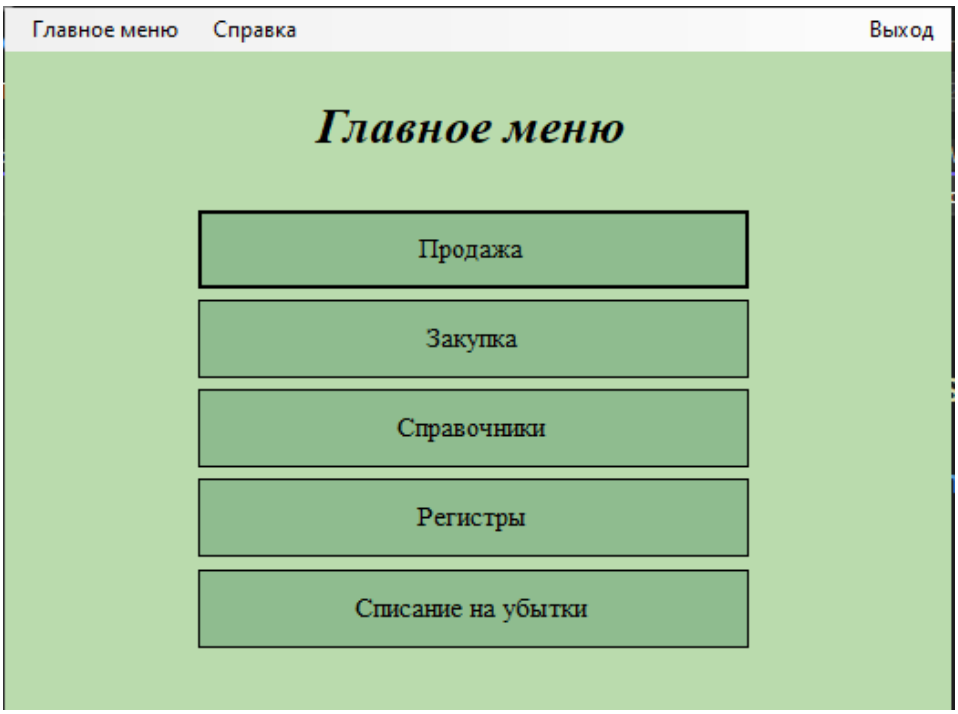

Рисунок 11 – Окно раздела «Главное меню»

В разделе «Продажа» пользователь имеет возможность создать новый заказ и в последующем сформировать подходящий ему документ. Кроме того, он может произвести поиск по дате и отобразить нужные ему заказы, находящиеся в заданном диапазоне дат. Кроме того, доступна функция просмотра содержимого заказа. В примере окна «Продажа» на рисунке 12 показаны разработанные элементы. Дополнительно пользователь может внести изменения в уже существующий заказ или удалить устаревший заказ.

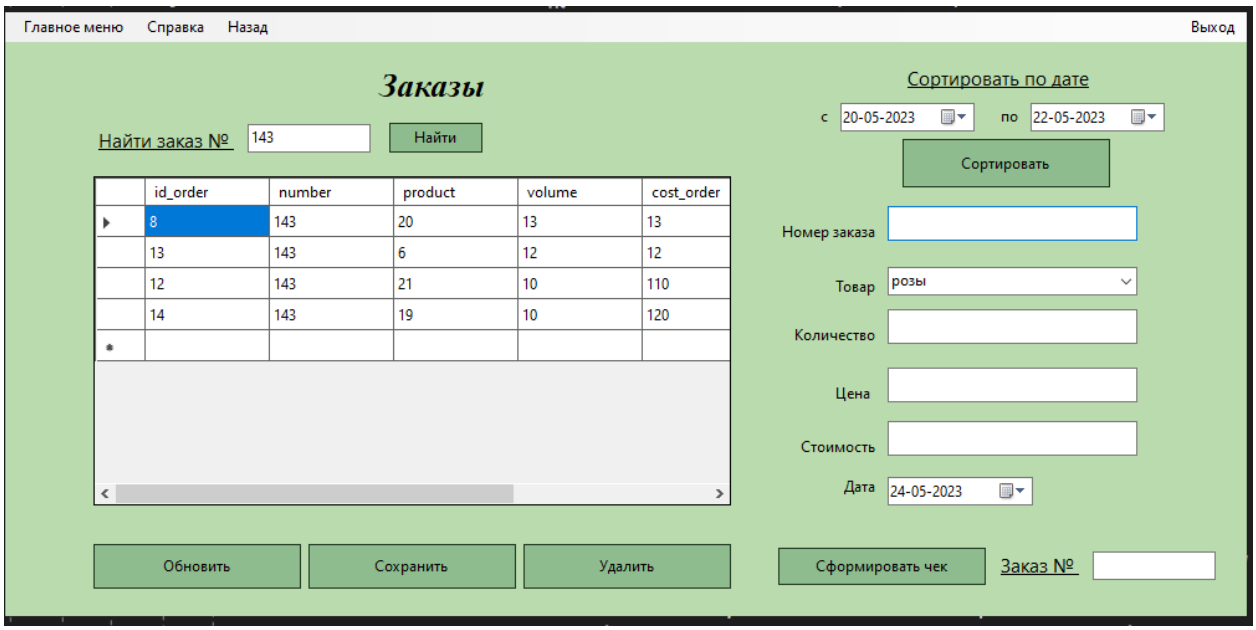

Рисунок 12 – Окно раздела «Продажа»

На рисунке 13 показан пример сортировки заказов по дате с 18.05.2023 по 24.05.2023.

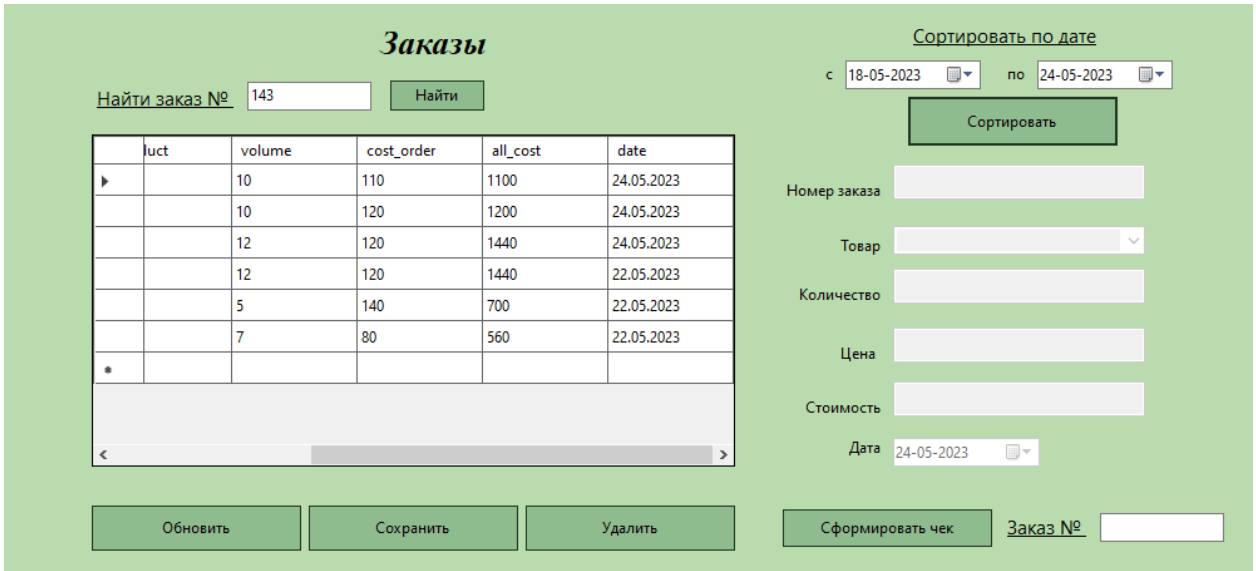

Рисунок 13 – Сортировка заказов по дате

Далее на рисунке 14 приведен товарный чек по заказу «143», который был сформирован по нажатию на кнопку «Сформировать чек».

Товарный чек

Заказ №143

Название	Количество	Цена	Стоимость
розы	12	120	1440
Игрушка Плюшевый медведь	13	13	13
С днем рождения	10	110	1100
Бумага крафтовая	10	120	1200

Итого: 3753

Дата оформления заказа: 24-05-2023

Рисунок 14 – Товарный чек по заказу №143

Так же пользователь может перейти в раздел «Справочники», где он может выбрать один из перечисленных разделов: «Номенклатура», «Поставщики» и «Пользователи». На рисунке 15 приведено окно раздела «Справочники».

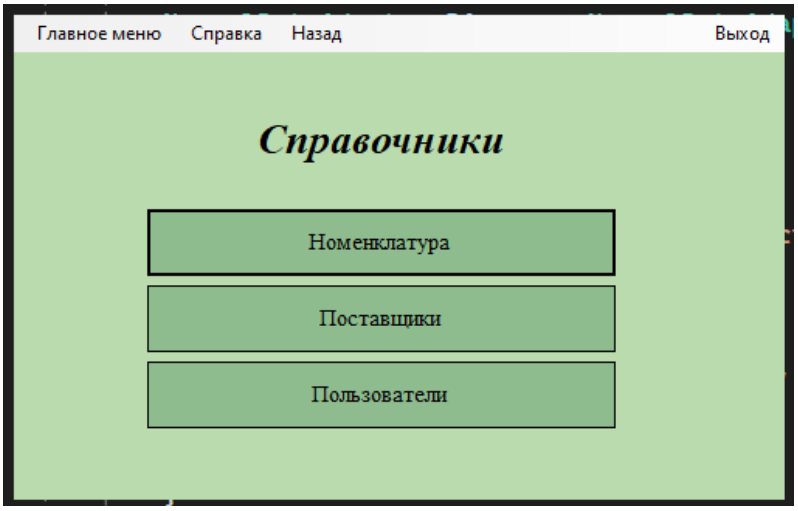

Рисунок 15 – Окно раздела «Справочники»

В разделе «Номенклатура», пользователю предоставляется выбор категории товаров. На рисунке 16 показано окно «Категории товаров».

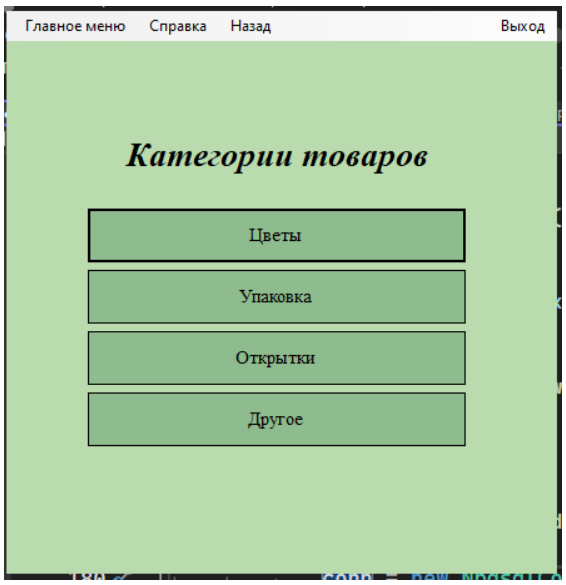

Рисунок 16 – Окно раздела «Категории товаров»

Как показано на рисунке 17, пользователь выбрал раздел «Цветы». В этой категории пользователь может ознакомиться со всей информацией о товарах, имеющихся в этой категории, а также добавить, изменить или удалить товары.

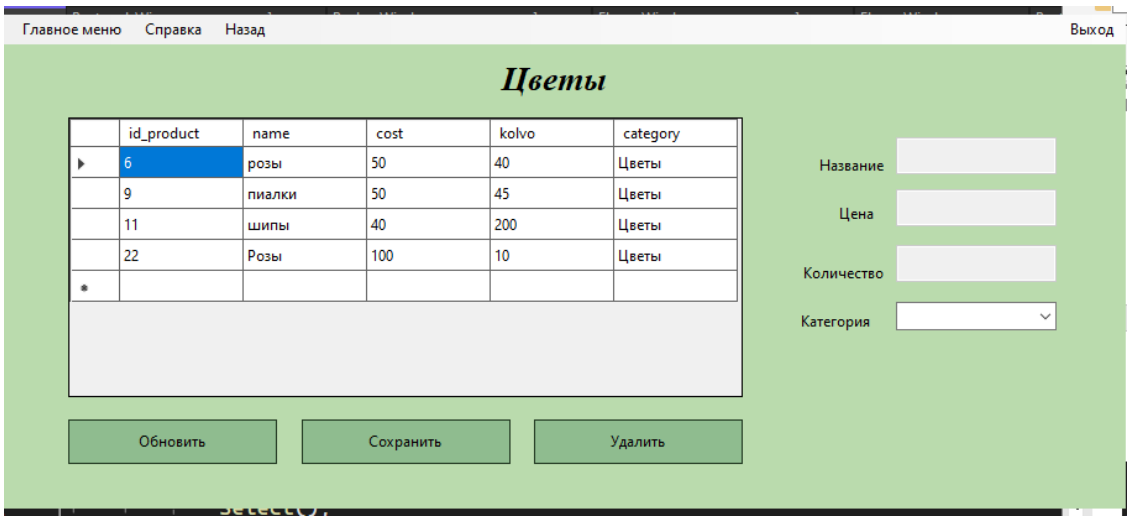

Рисунок 17 – Окно раздела «Цветы»

Затем клиент может просмотреть перечень всех поставщиков, перейдя в раздел "Поставщики". Там он может увидеть детали, а также добавить,

изменить или удалить информацию. Окно на рисунке 18 демонстрирует раздел «Поставщики».

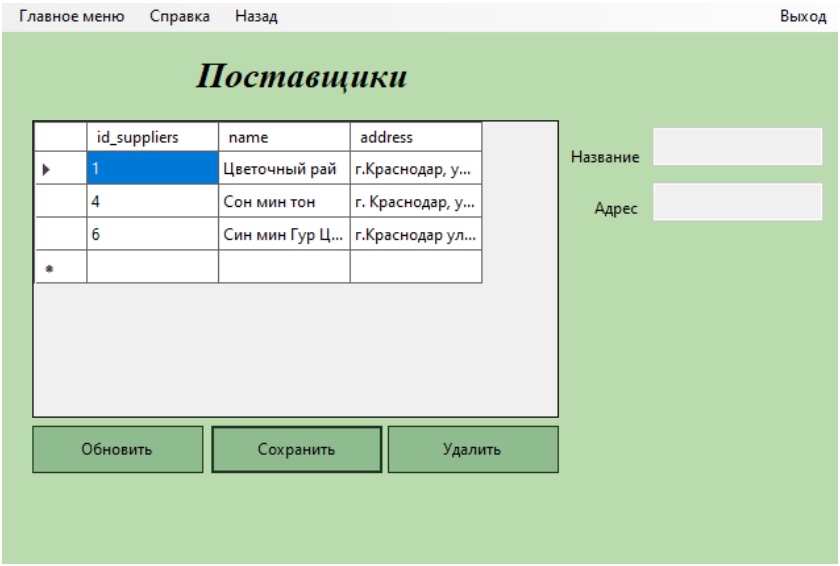

Рисунок 18 – Окно раздела «Поставщики»

Администратор системы может изменять, добавлять и удалять пользователей из системы, а также наделять их правами доступа. На рисунке 19 показано окно раздела «Пользователи».

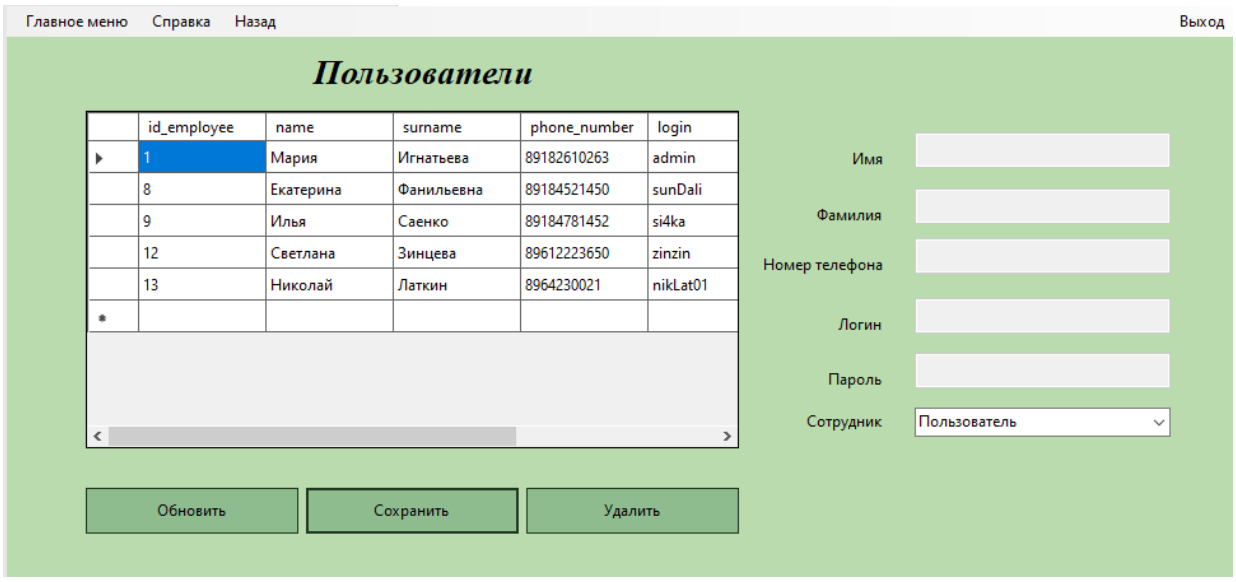

Рисунок 19 – Окно раздела «Пользователи»

Раздел «Закупки» предоставляет возможность сотруднику сформировать заказ на закупку и заполнить всю необходимую информацию. Кроме того, в разделе реализованы функции поиска нужного заказа на закупку или вывода всех заказов, выполненных за определенный период времени.

Также сотрудник может, по желанию, сформировать накладную на закупку. На рисунке 20 приведена работа окна «Закупки».

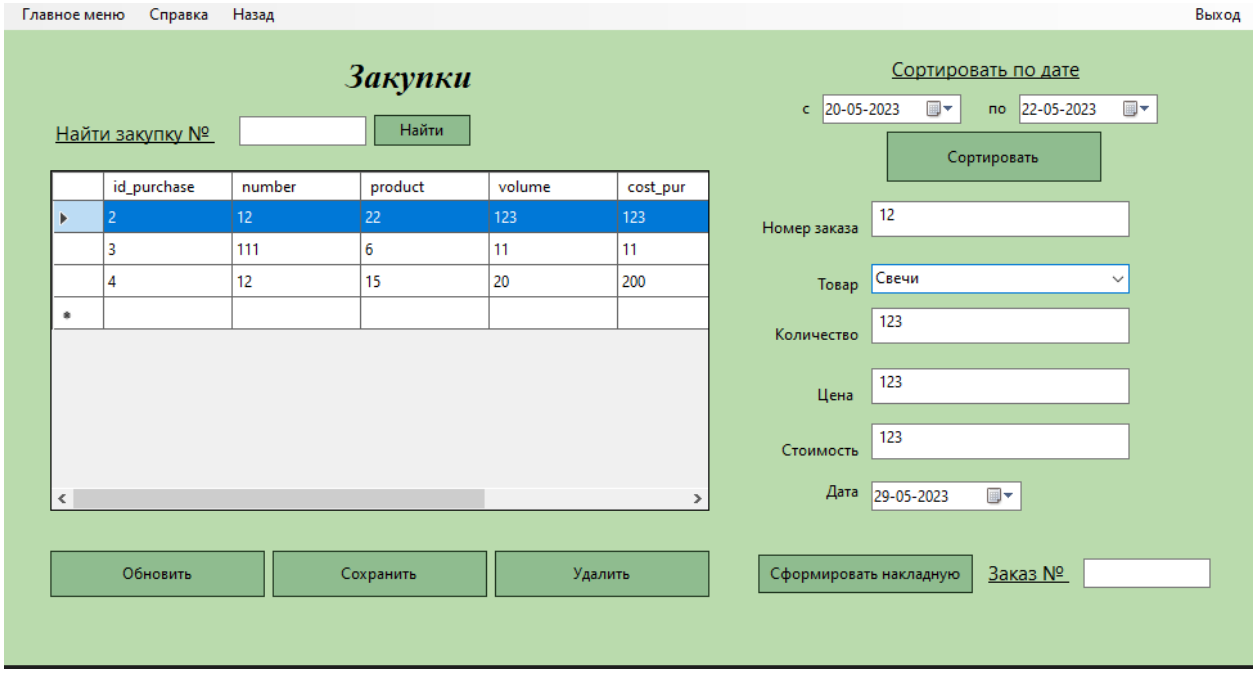

Рисунок 20 – Окно раздела «Закупки»

На рисунке 21 приведен пример накладной на закупку.

Накладная на закупку

Накладная №12

Название	Количество	Цена	Стоимость
Розы	123	123	123
Свечи	20	200	4000

Итого: 4123

Дата оформления накладной: 29-05-2023

Рисунок 21 – Накладная на закупку №12

На рисунке 22 приведен разработанный документ в разделе «Списание товаров».

Накладная на списание товаров

Документ №123

Название	Количество	Цена	Стоимость
фиалки	5	70	350
розы	10	100	1000

Итого: 1350

Дата оформления накладной: 29-05-2023

Рисунок 22 – Накладная на списание товаров №123

Через окно «Главное меню» пользователь системы может открыть раздел «Регистры» (показан на рисунке 23).

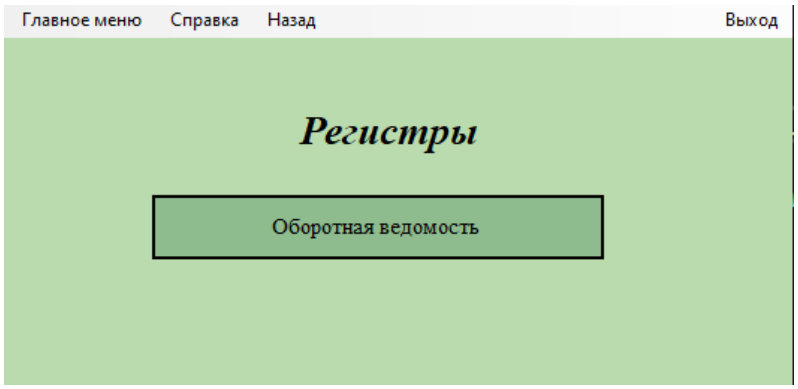

Рисунок 23 – Окно раздела «Регистры»

Затем, перейдя в раздел «Оборотная ведомость», сотрудник может выбрать нужные даты и вывести документ с остатками по выбранным датам и по выбранным продуктам. На рисунке 24 показано окно раздела «Оборотная ведомость».

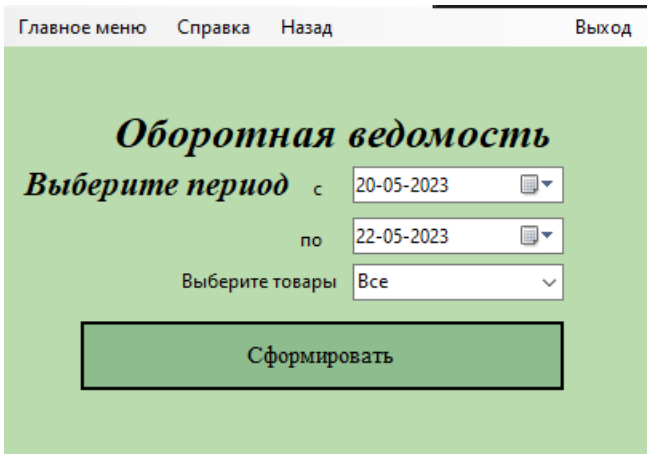

Рисунок 24 – Окно раздела «Оборотная ведомость»

На рисунке 25 приведен документ «Оборотная ведомость», а именно остатки на выбранный период.

Оборотная ведомость

По остаткам на складе

Период: 10.04.2023- 17.04.2023

Дата	№ документа	Наименование товара	Остаток на начало периода	Приход	Расход	Остаток на конец периода
10.04.2023	143	розы	0	200	0	200
12.04.2023	122	розы	200	0	21	179
12.04.2023	123	розы	179	0	101	68
17.04.2023	145	розы	68	50	0	118
17.04.2023	141	розы	118	0	70	48
10.04.2023	144	тюльпаны	0	200	0	200
12.04.2023	120	тюльпаны	200	0	100	100
16.04.2023	140	тюльпаны	100	0	70	30
17.04.2023	146	тюльпаны	30	50	0	80
17.04.2023	142	тюльпаны	80	0	30	50
10.04.2023	142	крафтовая бумага	0	100	0	100
12.04.2023	122	крафтовая бумага	100	0	2	98
12.04.2023	123	крафтовая бумага	98	0	2	96
17.04.2023	141	крафтовая бумага	96	0	2	94
12.04.2023	120	крафтовая бумага	94	0	2	92
16.04.2023	140	крафтовая бумага	92	0	2	90
17.04.2023	142	крафтовая бумага	90	0	2	88

Дата оформления документа: 17-04-2023

Рисунок 25 – Документ «Оборотная ведомость»

4.2 Оценка экономической эффективности используемой системы

Достижение максимальных результатов при минимальных затратах или уменьшение общих издержек на единицу продукции – такова экономическая эффективность. Для объективной оценки экономической эффективности основных компонентов автоматизированной информационной системы необходимо использовать различные методы оценки.

В первую очередь следует оценить энергопотребление рабочих мест сотрудников, которые занимаются своей деятельностью в цветочном магазине.

Затраты рассчитываются по следующей формуле:

$$Зэл = Ум * Праб * Цэ,$$

где Ум – употребляемая мощность персонального компьютера;

Праб – продолжительность работы персонального компьютера;

Цэ – стоимость киловатта электроэнергии.

В Краснодаре, где располагается цветочный магазин, стоимость 1 кВт электроэнергии для потребителей составляет 6 рублей по текущим тарифам. Компьютер в магазине работает в среднем 9 часов в день, соответствующих рабочим часам. Мощность, потребляемая персональным компьютером в магазине, составляет 0,22 кВт. Исходя из указанной формулы, средний расход электроэнергии в магазине за день составляет 6,6 рублей.

Таким образом, стоимость расхода электроэнергии составит: 6,6 * 25 = 165 рублей.

Стоимость оплаты труда сотрудника составляет 60000 рублей.

Рассчитаем стоимость 1 минуты сотрудника: 40000 / (9*25*60) = 2,96 рублей.

Таблица 1 – Сравнение эффективности подсистемы для сотрудников

Внедрение	Добавление заказа в базу, мин	Оформление накладной, мин	Внесение данных о товаре, мин
До	5	4	5
После	2	0,5	1,5

Один из сотрудников в среднем каждый день обрабатывает приблизительно 57 заявок. Для оформления накладной требуется около 4 минут на заявку, что в сумме составляет 228 минут в день.

Для внесения данных о товарах в базу требуется около 5 минут на заявку, что в сумме составляет 285 минут в день. Таким образом, до внедрения автоматизированной информационной системы сотрудник тратил на эти процессы 553 минут в день. В месяц этот же сотрудник занимался своей работой 13825 минут.

После внедрения АИС время, затрачиваемое на эти процессы, уменьшится: на оформление накладной будет тратиться 0,5*57 = 28,5 минут, а на внесение данных о товарах - 1,5*8 = 12 минут. Таким образом, после внедрения затраченное время составит 154,5 минут в день.

В первый месяц после внедрения АИС сотрудник займется работой, потратив 3862,5 минут. В результате экономия времени на сотрудника в год составит 119562 минуты (1992,7 часа).

Внедрение АИС обойдется в 43555, что будет окупаться за 1 месяц. Таким образом, разработанная информационная система документооборота способствует сокращению времени на оформление накладных, заказов и внесение данных о товарах, что экономически эффективно на долговременной перспективе.

Однако, для дальнейшего увеличения эффективности работы магазина по продаже цветов, следует изменить методику рационального использования рабочего времени.

ЗАКЛЮЧЕНИЕ

Таким образом, в ходе разработки автоматизированной системы документооборота для цветочного магазина были определены основные задачи и функциональные требования, которые необходимы для эффективной работы магазина. Созданная система позволяет ускорить процесс обработки заказов, уменьшить количество ошибок, связанных с неправильным заполнением документов, а также улучшить взаимодействие между сотрудниками.

В результате внедрения автоматизированной системы магазин стал работать более эффективно, быстро и качественно обрабатывать заказы, сократилось количество ошибок при оформлении документации.

Разработка автоматизированной системы документооборота для цветочного магазина имеет большое значение в условиях современного информационного общества, где компьютеризация и автоматизация процессов - ключ к успешности бизнеса. Использование такой системы позволяет значительно повысить эффективность работы бизнеса, сократить трудозатраты и снизить количество ошибок, связанных с ручной обработкой документов. Использование автоматизированной системы документооборота может быть востребовано в разных сферах бизнеса и позволит сократить расходы на административную работу.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Российская Федерация. Законы. Об информации, информационных технологиях и о защите информации : Федеральный закон № 149-ФЗ : текст с изменениями и дополнениями : принят Государственной Думой 8 июля 2006 года : одобрен Советом Федерации 14 июля 2006 года // КонсультантПлюс : справочная правовая система. – Москва, 1997.

2. Корчагин, А.Б. Модели данных для информационного взаимодействия / А.Б. Корчагин, И.Г. Лисьих, Д.А. Никифоров, Р.Л. Сиваков // International Journal of Open Information Technologies. 2017. №3. URL: https://cyberleninka.ru/article/n/modeli-dannyh-dlya-informatsionnogo-vzaimodeystviya (дата обращения: 09.02.2023).

3. Жалолов, О. И. Понятие SQL и реляционной базы данных / О. И. Жалолов, Х.У. Хаятов // Universum: технические науки. 2020. №6-1 (75). URL: https://cyberleninka.ru/article/n/ponyatie-sql-i-relyatsionnoy-bazy-dannyh (дата обращения: 09.03.2023).

4. Программа 1С: Розница 8: официальный сайт. – URL: https://www.vdgb.ru/katalog-1s/ torgovyy-i-skladskoy-uchet/torgovlya/1s-roznica-8/?cm_id = 70486325_ 47999 98233 (дата обращения: 08.02.2023).

5. Сайт компании СБИС: официальный сайт. – URL: https://sbis.ru/ (дата обращения: 08.02.2023)

6. Сайт компании Florapos: официальный сайт. – URL: https://posiflora.com/posiflora-florapoint-florapos/ (дата обращения: 08.02.2023)

7. Чеботарев, В. А. Выбор интегрированной среды разработки C#-приложений / В.А. Чеботарев, Н.П. Путивцева // Цифровая наука. 2021. №6-1. URL: https://cyberleninka.ru/article/n/vybor-integrirovannoy-sredy-razrabotki-c-prilozheniy (дата обращения: 02.03.2023).

8. Павловская, Т.А. C#. Программирование на языке высокого уровня. Учебник для вузов. / Т. А. Павловская. – Санкт Петербург: «Питер», 2009. – 432 с. – ISBN 978-5-91180-174-8.

9. Шамухамедов, Г.Х. Анализ современных методов хеширования / Г.Х. Шамухамедов, Н.К. Хыдыров, О.М. Союнова // Science Time. 2015. №6 (18). URL: https://cyberleninka.ru/article/n/analiz-sovremennyh-metodov-heshirovaniya (дата обращения: 10.03.2023).

10. Макаров, Н. С. Uml: поддержка проектирования и инструментальные среды / Н.С. Макаров // Прикладная информатика. 2007. №2. URL: https://cyberleninka.ru/article/n/uml-podderzhka-proektirovaniya-i-instrumentalnye-sredy (дата обращения: 03.02.2023).

11. Федосеева, А. С. Проектирование диаграммы прецедентов средствами UML на примере Web-приложения дома отдыха / А.С. Федосеева // Научные междисциплинарные исследования. 2021. №5. URL: https://cyberleninka.ru/article/n/proektirovanie-diagrammy-pretsedentov-sredstvami-uml-na-primere-web-prilozheniya-doma-otdyha (дата обращения: 03.02.2023).

12. Романов, С.С. Об инфологическом моделировании баз данных с помощью нормализации ER-диаграмм / С.С. Романов // Таврический научный обозреватель. 2017. №1 (18). URL: https://cyberleninka.ru/article/n/ob-infologicheskom-modelirovanii-baz-dannyh-s-pomoschyu-normalizatsii-er-diagramm (дата обращения: 09.03.2023).

ПРИЛОЖЕНИЕ А
Код формы «Заказы»

```
using Microsoft.EntityFrameworkCore.Metadata.Internal;
using Npgsql;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static
System.Windows.Forms.VisualStyles.VisualStyleElement;
using static
System.Windows.Forms.VisualStyles.VisualStyleElement.Window;
using Microsoft.Office.Interop.Excel;
using Microsoft.Office.Interop.Word;
using System.Xml;
using static
Microsoft.EntityFrameworkCore.DbLoggerCategory.Database;
using System.Reflection.Metadata;
using System.Reflection;
using System.Xml.Linq;
using System.CodeDom;
using System.Security.Cryptography;
using System.Drawing;
using static System.Collections.Specialized.BitVector32;
using Xceed.Words.NET;
namespace ProbaDiplom
{
    public partial class OrderWindow : Form
    {
        private string connstring =
String.Format("Server={0};Port={1};" +
        "User Id={2};Password={3};Database={4};",
        "localhost", 5432, "postgres",
        "root", "flowers_db");
        private NpgsqlConnection conn;
        private string sql;
        private string sql2;
        private NpgsqlCommand cmd;
        private System.Data.DataTable dt;
        private int rowIndex = -1;
        private string command;
        Connect connect = new Connect();
        public OrderWindow()
        {
```

```
            InitializeComponent();
            StartPosition = FormStartPosition.CenterParent; }

        private void Select()
        {
            try
            {
                conn.Open();
                sql = @"SELECT * FROM public.order ";
                cmd = new NpgsqlCommand(sql, conn);
                dt = new System.Data.DataTable();
                dt.Load(cmd.ExecuteReader());
                dgvDataOrders.DataSource = null; // reset
datagridiew
                dgvDataOrders.DataSource = dt;
                conn.Close();
            }
            catch (Exception ex)
            {
                conn.Close();
                MessageBox.Show("Error:" + ex.Message);
            }
        }

        private void OrderWindow_Load(object sender, EventArgs
e)
        {
            conn = new NpgsqlConnection(connstring);
            Select();
        }

        private void exsitToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            this.Close();
        }

        private void backToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            MainWindow mainWin = new MainWindow();
            mainWin.Show();
            this.Hide();
        }

        private void mainMenuToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            MainWindow mainWin = new MainWindow();
            mainWin.Show();
            this.Hide();
        }
```

```
        private void dgvDataOrders_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            if (e.RowIndex >= 0)
            {
                rowIndex = e.RowIndex;
                numberButton.Text =
dgvDataOrders.Rows[e.RowIndex].Cells["number"].Value.ToString();
                ProductcomboBox.Text =
dgvDataOrders.Rows[e.RowIndex].Cells["product"].Value.ToString()
;
                KolvoButton.Text =
dgvDataOrders.Rows[e.RowIndex].Cells["volume"].Value.ToString();
                PriceButton.Text =
dgvDataOrders.Rows[e.RowIndex].Cells["cost_order"].Value.ToStrin
g();
                all_priceButton.Text =
dgvDataOrders.Rows[e.RowIndex].Cells["all_cost"].Value.ToString(
);
            }
        }

        private void dgvDataOrders_CellContentClick(object
sender, DataGridViewCellEventArgs e)
        {

        }

        private void updateButton_Click(object sender, EventArgs
e)
        {
            rowIndex = -1;
            numberButton.Enabled = ProductcomboBox.Enabled =
KolvoButton.Enabled = PriceButton.Enabled =
all_priceButton.Enabled = dateTimePicker.Enabled = true;
            numberButton.Text = ProductcomboBox.Text =
KolvoButton.Text = PriceButton.Text = all_priceButton.Text =
dateTimePicker.Text = null;
            numberButton.Select();

            sql2 = @"SELECT * from product";
            System.Data.DataTable dt = new
System.Data.DataTable("product");
            conn.Open();
            NpgsqlDataAdapter DA = new NpgsqlDataAdapter(sql2,
conn);
            DA.Fill(dt);
            conn.Close();

            ProductcomboBox.DataSource = dt;
            ProductcomboBox.DisplayMember = "name";
```

```
            ProductcomboBox.ValueMember = "id_product";
        }

        private void safeButton_Click(object sender, EventArgs
e)
        {
            int result = 0;
            if (rowIndex < 0) // insert
            {
                try
                {
                    conn.Open();
                    sql = @"SELECT * from order_insert(:_number,
:_product, :_volume, :_cost_order, :_all_cost, :_date)";
                    cmd = new NpgsqlCommand(sql, conn);
                    cmd.Parameters.AddWithValue("_number",
numberButton.Text);
                    cmd.Parameters.AddWithValue("_product",
ProductcomboBox.SelectedValue);
                    cmd.Parameters.AddWithValue("_volume",
int.Parse(KolvoButton.Text));
                    cmd.Parameters.AddWithValue("_cost_order",
int.Parse(PriceButton.Text));
                    cmd.Parameters.AddWithValue("_all_cost",
int.Parse(all_priceButton.Text));
                    cmd.Parameters.AddWithValue("_date",
dateTimePicker.Value.Date);
                    result = (int)cmd.ExecuteScalar();
                    conn.Close();
                    if (result == 1)
                    {
                        MessageBox.Show("Inserted new product
seccessfully");
                        Select();
                    }
                    else
                    {
                        MessageBox.Show("Inserted fail");
                    }

                }
                catch (Exception ex)
                {
                    conn.Close();
                    MessageBox.Show("Произошла ошибка: " +
ex.Message);
                }
            }
            else //update
            {
                try
                {
```

```
                    conn.Open();
                    sql = @"SELECT *  from
order_update(:_id_order, :_number, :_product, :_volume,
:_cost_order, :_all_cost, :_date)";
                    cmd = new NpgsqlCommand(sql, conn);
                    cmd.Parameters.AddWithValue("_id_order",
int.Parse(dgvDataOrders.Rows[rowIndex].Cells["id_order"].Value.T
oString()));
                    cmd.Parameters.AddWithValue("_number",
numberButton.Text);
                    cmd.Parameters.AddWithValue("_product",
ProductcomboBox.SelectedValue);
                    cmd.Parameters.AddWithValue("_volume",
int.Parse(KolvoButton.Text));
                    cmd.Parameters.AddWithValue("_cost_order",
int.Parse(PriceButton.Text));
                    cmd.Parameters.AddWithValue("_all_cost",
int.Parse(all_priceButton.Text));
                    cmd.Parameters.AddWithValue("_date",
dateTimePicker.Value.Date);
                    result = (int)cmd.ExecuteScalar();
                    conn.Close();
                    if (result == 1)
                    {
                        MessageBox.Show("Update seccessfully");
                        Select();
                    }
                    else
                    {
                        MessageBox.Show("Updated fail");
                    }
                }
                catch (Exception ex)
                {
                    conn.Close();
                    MessageBox.Show("Update fail. Error: " +
ex.Message);
                }
            }
            result = 0;
            numberButton.Text = ProductcomboBox.Text =
KolvoButton.Text = PriceButton.Text = all_priceButton.Text =
dateTimePicker.Text = null;
            numberButton.Enabled = ProductcomboBox.Enabled =
KolvoButton.Enabled = PriceButton.Enabled =
all_priceButton.Enabled = dateTimePicker.Enabled = false;
        }

        private void deleteButton_Click(object sender, EventArgs
e)
        {
            if (rowIndex < 0)
```

```
            {
                MessageBox.Show("Выберите сотрудника которого
хотите удалить!");
                return;
            }
            try
            {
                conn.Open();
                sql = @"select * from order_delete(:_id_order)";
                cmd = new NpgsqlCommand(sql, conn);
                cmd.Parameters.AddWithValue("_id_order",
int.Parse(dgvDataOrders.Rows[rowIndex].Cells["id_order"].Value.T
oString()));
                if ((int)cmd.ExecuteScalar() == 1)
                {
                    MessageBox.Show("Заказ удален!");
                    rowIndex = -1;
                    conn.Close();
                    Select();
                }
                else
                {
                    conn.Close();
                }
            }
            catch (Exception ex)
            {
                conn.Close();
                MessageBox.Show("Произошла ошибка при удалении:
" + ex.Message);
            }
        }

        private void dateTimePicker1_ValueChanged(object sender,
EventArgs e)
        {
            dateTimePicker.CustomFormat = "dd-MM-yyyy";
        }

        private void flowLayoutPanel1_Paint(object sender,
PaintEventArgs e)
        {

        }

        private void textBox1_TextChanged(object sender,
EventArgs e)
        {

        }
```

```
        private void SortedButton_Click(object sender, EventArgs
e)
        {
            var FROMdate =
dateTimePicker2.Value.Date.ToString("yyyy-MM-dd");
            var TOdate =
dateTimePicker1.Value.Date.ToString("yyyy-MM-dd");

            try
            {
                conn.Open();
                sql = $"SELECT * FROM public.order WHERE date
BETWEEN '{FROMdate}' AND '{TOdate}'";
                cmd = new NpgsqlCommand(sql, conn);
                dt = new System.Data.DataTable();
                dt.Load(cmd.ExecuteReader());
                dgvDataOrders.DataSource = null; // reset
datagridiew
                dgvDataOrders.DataSource = dt;
                conn.Close();
            }
            catch (Exception ex)
            {
                conn.Close();
                MessageBox.Show("Error:" + ex.Message);
            }
        }

        private void dateTimePicker2_ValueChanged(object sender,
EventArgs e)
        {
            dateTimePicker2.CustomFormat = "dd-MM-yyyy";
        }

        private void dateTimePicker1_ValueChanged_1(object
sender, EventArgs e)
        {
            dateTimePicker1.CustomFormat = "dd-MM-yyyy";
        }

        private void SearchButton_Click(object sender, EventArgs
e)
        {
            var search = SearchTextBox.Text;

            try
            {
                conn.Open();
                sql = $"SELECT * FROM public.order WHERE number
= '{search}'";
                cmd = new NpgsqlCommand(sql, conn);
                dt = new System.Data.DataTable();
```

```
                dt.Load(cmd.ExecuteReader());
                dgvDataOrders.DataSource = null; // reset
datagridiew
                dgvDataOrders.DataSource = dt;
                conn.Close();
            }
            catch (Exception ex)
            {
                conn.Close();
                MessageBox.Show("Error:" + ex.Message);
            }
        }

        public int A()
        {
            var schet = NumberTextBox.Text;

            using (var conn = new NpgsqlConnection(connstring))
            {
                conn.Open();
                using (var cmd = new NpgsqlCommand($"SELECT
COUNT(*) FROM public.order WHERE number = '{schet}'", conn))
                {
                    int count =
Convert.ToInt32(cmd.ExecuteScalar()) + 1;
                    return count;
                }
            }
        }

        public int B()
        {
            var schet = NumberTextBox.Text;

            using (var conn = new NpgsqlConnection(connstring))
            {
                conn.Open();
                using (var cmd = new NpgsqlCommand($"SELECT
SUM(all_cost) FROM public.order WHERE number = '{schet}'",
conn))
                {
                    int count =
Convert.ToInt32(cmd.ExecuteScalar());
                    return count;
                }
            }
        }
        private void OrderButton_Click(object sender, EventArgs
e) {
            try
            {
                var schet = NumberTextBox.Text;
```

```
                conn.Open();
                command = $"SELECT name, volume, cost_order,
all_cost FROM public.order FULL JOIN product ON
public.order.product=product.id_product WHERE number =
'{schet}'";

                var datatable = new System.Data.DataTable();
                queryReturnData(command, datatable);
                ExportToWord(datatable);
                conn.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
        public System.Data.DataTable queryReturnData(string
command, System.Data.DataTable dataTable)
        {
            NpgsqlConnection myCon = new
NpgsqlConnection(connstring);
            myCon.Open();
            NpgsqlDataAdapter SDA = new
NpgsqlDataAdapter(command, conn);
            SDA.SelectCommand.ExecuteNonQuery();
            SDA.Fill(dataTable);
            return dataTable;
        }

        private void ExportToWord(System.Data.DataTable
dataTable)
        {
            if (dataTable.Rows.Count > 0)
            {
                object missing =
System.Reflection.Missing.Value;
                Microsoft.Office.Interop.Word.Application word =
new Microsoft.Office.Interop.Word.Application();
                Microsoft.Office.Interop.Word.Document document
= word.Documents.Add(ref missing, ref missing, ref missing, ref
missing);
                Microsoft.Office.Interop.Word.Paragraph para1 =
document.Content.Paragraphs.Add(ref missing);
                //Добавление текста со стилем Заголовок 1
                object styleHeading1 = "Заголовок 1";
                para1.Range.set_Style(styleHeading1);
                para1.Range.set_Style(styleHeading1);
                para1.Range.Text = "Товарный чек";
                para1.Alignment =
Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagr
aphCenter;
                para1.Range.InsertParagraphAfter();
```

```
                object styleHeading2 = "Заголовок 3";
                para1.Range.set_Style(styleHeading2);
                para1.Range.Text = "Заказ №" +
NumberTextBox.Text;
                para1.Range.InsertParagraphAfter();
                //Заголовок документа
                word.Documents.Application.Caption = "Товарный
чек";
                Microsoft.Office.Interop.Word.Table table =
document.Tables.Add(para1.Range, dataTable.Rows.Count + 1,
dataTable.Columns.Count, Type.Missing, Type.Missing);
                table.Borders.OutsideLineStyle =
Microsoft.Office.Interop.Word.WdLineStyle.wdLineStyleSingle;
                table.Borders.InsideLineStyle =
Microsoft.Office.Interop.Word.WdLineStyle.wdLineStyleSingle;
                table.Cell(1, 1).Range.Text = "Название";
                table.Cell(1, 2).Range.Text = "Количество";
                table.Cell(1, 3).Range.Text = "Цена";
                table.Cell(1, 4).Range.Text = "Стоимость";

                for (int i = 0; i < dataTable.Rows.Count; i++)
                {
                    for (int j = 0; j < dataTable.Columns.Count;
j++)
                    {
                        table.Cell(i + 2, j + 1).Range.Text =
dataTable.Rows[i][j].ToString();
                    }
                }
                Microsoft.Office.Interop.Word.Paragraph para2 =
document.Content.Paragraphs.Add(ref missing);
                object styleHeading4 = "Заголовок 3";
                para2.Range.set_Style(styleHeading4);
                para2.Range.set_Style(styleHeading4);
                DateTime myDateTime = DateTime.Now;
                string sqlFormattedDate =
myDateTime.ToString("dd-MM-yyyy");
                para2.Range.Text = "Итого: " + B();
                para2.Range.InsertParagraphAfter();
                object styleHeading5 = "Заголовок 3";
                para2.Range.set_Style(styleHeading5);
                para2.Range.Text = "Дата оформления заказа: " +
sqlFormattedDate;
                para2.Range.InsertParagraphAfter();
                word.Visible = true;}
            else
            {
                MessageBox.Show("No data to export!");
            }
        }
    }
}
```