МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение
высшего образования
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ
ГИДРОМЕТЕОРОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Экономики и управления на предприятии природопользования»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

(бакалаврская работа)
по направлению подготовки 09.03.03 Прикладная информатика
(квалификация – бакалавр)

На тему«Разработка информационной системы учёта товаров»

Исполнитель Овчинников Никита Сергеевич

Руководитель к.г.н., доцент по кафедре информатики и прикладной математике
Полупанов Владимир Николаевич

«К защите допускаю»

Руководитель кафедры _____

кандидат экономических наук

Майборода Евгений Викторович

«20» 01 2025 г.

Филиал Российского государственного гидрометеорологического университета в г. Туапсе
НОРМОКОНТРОЛЬ ПРОЙДЕН
«18» 01 2025 г.
Мартыненова Т.В.
ПОДПИСЬ РАСШИФРОВКА ПОДПИСИ

Туапсе
2025

ОГЛАВЛЕНИЕ

Введение

Развитие информационных технологий непрерывно меняет способы управления и организации бизнес-процессов в современных организациях. В полной мере это относится и к учёту товаров. Однако учёт малых и микропредприятий может существенно отличаться от учёта средних и, тем более, крупных предприятий, что отражается и на методах автоматизации учёта. Методыотличаются по функциональности, трудоёмкости использования и, пожалуй, самое главное, стоимости приобретения и эксплуатации.

Финансовое состояние микропредприятий в современный период таково, что для приобретения требуемой системы автоматизации складского учёта возможны лишь чрезвычайно ограниченные инвестиции. Снижение стоимости проектирования и эксплуатации такой системы в современных условиях представляется возможным путём использования открытых (opensoursesoftware)информационных технологий.

В настоящей выпускной квалификационной работе рассматривается возможность построения информационной системы, предназначенной для автоматизации учёта товаров малого предприятия, относящихся к категории микропредприятий.

Объект исследований –малое предприятие в г. Краснодаре, которое соответствии с нормативно утверждёнными критериями относится к микропредприятиям, так как его выручка не достигает 120 млн. руб. в год, а численность сотрудников менее 15 человек.

Предмет исследований – информационная системадля учёта товаров, предназначенная длямикропредприятия.

Актуальность работы обусловлена недостаточностью исследований по выбранной тематике в связи с появлением новых информационных технологий, которые могли бы повысить эффективность деятельности микропредприятий.

Цель выпускной квалификационной работы – проектирование информационной системы по учёту товаров микропредприятия.

Для достижения поставленной цели необходимо решить задачи:

— провести анализ объекта и предмета исследования;

— рассмотретьсуществующие решения для учёта товаров;

— составить обзор инструментария для реализации проекта;

— предложить проект приложения для микропредприятия

— оценить экономическую эффективность предложенного проекта.

Выполнениенастоящего исследования проводится с применением следующих методов: анализ и синтез теоретического материала в области складского учёта и соответствующих информационных технологий, моделирование функционирования составных элементов системы, оценка экономической эффективности проекта.

Теоретической и методической основой исследования служат отечественные и зарубежныеработыпо методологии и методамразработки информационных систем по складскому учёту, нормативно-справочная информация по информационным технологиям и инструментарию, используемым при разработке систем такого типа.

Проектирование информационной системы предполагается с использованием облегчённой методологиис ориентацией на задачи, с учётом масштабов микропредприятия и небольшого числа разработчиков.

Практическая значимость работы состоит в том, что её результаты могут быть использованы при проектировании информационной системы, ориентированной на повышение эффективности деятельности учёта товаров с учётом особенностей микропредприятия.

Работа состоит из введения, трёх основных разделов, заключения и списка использованной литературы. Общий объем работы составляет ?? страницы машинописного текста, которые содержат ? таблиц и ? рисунков.

1 Аналитическая часть

1.1 Анализ объекта и предмета исследования

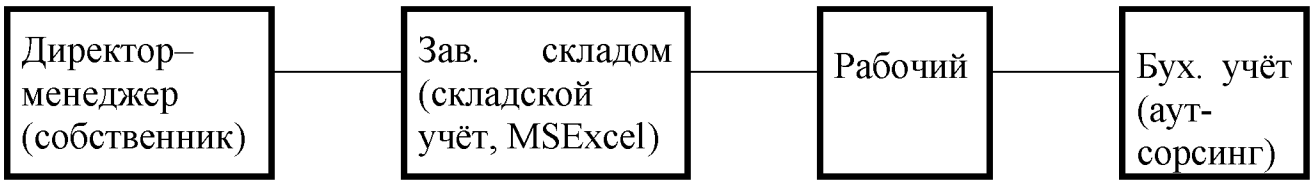
Объект исследования – малое предприятие, котороесоответствии с нормативно утверждёнными критериями относится к микропредприятиям, так как его выручка не достигает 120 млн. руб. в год, а численность сотрудников менее 15 человек.

По данным «Интерфакс» (spark-interfax.ru)организацийв Краснодаре, относящихся к микропредприятиям в настоящее время зарегистрировано около 130 единиц. Динамика их деятельности во многом похожа по основным финансовым показателям, так что ООО «МТ-ЮГ» можно считать типовым микропредприятием Краснодара.

На рисунок 1.1 показан ход выручки и чистой прибыли одного из таких предприятий (по данным экспертов компании TestFirm, которое можно взять в качестве типового.

Рисунок 1.1-Выручка и чистая прибыль микропредприятия

Сумма накопленной нераспределённой прибыли за 2018÷2023 г.г. составляет около 720 тыс. руб., а среднегодовая нераспределённая прибыль – 120 тыс. руб. При этом наблюдается тенденция к снижению прибыли. Собственник (он же –директор) в 2021 году сократил численность работников с пяти до трёх, себя наделяет функциями директора (управляющего), оставляет заведующего складом и рабочего, а бухгалтерский учёт отдаёт в аутсорсинг. В результате реорганизации структура предприятия приобретает вид:



Однако финансовые показатели продолжают снижаться.Анализ бизнес-процессов показывает, что одна из причин снижения финансовых показателей – складской учёт «вручную» с использованием MSExcel. Принимается решение о замене учёта «вручную» на автоматизированный с последующим сокращением должности зав. складом в расчёте на возможность выполнения задач учёта непосредственно директором.

На установку/разработку предполагаемой системы складского учёта собственник имеет возможность выделить 720 тыс. руб. (накопленная прибыль).

Предполагаемый срок установки/разработки и внедрения – в течение года.

По предварительным прикидкам становится понятно, за выделенную сумму можно установить/разработать систему с ограниченным функционалом, выполняющим лишь самые неотложные задачи учёта, но с уверенностью, что функционал системы поддаётся необходимому расширению в последующие годы.

Понятно также, что для выполнения работ по поставке и внедрению системы на таких условиях, можно привлечь лишь одного IT-специалиста, которым оказался студент последнего курса соответствующего образовательного учреждения.

В информационную систему, которая используется для того, чтобы вёлся учёт товаров, включены самые разнообразные компоненты. Такие компоненты могут быть как аппаратными, так и программными. Каждый компонент обеспечивает выполнение своей собственной функции, каждая из которых, в свою очередь, способствует увеличению уровня автоматизации учётных процессов, налаживаемых на предприятии. Конечная цель, которая

преследуется при создании в компании информационной учётной системы продукции – это обеспечение постоянного отслеживания статуса продукции, её количества, её качества, а также её местонахождения в конкретный момент времени. Получая информацию из соответствующей системы, уполномоченное лицо может принимать различные решения, касающиеся распределения продукции, а также её учёта.

Рассмотрим более подробно вопрос о том, какие конкретные функции могут реализовываться учётной системой, предназначенной для работы с продукцией предприятия. Так, во-первых, данная система ответственна за фиксацию факта поступления каждого товара на склад, где она будет размещаться до момента её востребования (в этот момент в систему вносятся данные, характеризующие объём поступившей продукции, её цену, а также её поставщика). Во-вторых, данная система ответственна за фиксацию факта отгрузки продукции покупателю. В-третьих, данная система ответственна за то, чтобы фиксировать текущее количество остатков продукции (в разбивке по отдельным видам). В-четвёртых, данная система ответственна за фиксацию стоимости, которая имеется у каждой продукции, поступившей на предприятие с целью её дальнейшей реализации. В-пятых, данная система ответственна за то, чтобы формировать, а также выводить отчётность, содержащую сведения о поступивших в компанию товарах, в распоряжение лиц, уполномоченных работать с таковой отчётностью.

Правильно выстроенное применение учётной системы способствует совершенствованию качества управления товарами на предприятии, а также сокращению объёма ошибок, допускаемых при управлении товарами на предприятии[1, с.44].

1.2 Обзор решений, имеющихся на рынке

В данной части работы мы кратко охарактеризуем те решения, которые уже имеются на рынке и эксплуатируются.

Среди преимуществ, которые характерны для таких продуктов, мы можем обратить внимание на следующие. Во-первых, они позволяют существенно нарастить степень автоматизации всех тех процессов, которые создаются в компании, чтобы она учитывала собственную продукцию, приобретаемую для дальнейшего использования, а также для дальнейшей реализации. Во-вторых, они предоставляют возможность вывести на принципиально новый уровень качество принимаемых и реализуемых управленческих решений, непосредственно касающихся продукции. В-третьих, благодаря им можно делать функционирование всего предприятия в большей степени производительным, эффективным. В-четвёртых, благодаря их внедрению можно сокращать временные затраты на выстраивание коммуникационных процессов между разными структурными подразделениями одной и той же компании, каждое из которых вносит свой вклад в процесс движения товара, поступающего на предприятие. В-пятых, автоматизированные учётные системы – это то, без чего немыслимо достижение высокого уровня клиентской удовлетворённости.

Однако, несмотря на все вышеперечисленные преимущества, характерные для уже появившихся решений по автоматизации процессов учёта продукции на предприятии, у них же имеются также и недостатки, также, в свою очередь, нуждающиеся в исследовании и анализе. Прежде всего обратим внимание на тот факт, что подобные системы крайне сложно внедрять в повседневное функционирование компании (в особенности это характерно для тех случае, когда на предприятии ранее не имелось никаких автоматизирующих продуктов). Кроме того, если компания делает решение в пользу внедрения полностью готового программного продукта, то она увеличивает свою степень зависимости от производителя данного продукта (и если по тем или иным причинам поставщик не сможет или не изъявит желания оказывать сервисные услуги, то на продолжительное время закупленное автоматизирующее решение не сможет полноценно эксплуатироваться). Ещё можно отметить, что увеличивается риск безопасности, если компания принимает решение о

внедрении автоматизирующего решения в свои процессы, связанные с перемещением продукции (ведь потенциально все данные могут оказаться в распоряжении третьих лиц) [2, с.101].

1.3Требования, которым должна соответствовать информационная учётная система, используемая с целью работы с товаром в организации

Требования, которым должна соответствовать информационная учётная система, используемая с целью работы с товаром в организации, будут нами рассмотрены в настоящей части исследования. Данные требования касаются, во-первых, того, насколько упомянутые системы надёжны; во-вторых, того, насколько упомянутые системы точно обрабатывают информацию, что в них вносится; в-третьих, того, насколько упомянутые системы автоматизированы; в-четвёртых, того, насколько упомянутые системы масштабируемы; в-пятых, насколько упомянутые системы удобны для того, чтобы их эксплуатацией мог заниматься персонал с разным уровнем и профилем подготовки; в-шестых, того, насколько упомянутые системы много позволяют составлять и анализировать отчётов; в-седьмых, того, насколько упомянутые системы гибки с точки зрения их применения как инструмента для управления ценами, устанавливаемыми на реализацию продаваемой компанией продукции. Только в том случае, если информационный продукт, решение о внедрении которого предприятие принимает, соответствует каждому из вышеперечисленных требований, становится реальной его быстрая адаптация к функционированию конкретной фирмы[3, с.186].

Отметим, что все требования, которые предъявляются к тому, как должны функционировать системы, используемые с целью автоматизации учётных процессов товаров предприятия, могут быть либо функциональными, либо нефункциональными. Функциональные требования – это такие требования, которые непосредственно касаются функционала, реализуемого конкретным программным продуктом. Когда формулируется то или иное функциональное

требование, фактически определяется, что именно автоматизирующая система должна делать в тех или иных ситуациях, как именно должно быть выстроено её функционирование в той или иной ситуации. Нефункциональные же требования не менее важны, чем функциональные. Они касаются того, какие параметры должны быть у системы, внедряемой в компанию для того, чтобы автоматизировать учётные процессы её продукции. Чем больше степень соответствия информационной системы предъявляемым по отношению к ней нефункциональным требованиям, тем качественнее она становится[4, с.232].

Проведём краткий обзор функциональных требований, которые должны выполняться информационной системой, эксплуатируемой с целью учёта товаров, поступающих на предприятие:

- требования, которые касаются учёта входящих товаров (выполняя их, система обеспечивает фиксацию каждого факта поступления товара на склад компании);

- требования, которые касаются учёта исходящих товаров (выполняя их, система обеспечивает фиксацию каждого факта убытия товара со склада компании);

- требования, которые касаются формирования отчётности о перемещении товаров на предприятии;

- требования, которые касаются управления ценами, по которым компания отпускает свои товары клиентам.

Проведём краткий обзор нефункциональных требований, которые должны выполняться информационной системой, эксплуатируемой с целью учёта товаров, поступающих на предприятие:

- требования, которые касаются безопасности сведений, вносимых в систему;

- требования, которые касаются производительности, присущей системе;

- требования, которые касаются надёжности, присущей системе;

- требования, которые касаются эргономичности, присущей системе;

- требования, которые касаются масштабируемости, присущей системе;

- требования, которые касаются доступности, присущей системе;
- требования, которые касаются аудита функционирования системы.

1.4 Архитектура информационной системы

Вариантов архитектуры, которые сегодня могут быть использованы для того, чтобы в компании была выстроена система, обеспечивающая автоматическое управление всеми процессами, касающимися товарного перемещения, существует множество. Самые распространённые из данных вариантов классифицируются на группы, в частности, это могут быть распределённые варианты, централизованные варианты, объектно-ориентированные варианты, а также микросервисные варианты. Все подобные варианты имеют как свои плюсы, так и свои минусы. И это, вне всякого сомнения, должно приниматься во внимание тогда, когда решается, какой именно из вариантов автоматизации будет применён в конкретном предприятии.

Прежде всего рассмотрим особенности, присущие такому варианту архитектуры, как централизованный. В том случае, когда в компании применяется централизованная архитектура, в ней создаётся и применяется общая база данных. К тем сведениям, которые находятся на хранении в такой базе данных, обращаются абсолютно все иные составляющие системы. Рассматриваемый вариант имеет такие преимущества, как относительная простота внедрения, а также относительная простота в управлении. Кроме того, при использовании систем, чья архитектура выстроена именно по такому варианту, можно сводить к минимуму ресурсные потери на обработку данных. Что же касается недостатков, то среди них выделяется прежде всего узость шины, из-за чего подобные системы практически лишены возможности масштабирования. Также следует отметить, что в случае появления отказов в работе централизованного хранилища данных фактически перестаёт работать вся система, что в определённых случаях может быть критически важным.

То, как обустроена система, имеющая централизованный тип архитектуры, продемонстрировано на рисунке 1.2. Как следует из тех данных, что продемонстрированы на рисунке 1.2, в системах имеется один центральный сервер, представляющий собой, таким образом, основной узел. Все остальные устройства, входящие в состав системы, имеют подключение к основному узлу (способов подобного подключения несколько, так, оно может быть реализовано либо через сеть, либо через Интернет). Каждая информация, которая вносится в систему, а также каждая команда, которая поступает в систему, сначала попадает на сервер, после чего он её обрабатывает. Функционал же клиентских устройств ограничен: фактически они необходимы только для того, чтобы выводить результаты обработки данных[5, с.21].

Рисунок 1.2–Обустройство архитектуры централизованного типа

Далее перейдём к изучению такого типа архитектуры, как распределённая. В том случае, когда используется распределённая архитектура, разные критические составляющие находятся не на одном сервере, а на разных. Благодаря такому типу архитектуры обработка данных осуществляется

параллельным образом. В отличие от централизованных архитектур, распределённые архитектуры хорошо поддаются масштабированию, в связи с чем их можно применять с целью разработки таких систем, что впоследствии рассматриваются под увеличение. Кроме того, системы, чьи архитектуры являются распределёнными, при возникновении сбоев утрачивают возможность выполнения лишь отдельных функций, а не всех функций сразу.

То, как именно выглядит обустройство системы, выстроенной в соответствии с распределённой архитектурой, показано на рисунке 1.3. Анализируя данные, которые показаны наи рисунке 1.3, мы можем отметить следующее: в системе всегда присутствует такой компонент, который ответственен за «балансировку» нагрузки. Данный компонент работает таким образом, чтобы все запросы, поступающие в систему для обработки, распределялись между серверами как можно более равномерным образом[6, с.33].

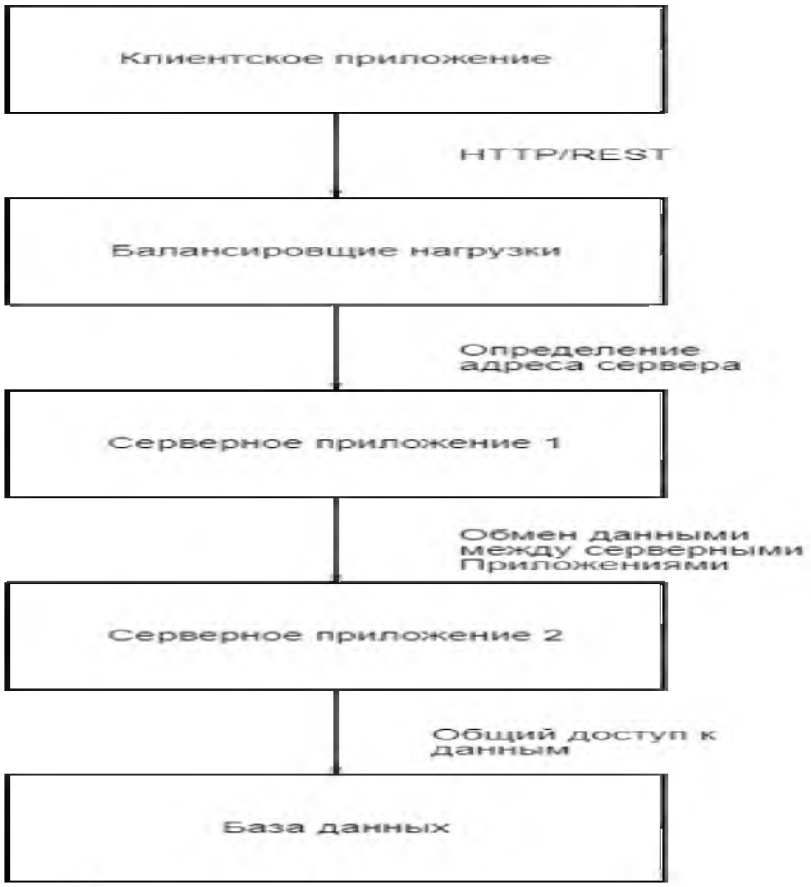


Рисунок 1.3–Обустройство системы, которая выстроена в соответствии с архитектурой распределённого типа

Тогда, когда при внедрении автоматизирующей системы принято решение о применении распределённой архитектуры, можно вносить изменения в перечень серверных приложений, если соответствующая необходимость появляется.

Микросервисная архитектура:

Суть архитектуры микросервисного типа заключается в том, что вся система оказывается разбитой на большое количество составляющих, каждая из которых представляет собой отдельно взятый сервис («микросервис»). Микросервис, в свою очередь, ответственен за то, чтобы обеспечивать нормальное функционирование конкретного процесса (или же выполнение конкретной функции). У тех систем, чья архитектура является микросервисной, имеется ряд преимуществ. Так, они достаточно гибки в своей работе, кроме того, они хорошо масштабируются и сопротивляются отказам. Обращая внимание на присущие им недостатки, мы можем отметить, что микросервисные системы достаточно сложно мониторить, кроме того, их постоянная работа приводит к появлению существенной нагрузки, приходящейся на сеть[17, с.273].

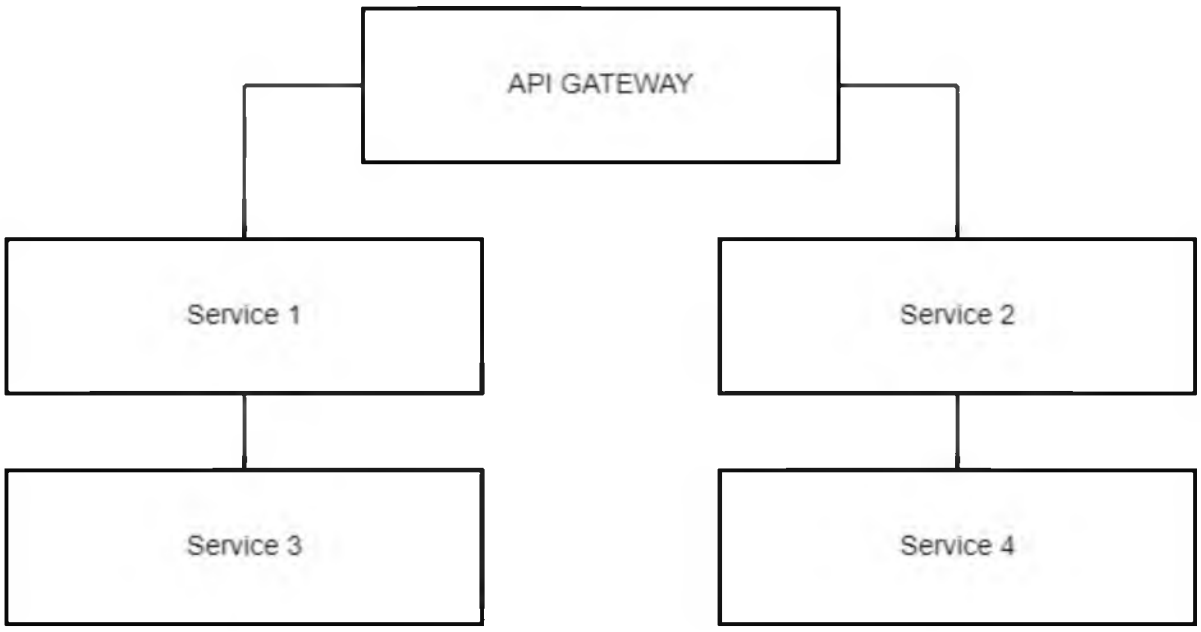


Рисунок 1.4 - Схема микросервисной архитектуры

Отличительной особенностью, которая характерна для той системы, чья архитектура – микросервисная, является наличие компонента API Gateway. Он отвечает за то, чтобы проводить обработку каждого запроса, поступающего в систему. После того, как поступивший запрос обработан, он распределяется на конкретный микросервис, относящийся к системе.

Для того, чтобы микросервисы, входящие в состав одной и той же системы, имели возможность взаимодействовать друг с другом, применяются разнообразные варианты. Например, у микросервисов может быть налажена прямая связь друг с другом. Кроме того, микросервисы могут контактировать также и посредством событийной шины. Конкретных способов обустройства микросервисной архитектуры существует великое множество, таким образом, их можно адаптировать к запросам и особенностям функционирования конкретной организации наиболее точно и тщательно[7, с.199].

Объектно-ориентированная архитектура:

У объектно-ориентированной архитектуры, как и у микросервисной архитектуры, присутствует особенность, заключающаяся в следующем: её можно разбивать на несколько отдельных структур, между которыми налажено взаимодействие. В совокупности все эти структуры могут обеспечивать постоянную реализацию сложных и многочисленных функций.

У объектно-ориентированной архитектуры имеется ряд преимуществ, о которых нами далее будет рассказано более подробно. Так, в тех случаях, когда архитектура является объектно-ориентированной, в ней код может быть применён повторно. Кроме того, системы, выстроенные в соответствии с такой архитектурой, хорошо поддаются масштабированию. А ещё при внедрении тех систем, у которых имеется такая архитектура, можно сокращать временные и ресурсные затраты на сопровождение функционирования [8, с.165]. Впрочем, у объектно-ориентированной архитектуры, как и у любого иного вида архитектур, имеются также и минусы. Так, они достаточно сложны (не только с точки зрения их внедрения, но также и с точки зрения развёртывания их дальнейшей эксплуатации). Кроме того, в некоторых случаях те возможности,

которые предлагаются архитектурами объектно-ориентированного типа, избыточны и не являются необходимыми. Обратим внимание ещё и на следующий факт: принятие решения в пользу внедрения системы с объектно-ориентированной архитектурой может иметь своим следствием рост накладных затрат, который не будет подтверждаться соответствующим увеличением производительности.

На рисунке 1.5 нами продемонстрировано обустройство той системы, чья архитектура характеризуется как объектно-ориентированная.

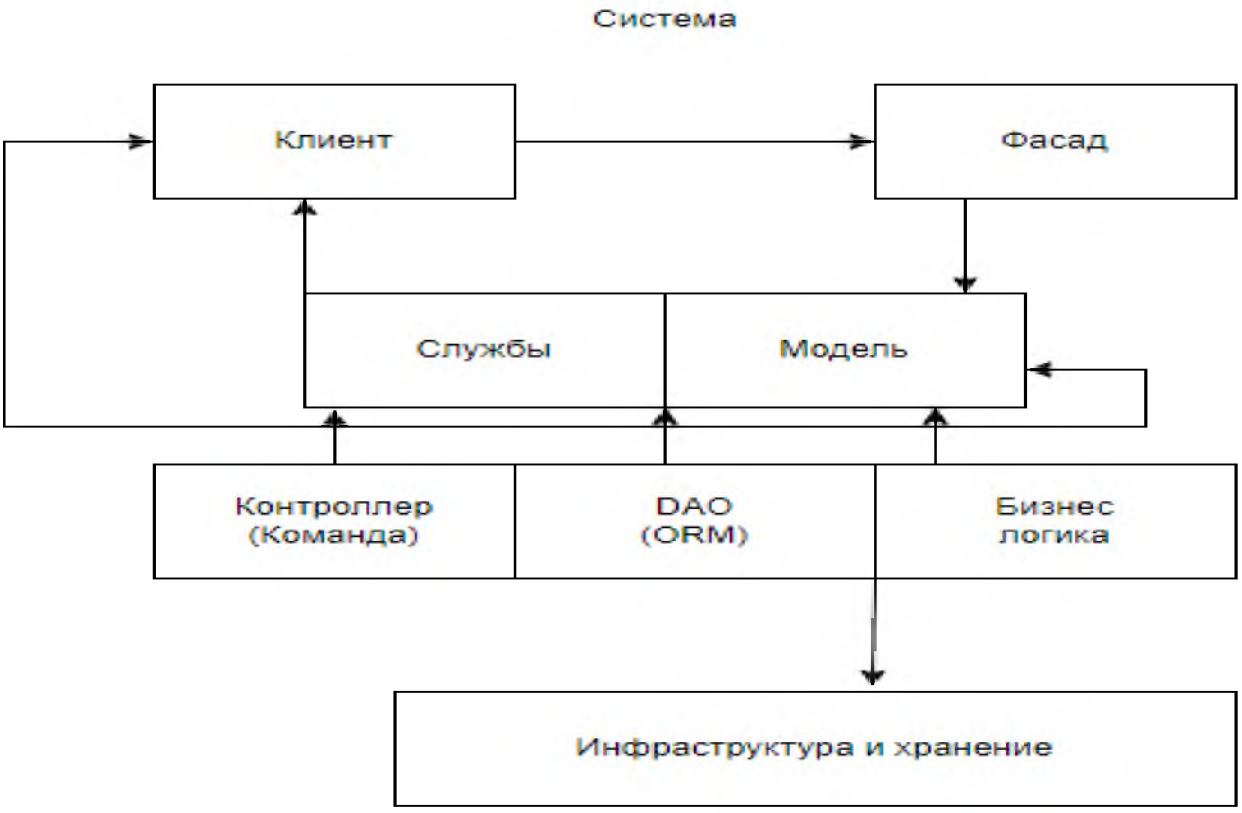


Рисунок 1.5 - Обустройство той системы, чья архитектура характеризуется как объектно-ориентированная

Таким образом, номенклатура компонентов, которые интегрированы в систему объектно-ориентированного типа, является достаточно многочисленной. Её основные составные части – это службы, клиент, фасад, контроллер, модель, инфраструктура, DAO, а также бизнес-логика. Компонент «клиент» не имеет непосредственного отношения к системе, поскольку это внешний пользователь, применяющий её в каких-либо своих целях. Компонент

«фасад» обеспечивает снижение сложностей в работе с объектно-ориентированной системой (в особенности это актуально для тех пользователей, которые не имеют соответствующего уровня подготовки). Компонент «служба» — это компонент, ответственный за реализацию конкретного функционала, выполняемого системой[16, с.102]. Компонент «модель» содержит в себе определённые правила, согласно которым производится обработка поступающих в систему сведений. Компонент «контроллер» является одним из наиболее важных, поскольку именно он в первую очередь работает с клиентскими запросами, выбирая, в частности, ту модель, согласно которой они будут обрабатываться в дальнейшем. Объект «бизнес-логика» содержит правила, устанавливающие, как именно производится генерация бизнес-данных, как именно осуществляется внесение изменений в подобные данные, как именно осуществляется их трансформация [9, с.30].

В таких системах, которые выстроены в соответствии с объектно-ориентированной логикой, объекты – это главные составляющие, интегрированные в систему. Фактически это «кирпичи», из которых состоит система. Каждый из подобных объектов характеризуется своим собственным состоянием, которое с течением времени может видоизменяться. Все объекты имеют отношение к тому или иному классу. Вхождение объекта в тот или иной класс зависит от того, какие именно свойства ему присущи. Если система является достаточно крупной, то несколько классов объединяются в иерархию.

Кратко изучим вопрос о том, на каких конкретных концепциях основывается функционирование объектно-ориентированных систем. Одна из самых важных среди вышеупомянутых концепций – это концепция наследования. Благодаря ей происходят переходы методов, а также свойств от одного класса к другому. Следующая концепция, на которую мы считаем необходимым обратить внимание – это концепция полиморфизма. Из-за того, что она существует, компоненты, относящиеся к отличающимся классам, могут вести себя одинаково. Третья концепция — это концепция инкапсуляции. Её

существование способствует интеграции информации в рамках одного и того же класса. Благодаря концепции инкапсуляции те сведения, которые должны сохраняться в режиме конфиденциальности, могут быть защищены от получения к ним несанкционированного доступа кем-либо. Последняя из тех концепций, которую мы считаем необходимым рассмотреть – это концепция абстракции. В связи с существованием концепции абстракции те данные, которые в своём изначально виде сложны для работы с ними неподготовленным человеком, упрощаются.

Благодаря тому, что в тех системах, которые являются объектно-ориентированными, объекты, а также классы могут эксплуатироваться повторно, с их применением можно разрабатывать интерфейсы открытого типа.

Среди тех достоинств, которые имеются у систем, относящихся к объектно-ориентированному типу, выделяются следующие. Во-первых, это высокий уровень гибкости, во-вторых, это высокий уровень масштабируемости, в-третьих, это высокий уровень безопасности, в-четвёртых, это наличие возможности несколько раз использовать один и тот же код, в-пятых, это относительно невысокая сложность реализации всех мероприятий, связанных с разработкой систем, а также с поддержкой их функционирования [10, с.214].

Следует отметить, что все рассмотренные нами типы обустройства систем имеют свои собственные преимущества, в связи с чем они пригодны для того, чтобы их можно было применять в тех или иных случаях. В каждом конкретном случае выбор в пользу того или иного архитектурного решения должен быть соответствующим образом обоснован.

2 Обзор и сравнение инструментов

2.1 Обзор существующих инструментов учета товара

Когда в компании, которая занимается работой с большим количеством продукции, используются соответствующие инструменты для её учёта, она приобретает возможность сделать всю работу с данными товарами более высокоэффективной.

Среди преимуществ, которые характерны для инструментов, применяемых с целью учёта продукции, мы можем обратить внимание на следующие.

Во-первых, они позволяют существенно нарастить степень автоматизации всех тех процессов, которые создаются в компании, чтобы она учитывала собственную продукцию, приобретаемую для дальнейшего использования, а также для дальнейшей реализации.

Во-вторых, они предоставляют возможность вывести на принципиально новый уровень качество принимаемых и реализуемых управленческих решений, непосредственно касающихся продукции.

В-третьих, благодаря им можно делать функционирование всего предприятия в большей степени производительным, эффективным.

В-четвёртых, благодаря их внедрению можно сокращать временные затраты на выстраивание коммуникационных процессов между разными структурными подразделениями одной и той же компании, каждое из которых вносит свой вклад в процесс движения товара, поступающего на предприятие.

В-пятых, автоматизированные учётные системы – это то, без чего немыслимо достижение высокого уровня клиентской удовлетворённости[15, с.163].

В таблице 2.1 нами представлены результаты краткого сопоставительного исследования, которое было нами проведено в отношении различных типов инструментов, применяемых на современных предприятиях с целью автоматизации учётных процессов с товарами.

Таблица 2.1 - Результаты краткого сопоставительного исследования, которое было нами проведено в отношении различных типов инструментов, применяемых на современных предприятиях с целью автоматизации учётных процессов с товарами.

Характеристики	POS-системы	ERP-системы	WMS-системы	SCM-системы	E-commerce платформы
Основное назначение	Учет продаж и обработка платежей в розничных магазинах.	Автоматизация и управление различными бизнес-процессами, включая учет товара, продаж, закупок и другие.	Управление процессами складского хранения, отгрузки товаров и инвентаризации.	Управление всей цепочкой поставок, включая учет товара на различных этапах поставки.	Управление электронной коммерцией, включая учет товара, заказов и инвентаризации.
Интеграция	Интеграция с другими системами, такими как учет, CRM и бухгалтерия.	Широкие возможности интеграции с другими системами в организации.	Требует интеграции с другими системами для полноценной работы.	Интеграция с другими системами для управления всей цепочкой поставок.	Интеграция различных функций управления электронной коммерцией.
Преимущества	Учет продаж и обработка платежей Интеграция с другими системами	Обширный функционал для автоматизации бизнес-процессов Широкие возможности интеграции	Управление процессами складского хранения и отгрузки товаров Фокус на инвентаризации	Управление всей цепочкой поставок Оптимизация процессов и управление запасами	Управление электронной коммерцией и учет товара
Недостатки	Могут быть неудобны для учета инвентаризации и работы на складе	Высокие затраты на внедрение и поддержку Могут быть сложными для малых предприятий	Могут быть перенасыщены функционалом для небольших предприятий Требует дополнительной интеграции с другими системами	Избыточны в функционале для организаций с менее сложными цепочками поставок	Могут быть менее подходящими для традиционной розничной торговли

Таким образом, у систем, которые относятся к разным группам, имеются свои собственные достоинства, свои собственные недостатки. Выбор в пользу систем соответствующей категории должен быть мотивирован, то есть он должен основываться, во-первых, на потребностях, имеющихся у бизнеса, во-вторых, на особенностях товаров, подлежащих учёту.

В таблице 2.2 нами показаны краткие описания самых распространённых типов систем, функционал которых заключается в том, чтобы осуществлять учёт товара, поступающего в компанию.

Таблица 2.2 - Краткие описания самых распространённых типов систем, функционал которых заключается в том, чтобы осуществлять учёт товара, поступающего в компанию.

Инструмент учета товара	Описание
POS-системы	Это программные комплексы, используемые в розничной торговле для учета продаж, инвентаризации и управления магазином. Они обеспечивают функции кассовой обработки и интегрируются с другими системами, такими как CRM и бухгалтерия.
ERP-системы	Enterprise Resource Planning (ERP) системы предназначены для автоматизации и управления различными бизнес-процессами, включая учет товара. Они обычно объединяют функции учета, закупок, производства, продаж и т. д.
WMS-системы	Warehouse Management System (WMS) системы предназначены для управления процессами складского хранения и отгрузки товаров. Они включают функции приемки и отгрузки, инвентаризации, планирования и оптимизации складских операций.
SCM-системы	Supply Chain Management (SCM) системы используются для управления всей цепочкой поставок, включая учет товара. Они включают в себя функции планирования спроса, управления запасами, совершенствования поставок и взаимодействия с поставщиками.
E-commerce платформы	Это коммерческие платформы для интернет-торговли, такие как Shopify, Magento, WooCommerce и другие, предоставляют возможности учета товара, заказов и инвентаризации. Они обычно включают функционал управления каталогом товаров, комплектование заказов, отслеживание запасов и другие инструменты, специализированные в области электронной коммерции.

Перейдём к более подробному изучению особенностей, которые присущи различным системам, функционал которых заключается в том, чтобы осуществлять учёт товара, поступающего в компанию.

POS-системы. POS-системы – это такие системы, которые используются в первую очередь в точках, обеспечивающих реализацию товара конечному клиенту. POS-системы – это специфические программные продукты, созданные для того, чтобы их можно было применять с целью фиксации данных о

транзакциях, происходящих в розничных точках реализации продукции предприятия. Современные POS-системы являются такими системами, функционал которых достаточно обширен. В частности, их можно применять для того, чтобы регистрировать факты продаж; для того, чтобы определять, какая выручка была получена благодаря продажам; для того, чтобы управлять инвентаризационными процессами на предприятии; для того, чтобы создавать чеки, подтверждающие факт покупки товара; для того, чтобы сохранять чеки, которые подтверждают факт состоявшейся покупки. POS-системы имеют достаточно широкие интеграционные возможности, благодаря чему можно налаживать обмен данными между ними и между бухгалтерскими системами, применяемыми в компании.

ERP-системы. ERP-системы – это комплексные системы, которые используются для того, чтобы автоматизировать функционирование всего предприятия в целом. ERP-системы пригодны для того, чтобы с их помощью можно было автоматизировать самые разные процессы, налаживаемые на предприятии. В первую очередь это касается процессов производства, процессов сбыта, процессов учёта денежных средств, а также процессов закупок продукции. Отличительная особенность, которая присуща ERP-системам, заключается в том, что они являются модульными. Их отдельные компоненты необходимы для того, чтобы осуществлять учёт товаров, поступающих на предприятие; для того, чтобы осуществлять менеджмент запасов, сформировавшихся на предприятии; для того, чтобы взаимодействовать с клиентами, которые обращаются в организацию.

WMS-системы. WMS-системы – это системы складского менеджмента. Они особенно актуальны для того бизнеса, которому необходимо хранить большое количество ресурсов, материалов, продукции на складах. WMS-системы – это инструменты для того, чтобы фиксировать факты поступления продукции на складскую инфраструктуру предприятия; того, чтобы фиксировать факты убытия продукции со складской инфраструктуры предприятия; для того, чтобы аккумулировать сведения обо всех перемещениях

продукции, ресурсов, материалов по объектам, относящимся к складской инфраструктуре фирмы. Когда в компании внедрена высококачественная и эффективная WSM-система, то обеспечивается сохранение полной информации о функционировании её складской инфраструктуры.

SCM-системы. SCM-системы – это такие системы, функционал которых заключается в том, чтобы осуществлять управление поставками на предприятии. Логика SCM-системы выстроена таким образом, что все поставки фирмы входят в состав соответствующих «цепочек». Эксплуатируя свою SCM-систему, компания может фиксировать данные о процессах поставок на предприятии, а значит, оно приобретает возможность планирования своей логистики, планирования спроса на свой товар. Кроме того, внедрение SCM-систем — это ещё и такой способ, задействуя который, компания может добиться увеличения эффективности взаимодействия с собственными поставщиками.

Платформы «e-commerce». Платформы «e-commerce» — это комплексные решения, которые могут быть использованы для того, чтобы создавать Интернет-магазины, а также для того, чтобы управлять их функционированием. Современные платформы «e-commerce» выстроены таким образом, чтобы их можно было применять как инструмент для управления каталогом продукции, поступающей в компанию; чтобы их можно было применять как инструмент для оформления заказов, поступивших в Интернт-магазин; чтобы их можно было применять как инструмент для проведения инвентаризационных процедур в Интернет-магазине; чтобы их можно было применять с целью осуществления оплаты товаров, передаваемых клиентам Интернет-магазина.

У всех тех инструментов, которые поименованы нами в таблице 2.2, имеются свои собственные преимущества, а также свои собственные недостатки. Выбор в пользу систем соответствующей категории должен быть мотивирован, то есть он должен основываться, во-первых, на потребностях, имеющихся у бизнеса, во-вторых, на особенностях товаров, подлежащих учёту [11, с.105].

2.2 SWOT анализ

SWOT-анализ является исследованием, которое проводится для того, чтобы выявить сильные и слабые стороны, присущие тому или иному объекту, процессу. Кроме того, SWOT-анализ – это такой инструмент, к которому можно прибегнуть тогда, когда требуется определить возможные негативные и позитивные последствия, возникающие из-за внешнего влияния на объект, процесс.

На рисунке 2.1 нами показаны итоги SWOT-анализа, который был осуществлён в отношении процесса складского учёта товаров, поступающих на предприятие.

Рисунок 2.1 –Итоги SWOT-анализа, который был осуществлён в отношении процесса складского учёта товаров, поступающих на предприятие

SWOT-анализ в современных реалиях является важным инструментом

абсолютно для любого предприятия. Прибегая к SWOT-анализу, фирма может выявить, что именно она делает качественно; что именно она делает недостаточно качественно; какие аспекты конкретного процесса требуют усовершенствования в первую очередь; какие возможности предоставлены в распоряжение фирмы для того, чтобы она могла развивать своё функционирование с течением времени. Таким образом, SWOT-анализ – это один из самых важных инструментов для того, чтобы в компании могла вестись деятельность в сфере стратегического планирования. У SWOT-анализа как у вида исследования имеется несколько преимуществ. Так, SWOT-анализ способен дать компании понять, какие плюсы, какие минусы имеются в её внутренней среде. Кроме того, SWOT-анализ способен дать компании понять, какие плюсы, какие минусы имеются в её внешней среде[13, с.95]. Применяя данные, которые были получены в рамках SWOT-анализа, фирма может сформировать и реализовать свою уникальную стратегию повышения уровня конкурентоспособности на рынке. Наконец, SWOT-анализ требуется также и для того, чтобы сформировать актуальный перечень угроз (рисков) для компании, получив, таким образом, в своё распоряжение всю необходимую информацию для того, чтобы обработать данные риски.

2.3 Обзор подходящих языков программирования

Когда принимается решение по вопросу о том, какой именно язык программирования будет использоваться, следует принимать во внимание многочисленные факторы. Во-первых, требуется учесть, к какому типу будет относиться создаваемая система. Во-вторых, требуется учесть, какие конкретные требования предъявляются по отношению к конечному результату проектной деятельности. В-третьих, необходимо учесть также и то, какие предпочтения высказываются разработчиками. В данной части настоящего исследования мы представим итоги сравнительного исследования, которое мы произвели в отношении самых популярных на сегодняшний момент времени

языков программирования.

Java: Java относится к числу тех языков программирования, которые имеют объектно-ориентированный характер. Java относится к числу тех языков программирования, которые можно применять на самых разнообразных платформах. Кроме того, Java относится к числу тех языков программирования, чьи синтактические особенности делают его одним из наиболее практичных и удобных при использовании специалистами с разным уровнем подготовки. Комментируя основные особенности, присущие Java, можно заострить внимание в том числе и на том, что у этого языка присутствует поддержка многопоточности. Наконец, объём той библиотеки, что присутствует у языка Java, среди всех остальных языков программирования является одним из самых крупных.

Второй язык программирования, который нами будет рассматриваться в рамках данной программы – это язык JavaScript. Количество уровней, которые имеются в данном языке программирования, таково, что его можно относить к числу высокоуровневых. Сфера применения соответствующего языка программирования – это создание страниц в Интернете, которые имеют высокий уровень интерактивности. Зафиксируем основные особенности, которые присущи JavaScript. Так, во-первых, JavaScript – это такой язык, который является интерпретируемым (соответственно, применяя его, можно корректировать содержимое страниц в Интернете, не инициируя процедуру компиляции). Кроме того, JavaScript, как и Java, является объектно-ориентированным языком программирования. Третья характеризующая JavaScript особенность – это наличие интеграции с CSS, а также cHTML. Проектировщиками JavaScript были предусмотрены все возможности для того, чтобы этот язык программирования был подготовлен к обработке разнообразных событий, появляющихся тогда, когда пользователь, находящийся на веб-странице, производит определённые целенаправленные действия (к примеру, кликает мышкой по кнопкам, а также отправляет разнообразные формы). Нельзя не обратить внимание на то, что JavaScript

полностью поддерживает асинхронное программирование, имеет все необходимые возможности для того, чтобы выполнение велось непосредственно в браузере клиента. Объём библиотеки JavaScript практически такой же значительный, как и у Java, что современными специалистами считается одним из самых важных преимуществ JavaScript.

Считаем необходимым более подробно остановиться на изучении вопроса о том, чем именно JavaScript отличается от такого распространённого языка программирования, как Java. Так, выбор в пользу Java чаще всего производится в тех случаях, когда это необходимо для того, чтобы создавались приложения серверного типа (что же касается JavaScript, то, как мы уже упомянули ранее, он проектировался в первую очередь для того, чтобы его можно было эксплуатировать с целью создания веб-страниц с разнообразными интерактивными пользовательскими решениями). Типизация языка Java – статическая, тогда как JavaScript отнесён к группе динамически типизированных. Выраженность типизации у языка Java достаточно существенная, тогда как у JavaScriptона, напротив, достаточно слабая. Особенности языка Java дозволяют рассматривать его в качестве инструмента, пригодного для того, чтобы с его помощью создавались приложения, занимающие большой объём памяти. В отличие от Java, JavaScript не может быть эффективно применён с целью создания таких приложений, которые занимают большой объём памяти. Таким образом, имеем возможность подытожить: несмотря на то, что у JavaScript и у Java имеются определённые схожие черты, сферы их практического использования не пересекаются.

Третий язык программирования, который нами будет изучаться в рамках соответствующей части работы – это язык C#. Среди всех тех языков программирования, которые сегодня известны, язык C# характеризуется как один из наиболее мощных. Он был предложен специалистами компании Microsoft, которые предусмотрели возможность его применения с целью разработки самых разнообразных приложений, отличающихся друг от друга, в частности, по сферам применения, а также по объёму. У языка C# имеется ряд

особенностей, которые являются характеризующими. Так, язык C# причисляется к группе объектно-ориентированных, что роднит его с уже изученными нами ранее JavaScript, а также Java. Вторая особенность, которая присуща C# — это многозадачность, то есть его можно использовать для того, чтобы создавать приложения, относящиеся к группе многопоточных. Создателями C# предусмотрено, что его можно применять на различных платформах, таким образом, мы причисляем его к группе платформенно независимых. Объём библиотеки у C#, как и у Java (JavaScript), значителен. Поскольку C# был создан специалистами Windows, у данного языка программирования имеется интеграция с различными программными решениями на основе Windows. Язык C# всегда характеризуется высокой степенью актуальности, поскольку разработавшей его компанией Windows перманентно выпускаются соответствующие обновления. А из-за того, что язык C# — это один из многочисленных компонентов общей экосистемы Windows, он продолжает оставаться практически безальтернативным при создании разных приложений и прочих решений к данной операционной системе.

Четвёртый язык программирования, который нами будет исследоваться в рамках данной работы – это язык PHP. PHP относится к числу так называемых «скриптовых» языков программирования. Соответственно, сфера, в которой он преимущественно применяется – это создание страниц в Интернете, которые имеют свойство динамичности, а также создание веб-приложений. Отметим основные особенности, которые присущи языку PHP. Во-первых, его можно интегрировать с HTML-страницами. Во-вторых, при проектировании языка PHP в него были заложены все необходимые возможности для того, чтобы с его помощью можно было управлять разнообразными базами данных. В-третьих, PHP является «многопоточным» языком. В-четвёртых, у PHP среди всех созданных языков программирования имеется одно из самых крупных комьюнити (а это означает, что при появлении тех или иных проблем, связанных с его использованием, можно получать ответы и рекомендации от представителей сообщества). Неоспоримое достоинство, которое присуще PHP,

заключается в том, что это полностью открытый язык программирования: так, его использование полностью бесплатно, вне зависимости от того, в каких целях планируется его применять. Наконец, PHP хорошо интегрируется с разными технологиями, а также с разнообразным серверным оборудованием. Все вышеперечисленные особенности, которые имеются у PHP, предопределяют, что уровень его распространения продолжает быть высоким[12, с.141].

Пятый язык программирования, который нами рассматривается в рамках настоящей работы – это язык Python. Среди всех известных на сегодняшний день языков программирования язык Python характеризуется как один из самых мощных. Следующая его отличительная черта – это наличие большого количества уровней. Соответственно, применять Python можно в целях решения самых разнообразных прикладных задач, от анализа данных до создания веб-страниц. Охарактеризуем основные особенности, которые присущи Python (и которые предопределяют его значительную степень распространения сегодня). Так, синтаксис, который имеется у Python, хорошо читаем, а также понятен даже неподготовленному пользователю. Следующая особенность, которая имеется у Python, заключается в том, что его можно применять в работе с самыми разнообразными платформами, поскольку он одинаково хорошо адаптирован к нескольким платформам. Как и у PHP, у Python имеется крупное комьюнити: этот язык программирования продолжает применяться в различных странах мира, и самые разные члены сообщества публикуют на форумах и прочих тематических ресурсах сведения о тех проблемах, с которыми они были вынуждены сталкиваться; сведения о тех решениях, которыми они воспользовались, чтобы преодолеть появившиеся проблемы. А ещё разработчиками Python было предусмотрено создание крупных и многочисленных библиотек, которые можно применять для того, чтобы решались самые разнообразные задачи программирования. В сегодняшних условиях популяризации машинного обучения актуальность Python не снижается, поскольку у этого языка программирования имеются все

необходимые возможности для того, чтобы его можно было использовать как инструмент для машинного обучения. Если появляется необходимость интеграции между тем программным продуктом, который был создан с применением Python, а также между тем программным продуктом, что был создан с применением иного языка программирования, то значимых сложностей и проблем не возникает, что можно считать ещё одним важным преимуществом, присущим Python. Самые распространённые сферы, в которых Python продолжает использоваться чаще, чем остальные языки программирования – это сфера анализа данных; сфера создания продуктов для машинного обучения; сфера автоматизации различных аспектов деятельности предприятий. Кроме того, именно к Python чаще всего считают возможным прибегать создатели разнообразных игровых приложений, в том числе и таких, которые являются достаточно крупными [18, с.88].

3 Разработка приложения

3.1 Разработка базы данных

С точки зрения SQL, «база данных» представляет собой совокупность данных, у которой имеется свойство структурированности. Вся та информация, которая относится к одной и той же базе данных, сведена в таблицы. Между таблицами налажено взаимодействие, которое реализуется посредством составления соответствующих запросов на языке SQL[28]. Таким образом, мы имеем возможность говорить о том, что таблица – это основной компонент базы данных в SQL. Для того, чтобы отличить отдельно взятую таблицу от всех остальных, для неё необходимо предусмотреть наименование. Кроме того, у каждой подобной таблицы также присутствует и своя собственная структура, пользуясь которой, её можно отличить от всех остальных. Как показано на рисунке 3.1, компонентами всех таблиц в SQL являются столбцы, а также ряды.

Существует ряд операций, прибегая к которым, можно осуществлять работу с информацией, находящейся на хранении в базе данных SQL. Рассмотрим данные операции более подробным образом:

- операция «создание таблицы». Сфера использования данной операции – это генерация новой таблицы, а также определение структуры, что имеется у данной таблицы;

- операция «вставка данных». Сфера использования данной операции – это расширение ранее созданной таблицы (посредством добавления в неё тех строк, которых в ней ранее не имелось);

- операция «актуализация данных». Сфера использования данной операции – это обновление той информации, что ранее появилась в той или иной таблице;

- операция «удаление данных». Сфера использования данной операции – это сокращение ранее созданной таблицы (посредством удаления из неё той или иной информации, которая в ней ранее появилась);

- операция «выборка данных». Сфера использования данной операции –

это подготовка выборок, которые будут содержать данные, вычлененные из таблицы по определённому правилу, по определённым признакам;

- операция «интеграция данных». Сфера использования данной операции – это соединение данных, изначально находящихся в разных таблицах, в одной и той же таблице.

Таким образом, имеем возможность подытожить, что SQL –это инструмент, предоставляющий возможность проводить разнообразные операции с информацией, хранящейся в базе данных. Прибегая к применению SQL, можно осуществлять работу с данными, прибегая к различным запросам.

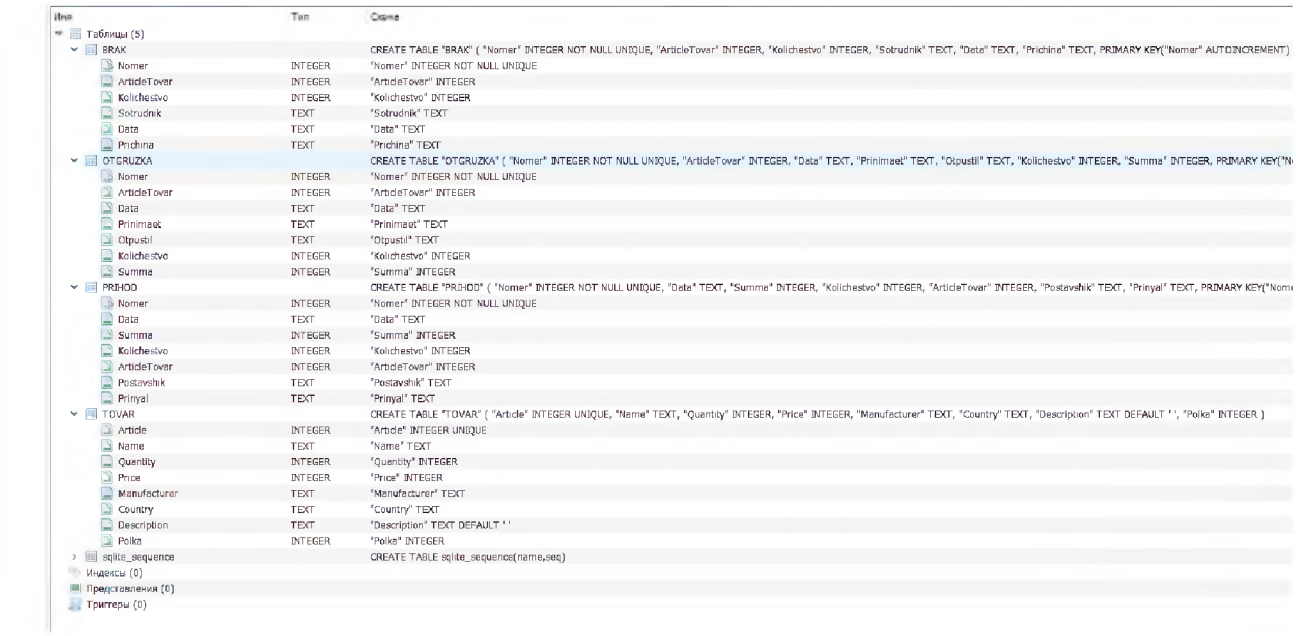


Рисунок 3.1– Внешний вид базы данных в SQL

Таблица: BRAK

	Nomer	ArticleTovar	Kolichestvo	Sotrudnik	Data	Prichina
	Филь...	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр
1	1	3	10	Вася	2022-02-17	Вскрыта упаковка
2	2	1	15	Лёша	2023-01-01	срок годности
3	3	1042	300	Дима	2022-11-01	срок годности
4	4	1029	10	Гриша	2022-01-01	упаковка
5	5	1042	20	Никита	2011-03-03	брак
6	6	1057	198	Никита	2022-01-01	упаковка
7	7	1042	1	Никита	2023-01-01	срок

Рисунок 3.2– Таблица «Брак»

На рисунке 3.2 нами показана таблица, которая имеет наименование «BRAK». Как становится ясно из её названия, она используется для того, чтобы сохранять данные о бракованной продукции, имеющейся у компании. Поля, вносимые в структуру вышеупомянутой таблицы, являются следующими:

- «Nomer»(в этом поле сохраняются данные об уникальном порядковом номере, который присвоен той или иной записи);

«ArticleTovar»(в этом поле сохраняются данные об артикуле, который имеется у бракованного товара);

«Kolichestvo»(в этом поле сохраняются данные об объёме товара, у которого был выявлен брак);

«Sotrudnik»(в этом поле сохраняются данные о работнике предприятия, которым был выявлен брак);

«Data»(в этом поле сохраняются данные о дате, а также о времени, когда был выявлен брак);

«Prichine»(в этом поле сохраняются данные о возможных причинах, которыми было обусловлено появление брака);

«Nomer»(данное поле используется для того, чтобы каждая запись, имеющаяся в таблице, была уникальной, то есть не могла дублироваться с какой-либо иной записью);

Таблица: OTGRUZKA

	Nomer	ArticleTovar	Data	Prinimaet	Otpustil	Kolichestvo	Summa
	Филь...	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр
1	1	2	2022-03-02	Пятёрочка	Ваня	1	120
2	6	1000	2023-12-10	Лента	никита	50	50
3	4	1015	2001-10-19	Магнит	Никита	45	5400
4	9	1015	2001-01-01	Лента	Максим	10	1200
5	7	1042	2023-12-10	Перекресток	Юрий	2	300
6	8	1042	2011-11-25	Теремок	Писцов	1	160
7	5	1057	2022-01-01	Пятёрочка	Олег	2	630

Рисунок 3.3–Таблица «Отгрузка»

На рисунке 3.3 нами показана таблица, которая имеет наименование «OTGRUZKA». Как становится ясно из её названия, она используется для того, чтобы сохранять данные о продукции, что была отгружена предприятием своим клиентам в течение того или иного временного промежутка. Поля, вносимые в структуру вышеупомянутой таблицы, являются следующими:

«Nomer»(в этом поле сохраняются данные об уникальном порядковом номере, который присвоен тому или иному товару, отгруженному предприятием клиенту);

«ArticleTovar»(в этом поле сохраняются данные о номере статьи, которой соответствует товар, отгруженный предприятием клиенту);

«Data» (в этом поле сохраняются данные о том, когда именно была произведена отгрузка продукции, выкупленной клиентом);

«Prinimaet»(в этом поле сохраняются данные о том, кому именно будет отправлен груз);

«Otpustil»(в этом поле сохраняются данные о том, кем именно был отправлен груз);

«Kolichestvo»(в этом поле сохраняются данные о том, сколько груза было отправлено);

«Summa» (в этом поле сохраняются данные о том, какой стоимостью характеризуется отправленный груз) [27].

Таблица: PRIHOD

	Nomer	Data	Summa	Kolichestvo	ArticleTovar	Postavshik	Prinyal
	Филь...	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр
1	1	2023-11-31	300	2	1059	ООО Русь	Андрей
2	2	2022-10-12	6000	40	1042	Склад Пром	Вася
3	3	2019-02-02	2000	10	1029	Мармелад	Андрей
4	4	2023-12-10	40000	200	1029	Мармелад	Гриша
5	6	2011-01-01	150	1	1042	ООО Мармелад	Лёша
6	7	2003-01-01	150	5	1042	ООО Мармелад	Ваня
7	8	2022-03-01	8000	50	1042	ООО Конфеты	Ваня
8	9	2023-12-26	26400	120	1059	Склад Пром	Вася Пупов

Рисунок 3.4–Таблица «Приход»

На рисунке 3.4 нами показана таблица, которая имеет наименование «PRIKHOD». Как становится ясно из её названия, она используется для того, чтобы сохранять данные о продукции, что поступила на предприятие в течение того или иного временного промежутка.

Поля, вносимые в структуру вышеупомянутой таблицы, являются следующими:

«Nomer»(данный столбец используется для того, чтобы сделать каждую запись уникальной);

«Data»(данный столбец используется для того, чтобы зафиксировать с его помощью сведения, характеризующие дату поступления товара в компанию);

«Summa»(данный столбец используется для того, чтобы зафиксировать с его помощью сведения, характеризующие стоимость закупленного компанией товара);

«Kolichestvo»(данный столбец используется для того, чтобы зафиксировать с его помощью сведения, характеризующие объём закупленного компанией товара);

«ArticleTovar»(данный столбец используется для того, чтобы зафиксировать с его помощью сведения, характеризующие идентификатор закупленного компанией товара);

«Postavschik»(данный столбец используется для того, чтобы зафиксировать с его помощью сведения, характеризующие поставщика закупленного компанией товара);

«Prinyal»(данный столбец используется для того, чтобы зафиксировать с его помощью сведения, характеризующие того, кто произвёл приёмку закупленного компанией товара);

На рисунке 3.5 нами показана таблица, которая имеет наименование «TOVAR».

Как становится ясно из её названия, она используется для того, чтобы сохранять данные о продукции, что хранится на предприятии в течение того или иного временного промежутка.

Таблица: TOVAR

Filter in any column

	Article	Name	Quantity	Price	Manufacturer	Country	Description	Polka
	Фильтр	Фильтр	Фильтр	Фи...	Фильтр	Фильтр	Фильтр	Фил...
1	1042	Мармелад Червячки зеленые	250	115	HARIBO	США	с ягодой	155
2	1015	Мармелад червячки красные	900	120	HARIBO	Россия		25
3	1029	Мармелад червячки желтые	400	200	HARIBO	США	Оригинальный вкус	23
4	1043	Мармелад червячки синие	2	150	МармеладПром	Египет		7
5	1055	Мишки клубничные 300гр	2000	500	МармеладПром	Германия	300гр	16
6	1057	Палочки мармеладные арбуз 200гр	300	315	МармеладПром	Китай	банан/черника	3
7	1000	Палочки мармеладные дыня 500гр	200	340	HARIBO	Китай	1	2
8	1044	Палочки мармеладные фруктовые 100гр	150	140	МармеладПром	Россия		1
9	1056	Подушечки мармеладные 50гр дыня	1000	150	HARIBO	Китай		6
10	1058	Подушечки мармеладные 150гр дыня	2000	400	МармеладПром	Россия		5
11	1059	Подушечки мармеладные 500гр банан	420	220	HARIBO	Россия		4
12	1050	Подушечки мармеладные 300гр черника	15	280	HARIBO	Россия		8
13	1051	Леденцы с вишней 50гр	123	130	МармеладПром	Китай		9
14	1014	Леденцы с вишней 30гр	114	312	МармеладПром	Китай		10
15	1018	Мармелад арбузный 1кг	46	156	HARIBO	Россия		11
16	1091	Мармелад фруктовый 1кг	321	311	МармеладПром	Китай		12
17	1095	Мармелад черничный 1кг	648	123	HARIBO	Китай		13
18	1082	Мармелад клубничный 2кг	967	333	МармеладПром	Германия		14
19	1073	Палочки мармеладные черника 150гр	345	111	МармеладПром	Китай		15
20	1074	Палочки мармеладные клубника 100гр	53	100	HARIBO	Германия		130

Рисунок 3.5 – таблица «Товар»

Поля, вносимые в структуру вышеупомянутой таблицы, являются следующими:

«Article»(поле, которое представляет собой уникальный идентификатор для товара, поступившего на склад предприятия);

«Name»(поле, которое представляет собой наименование товара, хранящегося на складе компании);

«Quantity»(поле, которое используется для того, чтобы сохранять данные о количестве товара, переданного для его хранения на склад компании);

«Price»(поле, которое используется для того, чтобы сохранять данные о стоимости товара, переданного для его хранения на склад компании);

«Manufacturer»(поле, которое используется для того, чтобы сохранять данные о поставщике товара, переданного для его хранения на склад компании);

«Country»(поле, которое используется для того, чтобы сохранять данные о государстве происхождения товара, переданного для его хранения на склад компании);

«Description»(поле, которое используется для того, чтобы сохранять подробные данные с целью описания товара, переданном для его хранения на склад компании);

«Polka»(поле, которое используется для того, чтобы сохранять данные о полке, выделенной для размещения товара, переданного для его хранения на склад компании)

3.2 Разработка информационной системы

Функция authentification() (Приложение А)используется для проверки логина и пароля пользователя, чтобы позволить или запретить доступ к основной программе.

Сначала она получает значения, введенные пользователем, из полей ввода логина и пароля. Затем происходит проверка этих значений на соответствие заранее заданным логину «admin» и паролю «123». Если логин и пароль совпадают с заданными значениями, то окно аутентификации закрывается, и функция main_program() вызывается для запуска основной программы. Если данные не соответствуют ожидаемым значениям, то появляется всплывающее сообщение с предупреждением об ошибке ввода [26].

Аутентификация нужна для обеспечения безопасности и защиты программы от несанкционированного доступа. Пользователь должен ввести правильный логин и пароль, чтобы получить доступ к основной программе.
Функция main_program() представляет собой главную программу приложения «Мармеладный склад». В этой функции мы создаем основное окно приложения, устанавливаем его заголовок и размер, а также делаем его окно полноэкранным (с помощью метода state('zoomed')). Затем мы устанавливаем

соединение с базой данных 'sklad.db' и создаем курсор для выполнения SQL-запросов.

Tk() - это класс из библиотеки tkinter, который позволяет создавать графические интерфейсы.

root.title(«Мармеладный склад») - устанавливает заголовок окна приложения.

root.geometry(«1300x600») - устанавливает размер окна приложения.

root.state('zoomed') - делает окно полноэкранным.

sql.connect('sklad.db') - устанавливает соединение с базой данных 'sklad.db'.

connection.cursor() - создает курсор для выполнения SQL-запросов.

Таким образом, функция main_program() выполняет все необходимые инициализационные шаги для запуска приложения «Мармеладный склад». Функция info_tovar() выполняет запрос к базе данных и извлекает информацию о товарах. Затем она вставляет эту информацию в виде строк в виджет tovar_tree.

Функция info_postavka() выполняет запрос к базе данных и извлекает информацию о поставках. Затем она вставляет эту информацию в виде строк в виджет postavka_tree.

Функция info_brak() выполняет запрос к базе данных и извлекает информацию о браке. Затем она вставляет эту информацию в виде строк в виджет brak_tree.

Функция info_otgruzka() выполняет запрос к базе данных и извлекает информацию о отгрузках. Затем она вставляет эту информацию в виде строк в виджет otgruzka_tree.

Описание функций указывает, что они извлекают информацию из базы данных и отображают ее в соответствующих виджетах. Например, функция info_tovar() извлекает информацию о товарах и отображает ее в виджете tovar_tree.

Функция refresh() (Приложение А) предназначена для обновления данных

в графическом пользовательском интерфейсе.

Функция refresh() содержит вызовы методов delete() для удаления всех элементов из таблиц (tovar_tree, postavka_tree, brak_tree, otgruzka_tree) и последующие вызовыфункций info_tovar(), info_postavka(), info_brak(), info_otgruzka(), чтобы добавить обновленные данные в таблицы.

Таким образом, после выполнения функции refresh(), таблицы tovar_tree, postavka_tree, brak_tree и otgruzka_tree будут обновлены новыми данными.

Функция find_tovar() (Приложение А) выполняет поиск товара в базе данных по введенному пользователем значению search_tovar.

Сначала функция очищает все таблицы из базы данных (tovar_tree, postavka_tree, brak_tree, otgruzka_tree) от предыдущих результатов поиска с помощью метода delete(*tovar_tree.get_children()).

Затем функция извлекает все строки из таблицы TOVAR и сохраняет их в переменную tovars с помощью команды SQL SELECT. Аналогично происходит извлечение данных из таблиц PRIHOD, BRAK, OTGRUZKA, которые сохраняются соответственно в переменные postavki, braki, otgruzki.

Далее происходит поиск товара в таблице TOVAR по всем полям строки. Если строка содержит введенное значение search_tovar, то она добавляется в таблицу tovar_tree с помощью метода insert('', END, values=tovar). Если значение search_tovar содержится в поле Name или Description, то она также добавляется в таблицу tovar_tree. Также проверяется, является ли search_tovar числом (is_digit = search_tovar.isdigit()), и если это так, то проверяется, присутствует ли число в строке. Если число присутствует, то оно также добавляется в таблицу tovar_tree.

То же самое происходит для таблиц PRIHOD, BRAK, OTGRUZKA.

Данные из таблиц добавляются в соответствующие виджеты postavka_tree, brak_tree, otgruzka_tree.

В конце функция вызывает функцию refresh(), которая обновляет все таблицы с исходными данными, если поле поиска осталось пустым.

Таким образом, функция find_tovar() выполняет поиск товаров в базе данных и отображает соответствующие результаты в таблицах виджетов [25].

Функция otchet()в данном коде формирует отчет о браке.

Она создает окно приложения с возможностью выбора категории, даты начала и даты окончания для фильтрации данных. При нажатии кнопки «Сформировать отчет» происходит выполнение функции otchet_go(), которая открывает новое окно, в котором отображается таблица с данными о браке.

В функции otchet_go() есть вложенная функция brack(), которая формирует окно для отображения отчета. Она создает таблицу с заданными столбцами и заполняет ее данными из базы данных. После этого происходит захват экрана данного окна и сохранение его в файл формата PNG.

После выполнения функции brack(), открывается окно с отчетом о браке, в котором пользователь может просмотреть и сохранить полученные данные.

Данный код (приложение А)описывает функцию otgruzochka(), которая отображает отчет об отгрузке товаров на экране и сохраняет его в файл.

Функция otgruzochka() использует библиотеку Tkinter для создания графического интерфейса и библиотеку PIL для работы с изображениями.

Вначале функция создает новое окно с заданными размерами и устанавливает его заголовок. Затем создается таблица tree_otgruzka с заданными столбцами ('Nomer', 'ArticleTovar', 'Data', 'Prinimaet', 'Otpustil', 'Kolichestvo', 'Summa').

Далее, функция выполняет запрос к базе данных для получения данных об отгрузке, которые находятся в диапазоне между двумя указанными датами (date_ot и date_do). Результаты запроса записываются в переменную dates1.

Затем функция отображает полученные данные в таблице tree_otgruzka с помощью метода insert().

После отображения данных функция вызывает функцию screen(), которая выполняет скриншот экрана и сохраняет его в файл otgruzka.jpeg.

Затем функция вызывает диалоговое окно сохранения файла, где пользователь может выбрать место сохранения и имя файла. Если пользователь указывает имя файла, то скриншот также сохраняется в выбранное место.

В результате выполнения функции otgruzochka(), на экране отображается таблица с данными об отгрузке товаров, а также создается скриншот экрана и сохраняется в файл. Это позволяет пользователям легко получать отчеты об отгрузке и сохранять их для последующего использования.

Код postavochka() (приложение А) представляет собой функцию для создания и отображения таблицы с данными о поставках. Здесь используется библиотека Tkinter для создания графического интерфейса пользователя (GUI), а именно окна, в котором будет отображаться таблица.

Функция screen() используется для создания скриншота текущего состояния окна и сохранения его в файл с расширением .jpeg. После этого пользователю предлагается выбрать файловое имя и сохранить скриншот в формате .png.

Затем создается окно с заголовком «Поставка | Отчёт» и задаются столбцы для отображения данных в таблице. В этом случае у таблицы есть столбцы «Номер», «Дата», «Сумма», «Количество», «Артикул», «Поставщик» и «Принял».

Затем выполнится запрос в базу данных SQLite для выборки данных о поставках, в которых дата попадает в заданный диапазон между date_ot и date_do. Полученные данные будут добавляться в таблицу.

Наконец, таблица будет упакована и отображена в окне GUI. Затем будет вызвана функция screen() для создания скриншота окна с таблицей и сохранения его в файл.

Данный код (Приложение А)представляет реализацию функционала по принятию поставки товара. Он создает графическое окно с формой, где пользователь может ввести необходимую информацию о поставке, включая артикул товара, дату, поставщика, ответственного за приемку товара и количество принимаемого товара [23].

После нажатия кнопки «Принять», код проверяет, что все поля формы заполнены. Если хотя бы одно поле пустое, выводится предупреждение. Затем код проверяет, существует ли товар с указанным артикулом в базе данных. Если товар не найден, выводится предупреждение об ошибке. Далее код извлекает текущее количество и цену товара из базы данных и рассчитывает общую сумму поставки. Затем код проверяет, что введенная дата соответствует формату «ГГГГ-ММ-ДД». Если введенная дата не соответствует формату, выводится предупреждение. Если все данные корректны, код обновляет количество товара в базе данных, добавляет запись о поставке в таблицу «PRIHOD» и сохраняет изменения в базе данных. Кроме того, информация о поставке записывается в файл журнала.

В целом, данный код предоставляет пользователю возможность принять поставку товара, заполнив соответствующую форму, а затем обновляет базу данных и журнал событий при успешной обработке поставки.

Функция otgruzit() (Приложение А)представляет собой окно для отгрузки товара. Она создает графический интерфейс с использованием библиотеки tkinter и содержит поле для ввода артикула товара, даты отгрузки, получателя, сотрудника, который выполнил отгрузку, а также количество отгружаемого товара.

При нажатии кнопки «Отгрузить» вызывается функция sdelat_otgruzku(), которая извлекает значения из полей ввода и выполняет следующие действия:

Проверяет, заполнены ли все поля. Если нет, выводится предупреждение о необходимости заполнить все поля.

Извлекает все артикулы товаров из базы данных и проверяет, существует ли введенный артикул.

Если артикул не найден, выводится предупреждение о том, что товар не найден.

Если артикул найден, получает текущее количество и цену товара из базы данных.

Вычисляет общую сумму отгрузки на основе цены и количества товара.

Проверяет, соответствует ли введенная дата формату «ГГГГ-ММ-ДД». Если нет, выводится предупреждение о необходимости ввести корректную дату.

Если дата введена в правильном формате и количество товара достаточно, обновляет количество товара в базе данных, вставляет новую запись о отгрузке в таблицу «OTGRUZKA» и закрывает окно отгрузки.

Записывает дату и время отгрузки, а также информацию о выполненной операции в файл журнала.

Функция vrychnyy() отвечает за внесение нового товара в базу данных вручную. Она создает окно window_vnesti, где пользователь может ввести данные о новом товаре, такие как артикул, название, количество, цена, производитель, страна, описание и полка.

В целом, функция ручного ввода предоставляет пользователям больше контроля и возможностей взаимодействия с приложением[20].

При нажатии на кнопку «Внести», функция vnesti_vrychnyy() получает значения, введенные пользователем, и выполняет следующие действия:

1) Проверяет, заполнены ли все поля. Если хотя бы одно поле не заполнено, выводится предупреждение.

2) Проверяет, существует ли товар с таким же артикулом. Если товар с таким артикулом уже существует, выводится предупреждение.

Если все проверки прошли успешно, данные о новом товаре добавляются в базу данных.

Окно window_vnesti закрывается.

Записывается информация о добавлении товара в лог-файл logs.txt.

Таким образом, функция vrychnyy() предоставляет пользователю возможность внести новый товар в базу данных вручную, обеспечивая проверку заполнения полей и предотвращение дублирования товаров.

Данный код (Приложение А) представляет собой функцию «peremestit()», которая реализует окно для перемещения товара. В окне пользователю предлагается ввести артикул товара и номер полки, на которую нужно

переместить данный товар. После нажатия на кнопку «Переместить» происходит обновление записи в базе данных, где устанавливается новое значение полки для указанного артикула товара. При успешном выполнении операции происходит запись информации о перемещении товара в лог-файл.

1) Функция «peremestit()» объявляется.

2) Внутри функции определяется еще одна вложенная функция «peremeshenie()», которая будет вызываться при нажатии на кнопку «Переместить». Эта функция считывает введенные значения артикула товара и номера полки.

3) Выполняется SQL-запрос для получения всех артикулов товаров из базы данных.

4)Проверяется, заполнены ли оба поля ввода. Если хотя бы одно поле не заполнено, то выводится предупреждающее сообщение.

5) Если оба поля заполнены, то происходит проверка наличия введенного артикула в базе данных. Если артикул найден, то происходит обновление записи в базе данных, устанавливая новое значение полки для указанного артикула.

6) Если успешно выполнено обновление, то окно перемещения товара закрывается. Также происходит запись информации о перемещении товара в лог-файл, где указывается дата и время операции, а также SQL-запрос, осуществленный для перемещения товара.

7) Если артикул товара не найден, то выводится предупреждающее сообщение.

8) Функция «peremestit()» создает окно с элементами для ввода артикула и номера полки, а также кнопкой «Переместить». При нажатии на кнопку вызывается вложенная функция «peremeshenie()».

9) Запускается основной цикл обработки событий окна.

Функция tsena()отвечает за изменение цены товара в базе данных. Она открывает новое окно с формой, содержащей поля для ввода артикула товара и новой цены. Когда пользователь нажимает кнопку «Поменять», функция

peremeshenie() вызывается. Функция проверяет, заполнены ли оба поля, и если нет, выводит сообщение об ошибке. Если поля заполнены, функция выполняет SQL-запрос к базе данных для обновления цены товара с заданным артикулом. При успешном обновлении цены, окно закрывается и добавляется запись в файл журнала изменений.

В данном коде (Приложение А) создается интерфейс с использованием виджетаTreeview в библиотеке tkinter. ВиджетTreeview позволяет отображать данные в виде таблицы с возможностью сортировки и выбора строк.

В каждом из четырех блоков кода создается отдельный экземпляр Treeview. Для каждой таблицы задаются столбцы и их заголовки с помощью кортежей columns_tovar, columns_postavka, columns_brak и columns_otgruzka.

Затем каждый экземпляр Treeview привязывается к соответствующему фрейму на графическом интерфейсе с помощью метода pack. Фреймы tovar_frame, postavka_frame, brak_frame и otgruzka_frame используются для размещения таблиц на графическом интерфейсе[19].

Для каждого Treeview задаются заголовки столбцов с помощью метода heading и их ширина с помощью метода column. Последующие параметры методов heading и column определяют текст заголовка и ширину столбца соответственно. ВиджетыTreeview привязываются к указанным столбцам и настраиваются для растягивания с помощью атрибута stretch=YES.

Приведенный код (Приложение А)создает графическое окно авторизации (authentication_window) с использованием библиотеки Tkinter. Окно имеетразмер 220x140 пикселей и название «Авторизация».

В окне есть метки для полей ввода логина и пароля. Метка «Логин» расположена в столбце 0 и строке 0, а поле ввода логина расположено в столбце 1 и строке 0. Аналогичным образом, метка «Пароль» располагается в столбце 0 и строке 1, а поле ввода пароля - в столбце 1 и строке 1.

Затем создается кнопка «Войти» с помощью элемента ttk.Button и связывается с функцией authentification. Кнопка занимает 4 столбца, начиная с 0 и заканчивая 3, а также находится в строке 2.

После этого происходит запуск главного цикла окна (mainloop()), который позволяет пользователю взаимодействовать с окном[21].

3.3Разработка пользовательского интерфейса.

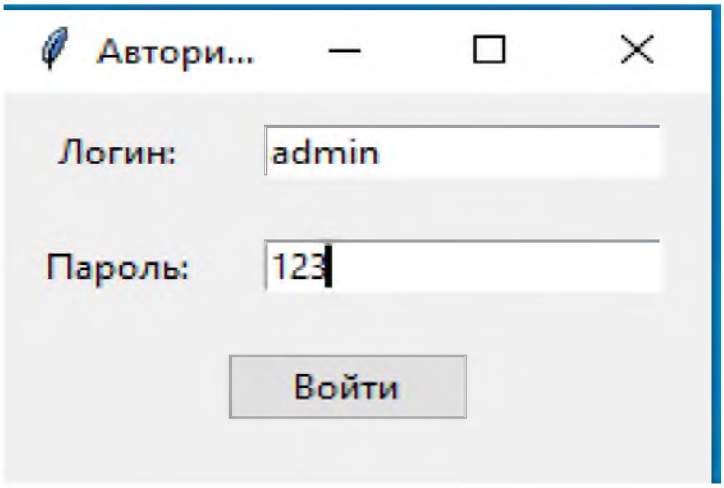


Рисунок 3.6– Графическое окно «Окно авторизации»

На рисунке 3.6 нами продемонстрировано, как именно выглядит всплывающее окно, необходимое для окончания процедуры авторизации. Когда перед пользователем появляется данное окно, он может, используя его, внести в систему данные о том, каковы его логин, а также пароль. Благодаря соответствующему окну появляется возможность предоставления пользователю в работу тех или иных систем, модулей. Благодаря соответствующему окну появляется возможность сохранения безопасности тех данных, которые являются конфиденциальными, в связи с чем их требуется соответствующим образом охранять. Логин, который присваивается каждому пользователю, прошедшему процедуры, связанные с его регистрацией в системе – это не что иное, как инструмент для его идентификации. Что же касается пароля, то он является уникальной и известной только самому пользователю комбинацией символов, без ввода которой в систему осуществить вход в учётную запись будет невозможно[22]. Как только пользователем произведено нажатие кнопки «Найти», осуществляется проверка тех данных, что были введены. В том случае, если они являются корректными, пользователю предоставляется

возможность входа в личный кабинет, зарегистрированный на его имя. В том случае, если они являются некорректными, пользователю не предоставляется возможность входа в личный кабинет, зарегистрированный на его имя.

На рисунке 3.7 нами показано, как именно выглядит окно с наименованием «Главное меню».

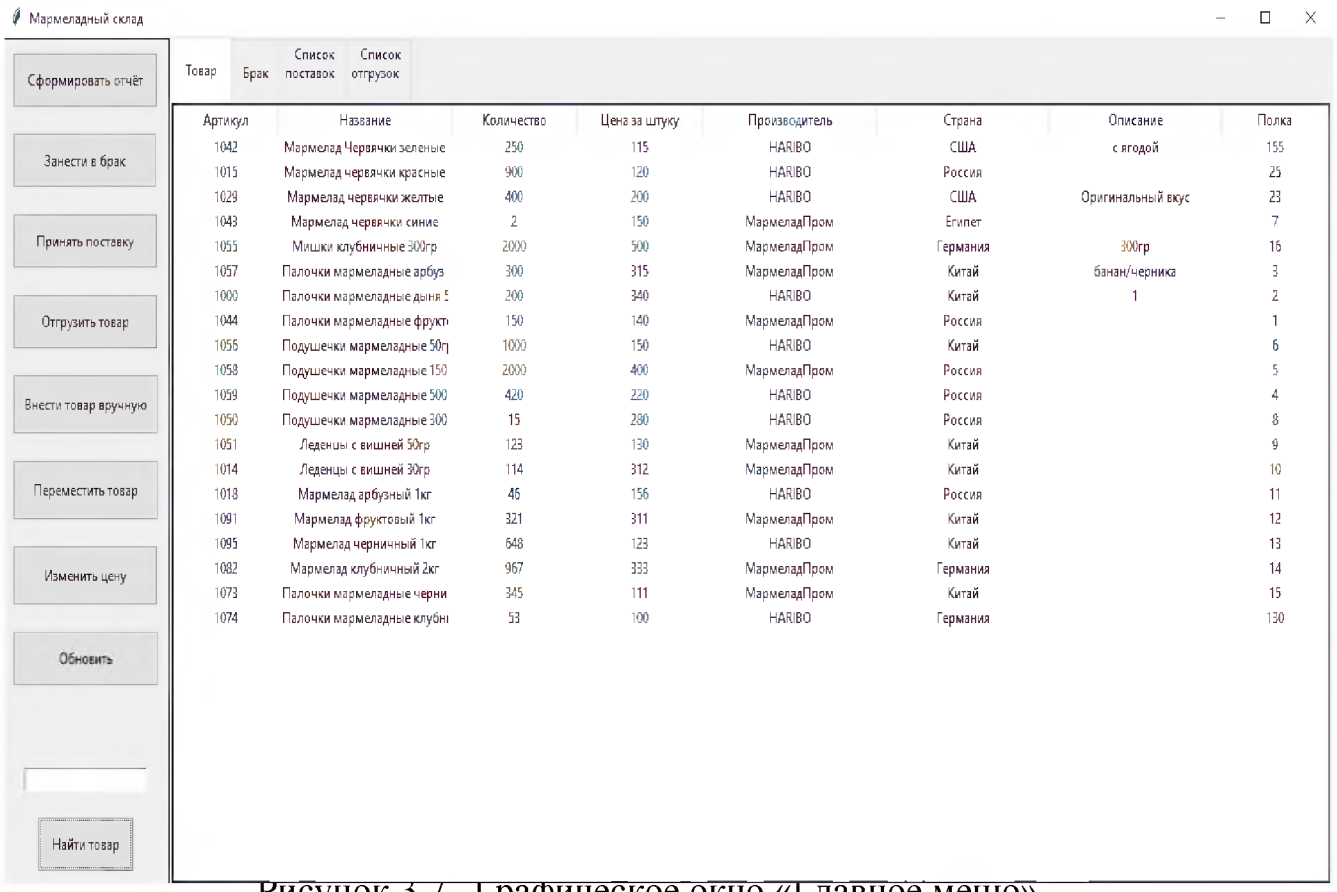


Рисунок 3.7 – Графическое окно «Главное меню»

Фактически на рисунке 3.7 мы показали, как именно выглядит интерфейс программного решения, разработанного для того, чтобы в компании вёлся постоянный учёт продукции, переданной для её хранения на объекты складской инфраструктуры. Данный интерфейс характеризуется наличием многочисленных кнопок и клавиш, у каждой из которых имеется своя собственная функция. Так, применяя клавишу «Формирование отчёта», пользователь может выгрузить интересующий его отчёт, содержащий массив данных, описывающих хранение продукции, материалов на складе. Используя клавишу «Занесение в брак», пользователь может присвоить той или иной продукции (тому или иному материалу) статус бракованного. Применяя

клавишу «Принять поставку», пользователь может зафиксировать результат поставки материалов, продукции на склад предприятия. Используя кнопку «Отгрузка товара», пользователь может зафиксировать результат отгрузки продукции, которая состоялась со склада фирмы. Применяя кнопку «Ручное добавление товара», пользователь может самостоятельно внести в систему сведения о товаре, которые по каким-либо причинам не были ранее добавлены в автоматическом режиме. Пользуя кнопкой «Перемещение товара», пользователь может изменить данные о местоположении того или иного товара, что ранее были введены в систему. Прибегая к кнопке «Корректировка цены», пользователь может изменить данные о цене того или иного товара, что ранее были введены в систему. Клавиша «Обновление» необходима для того, чтобы на устройстве пользователя осуществилась актуализация данных. Если пользователю требуется произвести информационный поиск по системе, он может воспользоваться такой кнопкой, как «Поиск информации».

У таблицы, которая используется для того, чтобы зафиксировать результат складского учёта на предприятии, имеются столбцы, обладающие уникальными наименованиями. Такие наименования – это «Количество», «Название», «Стоимость», «Государство происхождения», «Описание», «Номер полки», а также некоторые иные. При появлении соответствующей необходимости номенклатура столбцов в таблице может быть подвергнута пересмотру.

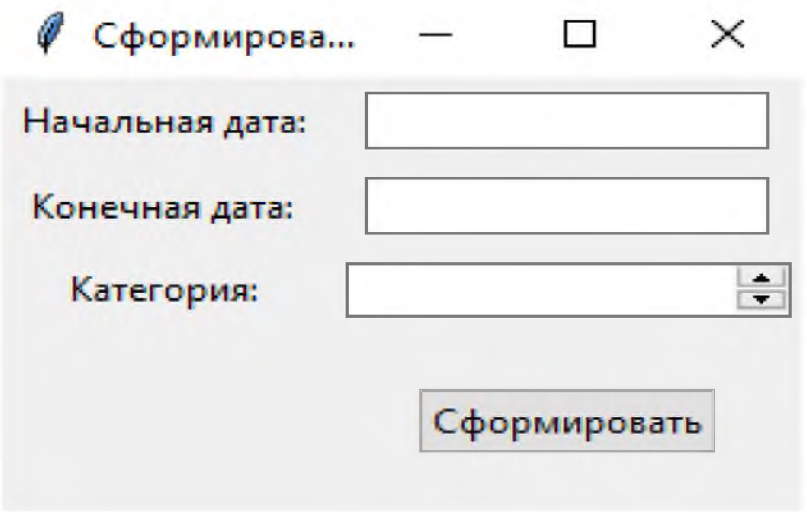


Рисунок 3.8– Графическое окно «Сформировать отчёт»

На рисунке 3.8 нами показано, как выглядит окно, выводимое перед пользователем тогда, когда ему требуется сформировать ту или иную отчётность. В данном окне имеется три основных поля. Первое поле используется для того, чтобы зафиксировать дату, начиная с которой, в отчётность будут вноситься данные. Второе поле используется для того, чтобы зафиксировать конечную дату (то есть ту дату, после которой сведения уже не будут вноситься в отчётность) [29].

Наименование третьего поля – это «Категория». Когда пользователь начинает работать с соответствующим полем, система выдаёт ему список из трёх отдельных граф. Они именуются как «Брак», «Отгрузка», а также «Поставка». Таким образом, пользователю даётся возможность самостоятельно выбрать графы, сведения о которых будут сведены в формируемый отчёт. Только после того, как пользователь внёс все соответствующие данные в каждую из вышеперечисленных граф, система предоставит ему возможность формирования отчёта. Вариантов вывода отчёта имеется два. Во-первых, он может быть показан непосредственно на экране, во-вторых, он может быть выгружен в отдельно взятый файл [24]. Обобщая вышесказанное, мы можем отметить, что функция формирования отчётности – это удобный инструмент для генерации разнообразных отчётов, отличающихся друг от друга по периодам данных, а также по видам сохраняемой информации.

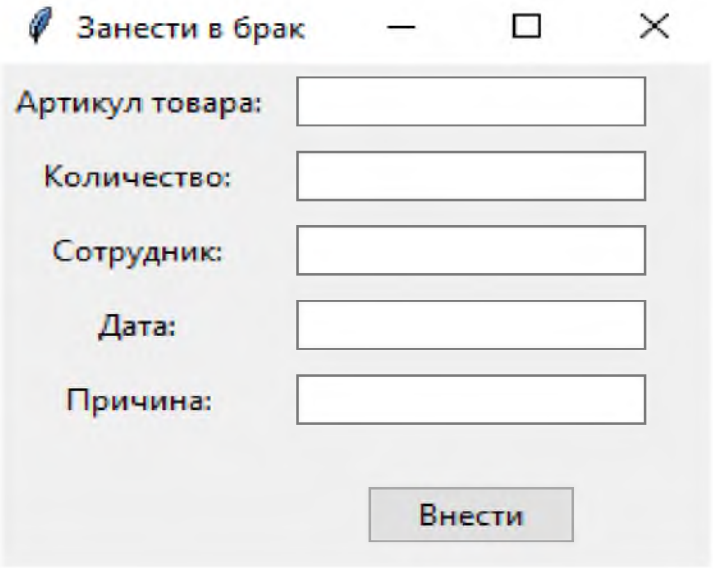


Рисунок 3.9- Графическое окно «Занести в брак»

На рисунке 3.9 нами показано, как выглядит окно, применяемое для того, чтобы актуализировать в системе информацию о бракованной продукции. Поля, которые имеются в данном окне, являются следующими:

Поле «Артикул»(необходимо для того, чтобы зафиксировать сведения об артикуле продукции, у которой был выявлен брак);

Поле«Количество»(необходимо для того, чтобы зафиксировать сведения об объёме продукции, в чьём отношении принято решение о её признании бракованной);

Поле «Сотрудник (необходимо для того, чтобы зафиксировать данные о работнике предприятия, принявшем решение о том, чтобы отнести ту или иную продукцию к категории бракованной);

Поле «Дата» (необходимо для того, чтобы зафиксировать сведения о дате и о времени, когда было принято решение о присвоении той или иной продукции статуса бракованной).

Поле «Причина» (необходимо для того, чтобы зафиксировать сведения о причинах, способствовавших вынесению в отношении продукции решения об её отнесении к бракованной).

После того, как произведено заполнение всех перечисленных выше полей, пользователю требуется кликнуть по кнопке «Внести». После клика по данной кнопке сведения будут обновлены в системе. Рассмотренное выше графическое окно позволяет всем, кому дана возможность пользоваться программой, актуализировать хранящиеся в ней сведения о том, какой именно продукции был присвоен статус бракованной. Удобство этой формы подтверждается данными о практической эксплуатации программы. Так, работникам предприятия, прибегающим к использованию соответствующей формы, требуется минимальное количество времени, чтобы обновить информацию в системе о продукции, в чьём отношении было принято решение о её признании бракованной.

На рисунке 3.10 нами показано, как выглядит окно, применяемое для того, чтобы актуализировать в системе информацию о товарах, поступивших на

предприятие.

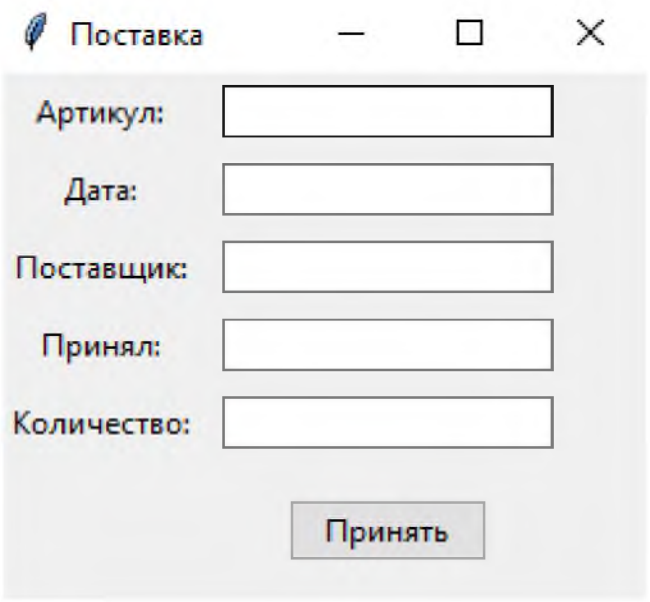


Рисунок 3.10-Графическое окно «Поставка»

Поля, которые имеются в данном окне, являются следующими:

Поле «Артикул»(необходимо для того, чтобы отразить в информационной системе данные о том, каков артикул поступившей в компанию продукции);

Поле «Дата»(необходимо для того, чтобы отразить в информационной системе данные о том, когда именно предприятие произвело приёмку поступившего товара);

Поле «Поставщик»(необходимо для того, чтобы отразить в информационной системе данные о том, какой конкретный поставщик передал товары компании);

Поле «Сотрудник»(необходимо для того, чтобы отразить в информационной системе данные о том, кто именно из уполномоченных работников предприятия произвёл приёмку поступившей продукции);

Поле «Количество»(необходимо для того, чтобы отразить в информационной системе данные о том, в каком именно объёме продукция была принята для её последующего хранения).

Как только каждое из вышеперечисленных полей заполнено, система предложит пользователю кликнуть по клавише «Принять». Сразу после того,

как это сделано, в информационную систему будут заедены актуализированные данные о состоявшейся поставке.

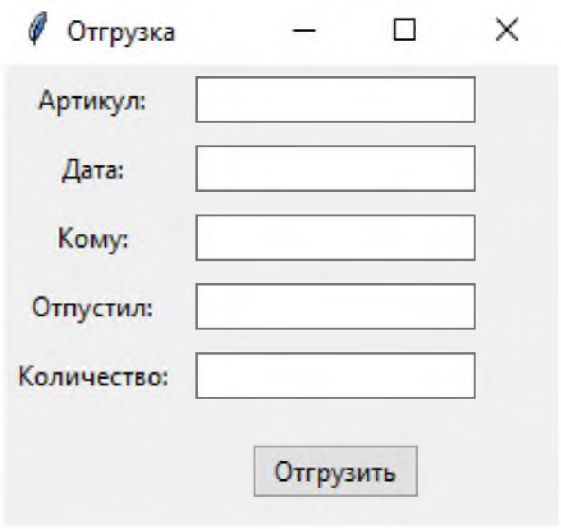


Рисунок 3.11–Графическое окно «Отгрузка»

На рисунке 3.11 нами показано, как выглядит окно, применяемое для того, чтобы актуализировать в системе информацию о товарах, отгруженных компанией. Поля, которые имеются в данном окне, являются следующими:

Поле «Артикул» (необходимо для того, чтобы отразить в информационной системе данные о том, каков артикул продукции, что была отгружена предприятием);

Поле «Дата» (необходимо для того, чтобы отразить в информационной системе данные о том, когда именно предприятие произвело отгрузку товара);

Поле «Кому» (необходимо для того, чтобы отразить в информационной системе данные о том, кому именно была осуществлена отгрузка товара, поставленного предприятием);

Поле «Сотрудник» (необходимо для того, чтобы отразить в информационной системе данные о том, кто именно из уполномоченных работников предприятия произвёл отгрузку продукции);

Поле «Количество» (необходимо для того, чтобы отразить в информационной системе данные о том, в каком именно объёме продукция была отгружена).

Как только каждое из вышеперечисленных полей заполнено, система предложит пользователю кликнуть по клавише «Отгрузить». Сразу после того, как это сделано, в информационную систему будут заедены актуализированные данные о состоявшейся отгрузке

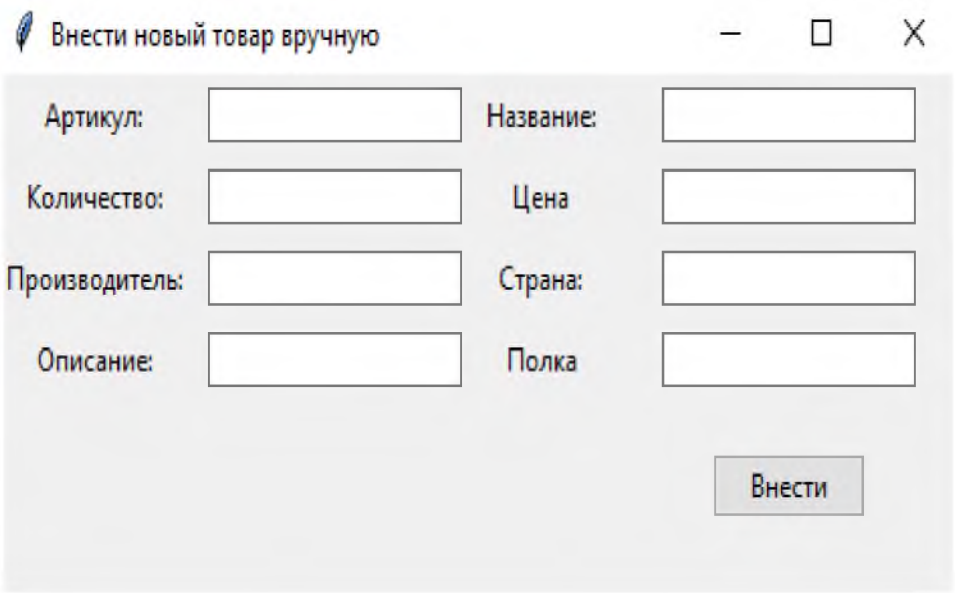


Рисунок 3.12-Графическое окно «Внести новый товар вручную»

На рисунке 3.12 нами показано, как выглядит окно, применяемое для того, чтобы актуализировать в системе информацию о товарах, находящихся на хранении на предприятии, которые имеются в данном окне, являются следующими:

Поле «Артикул» (необходимо для того, чтобы отразить в информационной системе данные о том, каков артикул продукции, что была зафиксирована в системе);

Поле «Количество» (предоставляет возможность отразить в информационной системе данные, характеризующие объём вручную вводимого товара);

Поле «Производитель» (предоставляет возможность отразить в информационной системе данные, характеризующие поставщика поступившего в компанию товара);

Поле «Описание» (предоставляет возможность отразить в информационной системе данные, характеризующие поступивший товар, в частности, с точки зрения его основного предназначения);

Поле «Цена» (предоставляет возможность отразить в информационной системе данные, характеризующие цену, по которой товар, закупленный организацией, был ею приобретён);

Поле «Государство происхождения» (предоставляет возможность отразить в информационной системе данные, характеризующие страну, откуда поступил товар);

Поле «Полка» (предоставляет возможность отразить в информационной системе данные, характеризующие номер полки, куда был помещён товар, поступивший в компанию)/

Как только каждое из вышеперечисленных полей заполнено, система предложит пользователю кликнуть по клавише «Внести». Сразу после того, как это сделано, в информационную систему будут заедены актуализированные данные о товаре, что был поставлен в компанию.

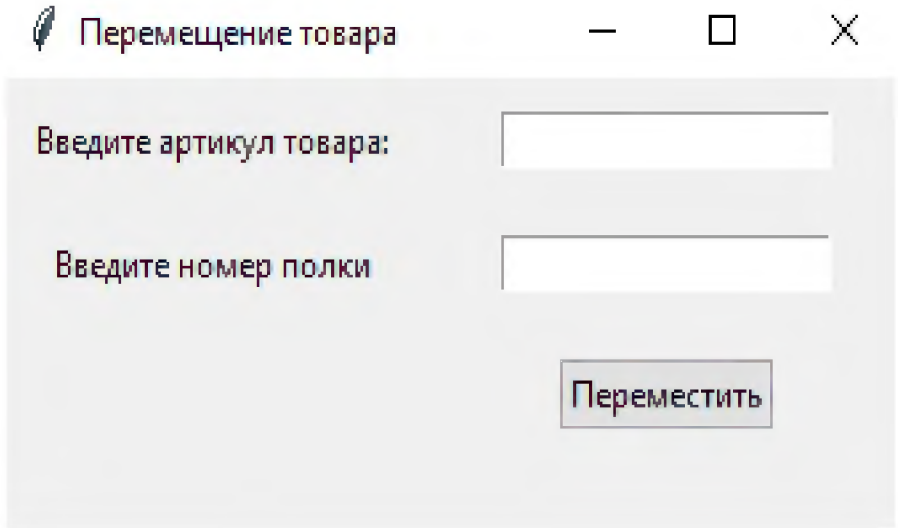


Рисунок 3.13-Графическое окно «Перемещение товара»

На рисунке 3.13 нами показано, как выглядит окно, применяемое для того, чтобы актуализировать в системе информацию о перемещённых товарах. В рассматриваемом окне присутствует несколько полей, в каждое из которых требуется внести соответствующие сведения. В частности, это такие сведения,

которые характеризуют артикул продукции, подвергнутой перемещению, а также сведения, позволяющие отразить, на какой именно поле располагался товар, подвергнутый перемещению; на какой именно поле данный товар станет располагаться после окончания перемещения [30].

Как только каждое из вышеперечисленных полей заполнено, система предложит пользователю кликнуть по клавише «Переместить».

В этом случае в системе появятся актуализированные данные о том, на какой именно полке будет располагаться перемещённый товар.

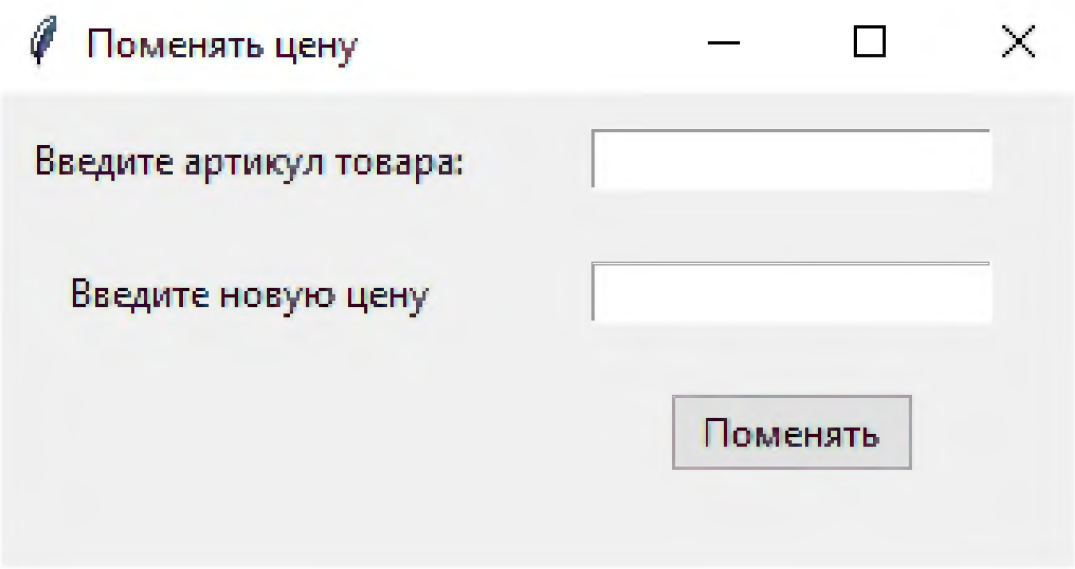


Рисунок 3.14-Графическое окно «Поменять цену»

На рисунке 3.14 нами показано, как выглядит окно, применяемое для того, чтобы актуализировать в системе информацию о стоимости продукции. В рассматриваемом окне присутствует несколько полей, в каждое из которых требуется внести соответствующие сведения.

В частности, это такие сведения, которые характеризуют артикул продукции, чья стоимость была изменена. Кроме того, это также и сведения, которые описывают новую стоимость товара.

Как только каждое из вышеперечисленных полей заполнено, система предложит пользователю кликнуть по клавише «Изменить». В этом случае в системе появятся актуализированные данные о том, какова стоимость того или иного товара, находящегося на хранении.

3.4 Оценка экономической эффективности.

При оценке экономической эффективности разработки и внедрения будем исходить из следующего:

1) времени на разработку приложения отводится 3 мес., остальные 9 мес. – опытная эксплуатация;
2) с начала опытной эксплуатации обязанности зав. складом совмещает директор, а зав. складом сокращается (зарплата - 75 тыс. руб.);
3) группа разработки – 1человека;
4) чистая прибыль предприятия до внедрения приложения: за год – 120 тыс. руб.; 120/12 = 10 тыс. руб. в месяц;
5) общая стоимость разработки – 720 тыс. руб. – это нераспределённая прибыль, накопленная грузоперевозчиком за 5 лет (120 тыс. руб. за год).
6) разработка и внедрение приложения выполняется по договору овозмездном оказании услугиз расчёта 60 тыс. руб. х 12 мес. = 720 тыс.руб, оформившему «самозанятость» (с учётом поощрительной первоначальной суммы – налог – 0 руб.). Это интересно разработчику - ничего не теряет на отчисления;
7) затраты на оборудование и программное обеспечение отсутствуют, поскольку используется ранее установленная локальная сеть компьютеров, а разработка ведётся наopensoft-инструментарии, поставляемом бесплатно.
8) По окончании года программа передаётся заказчику в пользование по лицензионному договору, в который входит сопровождение программы «как есть», то есть расширение возможностей программы - это за дополнительную оплату. Стоимость лицензии определим в размере 10 % от стоимости разработки – это 60 тыс. руб. за год (по 5тыс. руб. в месяц).
9) В период опытной эксплуатации разработанное приложение уже

должно повышать прибыльность предприятия за счёт более эффективного управления перевозками. Вполне оправданно ожидать, что наше приложение должно повышать прибыльность не меньше, чем другие приложения. Возьмём от 10% это средняя величина успешных внедрений.

Ход прибыли/убытков по этапам: разработка, опытная эксплуатация приводится в таблице и показан на графике:

Ход убытков/прибыли в период разработки и опытной эксплуатации приложения												
Чистая прибыль до внедрения приложения за год. тыс. р.:	120											
Чистая прибыль до внедрения приложения за месяц, тыс. р.:	10,0											
Оплата разработчику, тыс. р.:	60											
Зарплата зав.складом, тыс. р.:	75											
Стоимость лицензии на приложение, тыс. р.:	5											
Этап	Разработка			Опытная эксплуатация								
Месяц	1	2	3	4	5	6	7	8	9	10	11	12
Увеличение прибыли при использовани приложения, %:	-	-	-	10	10	10	10	10	10	10	10	10
Прибыль без приложения, тыс. р.	10	20	30	40	50	60	70	80	90	100	110	120
Убыток - оплаты разработчику, тыс. р.	-60	-120	-180	-240	-300	-360	-420	-480	-540	-600	-660	-720
Убыток - оплаты лицензии, тыс. р.	-	-	-	-5	-10	-15	-20	-25	-30	-35	-40	-45
Прибавка к прибыли при сокращении зав.складом, тыс. р.	0	0	0	75	150	225	300	375	450	525	600	675
Прибавка к прибыли за счёт приложения, тыс. р.	0	0	0	1	2	3	4	5	6	7	8	9
Прибыль/убыток, итоговая, тыс. р.	-50	-100	-150	-129	-108	-87	-66	-45	-24	-3	18	39

Рисунок 3.15 - Расчёт экономической эффективности приложения

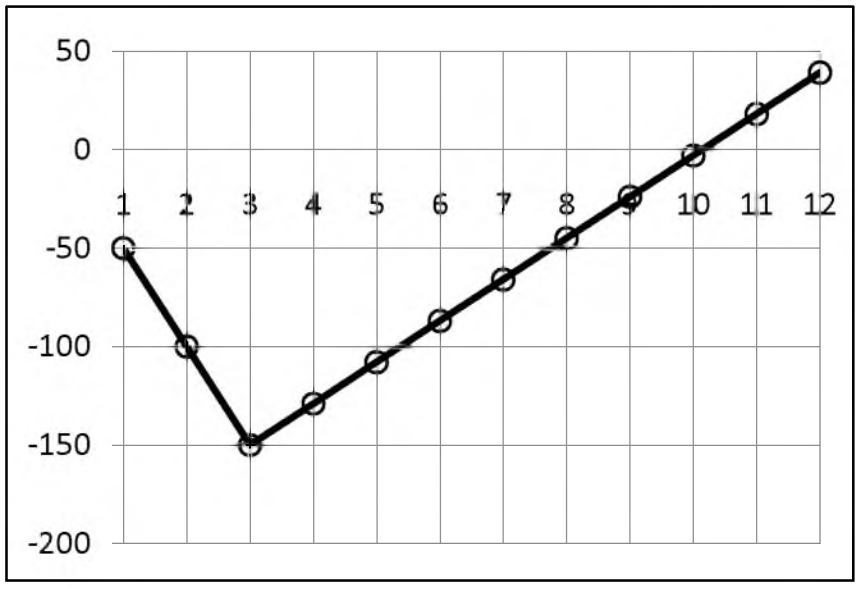


Рисунок 3.16 - Ход прибыли на этапах разработки и внедрения приложения

Таким образом, разработка инф. системы по силам малому предприятию грузоперевозчику и экономически оправдана.

Экономическая эффективность деятельности существенно повышается и можно ожидать существенный прирост чистой прибыли.

Расходы на разработку и внедрение окупаются уже к седьмому месяцу опытной эксплуатации системы.

Заключение

В настоящем исследовании нами было представлено обустройство автоматизированной системы, которая обеспечивает повышение качества учётной деятельности в организации, занимающейся обработкой большого количества разнообразных видов продукции. Созданная нами система имеет большое количество инструментов для автоматизации повседневных, рутинных операций, которые в большом количестве возникают в деятельности предприятия, работающего с широким ассортиментом продукции. Благодаря предусмотренным нами возможностям система может эксплуатироваться таким образом, чтобы её применение позволяло учитывать различные детали товаров, поступающих в компанию, вплоть до конкретного номера даже небольших образцов продукции. Кроме того, система, которая нами была представлена в настоящей работе, имеет простой для эксплуатации интерфейс, в связи с чем заниматься работой с ней могут даже лица с низким уровнем подготовки.

Одним из ключевых этапов разработки был анализ и изучение инструментов, необходимых для создания функциональной и эффективной системы учета. В рамках этого процесса было уделено особое внимание изучению DB Browser (SQLite), на основе которого была создана база данных. Этот шаг позволил углубить понимание принципов работы с базами данных и использовать их для хранения и обработки информации о товарах на складе.

Кроме того, для разработки информационной системы был изучен язык программирования Python 3. На его основе было написано приложение с графическим интерфейсом, обеспечивающее удобное взаимодействие с системой. Это позволяет пользователям производить операции по учету товаров на складе.

Так же были изучены следующие аспекты:

1)Изучены требования организации к системе учета товаров.

2) Проанализированы существующие информационные системы учета товаров на рынке.

3) Определены функциональные возможности и необходимые модули информационной системы.

4)Разработаны архитектура и прототип системы.

5)Оптимизирована функциональность системы.

Все поставленные цели и задачи для данной работы были выполнены в полном объёме.

Исходя из вышеизложенного, разработка информационной системы учёта товаров имеет значительный потенциал для улучшения управления запасами, оптимизации процессов и повышения эффективности работы в организации. Она поможет сократить временные и финансовые затраты, снизить риск ошибок и усилить контроль над учётными процессами. Введение данной системы является важным шагом в цифровой трансформации организации и обеспечит её конкурентоспособность и устойчивость на рынке.

Список литературы

1. Белов, В.В. Проектирование информационных систем: учеб. / В.В. Белов. - М.: Академия, 2018. - 144 с.

2. Гвоздева, Т.В. Проектирование информационных систем: технология автоматизированного проектирования. Лабораторный практикум. Учебно-справочное пособие / Т.В. Гвоздева, Б.А. Баллод. - СПб.: Лань, 2018. - 156 с.

3. Гвоздева, Т.В. Проектирование информационных систем. Стандартизация: учеб. пособие / Т.В. Гвоздева, Б.А. Баллод. - СПб.: Лань, 2019. - 252 с.

4. Дыбская, В.В. Проектирование системы распределения в логистике: Монография / В.В. Дыбская. - М.: Инфра-М, 2019. - 277 с.

5. Зырянов, Ю.Т. Проектирование радиопередающих устройств для систем подвижной радиосвязи: учеб. пособие / Ю.Т. Зырянов, П.А. Федюнин, О.А. Белоусов. - СПб.: Лань, 2018. - 116 с.

6. Конюх, В.Л. Проектирование автоматизир. систем производст.: учеб. пособие / В.Л. Конюх. - М.: Курс, 2018. - 64 с.

7. Конюхова, Е.А. Проектирование систем электроснабжения промышленных предприятий (теория и примеры) / Е.А. Конюхова. - М.: Русайнс, 2018. - 224 с.

8. Коршак, А.А. Проектирование систем газораспределения: учеб. пособие / А.А. Коршак. - Рн/Д: Феникс, 2018. - 192 с.

9. Мартишин, С.А. Проектирование и реализация баз данных в СУБД MySQL с использованием MySQLWorkbench: Методы и средства проектирования информационных систем и техноло / С.А. Мартишин, В.Л. Симонов, М.В. Храпченко. - М.: Форум, 2018. - 61 с.

10.Остроух, А.В. Интеллектуальные информационные системы и технологии: Монография / А.В. Остроух, А.Б. Николаев. - СПб.: Лань, 2019. - 308 с.

11. Перлова, О.Н. Проектирование и разработка информационных систем:

учеб. / О.Н. Перлова. - М.: Академия, 2018. - 272 с.

12. Федоренко, И.Я. Проектирование технических устройств и систем: принципы, методы, процедуры: учеб. пособие / И.Я. Федоренко, А.А. Смышляев. - М.: Форум, 2018. - 176 с.

13. Хорольский, В.Я. Проектирование и эксплуатация энергоустановок телекоммуникационных систем: Учеб. пособие / В.Я. Хорольский, А.Б. Ершов. - М.: Форум, 2019. - 285 с.

14. Лашина, М.В. Информационные системы и технологии в экономике и маркетинге: Учеб. пособие / М.В. Лашина, Т.Г. Соловьев. - М.: КноРус, 2018. - 480 с.

15. Сулейманова, Д.Ю. Информационные системы управления инновационными процессами / Д.Ю. Сулейманова. - М.: Русайнс, 2018. - 224 с.

16.Уткин, В.Б. Информационные системы в экономике / В.Б. Уткин. - М.: Academia, 2018. - 189 с.

17. Федорова, Г.Н. Информационные системы: учеб. / Г.Н. Федорова. - М.: Academia, 2018. - 384 с.

18. Чистов, Д.В. Информационные системы в экономике: учеб. пособие / Д.В. Чистов. - М.: Инфра-М, 2019. - 248 с.

19. Официальный сайт python [Электронный ресурс]. URL: https://www.python.org/ (Дата обращения 13.12.2023)

20. Официальный сайт группы компаний «АйТи» [Электронный ресурс].URL:http://www.it.ru(Дата обращения: 10.11.2023).

21. Сайт Professorweb [Электронный ресурс].URL: http://professorweb.ru (Дата обращения: 20.11.2023).

22. Официальный сайт компании «Новософт» [Электронный ресурс]. URL:http://www.novosoft.ru (Дата обращения 21.11.2023).

23. Официальный сайт компании «Basecamp» [Электронный ресурс]. URL: https://basecamp.com (Дата обращения 24.11.2023).

24. Официальный сайт компании «Инфоком» [Электронный ресурс]. URL: https://infocom-s.ru/ (Дата обращения 02.12.2023).

25. Сайт codecademy. [Электронный ресурс]. URL: https://www.codecademy.com (Дата обращения 02.12.2023).

26. Сайтtutorialspoint [Электронный ресурс]. URL: https://www.tutorialspoint.com/ (Дата обращения 15.12.2023).

27. Сайтlearnpython [Электронный ресурс]. URL: https://learnpython.org/ (Дата обращения 16.12.2023).

28. Сайтstepik [Электронный ресурс]. URL:https://stepik.org/ (Дата обращения 1.12.2023).

29. Сайтsqlfiddle [Электронный ресурс]. URL:https://sqlfiddle.com/ (Дата обращения 19.12.2023).

30. Сайтhackerrank [Электронный ресурс]. URL: https://www.hackerrank.com/(Дата обращения 22.12.2023).

Приложение А

Коды программы

```
# 1
pattern = r'^\d{4}-\d{2}-\d{2}$'

# 2
def authentification():
    login = username_entry.get()
    password = password_entry.get()

    if(login == "admin") and (password == "123"):
        authentification_window.destroy()
        main_program()
```

1. Функция аутентификации

```
# 3
def main_program():
    #4
    root = Tk()
    root.title("Мармеладный склад")
    root.geometry("1300x600")
    root.state('zoomed')
    # 5 connection to sklad.db
    connection = sql.connect('sklad.db')
    cursor = connection.cursor()
```

2. Основное окно

```
#6
def info_tovar():
    cursor.execute('SELECT Article, Name, Quantity, Price, Manufacturer, Country, Description, Polka FROM TOVAR')
    tovars = cursor.fetchall()
    for tovar in tovars:
        tovar_tree.insert("", END, values=tovar)

def info_postavka():
    cursor.execute('SELECT Nomer, Data, Summa, Kolichestvo, ArticleTovar, Postavshik, Prinyal FROM PRIHOD')
    postavki = cursor.fetchall()
    for postavka in postavki:
        postavka_tree.insert("", END, values=postavka)

def info_brak():
    cursor.execute('SELECT Nomer, ArticleTovar, Kolichestvo, Sotrudnik, Data, Prichina FROM BRAK')
    braki = cursor.fetchall()
    for brak in braki:
        brak_tree.insert("", END, values=brak)

def info_otgruzka():
    cursor.execute('SELECT Nomer, ArticleTovar, Data, Prinimaet, Otpustil, Kolichestvo, Summa FROM OTGRUZKA')
    otgruzki = cursor.fetchall()
    for otgruzka in otgruzki:
        otgruzka_tree.insert("", END, values=otgruzka)
```

3. Основные функции

Продолжение приложения А

```
# 7
def refresh():
    tovar_tree.delete(*tovar_tree.get_children())
    postavka_tree.delete(*postavka_tree.get_children())
    brak_tree.delete(*brak_tree.get_children())
    otgruzka_tree.delete(*otgruzka_tree.get_children())
    info_tovar()
    info_postavka()
    info_brak()
    info_otgruzka()
```

4. Функция обновления

```
#8
def find_tovar():
    tovar_tree.delete(*tovar_tree.get_children())
    tovars = []
    cursor.execute('SELECT Article, Name, Quantity, Price, Manufacturer, Country, Description, Polka FROM TOVAR')
    tovars = cursor.fetchall()

    postavka_tree.delete(*postavka_tree.get_children())
    postavki = []
    cursor.execute('SELECT Nomer, Data, Summa, Kolichestvo, ArticleTovar, Postavshik, Prinyal FROM PRIHOD')
    postavki = cursor.fetchall()

    brak_tree.delete(*brak_tree.get_children())
    braki = []
    cursor.execute('SELECT Nomer, ArticleTovar, Kolichestvo, Sotrudnik, Data, Prichina FROM BRAK')
    braki = cursor.fetchall()

    otgruzka_tree.delete(*otgruzka_tree.get_children())
    otgruzki = []
    cursor.execute('SELECT Nomer, ArticleTovar, Data, Prinimaet, Otpustil, Kolichestvo, Summa FROM OTGRUZKA')
    otgruzki = cursor.fetchall()

    search_tovar = find_entry.get()
    if search_tovar == '':
        refresh()
        return
    for tovar in tovars:
        if (search_tovar in tovar):
            tovar_tree.insert('', END, values=tovar)
        else:
            name = tovar[1]
            opisanie = tovar[6]
            name.split()
            if search_tovar in name or search_tovar in opisanie:
                    tovar_tree.insert('', END, values=tovar)

    is_digit = search_tovar.isdigit()
    if is_digit == True and search_tovar != '':
        for tovar in tovars:
            if int(search_tovar) in tovar:
                tovar_tree.insert('', 'end', values=tovar)
            else:
                continue

    #
    if search_tovar == '':
        refresh()
        return
    for postavka in postavki:
        if (search_tovar in postavka):
            postavka_tree.insert('', END, values=postavka)
```

5. Функция поиска товара

Продолжение приложения А

```
def otchet():
    window_otchet = Tk()
    window_otchet.geometry("250x160")
    window_otchet.title("Сформировать отчёт")

    def otchet_go():
        category = spinbox_otchet.get()
        date_do = entry_do.get()
        date_ot = entry_ot.get()

        def brack():

            window_go = Tk()
            window_go.geometry("1200x568")
            def screen():
                window_go.update()
                time.sleep(1)
                window_go.focus_set()
                window_go.focus_force()
                #cap = tkcap.CAP(window_go)
                #cap.capture("photo.png", window_go)
                x = window_go.winfo_rootx()
                y = window_go.winfo_rooty()
                width = window_go.winfo_width()
                height = window_go.winfo_height()
                screen = ImageGrab.grab(bbox=(x, y, x+width, y+height),include_layered_windows = True)

                fllename = filedialog.asksaveasfilename(defaultextension=".png", filetypes=[("PNG files", "*.png")])
                if fllename:
                    screen.save(fllename)

            #window_go.title("Отчёт")
            #date_do = entry_do.get()
            #date_ot = entry_ot.get()

            window_go.title("Брак | Отчёт")
            #window_go.title("brq")
            brak_columns = ('Nomer', 'ArticleTovar', 'Kolichestvo', 'Sotrudnik', 'Data', 'Prichina')
            tree_brak = ttk.Treeview(window_go, columns=brak_columns, show="headings", height=70, selectmode=None)
            tree_brak.heading("Nomer", text="Номер")
            tree_brak.heading("ArticleTovar", text="Артикул")
            tree_brak.heading("Kolichestvo", text="Количество")
            tree_brak.heading("Sotrudnik", text="Знёс")
            tree_brak.heading("Data", text="Дата")
            tree_brak.heading("Prichina", text="Причина")
            tree_brak.column("#1", stretch=YES, width=80, anchor='c')
            tree_brak.column("#2", stretch=YES, width=100, anchor='c')
            tree_brak.column("#3", stretch=YES, width=100, anchor='c')
            tree_brak.column("#4", stretch=YES, width=170, anchor='c')
            tree_brak.column("#5", stretch=YES, width=140, anchor='c')
            tree_brak.column("#6", stretch=YES, width=210, anchor='c')

            cursor.execute('SELECT * FROM BRAK WHERE Data BETWEEN (?) AND (?)', (date_ot, date_do))
            dates = cursor.fetchall()
                #print(dates)
            for date in dates:
                tree_brak.insert("", END, values=date)


            tree_brak.pack(fill=BOTH, expand=1)
            screen()
```

6. Отчёт и функции для него

Приложение Б

Коды составления отчётов

```
def otgruzochka():
    window_go = Tk()
    window_go.geometry("1200x568")
    def screen():
        window_go.update()
        time.sleep(1)
        window_go.focus_set()
        window_go.focus_force()
        #cap = tkcap.CAP(window_go)
        #cap.capture("photo.png", window_go)
        x = window_go.winfo_rootx()
        y = window_go.winfo_rooty()
        width = window_go.winfo_width()
        height = window_go.winfo_height()
        screen = ImageGrab.grab(bbox=(x, y, x+width, y+height),include_layered_windows = True)
        screen.save("otgruzka.jpeg")

        fllename = filedialog.asksaveasfilename(defaultextension=".png", filetypes=[("PNG files", "*.png")])
        if fllename:
            screen.save(fllename)


    #date_do = entry_do.get()
    #date_ot = entry_ot.get()
    window_go.title("Отгрузка | Отчет")
    otgruzka_columns = ('Nomer', 'ArticleTovar', 'Data', 'Prinimaet', 'Otpustil', 'Kolichestvo', 'Summa')
    tree_otgruzka = ttk.Treeview(window_go, columns=otgruzka_columns, show="headings", height=70, selectmode=None)

    tree_otgruzka.heading("Nomer", text="Номер")
    tree_otgruzka.heading("ArticleTovar", text="Артикул")
    tree_otgruzka.heading("Data", text="Дата")
    tree_otgruzka.heading("Prinimaet", text="Заказчик")
    tree_otgruzka.heading("Otpustil", text="Отпустил")
    tree_otgruzka.heading("Kolichestvo", text="Количество")
    tree_otgruzka.heading("Summa", text="Сумма")
    tree_otgruzka.column("#1", stretch=YES, width=150, anchor='c')
    tree_otgruzka.column("#2", stretch=YES, width=150, anchor='c')
    tree_otgruzka.column("#3", stretch=YES, width=150, anchor='c')
    tree_otgruzka.column("#4", stretch=YES, width=150, anchor='c')
    tree_otgruzka.column("#5", stretch=YES, width=150, anchor='c')
    tree_otgruzka.column("#6", stretch=YES, width=150, anchor='c')
    tree_otgruzka.column("#7", stretch=YES, width=150, anchor='c')

    cursor.execute('SELECT * FROM OTGRUZKA WHERE Data BETWEEN (?) AND (?)', (date_ot, date_do))
    datesl = cursor.fetchall()
    for datel in datesl:
        tree_otgruzka.insert("", END, values=datel)

    tree_otgruzka.pack(fill=BOTH, expand=1)
    screen()
```

7. Отчёт отгрузки

Продолжение приложения Б

```
def postavochka():
    window_go = Tk()
    window_go.geometry("1200x568")
    def screen():
        window_go.update()
        time.sleep(1)
        window_go.focus_set()
        window_go.focus_force()
        #cap = tkcap.CAP(window_go)
        #cap.capture("photo.png", window_go)
        x = window_go.winfo_rootx()
        y = window_go.winfo_rooty()
        width = window_go.winfo_width()
        height = window_go.winfo_height()
        screen = ImageGrab.grab(bbox=(x, y, x+width, y+height),include_layered_windows = True)
        screen.save("postavka.jpeg")

        filename = filedialog.asksaveasfilename(defaultextension=".png", filetypes=[("PNG files", "*.png")])
        if filename:
            screen.save(filename)

    #date_do = entry_do.get()
    #date_ot = entry_ot.get()
    window_go.title("Поставка | Отчет")
    postavka_columns = ('Nomer', 'Data', 'Summa', 'Kolichestvo', 'ArticleTovar', 'Postavshik', 'Prinyal')
    tree_postavka = ttk.Treeview(window_go, columns=postavka_columns, show="headings", height=70, selectmode=None)

    tree_postavka.heading("Nomer", text="Номер")
    tree_postavka.heading("Data", text="Дата")
    tree_postavka.heading("Summa", text="Сумма")
    tree_postavka.heading("Kolichestvo", text="Количество")
    tree_postavka.heading("ArticleTovar", text="Артикул")
    tree_postavka.heading("Postavshik", text="Поставщик")
    tree_postavka.heading("Prinyal", text="Принял")
    tree_postavka.column("#1", stretch=YES, width=80, anchor='c')
    tree_postavka.column("#2", stretch=YES, width=150, anchor='c')
    tree_postavka.column("#3", stretch=YES, width=80, anchor='c')
    tree_postavka.column("#4", stretch=YES, width=80, anchor='c')
    tree_postavka.column("#5", stretch=YES, width=150, anchor='c')
    tree_postavka.column("#6", stretch=YES, width=150, anchor='c')
    tree_postavka.column("#7", stretch=YES, width=150, anchor='c')

    cursor.execute('SELECT * FROM PRIHOD WHERE Data BETWEEN (?) AND (?)', (date_ot, date_do))
    dates2 = cursor.fetchall()
    for date2 in dates2:
        tree_postavka.insert("", END, values=date2)

    tree_postavka.pack(fill=BOTH, expand=1)
    screen()
```

8. Отчёт поставок

Приложение В

Редактирование таблиц учёта товара

```
def prinyat():
    window_prinyat = Tk()
    window_prinyat.title("Поставка")
    window_prinyat.geometry("250x210")

    def prinyat_logic():
            article = entry_article.get()
            data = entry_data.get()
            otkogo = entry_company.get()
            prinyal = entry_prinyal.get()
            kolvo = entry_kolvo.get()
            summa = 0
            cursor.execute('SELECT Article FROM TOVAR')
            articles = cursor.fetchall()
            flag = False
            pattern = r'^\d{4}-\d{2}-\d{2}$'


            if (article == '') or (data == '') or (otkogo == '') or (prinyal == '') or (kolvo == ''):
                messagebox.showwarning("Ошибка", "Заполните все поля!")
                window_prinyat.lift()
            else:
                for x in articles:
                    if int(article) in x:
                        cursor.execute('SELECT Quantity FROM TOVAR WHERE Article = (?)', (article,))
                        quantity = cursor.fetchall()
                        quantity = quantity[0][0]
                        print(quantity)
                        cursor.execute('SELECT Price FROM TOVAR WHERE Article = (?) ', (article,))
                        price = cursor.fetchall()
                        price = price[0][0]
                        print(price)
                        summa = price * int(kolvo)

                        flag = True
                        if re.match(pattern, data):
                            cursor.execute('UPDATE TOVAR SET Quantity = Quantity + (?) WHERE Article = (?)', (kolvo, article))
                            cursor.execute('INSERT INTO PRIHOD (ArticleTovar, Data, Postavshik, Prinyal, Kolichestvo, Summa) VALUES (?, ?, ?, ?, ?, ?)', (article, data, otkogo, prinyal, kolvo, summa))
                            connection.commit()
                            window_prinyat.destroy()

                            nor = datetime.now()
                            string = adm + str(nor) + " | " + "INSERT INTO PRIHOD (ArticleTovar, Data, Postavshik, Prinyal, Kolichestvo, Summa) VALUES (" + str(article) + ", " + data + ", " + otkogo +
                            file = open("logs.txt", "a")
                            file.write(string)
                            file.close()
                        else:e = ope
                            messagebox.showwarning("Ошибка", "Введите корректную дату - 'ГГГГ-ММ-ДД'")
                            window_prinyat.lift()


                if flag == False:
                    messagebox.showwarning("Ошибка", "Товар не найден")
```

9. Логика принятия поставки

Продолжение приложения В

```
def otgruzit():
    window_otgruz = Tk()
    window_otgruz.title("Отгрузка")
    window_otgruz.geometry("250x210")

    def sdelat_otgruzku():
        article = entry_article.get()
        data = entry_data.get()
        komu = entry_company.get()
        otpustil = entry_otpustil.get()
        kolvo = entry_kolvo.get()
        summa = 0
        cursor.execute('SELECT Article FROM TOVAR')
        articles = cursor.fetchall()
        flag = False
        pattern = r'^\d{4}-\d{2}-\d{2}$'

        if (article == '') or (data == '') or (komu == '') or (otpustil == '') or (kolvo == ''):
            messagebox.showwarning("Ошибка", "Заполните все поля!")
            window_otgruz.lift()
        else:
            for x in articles:
                if int(article) in x:
                    cursor.execute('SELECT Quantity FROM TOVAR WHERE Article = (?)', (article,))
                    quantity = cursor.fetchall()
                    quantity = quantity[0][0]
                    print(quantity)
                    cursor.execute('SELECT Price FROM TOVAR WHERE Article = (?) ', (article,))
                    price = cursor.fetchall()
                    price = price[0][0]
                    print(price)
                    summa = price * int(kolvo)

                    flag = True
                    if re.match(pattern, data):

                        if(quantity >= int(kolvo)):
                            cursor.execute('UPDATE TOVAR SET Quantity = Quantity - (?) WHERE Article = (?)', (kolvo, article))
                            cursor.execute('INSERT INTO OTGRUZKA (ArticleTovar, Data, Prinimaet, Otpustil, Kolichestvo, Summa) VALUES (?, ?, ?, ?, ?, ?)', (article, data, komu, otpustil, kolvo, summa))
                            connection.commit()
                            window_otgruz.destroy()

                            nor = datetime.now()
                            string = adm +  str(nor) + " | " + "INSERT INTO OTGRUZKA (ArticleTovar, Data, Prinimaet, Otpustil, Kolichestvo, Summa) VALUES (" + str(article) + ", " + data + ", " + komu +
                            file = open("logs.txt", "a")
                            file.write(string)
                            file.close()

                        else:
                            messagebox.showwarning("Ошибка", "Не хватает товаров!")
                            window_otgruz.lift()
                    else:
                        messagebox.showwarning("Ошибка","Введите корректную дату - 'ГГГГ-ММ-ДД'" )
                        window_otgruz.lift()
```

10. Функция отгрузки

Продолжение приложения В

```
def vruchnuy():
    window_vnesti = Tk()
    window_vnesti.title("Внести новый товар вручную")
    window_vnesti.geometry("480x200")

    def vnesti_vruchnuy():
        article = entry_article.get()
        name = entry_name.get()
        kolvo = entry_kolvo.get()
        price = entry_price.get()
        manu = entry_manu.get()
        strana = entry_strana.get()
        opisani = entry_opisanie.get()
        polka = entry_polka.get()


        cursor.execute('SELECT Article FROM TOVAR')
        artikyls = cursor.fetchall()
        if (article == '') or (name == '') or (kolvo == '') or (price == '') or (manu == '') or (strana =='') or (polka == ''):
            messagebox.showwarning("Ошибка", "Запомните все поля!")
            window_vnesti.lift()
        else:
            try:
                cursor.execute('INSERT INTO TOVAR (Article, Name, Quantity, Price, Manufacturer, Country, Description, Polka) VALUES (?, ?, ?, ?, ?, ?, ?, ?)',(article, name, kolvo, price, manu, strana, opisani, polka) )
                connection.commit()
                window_vnesti.destroy()

                now = datetime.now()
                string = adm +  str(now) + " | " + "INSERT INTO TOVAR (Article, Name, Quantity, Price, Manufacturer, Country, Description, Polka) VALUES (" + str(article) + ", " + name + ", " + str(kolvo) + ", " + str(price) +
                file = open("logs.txt", "a")
                file.write(string)
                file.close()
            except sql.IntegrityError:
                messagebox.showwarning("Ошибка", "Такой товар уже существует")
                window_vnesti.lift()

    label_article = ttk.Label(window_vnesti, text="Артикул:")
    label_article.grid(column=0, row=0, padx=5, pady=5)
    entry_article = ttk.Entry(window_vnesti)
    entry_article.grid(column=1, row=0, padx=5, pady=5)

    label_name = ttk.Label(window_vnesti, text="Название:")
    label_name.grid(column=2, row=0, padx=5, pady=5)
    entry_name = ttk.Entry(window_vnesti)
    entry_name.grid(column=3, row=0, padx=5, pady=5)

    label_kolvo = ttk.Label(window_vnesti, text="Количество:")
    label_kolvo.grid(column=0, row=1, padx=5, pady=5)
    entry_kolvo = ttk.Entry(window_vnesti)
    entry_kolvo.grid(column=1, row=1, padx=5, pady=5)
```

11.Функция ручного ввода

Продолжение приложения В

```
def peremestit():
    def peremeshenie():
        artikyl = entry_peremestit_a.get()
        polka = entry_peremestit_p.get()

        cursor.execute('SELECT Article FROM TOVAR')
        articles = cursor.fetchall()

        if artikyl == '' or polka == '':
            messagebox.showwarning("Ошибка", "Заполните все поля!")
            window_peremestit.lift()
        else:
            flag = False
            for x in articles:
                if int(artikyl) in x:
                    cursor.execute('UPDATE TOVAR SET Polka = ? WHERE Article = ?', (polka,artikyl))
                    connection.commit()
                    window_peremestit.destroy()
                    flag = True
                    nor = datetime.now()
                    string = adm +  str(nor) + " | " + "UPDATE TOVAR SET Polka = " + polka + " WHERE Article = " + artikyl + '\n'
                    file = open("logs.txt", "a")
                    file.write(string)
                    file.close()
            if flag == False:
                messagebox.showwarning("Ошибка", "Товар не найден")
                window_peremestit.lift()

    window_peremestit = Tk()
    window_peremestit.title("Перемещение товара")
    window_peremestit.geometry("340x150")
    label_peremestit_a = Label(window_peremestit, text="Введите артикул товара:")
    label_peremestit_a.grid(column=0, row=0, padx=10, pady=10)
    entry_peremestit_a = Entry(window_peremestit)
    entry_peremestit_a.grid(column=1, row=0, padx=10, pady=10)

    label_peremestit_p = Label(window_peremestit, text="Введите номер полки")
    label_peremestit_p.grid(column=0, row=1, padx=10, pady=10)
    entry_peremestit_p = Entry(window_peremestit)
    entry_peremestit_p.grid(column=1, row=1, padx=10, pady=10)

    button_peremestit = ttk.Button(window_peremestit, text="Переместить", command=peremeshenie)
    button_peremestit.grid(column=1, row=2, padx=50, pady=10)
    window_peremestit.mainloop()
```

12.Перемещение товара

Продолжение приложения В

```
def tsena():

    def peremeshenie():
        artikyl = entry_peremestit_a.get()
        tsena = entry_peremestit_p.get()

        cursor.execute('SELECT Article FROM TOVAR')
        articles = cursor.fetchall()

        if artikyl == '' or tsena == '':
            messagebox.showwarning("Ошибка", "Заполните все поля!")
            window_peremestit.lift()
        else:
            flag = False
            for x in articles:
                if int(artikyl) in x:
                    cursor.execute('UPDATE TOVAR SET Price = ? WHERE Article = ?', (tsena,artikyl))
                    connection.commit()
                    window_peremestit.destroy()
                    flag = True
                    nor = datetime.now()
                    string = adm +  str(nor) + " | " + "UPDATE TOVAR SET Price = " + tsena + " WHERE Article = " +
                    file = open("logs.txt", "a")
                    file.write(string)
                    file.close()
            if flag == False:
                messagebox.showwarning("Ошибка", "Товар не найден")
                window_peremestit.lift()

    window_peremestit = Tk()
    window_peremestit.title("Поменять цену")
    window_peremestit.geometry("340x150")
    label_peremestit_a = Label(window_peremestit, text="Введите артикул товара:")
    label_peremestit_a.grid(column=0, row=0, padx=10, pady=10)
    entry_peremestit_a = Entry(window_peremestit)
    entry_peremestit_a.grid(column=1, row=0, padx=10, pady=10)

    label_peremestit_p = Label(window_peremestit, text="Введите новую цену")
    label_peremestit_p.grid(column=0, row=1, padx=10, pady=10)
    entry_peremestit_p = Entry(window_peremestit)
    entry_peremestit_p.grid(column=1, row=1, padx=10, pady=10)

    button_peremestit = ttk.Button(window_peremestit, text="Поменять", command=peremeshenie)
    button_peremestit.grid(column=1, row=2, padx=50, pady=10)
    window_peremestit.mainloop()

#def logging():
```

13. Функция изменения цен

Продолжение приложения В

```
columns_tovar = ('Article', 'Name', 'Quantity', 'Price', 'Manufacturer', 'Country', 'Description', 'Polka')
columns_postavka = ('Nomer', 'Data', 'Summa', 'Kolichestvo', 'ArticleTovar', 'Postavshik', 'Prinyal')
columns_brak = ('Nomer', 'ArticleTovar', 'Kolichestvo', 'Sotrudnik', 'Data', 'Prichina')
columns_otgruzka = ('Nomer', 'ArticleTovar', 'Data', 'Prinimaet', 'Otpustil', 'Kolichestvo', 'Summa')

tovar_tree = ttk.Treeview(tovar_frame, columns=columns_tovar, show="headings", height=70, selectmode=None)
postavka_tree = ttk.Treeview(postavka_frame, columns=columns_postavka, show="headings", height=70, selectmode=None)
brak_tree = ttk.Treeview(brak_frame, columns=columns_brak, show="headings", height=70, selectmode=None)
otgruzka_tree = ttk.Treeview(otgruzka_frame, columns=columns_otgruzka, show="headings", height=70, selectmode=None)

tovar_frame.pack(pady=1, padx=1, fill=BOTH)
postavka_frame.pack(pady=1, padx=1, fill=BOTH)
brak_frame.pack(pady=1, padx=1, fill=BOTH)
otgruzka_frame.pack(pady=1, padx=1, fill=BOTH)

tovar_tree.heading("Article", text="Артикул")
tovar_tree.heading("Name", text="Название")
tovar_tree.heading("Quantity", text="Количество")
tovar_tree.heading("Price", text="Цена за штуку")
tovar_tree.heading("Manufacturer", text="Производитель")
tovar_tree.heading("Country", text="Страна")
tovar_tree.heading("Description", text="Описание")
tovar_tree.heading("Polka", text="Полка")
tovar_tree.column("#1", stretch=YES, width=80, anchor='c')
tovar_tree.column("#2", stretch=YES, width=150, anchor='c')
tovar_tree.column("#3", stretch=YES, width=100, anchor='c')
tovar_tree.column("#4", stretch=YES, width=100, anchor='c')
tovar_tree.column("#5", stretch=YES, width=150, anchor='c')
tovar_tree.column("#6", stretch=YES, width=150, anchor='c')
tovar_tree.column("#7", stretch=YES, width=150, anchor='c')
tovar_tree.column("#8", stretch=YES, width=80, anchor='c')

postavka_tree.heading("Nomer", text="Номер")
postavka_tree.heading("Data", text="Дата")
postavka_tree.heading("Summa", text="Сумма")
postavka_tree.heading("Kolichestvo", text="Количество")
postavka_tree.heading("ArticleTovar", text="Артикул")
postavka_tree.heading("Postavshik", text="Поставщик")
postavka_tree.heading("Prinyal", text="Принял")
postavka_tree.column("#1", stretch=YES, width=80, anchor='c')
postavka_tree.column("#2", stretch=YES, width=150, anchor='c')
postavka_tree.column("#3", stretch=YES, width=80, anchor='c')
postavka_tree.column("#4", stretch=YES, width=80, anchor='c')
postavka_tree.column("#5", stretch=YES, width=150, anchor='c')
postavka_tree.column("#6", stretch=YES, width=150, anchor='c')
postavka_tree.column("#7", stretch=YES, width=150, anchor='c')
```

14. Использование виджета Treeview

Продолжение приложения В

```
# Окно авторизации
authentification_window = Tk()
authentification_window.geometry("220x140")
authentification_window.title("Авторизация")

# Логин
username_label = Label(authentification_window, text="Логин:")
username_label.grid(padx=10, pady=10, column = 0, row = 0)
username_entry = Entry(authentification_window)
username_entry.grid(padx=5, pady=10, column = 1, row = 0)

# Пароль
password_label = Label(authentification_window, text="Пароль:")
password_label.grid(padx=10, pady=10, column = 0, row = 1)
password_entry = Entry(authentification_window)
password_entry.grid(padx=10, pady=10, column=1,row =1)

#Кнопка
authentification_button = ttk.Button(authentification_window, text="Войти", command=authentification)
authentification_button.grid(columnspan=4, padx=10, pady = 10, column = 0, row = 2)
authentification_window.mainloop()
```

15. Графическое окно авторизации