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 Сковороднев Юрий Олегович

Руководитель ст. преподаватель Сафонова Татьяна Владимировна

«К защите допускаю»

Руководитель кафедры _____

кандидат экономических наук

Майборода Е.В.

«13» января 2024 г.

Филиал Российского государственного гидрометеорологического университета в г. Туапсе
НОРМОКОНТРОЛЬ ПРОЙДЕН
«22» января 2024 г.
Мартынова Ю. В.
ПОДПИСЬ РАСШИФРОВКА ПОДПИСИ

Туапсе
2024

ОГЛАВЛЕНИЕ

Введение

Развитие информационных технологий непрерывно меняет способы управления и организации бизнес-процессов в современных организациях. Одной из главных составляющих успешного ведения дел является точный учет товаров. В связи с этим, разработка информационной системы учета товаров является востребованной темой.

Актуальность темы «разработка информационной системы учета товаров» обусловлена несколькими факторами:

1)Оптимизация бизнес-процессов: Информационная система учета товаров позволяет автоматизировать и оптимизировать процессы учета, отслеживания и контроля товаров, что способствует улучшению эффективности бизнес-процессов. Правильный учет товаров обеспечивает точное планирование производства, закупки, хранения и продажи товаров, что позволяет избежать излишних затрат и минимизировать потери.

2)Увеличение конкурентоспособности: Современный бизнес требует высокой скорости и эффективности учета товаров. Информационная система учета товаров позволяет оперативно реагировать на изменения в спросе и поставках, прогнозировать и планировать потребности в товарах, что способствует улучшению сервиса и повышению конкурентоспособности компании.

3)Минимизация ошибок и улучшение точности: Вручную вести учет товаров чревато ошибками в данных, дублированием информации и потерей ценного времени. Информационная система учета товаров позволяет автоматизировать и структурировать процессы учета, устраняя возможность человеческого фактора и повышая точность и достоверность данных.

4)Удовлетворение запросов потребителей: Современные потребители требуют оперативности, доступности и адаптивности сервисов. Информационная система учета товаров позволяет обеспечить оперативность и достоверность информации о наличии товаров, их характеристиках и ценах, что

способствует повышению удовлетворенности потребителя и формированию лояльности к компании.

5)Рост объемов и сложности бизнес-процессов: с появлением электронной коммерции, увеличением числа клиентов и расширением ассортимента товаров, организации сталкиваются с растущими объемами данных, которые необходимо отслеживать и учитывать. Информационная система учета товаров позволяет эффективно организовать и обрабатывать эти данные, упрощая процессы учета и повышая производительность.

6)Необходимость точности и актуальности информации: в условиях быстро меняющегося рынка и сильной конкуренции, точный учет товаров становится критически важным. Ошибки или задержки в учете могут привести к излишним запасам, потере клиентов, а также к финансовым потерям. Разработка информационной системы учета товаров помогает обеспечить актуальность и точность информации о наличии товаров, состоянии запасов и заказах, минимизируя риски и повышая эффективность бизнеса.

7)Необходимость автоматизации и интеграции бизнес-процессов: Современные организации активно внедряют информационные системы, чтобы автоматизировать и интегрировать бизнес-процессы. Разработка информационной системы учета товаров позволяет связать данные о товарах с другими системами, такими как системы управления складом или системы электронной коммерции, обеспечивая единое информационное пространство и повышая эффективность взаимодействия всех компонентов бизнеса.

8)Возросшая потребность в анализе данных: Информационная система учета товаров может собирать и анализировать данные о продажах, складских запасах, потребительском спросе и других факторах, позволяя принимать обоснованные бизнес-решения. Это особенно важно для прогнозирования спроса, оптимизации запасов и планирования производства, что позволяет сократить затраты и повысить эффективность работы.

Объект исследования: Бизнес-процессы по реализации деятельности подразделения складского учета.

Предмет исследования: Оптимизация и автоматизация деятельности подразделения складского учета.

Цель: Разработать эффективную информационную систему, которая позволит организации эффективно управлять учетом товаров, отслеживать их перемещение, контролировать запасы и оптимизировать процессы управления.

Задачи исследования:

1) Изучение требований организации к системе учета товаров.
2) Анализ существующих информационных систем учета товаров на рынке.
3) Определение функциональных возможностей и необходимых модулей информационной системы.
4) Разработка архитектуры и прототипа системы.
5) Оптимизация функциональности системы.

Выполнение этих задач позволит создать эффективную информационную систему учета товаров, которая улучшит управление запасами, сократит издержки и повысит эффективность работы организации.

Таким образом, разработка информационной системы учета товаров имеет высокую актуальность и значимость для современных организаций, помогая оптимизировать бизнес-процессы, повысить конкурентоспособность и удовлетворить запросы потребителей.

1 Теоретические аспекты разработки информационной системы

1.1 Определение информационной системы для учета товара

Информационная система для учета товара - это комплекс программных и аппаратных средств, специально разработанных для автоматизации процесса учета и управления товарными запасами в организации. Ее целью является обеспечение эффективного и точного контроля за движением товаров от момента поступления на склад до реализации, а также поддержка принятия управленческих решений в сфере учета товаров.

Основные функции информационной системы для учета товара могут включать:

1) Отслеживание поступления и отправки товаров: система должна фиксировать все поступления товаров на склад, включая информацию о количестве, стоимости, поставщике, а также дате и времени поступления. Аналогично, система должна отслеживать отгрузки товаров клиентам.

2) Контроль остатков товара: система должна автоматически вычитать из остатка товара каждую продажу или отгрузку и обновлять информацию о текущем количестве товаров на складе. Такой контроль позволит владельцу или менеджеру оперативно реагировать на необходимость дополнительных закупок или перераспределения товаров.

3) Учет стоимости товара: система должна хранить информацию о стоимости каждого товара, а также общую стоимость запасов на складе. Это поможет управленческому персоналу определить точные затраты на запасы и оценить их рентабельность.

4) Генерация отчетов: информационная система должна иметь возможность генерировать различные отчеты. Это позволяет анализировать данные, определять тенденции и принимать обоснованные решения на основе данных.

В целом, информационная система для учета товара предназначена для автоматизации и оптимизации процесса учета и управления товарами,

снижения рисков ошибок и улучшения эффективности организации [1, с.44].

1.2 Преимущества и недостатки существующих решений

Преимущества существующих решений в информационных системах:

1) Автоматизация бизнес-процессов: Информационные системы позволяют автоматизировать повседневные задачи, упрощая их выполнение и освобождая ресурсы для выполнения более сложных и стратегически важных задач.

2) Улучшение принятия решений: Информационные системы обеспечивают доступ к актуальным данным и аналитическим инструментам, позволяющим принимать обоснованные решения на основе фактов и данных. Это помогает снизить риски и повысить эффективность принимаемых решений.

3) Увеличение производительности и эффективности: Информационные системы помогают оптимизировать бизнес-процессы и повышать производительность сотрудников, сокращая время выполнения задач и устраняя дублирование работ.

4) Улучшение коммуникации и сотрудничества: Информационные системы обеспечивают средства для легкого обмена информацией, документами и переговорами между сотрудниками. Это способствует лучшей коммуникации и сотрудничеству внутри организации.

5) Улучшение доступности и удовлетворенности клиентов: Информационные системы позволяют организациям предлагать лучший уровень обслуживания клиентов, предоставляя им удобные способы взаимодействия, быстрый доступ к информации и возможность получать персонализированный опыт обслуживания.

Недостатки существующих решений в информационных системах:

1) Сложность и затратность внедрения: Внедрение информационной системы может быть сложным и требовать значительных затрат, как финансовых, так и временных. Оно может потребовать перестройки бизнес-

процессов, обучения персонала и интеграции с другими существующими системами.

2) Зависимость от поставщиков: При использовании готового решения организация становится зависимой от поставщика, что может ограничивать ее гибкость и свободу действий. Поддержка, обновления и расширение системы могут зависеть от поставщика, что может повлечь дополнительные затраты и неудобства.

3) Риски безопасности: Информационные системы подвержены различным видам угроз безопасности, включая хакерские атаки, вирусы, утечки данных и другие. Недостатки в системе безопасности и ошибки в ее эксплуатации могут привести к уязвимостям и потере конфиденциальности данных.

4) Интеграционные сложности: Существующие системы могут не соответствовать требованиям и стандартам интеграции с другими системами, что может создавать сложности при обмене данными и взаимодействии с внешними системами [2, с.101].

1.3 Основные требования к информационной системе для учета товара

Основные требования к информационной системе для учета товара могут включать следующие аспекты:

1) Надежность: Информационная система должна быть надежной и стабильной, чтобы обеспечить бесперебойную работу и минимизировать риск потери данных или сбоев.

2) Точность данных: Система должна обеспечивать точность данных о товарах, поставках, отгрузках, остатках и других параметрах, так как на основе этих данных принимаются управленческие решения и планируются операции.

3) Автоматизация: Информационная система должна автоматизировать процессы учета товаров, минимизируя ручной ввод данных и предоставляя функции по автоматической обработке и анализу информации.

4) Масштабируемость: Система должна обладать возможностью масштабирования, чтобы поддерживать учет товаров в зависимости от роста бизнеса и увеличения объема операций.

5) Удобный интерфейс пользователя: Система должна иметь интуитивно понятный и удобный интерфейс, что позволит пользователям легко осваивать и работы с ней, а также повысит эффективность использования системы.

6) Генерация отчетов: Информационная система должна предоставлять возможность генерации разнообразных отчетов о товарах, движении запасов, обороте, остатках и др. Качественные отчеты помогут анализировать эффективность управления запасами, выявлять закономерности и тенденции, планировать деятельность и принимать обоснованные решения.

7) Гибкость в управлении ценами: Информационная система должна предоставлять возможность гибкого изменения цен на товары.

Эти требования к информационной системе для учета товара помогут обеспечить эффективное управление запасами, точность учета и анализ данных, улучшение процессов управления и принятие обоснованных решений в сфере учета товаров [3, с.186].

Функциональные и нефункциональные требования. Функциональные и нефункциональные требования являются основной основой при разработке программного обеспечения или системы.

Функциональные требования определяют, что система должна делать и какие функции она должна выполнять. Они описывают, каким образом система должна вести себя в различных ситуациях и как она должна взаимодействовать с пользователями и другими системами.

Нефункциональные требования определяют, какими свойствами и характеристиками должна обладать система. Они описывают качество, производительность, надежность, безопасность и другие аспекты системы. Оба вида требований важны для того, чтобы система была разработана и реализована соответствующим образом. Функциональные требования определяют желаемую функциональность, а нефункциональные требования

обеспечивают ее качество и эффективность [4, с.232].

Функциональные требования:

1) Регистрация поступления товаров: Система должна обеспечить учет и регистрацию всех поступающих товаров на склад, включая данные о количестве, стоимости, поставщике, дате поступления и другие характеристики.

2) Учет отгрузки товаров: Система должна регистрировать отгрузку товаров клиентам с указанием количества, цены, даты отгрузки, а также информации о клиенте.

3) Формирование отчетов о движении товаров: Система должна генерировать отчеты о движении товаров, включая приход и расход товаров, предоставляя детализированную информацию об операциях.

4) Управление стоимостью товаров: Система должна вести учет стоимости товаров, автоматически пересчитывая их принимая во внимание количество.

Нефункциональные требования:

1) Безопасность данных: Система должна обеспечивать высокий уровень защиты данных о товарах, а также контроля доступа к информации и логгирования изменений.

2) Производительность: Система должна обеспечивать высокую производительность при обработке большого объема данных и генерации отчетов.

3) Надежность: Обеспечение непрерывной работы системы, минимизация риска сбоев и потери данных.

4) Эргономика интерфейса: Разработка удобного пользовательского интерфейса, позволяющего эффективно выполнять все операции с товарами.

5) Масштабируемость: Возможность расширения системы в случае увеличения числа пользователей или объема операций.

6) Доступность: Обеспечение доступности системы для пользователей в различных часовых поясах и географических распределений.

7) Аудит: Ведение журнала аудита для отслеживания и анализа действий пользователей в системе.

1.4 Архитектура информационной системы

Для разработки информационной системы для учета товаров существует несколько подходов к архитектуре, включая централизованные, распределенные,микросервисные и объектно-ориентированныеархитектуры. Каждый из них имеет свои преимущества и недостатки, и выбор архитектуры зависит от конкретных бизнес-потребностей и технических требований компании. Краткое описание каждой из них:

Централизованная архитектура:

Принцип работы: Централизованная система учета товаров предполагает наличие единой базы данных, к которой обращаются все модули и компоненты системы. Вся логика бизнес-процессов и функциональности находится в одном центральном месте.

Преимущества: Простота разработки, интеграции и управления. Однородная и последовательная обработка данных. Относительно простая масштабируемость.

Недостатки: Узкая шина данных может привести к проблемам производительности и масштабируемости в крупных системах. Один центральный узел - единственная точка отказа.

На этой схеме(рисунок 1.1) видно, что есть центральный сервер/компьютер, который является основным узлом архитектуры. Клиентские устройства подключаются к этому серверу через сеть или Интернет. Все команды и информация обрабатываются на сервере, а клиентские устройства служат только для отображения результатов и взаимодействия с сервером. Это обеспечивает более простую управляемость и контроль над системой, так как все данные и логика хранятся и исполняются на сервере [5, с.21].

Рисунок 1.1 - Схема централизованной архитектуры

Распределенная архитектура:

1) Принцип работы: Распределенная архитектура размещает компоненты системы на различных узлах или серверах, что позволяет обрабатывать данные параллельно и децентрализованно.

2) Преимущества: Улучшенная производительность и масштабируемость. Высокая отказоустойчивость, так как система продолжает работу при отказе отдельных узлов.

3) Недостатки: Сложность управления и согласования данных между различными узлами. Необходимость внешней компоновки и мониторинга системы.

В этой схеме(рисунок 1.2) клиентское приложение взаимодействует с сервисом через балансировщик нагрузки, который распределяет запросы на доступные серверы в распределенной системе. Серверное приложение 1 и

серверное приложение 2 обмениваются данными между собой и имеют общий доступ к базе данных [6, с.33].

Рисунок 1.2 - Схема распределенной архитектуры

Такая архитектура позволяет улучшить масштабируемость и отказоустойчивость системы, так как можно добавлять или удалять серверные приложения при необходимости, а также распределить нагрузку на несколько серверов.

Микросервисная архитектура:

– Принцип работы: Микросервисная архитектура разделяет систему на небольшие самостоятельные сервисы, каждый из которых обслуживает

конкретную функцию или процесс.

– Преимущества: Гибкость, масштабируемость и отказоустойчивость. Удобство внедрения новых функциональностей и поддержки. Легкое обновление и замена компонентов [17, с.273].

– Недостатки: Усложнение развертывания и мониторинга, а также увеличение нагрузки на сеть из-за взаимодействия между сервисами.

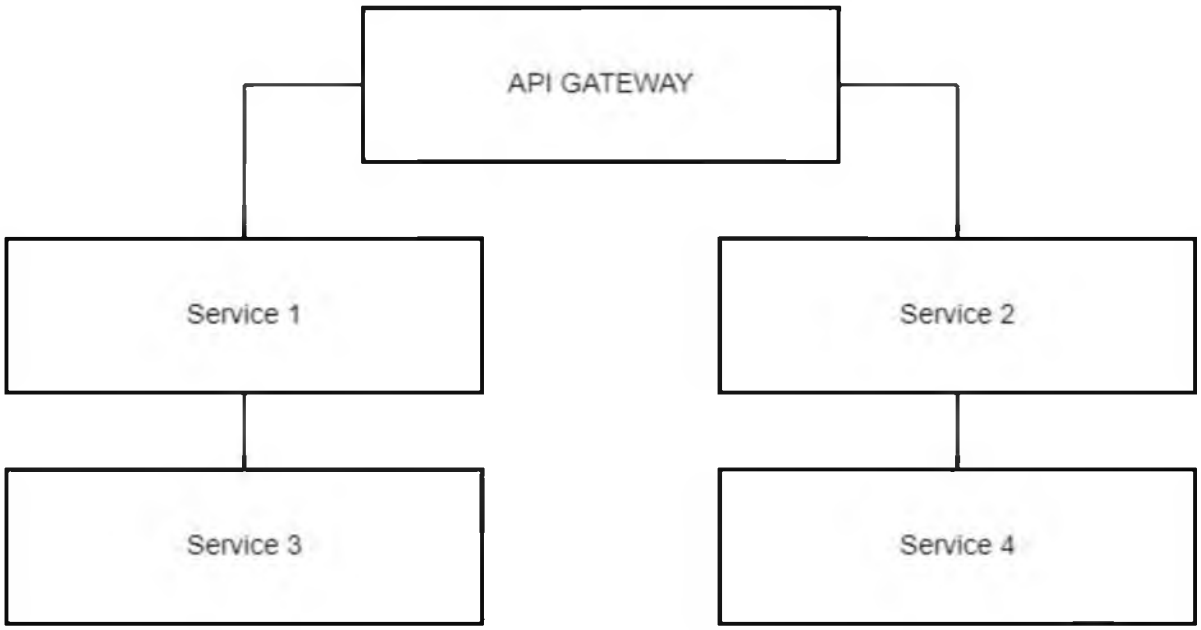


Рисунок 1.3 - Схема микросервисной архитектуры

В этой схеме(рисунок1.3) есть API Gateway, которая является центральным узлом и точкой входа для клиентов системы. API Gateway обрабатывает все внешние запросы и направляет их к соответствующим микросервисам.

Микросервисная архитектура состоит из нескольких сервисов каждый из которых отвечает за определенную функциональность системы. Каждый микросервис имеет свою собственную базу данных и отвечает только за свою область ответственности.

Микросервисы могут общаться между собой напрямую или через событийную шину, в зависимости от требований системы. Они могут быть развернуты на разных серверах или контейнерах для обеспечения

горизонтальной масштабируемости и независимого развертывания.

Это лишь пример схемы микросервисной архитектуры, и реальная архитектура может быть более сложной и включать дополнительные компоненты, такие как сервисы аутентификации, мониторинга, балансировки нагрузки и другие[7, с.199].

Объектно-ориентированная архитектура:

– Принцип работы: Принципы объектно-ориентированной архитектуры позволяют разделять программу на небольшие, независимые части (объекты), которые взаимодействуют друг с другом, образуя сложный функционал. Это позволяет сокращать сложность программы, повышать ее гибкость и повторно использовать код.

– Преимущества:

1)Повторное использование кода: ООА позволяет создавать классы и объекты, которые могут использоваться повторно в различных проектах. Это значительно упрощает процесс разработки и улучшает производительность.

2)Масштабируемость: ООА обеспечивает гибкость и масштабируемость разработки. Объекты можно легко расширять и изменять без воздействия на другие части системы.

3)Упрощение сопровождения: ООА упрощает сопровождение кода, так как различные компоненты системы могут быть разделены и модифицированы независимо друг от друга.

4)Улучшение модульности: ООА позволяет создавать модули, которые являются самодостаточными и могут взаимодействовать только с определенными объектами. Это способствует улучшению структуры программы[8, с.165].

– Недостатки:

1)Сложность: ООА может быть сложным для понимания и внедрения для некоторых разработчиков. Не всегда легко определить, какие объекты создавать и каким образом они должны взаимодействовать.

2)Избыточность: Иногда объекты могут быть избыточными и приводить к

увеличению сложности системы. Неправильное использование ООА может привести к созданию множества ненужных классов и объектов.

3)Производительность: В некоторых случаях ООА может приводить к увеличению накладных расходов на выполнение программы. Создание и управление большим количеством объектов может замедлить работу системы.

4)Сложность отладки: Ошибка в одном объекте может повлиять на работу других объектов, что делает отладку сложнее.

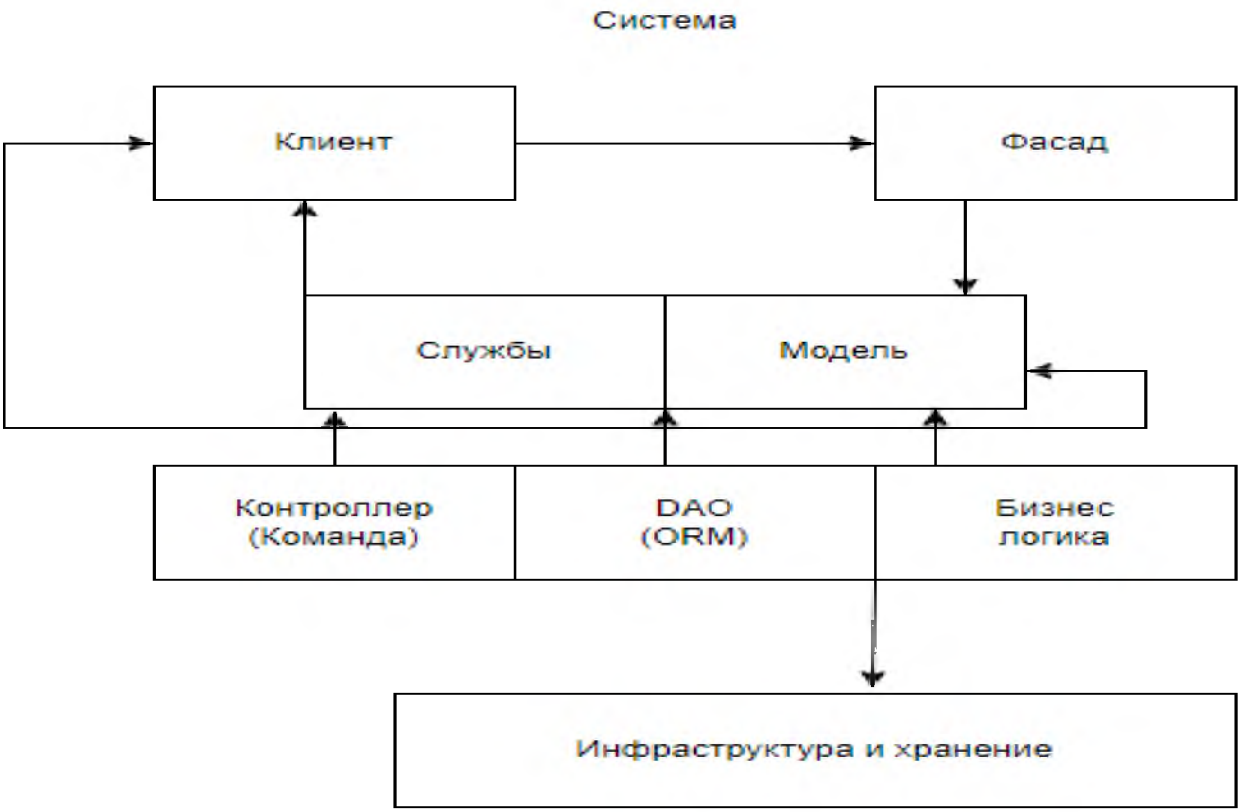


Рисунок 1.4 - Схема Объектно-ориентированной архитектуры

Схема (рисунок 1.4) включает:

1) Клиент: Внешний пользователя или система, которая взаимодействует с системой.

2) Фасад: Упрощает интерфейсы сложной подсистемы для внешних клиентов. Может работать как точка входа в систему, скрывая сложность и предоставляя простой интерфейс для клиентов.

3) Службы: Компоненты, которые предоставляют определенные функциональности (например, аутентификация, логирование и т. д.), могут

использоваться в различных частях системы [16, с.102].

4) Модель: Представляет бизнес-данные и правила, при помощи которых данные обрабатываются. Это может включать объекты данных и их поведение.

5)Контроллер (Команда): В MVC-архитектуре контроллер обрабатывает входящие запросы от клиента, изменяет модель и выбирает представление для ответа. В паттерне «Команда» контроллер может быть посредником для выполнения операций, инкапсулируя действие и его параметры.

6) DAO (DataAccessObject): Объект доступа к данным служит для абстракции и инкапсуляции всех доступов к источнику данных. DAO управляет соединением с базой данных для сохранения и загрузки данных.

7) Бизнес логика: Содержит бизнес-правила, которые определяют, как бизнес-данные создаются, хранятся и изменяются. Она является центральной частью программной системы.

8) Инфраструктура и хранение: Включает в себя все, что необходимо для поддержки разработки, развертывания, выполнения и поддержки бизнес-приложений. Это может включать базы данных, серверы, сетевые компоненты, системы мониторинга и др.

Объектно-ориентированная архитектура (ООА) является подходом к проектированию информационных систем, основанном на концепциях объектов, классов и их взаимодействия. В этом подходе система представляется в виде набора взаимосвязанных объектов, каждый из которых выполняет определенные функции и имеет свои свойства [9, с.30].

В ООА объекты являются основными строительными блоками системы. Каждый объект имеет свое состояние (переменные), поведение (методы) и связи с другими объектами. Объекты группируются в классы, которые определяют общие свойства и поведение для всех объектов этого класса.

Классы могут быть упорядочены в иерархию, где один класс наследует свойства и методы от другого класса.

Основные концепции, применяемые в ООА, включают наследование, полиморфизм, инкапсуляцию и абстракцию.

Наследование позволяет классам наследовать свойства и методы от других классов. Это позволяет повторно использовать код, упрощает структуру системы и способствует гибкости и расширяемости.

Полиморфизм позволяет объектам различных классов иметь одинаковое поведение. Например, несколько классов могут иметь метод с одним именем, но с различными реализациями.

Инкапсуляция позволяет объединить данные и методы внутри класса и скрыть их от внешнего доступа. Только публичные методы могут быть доступны для использования вне класса. Это способствует безопасности и защите данных.

Абстракция предоставляет способ представления сложных систем и концепций в более простых и удобных терминах. Абстрактные классы и интерфейсы определяют общие свойства и методы, но не предоставляют конкретную реализацию.

Потенциальное повторное использование классов и объектов позволяет реализовывать открытые интерфейсы, которые могут быть использованы в других системах и интегрированы с другими приложениями.

Выбор ООА для информационной системы обладает множеством преимуществ, включая модульность, гибкость, управление сложностью, масштабируемость, безопасность, повторное использование кода, упрощение разработки и поддержки системы. Эти преимущества содействуют эффективности, надежности и сопровождаемости системы на протяжении ее жизненного цикла [10, с.214].

Каждая из этих архитектур подходит для разных сценариев использования и имеет свои преимущества и недостатки. Выбор архитектуры должен быть обоснован потребностями бизнеса, технологическими требованиями и перспективами развития системы.

2 Обзор и сравнение инструментов

2.1 Обзор существующих инструментов учета товара

Инструменты учета товаров имеют центральное значение для эффективного управления запасами и процессом продажи товаров. Они предоставляют компаниям и предприятиям следующие преимущества:

1) Автоматизация учета товаров: Использование информационных систем для учета товаров позволяет автоматизировать процессы отслеживания, фиксации и контроля за движением товаров. Это позволяет компаниям сократить объем ручной работы, уменьшить вероятность ошибок и увеличить точность учета [15, с.163].

2) Улучшение планирования и управления запасами: Инструменты учета товаров предоставляют комплексную информацию о количестве, распределении и перемещении товаров внутри компании. Это позволяет лучше планировать потребности в запасах и оптимизировать процессы закупок, минимизируя излишние запасы или нехватку товаров.

3) Оптимизация процесса продажи: Учет товаров в информационной системе позволяет ускорить процесс продажи путем быстрого определения наличия товара на складе и его быстрого отображения в системе оформления заказов. Как результат, клиенты получают более точную и актуальную информацию о наличии товаров и их доставке.

4) Анализ и отчетность: Информационные системы учета товаров предоставляют возможность анализировать данные о продажах, запасах и потребительском поведении. Это позволяет компаниям получать полезные отчеты и аналитическую информацию для принятия решений, планирования, оптимизации бизнес-процессов и улучшения эффективности.

5) Интеграция со смежными системами: Инструменты учета товаров часто интегрируются с другими системами в компании, такими как системы управления складом, системы планирования ресурсов предприятия (ERP) и системы продажи (POS). Это позволяет обеспечить единый и согласованный

поток данных между различными функциональными областями компании.

6) Безопасность и контроль: Инструменты учета товаров также предоставляют контрольные механизмы для предотвращения краж и несанкционированного доступа к товарам. Такие механизмы включают в себя маркировку товаров, системы видеонаблюдения, а также аудит и отслеживание действий пользователей [14, с.351].

В целом, инструменты учета товаров в информационных системах играют важную роль в эффективном управлении запасами, оптимизации процесса продажи и принятии рациональных бизнес-решений. Они помогают компаниям повысить эффективность, улучшить обслуживание клиентов и обеспечить контроль и безопасность товаров внутри организации.

Таблица – 2.1 Сравнение систем

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

Таблица 2.1 дает обзор основных характеристик и преимуществ каждой из систем учета товара. Выбор конкретной системы будет зависеть от специфики бизнеса, его потребностей и бюджета.

Таблица – 2.2 Описание систем

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.

POS-системы (системы точек продаж):

POS-системы представляют собой программное обеспечение, специально разработанное для учета продаж и обработки транзакций в розничных магазинах. Они обеспечивают функционал для регистрации продаж, подсчета денег, управления инвентаризацией, создания чеков и прочих кассовых операций. Они могут также интегрироваться с другими системами, такими как

бухгалтерия, управление ресурсами компании (CRM), учет запасов и т.д.

ERP-системы (системы управления предприятием):

ERP-системы представляют собой программные комплексы, предназначенные для автоматизации и управления различными бизнес-процессами, включая учет товара, финансов, закупок, производства, продаж и других. Они обычно включают в себя модули, такие как учет товара, управление запасами, финансы, управление отношениями с клиентами (CRM) и другие.

WMS-системы (системы управления складом):

WMS-системы предназначены для управления складскими операциями, включая принятие и отгрузку товаров, инвентаризацию, планирование, оптимизацию и отчетность. Они обеспечивают контроль над перемещением товаров на складе, управление распределением товаров по местам хранения, отслеживание сроков годности и другие функции, связанные с управлением запасами на складе.

SCM-системы (системы управления цепочкой поставок):

SCM-системы предназначены для управления всей цепочкой поставок, что включает учет товара на различных этапах поставки, планирование спроса, управление логистикой, управление запасами, сотрудничество с поставщиками и другие функции, связанные с оптимизацией процессов поставок.

E-commerce платформы:

E-commerce платформы предоставляют возможности для создания и управления интернет-магазинами. Они включают функционал управления каталогом товаров, оформления заказов, учета инвентаризации, интегрированные платежные системы, маркетинговые инструменты, аналитику и другие инструменты, специализированные для интернет-торговли.

Каждый из этих инструментов (таблица 2.2) обладает своими характеристиками, преимуществами и недостатками, и выбор конкретного инструмента учета товара зависит от конкретных требований и потребностей бизнеса [11, с.105].

2.2 SWOT анализ

SWOT-анализ (анализ сильных сторон, слабых сторон, возможностей и угроз) может помочь в оценке текущего состояния и перспектив учета товаров. SWOT-анализа для учета товаров:

Рисунок 2.1 - SWOTанализ для учёта товаров на складе

Анализ этих четырех аспектов (рисунок 2.1) помогает компании выявить, что она делает хорошо или плохо, какие возможности у нее есть для роста, и какие угрозы могут возникнуть в будущем. Это позволяет разработать стратегии для усиления сильных сторон, устранения слабых, использования возможностей и защиты от угроз. SWOT-анализ является важным

инструментом для стратегического планирования и принятия обоснованных решений. Основные плюсы:

1) Оценка внутренней среды: SWOT-анализ позволяет оценить внутреннюю среду компании, выявив ее сильные и слабые стороны. Это включает понимание бизнес-процессов, структуры управления, культуры организации, финансовых ресурсов и других аспектов, которые могут влиять на ее конкурентоспособность.

2) Выявление внешних факторов: SWOT-анализ также помогает определить внешние возможности и угрозы, включая изменения в законодательстве, экономические изменения, конкуренцию, технологические инновации и изменения в потребительском спросе [13, с.95].

3) Выработка стратегии: Используя результаты SWOT-анализа, компании могут разработать стратегии, которые позволят им максимально использовать свои сильные стороны, минимизировать слабые стороны, воспользоваться возможностями и защититься от угроз.

4) Принятие обоснованных решений: SWOT-анализ обеспечивает информацию и понимание, необходимые для принятия обоснованных решений о том, как компания может улучшить свою позицию на рынке, адаптироваться к переменам и реагировать на конкуренцию.

5) Управление рисками: Анализ угроз позволяет руководству определить, какие угрозы могут повлиять на бизнес, и создать стратегии для уменьшения рисков и преодоления проблем, связанных с ними.

Таким образом, SWOT-анализ является мощным инструментом для помощи компаниям в понимании своего положения на рынке, планировании стратегий и принятии обоснованных решений.

2.3 Обзор подходящих языков программирования

При выборе языка программирования для разработки информационной системы необходимо учитывать множество факторов, включая тип системы,

требования проекта, предпочтения разработчиков и многое другое. Ниже приведены некоторые из наиболее популярных языков программирования, которые могут быть подходящими для создания информационной системы:

Java: Java - это объектно-ориентированный, класс-ориентированный язык программирования, который разработан компанией SunMicrosystems (после приобретения - Oracle). Его основные характеристики:

1) Платформенная независимость: Код, написанный на Java, компилируется в промежуточный байт-код, который может выполняться на любой виртуальной машине Java (JVM), что делает Java платформенно-независимой.

2) Объектно-ориентированный: Java полностью поддерживает парадигму объектно-ориентированного программирования, включая абстракцию, наследование, полиморфизм и инкапсуляцию.

3) Простой и понятный синтаксис: Синтаксис Java схож с синтаксисом языков C и C++, что делает его относительно легким для изучения и использования.

4) Автоматическое управление памятью: Java предоставляет абстракцию управления памятью через механизм сборки мусора, который автоматически удаляет неиспользуемые объекты из памяти.

5) Многопоточность: Java предоставляет встроенную поддержку многопоточности, что позволяет разрабатывать многопоточные приложения и управлять потоками выполнения.

6) Большая стандартная библиотека: Java поставляется с обширной стандартной библиотекой, которая включает в себя различные классы и методы для решения разнообразных задач, таких как работа с файлами, сетевое взаимодействие, обработка данных и многие другие.

7) Потоко-ориентированный: Java обладает встроенной поддержкой потоков ввода-вывода (I/O), что облегчает обработку файлов и сетевое взаимодействие.

8) Платформа для веб-приложений: С помощью технологий таких как

JavaEnterpriseEdition (Java EE), JavaServerPages (JSP), и сервлетов, Java часто используется для создания масштабируемых веб-приложений и сервисов.

9) Инструменты разработки: Существует широкий выбор интегрированных сред разработки (IDE), таких как Eclipse, IntelliJ IDEA, NetBeans, которые облегчают создание, отладку и тестирование приложений на Java.

JavaScript: JavaScript - это высокоуровневый интерпретируемый язык программирования, который широко используется для создания динамических интерактивных веб-страниц. Ниже приведены основные характеристики и функции JavaScript:

1) Интерпретируемый язык программирования: JavaScript интерпретируется прямо во время выполнения, что позволяет создавать динамические изменения на веб-страницах без необходимости повторной компиляции.

2) Объектно-ориентированный: JavaScript основан на объектной модели, что позволяет создавать объекты, их методы и свойства, а также наследование и полиморфизм.

3) Интеграция с HTML/CSS: JavaScript эффективно интегрируется с HTML и CSS, обеспечивая динамическое изменение содержимого и стилей веб-страниц.

4) Событийно-ориентированный: JavaScript позволяет обрабатывать события веб-страницы, такие как клики, наведение курсора, отправка форм, что делает взаимодействие пользователя с веб-страницей более динамичным.

5) Асинхронное программирование: JavaScript обеспечивает возможности для асинхронного программирования, включая работы с коллбэками, промисами и асинхронными функциями, что позволяет обрабатывать операции ввода-вывода без блокировки основного потока выполнения.

6)Браузерное выполнение: JavaScript выполняется в браузере пользователя, что позволяет создавать интерактивные пользовательские интерфейсы и обеспечивать более динамичный опыт использования веб-сайтов.

7) Поддержка серверных технологий: JavaScript также может быть использован на серверной стороне с использованием платформы Node.js для создания масштабируемых и быстрых серверных приложений.

8) Богатая стандартная библиотека: JavaScript поставляется с обширной стандартной библиотекой, содержащей различные методы, объекты и функции для работы с DOM, обработки событий, работы с массивами, строками, временем и другими операциями.

9) Динамическая типизация: JavaScript является языком с динамической типизацией, что означает, что не требуется явного объявления типов переменных, что обеспечивает гибкость и упрощает разработку.

JavaScript является ключевым компонентом для разработки веб-приложений и дает разработчикам мощный инструмент для создания динамических и интерактивных пользовательских интерфейсов.

Java и JavaScript - это два разных языка программирования с разными назначениями и характеристиками. Основные отличия между ними:

1) Назначение:

Java: Java часто используется для создания серверных приложений, мобильных приложений (Android), настольных приложений и больших корпоративных приложений.

JavaScript: JavaScript используется веб-разработкой для создания динамических интерактивных веб-страниц, клиентских веб-приложений и в последнее время также серверной разработки с использованием Node.js.

2) Техническая реализация:

Java: Java является статически типизированным языком программирования, выполняемым на виртуальной машине Java (JVM).

JavaScript: JavaScript является динамически типизированным языком программирования, интерпретируемым в браузере.

4) Синтаксис и особенности:

Java: Java использует сильную статическую типизацию и объектно-ориентированный подход. Он предоставляет строгие правила для объявления

переменных и классов.

JavaScript: JavaScript, с другой стороны, использует слабую динамическую типизацию и объектно-ориентированный стиль программирования. Он также поддерживает функциональное программирование.

5) Использование:

Java: Java часто используется для создания более крупных приложений, требующих высокой надежности и производительности, таких как серверные приложения или приложения для мобильных устройств.

JavaScript: JavaScript широко применяется во веб-разработке для создания динамических элементов на веб-страницах, формы обратной связи, клиентской валидации и других задач, связанных с интерактивной частью веб-приложений.

Хотя у Java и JavaScript есть общие черты (например, синтаксический сахар), они предназначены для разных сфер применения и обладают различными особенностями и характеристиками.

C#: Язык программирования C# - это мощный и универсальный язык программирования, разработанный корпорацией Microsoft. Он предназначен для создания различных типов приложений, от настольных программ до веб-приложений и приложений для мобильных устройств. Вот несколько ключевых характеристик и возможностей C#:

Основные характеристики:

1) Объектно-ориентированный: C# полностью объектно-ориентированный язык программирования, который поддерживает основные концепции ООП, такие как наследование, полиморфизм, инкапсуляцию и абстракцию.

2) Многозадачность и асинхронное программирование: C# имеет встроенную поддержку многопоточности и асинхронного программирования, что позволяет создавать многопоточные приложения и обрабатывать асинхронные операции.

3) Платформенно-независимый: C# в первую очередь ассоциируется с

платформой .NET, которая обеспечивает кроссплатформенную совместимость для выполнения приложений на различных операционных системах.

4) Богатая стандартная библиотека: C# поставляется с обширной стандартной библиотекой, содержащей широкий спектр классов и методов для создания разнообразных типов приложений.

5) Интеграция с Windows: C# тесно интегрирован с платформой Windows, позволяя создавать настольные приложения, использующие функциональность операционной системы.

6) Современные особенности: C# постоянно обновляется, вводя новые особенности и инструменты. Например, введение асинхронных методов, анонимных типов, LINQ (LanguageIntegratedQuery) и т. д.

7) Использование LINQ: C# предоставляет возможность использовать язык интегрированных запросов (LINQ) для упрощения обращения к источникам данных и их обработки.

C# широко используется для разработки настольных приложений под платформу Windows, веб-приложений, веб-служб, игр и мобильных приложений для платформы WindowsPhone.

Он также может быть использован для создания программного обеспечения для встраиваемых систем, как часть технологий, связанных с платформой .NET, и для разработки серверных приложений.

C# является интегральной частью технологий, связанных с платформой .NET, таких как ASP.NET для веб-разработки, WinForms и WPF для настольных приложений, Xamarin для мобильной разработки и других технологий.

Этот язык представляет собой мощный инструмент для разработки различных типов приложений и остается популярным среди разработчиков и организаций, благодаря своей универсальности и обширному экосистеме инструментов и технологий.

PHP: PHP (HypertextPreprocessor) - это скриптовый язык программирования общего назначения, который в основном используется для разработки веб-приложений и динамических веб-страниц. Вот несколько ключевых

характеристик и особенностей PHP:

1) Интеграция с HTML: PHP может быть встроен в HTML-страницы, что позволяет создавать динамические веб-страницы, где PHP-скрипты выполняются на сервере перед тем, как страница отправляется браузеру пользователя.

2) Базы данных: PHP обладает мощной поддержкой для работы с различными системами управления базами данных (например, MySQL, PostgreSQL, SQLite), что позволяет взаимодействовать с базами данных для создания динамических сайтов и веб-приложений.

3) Многопоточность: PHP позволяет создавать многопоточные приложения, хотя его модель выполнения, обычно, предпочтительно основывать на модели с однопоточным рабочим процессом и многопоточным распределением запросов, например, с использованием сервера Apache.

4) Широкое использование: PHP широко используется для разработки веб-приложения, динамических сайтов, а также интеграции с другими технологиями веб-разработки, такими как HTML, CSS, JavaScript и т.д.

5) Большое сообщество и поддержка: PHP имеет большое сообщество разработчиков и изобилие библиотек и фреймворков, что делает его привлекательным для разработчиков и обеспечивает доступ к множеству руководств, документации и примеров кода.

6) Открытый и бесплатный: PHP является открытым и бесплатным программным обеспечением, что делает его доступным для широкого круга разработчиков.

7) Встраиваемость: PHP можно встраивать в различные сторонние технологии и серверы - это язык серверной стороны, который может использоваться с Apache, Nginx, IIS и другими веб-серверами.

8) Поддержка различных протоколов: PHP обладает функциональностью для работы с различными сетевыми протоколами, такими как HTTP, FTP, IMAP, и т.д.

PHP продолжает оставаться одним из наиболее популярных языков

программирования для веб-разработки, благодаря мощным возможностям в создании динамических сайтов, широкой поддержке и большому сообществу разработчиков[12, с.141].

Основания выбора Python. Python - мощный и высокоуровневый интерпретируемый язык программирования, который применяется во многих областях, от веб-разработки до анализа данных и машинного обучения. Вот более подробная информация о Python:

1) Читаемый и понятный синтаксис: Python славится своим простым и понятным синтаксисом, который делает код легко читаемым, что особенно полезно для начинающих программистов.

2) Кроссплатформенность: Python поддерживает работу на различных операционных системах, включая Windows, macOS и Linux, что делает его универсальным инструментом для разработки кроссплатформенных приложений.

3) Большое сообщество и поддержка: Python имеет огромное и активное сообщество разработчиков, что обеспечивает обширную документацию, богатый выбор библиотек и обучающие ресурсы.

4) Множество библиотек и фреймворков: Python обладает обширным экосистемой библиотек и фреймворков для широкого спектра задач, включая веб-разработку (Django, Flask), анализ данных (pandas, NumPy), машинное обучение (TensorFlow, PyTorch), научные вычисления, автоматизацию и другие области.

5) Машинное обучение и анализ данных: Python является одним из основных языков программирования в области анализа данных и машинного обучения, благодаря мощным библиотекам, таким как TensorFlow, PyTorch, scikit-learn, pandas и другим.

6) Простота расширения: Python легко интегрируется с кодом, написанным на других языках программирования, что делает его универсальным инструментом для создания сложных систем и приложений.

7) Широкое применение: Python используется для создания веб-

приложений, настольных приложений, игр, автоматизации задач, научных исследований, анализа данных, машинного обучения и для решения многих других задач.

Python остается одним из самых популярных языков программирования благодаря своей мощности, гибкости, поддержке и удобству использования.

Использование языка программирования Python охватывает широкий спектр областей и сфер деятельности, от веб-разработки и анализа данных до машинного обучения, научных вычислений и многое другое. Python остается популярным в силу своей простоты, гибкости, мощных библиотек и обширной экосистемы.

Python используется для создания веб-приложений и веб-сайтов, анализа данных, научных исследований, разработки машинного обучения и искусственного интеллекта, скриптинга и автоматизации задач, а также для игровой разработки и создания настольных приложений. Его простой и понятный синтаксис, обширные библиотеки и мультипарадигменный подход делают Python универсальным инструментом для разработки программного обеспечения в различных областях.

Благодаря своему многообразию и обширным возможностям, Python продолжает оставаться одним из наиболее популярных языков программирования и оставляет глубокий след в области разработки программного обеспечения.

Наряду с активным сообществом и подкреплением фундаментальных принципов, Python остается лидером в создании программных решений для разнообразных отраслей, и его влияние остается значительным в различных областях технологий [18, с.88].

3 Разработка приложения

3.1 Разработка базы данных

База данных в SQL (StructuredQueryLanguage) - это структурированная совокупность данных, организованных в таблицы, которые взаимодействуют друг с другом с помощью языка запросов SQL [28].

База данных в SQL состоит из таблиц, каждая из которых имеет свое имя и структуру. Таблицы состоят из рядов (строк) и столбцов (рисунок 3.1). Строки представляют отдельные элементы данных, а столбцы определяют тип данных, который содержится в каждой ячейке таблицы.

В SQL определены основные операции, позволяющие работать с базой данных:

1) Создание таблиц - операция, которая позволяет создать новую таблицу, задать ее структуру (имена и типы столбцов) и связи с другими таблицами.

2) Вставка данных - операция, которая позволяет добавить новые строки данных в таблицу.

3) Обновление данных - операция, которая позволяет изменить существующие данные в таблице.

4) Удаление данных - операция, которая позволяет удалить данные из таблицы.

5) Выборка данных - операция, которая позволяет извлечь данные из таблицы с использованием различных условий и фильтров.

6) Слияние данных - операция, которая позволяет объединить данные из нескольких таблиц в один результат.

База данных в SQL также поддерживает концепцию отношений между таблицами с помощью операций объединения (JOIN), которые позволяют объединять данные из разных таблиц на основе их общих столбцов.

SQL предоставляет мощный инструментарий для работы с базой данных, позволяя создавать, изменять, извлекать и управлять данными с помощью простого и понятного языка запросов.

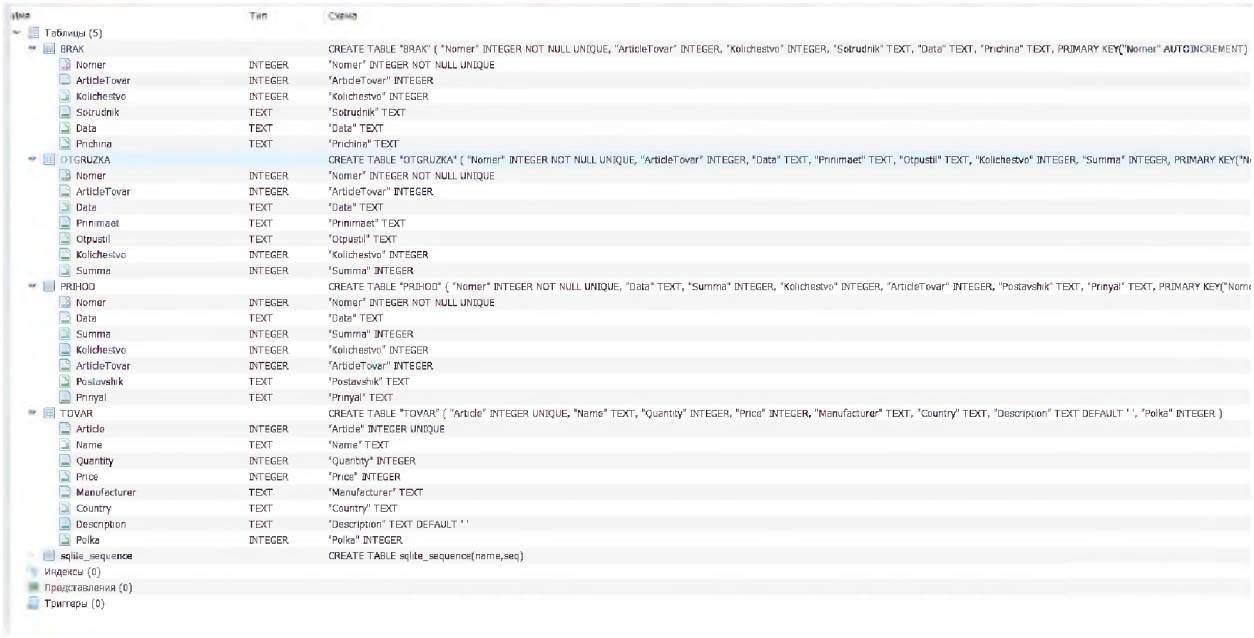
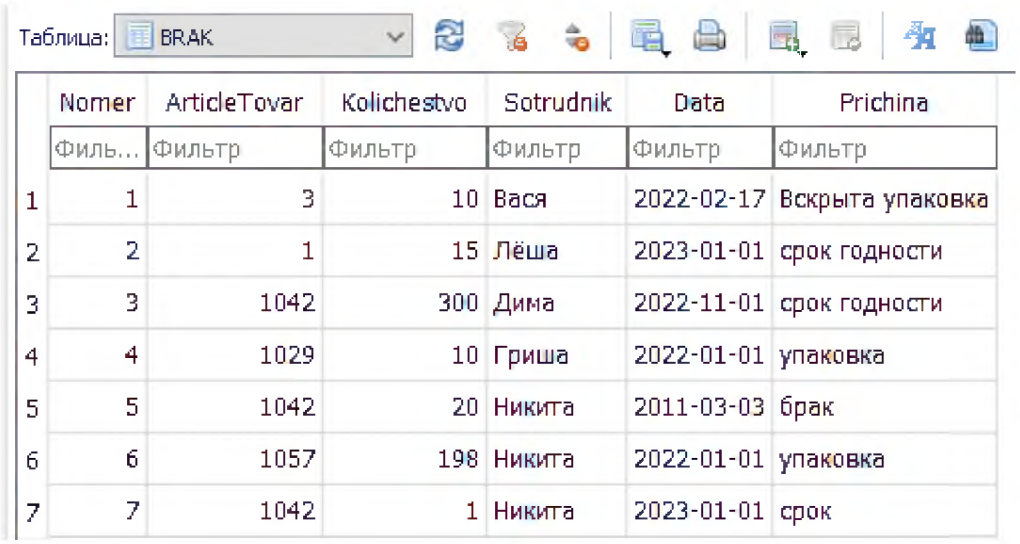


Рисунок 3.1– Общая картина базы данных



	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»

Таблица «BRAK» (рисунок 3.2) представляет собой структуру для хранения информации о браке товара в компании. Она включает следующие поля:

«Nomer» (INTEGER): Уникальный номер записи о браке.

«ArticleTovar» (INTEGER): Номер артикула бракованного товара.

«Kolichestvo» (INTEGER): Количество бракованного товара.

«Sotrudnik» (TEXT): ФИО сотрудника, который обнаружил брак.

«Data» (TEXT): Дата, когда был обнаружен брак.

«Prichine» (TEXT): Описание причин брака.

«Nomer» задан как INTEGER и определен как NOT NULL и UNIQUE, что означает, что каждая запись в таблице будет иметь уникальный номер и этот номер не может быть неопределен или дублирован.

«ArticleTovar» и «Kolichestvo» также заданы как INTEGER, предполагая, что артикул и количество бракованного товара будут представлены числами.

«Sotrudnik» и «Prichine» определены как TEXT, что позволяет хранить текстовую информацию, такую как ФИО сотрудника и описание причин брака.

«Data» также определена как TEXT, предполагая хранение даты в текстовом формате.

Ключевое слово «AUTOINCREMENT» в PRIMARY KEY(«Nomer» AUTOINCREMENT) указывает на автоматическое инкрементирование значения поля «Nomer» при вставке новой записи, гарантируя уникальность номеров.

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»

Таблица «OTGRUZKA»(рисунок 3.3) представляет собой хранилище информации о грузах, отправленных и принятых. Она содержит следующие столбцы:

«Nomer» (INTEGER): уникальный номер груза, не может быть пустым;

«ArticleTovar» (INTEGER): номер статьи вещества, связанного с грузом;

«Data» (TEXT): дата исполнения груза;

«Prinimaet» (TEXT): имя получателя груза;

«Otpustil» (TEXT): имя отправителя груза;

«Kolichestvo» (INTEGER): количество груза;

«Summa» (INTEGER): сумма груза;

Данная таблица предназначена для хранения информации о грузах, таких как номер груза, дата исполнения, описание предпросмотра и имена отправителя и получателя груза. Количество и сумма груза также записываются в таблицу [27].

Столбец «Nomer» является основным ключом таблицы, что означает, что он должен быть уникальным для каждой записи. Чтобы гарантировать уникальность, он также имеет ограничение «UNIQUE», которое не позволяет иметь одинаковые значения в этом столбце.

Ограничение «NOT NULL» указывает, что значение должно быть обязательно заполнено для каждой записи в столбце «Nomer».

Все остальные столбцы определены с определенными типами данных. «INTEGER» используется для числовых значений, таких как номер статьи, количество груза и сумма, а «TEXT» используется для строковых значений, таких как дата исполнения, описание предпросмотра, имя отправителя и получателя груза.

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»

Таблица «PRIKHOD»(рисунок 3.4) представляет собой таблицу базы данных, содержащую информацию о приходе товаров. Таблица содержит следующие столбцы:

«Nomer» - столбец уникальных идентификаторов прихода товара. NOT NULL означает, что каждая запись должна иметь значение в этом столбце, и значение должно быть уникальным.

«Data» - столбец, в котором содержится информация о дате прихода товара.

«Summa» - столбец для хранения суммы стоимости прихода товара.

«Kolichestvo» - столбец, содержащий информацию о количестве пришедшего товара.

«ArticleTovar» - столбец, в котором хранится идентификатор товара.

«Postavschik» - столбец, содержащий информацию о поставщике товара.

«Prinyal» - столбец, в котором хранится информация о том, кто принял товар.

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»

Таблица «TOVAR» (рисунок 3.5) представляет собой базу данных, предназначенную для хранения информации о товарах. В таблице присутствуют следующие поля:

«Article» - целочисленное поле, уникальное для каждого товара, служит в качестве идентификатора товара.

«Name» - текстовое поле, содержащее название товара.

«Quantity» - целочисленное поле, указывающее количество товара на складе.

«Price» - целочисленное поле, представляющее цену товара.

«Manufacturer» - текстовое поле, содержащее информацию о производителе товара.

«Country» - текстовое поле, указывающее страну происхождения товара.

«Description» - текстовое поле, содержащее описание товара. Поле имеет значение по умолчанию равное пустой строке.

«Polka» - целочисленное поле, представляющее информацию о полке, на которой находится товар.

Поле «Article» объявлено как UNIQUE, что означает, что каждое значение этого поля должно быть уникальным для каждой записи в таблице.

Поле «Description» имеет значение по умолчанию в виде пустой строки. Это означает, что если при вставке новой записи не указано значение для поля «Description», то оно будет автоматически установлено в пустую строку.

Поле «Polka» представляет информацию о том, на какой полке находится товар. Тип данных INTEGER используется для хранения числовых значений, которые могут быть использованы для идентификации полки внутри системы складского хранения.

Таким образом, таблица «TOVAR» предоставляет структуру для хранения и управления информацией о товарах, включая их артикул, название, количество, цену, производителя, страну происхождения, описание и место хранения на складе.

3.2 Разработка информационной системы

```
# 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()
```

Рисунок 3.6 - Функция аутентификации

Функция authentification() (рисунок 3.6) 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() (рисунок 3.7) 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.

```
# 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()
```

Рисунок 3.7 – Основное окно

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».

```
#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)
```

Рисунок 3.8 – основные функции

Функция info_tovar() (рисунок 3.8) 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.

```
# 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()
```

Рисунок 3.9 - Функция обновления

Функция refresh() (рисунок 3.9) 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.

```
#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)
```

Рисунок 3.10 - Функция поиска товара

Функция find_tovar() (рисунок 3.10) 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() (рисунок 3.11) 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.

```
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="Внёс")
            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()
```

Рисунок 3.11 – Отчёт и функции для него

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.

```
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 | Отчёт")
    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))
    dates1 = cursor.fetchall()
    for date1 in dates1:
        tree_otgruzka.insert("", END, values=date1)

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

Рисунок 3.12 – Отчёт отгрузки

Данный код (рисунок 3.12) 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.

```
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")

        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("Поставка | Отчёт")
    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()
```

Рисунок 3.13 – Отчёт поставок

Код postavochka() (рисунок 3.13) 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.

```
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")
```

Рисунок 3.14 -Логика принятия поставки

Данный код (рисунок 3.14) 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.

Рисунок 3.15 - Функция отгрузки

Функция otgruzit() (рисунок 3.15) 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() (рисунок 3.16) 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].

```
def vrychnyy():
    window_vnesti = Tk()
    window_vnesti.title("Внести новый товар вручную")
    window_vnesti.geometry("490x300")

    def vnesti_vrychnyy():
        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)
```

Рисунок3.16 – Функция ручного ввода

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.

```
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()
```

Рисунок 3.17 – Перемещение товара

Данный код (рисунок 3.17) 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.

```
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():
```

Рисунок 3.18 – Функция изменения цен

Функция tsena() (рисунок 3.18) 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.

```
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')
```

Рисунок 3.19 – Использование виджетаTreeview

В данном коде (рисунок 3.19) 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.

```
# 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()
```

Рисунок 3.20 –Графическое окно авторизации

Приведенный код (рисунок 3.20) создает графическое окно авторизации (authentication_window) с использованием библиотеки Tkinter. Окно имеет размер 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.

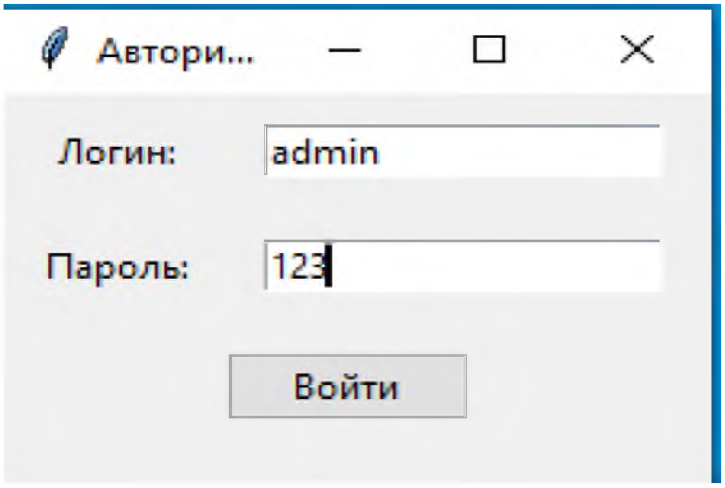


Рисунок 3.21 – Графическое окно «Окно авторизации»

Графическое окно авторизации (рисунок 3.21) - это компонент интерфейса, который предоставляет пользователю возможность ввести свои учетные данные для доступа к определенному приложению, сайту или системе. Обычно окно авторизации содержит поля для ввода логина и пароля, а также кнопку для входа.

Графическое окно авторизации служит для обеспечения безопасности и ограничения доступа к контенту или функциям определенной системы. Когда пользователь пытается получить доступ к приложению или сайту, окно авторизации появляется перед ним.

В окне авторизации пользователь должен ввести свой логин и пароль, которые заранее были зарегистрированы в системе. Логин (иногда называемый также именем пользователя) обычно является уникальным идентификатором пользователя, а пароль - секретной комбинацией символов, используемой для проверки подлинности пользователя[22].

После нажатия кнопки «Войти» система проверяет введенные данные. Если логин и пароль совпадают с данными, хранящимися в системе, пользователь получает доступ к приложению или сайту. В противном случае система может выдать сообщение об ошибке, указав неверный логин или пароль.

Графическое окно авторизации - это важный элемент пользовательского интерфейса, который помогает обеспечить безопасность и контроль доступа к системе, защищая конфиденциальные данные и предотвращая несанкционированный доступ.

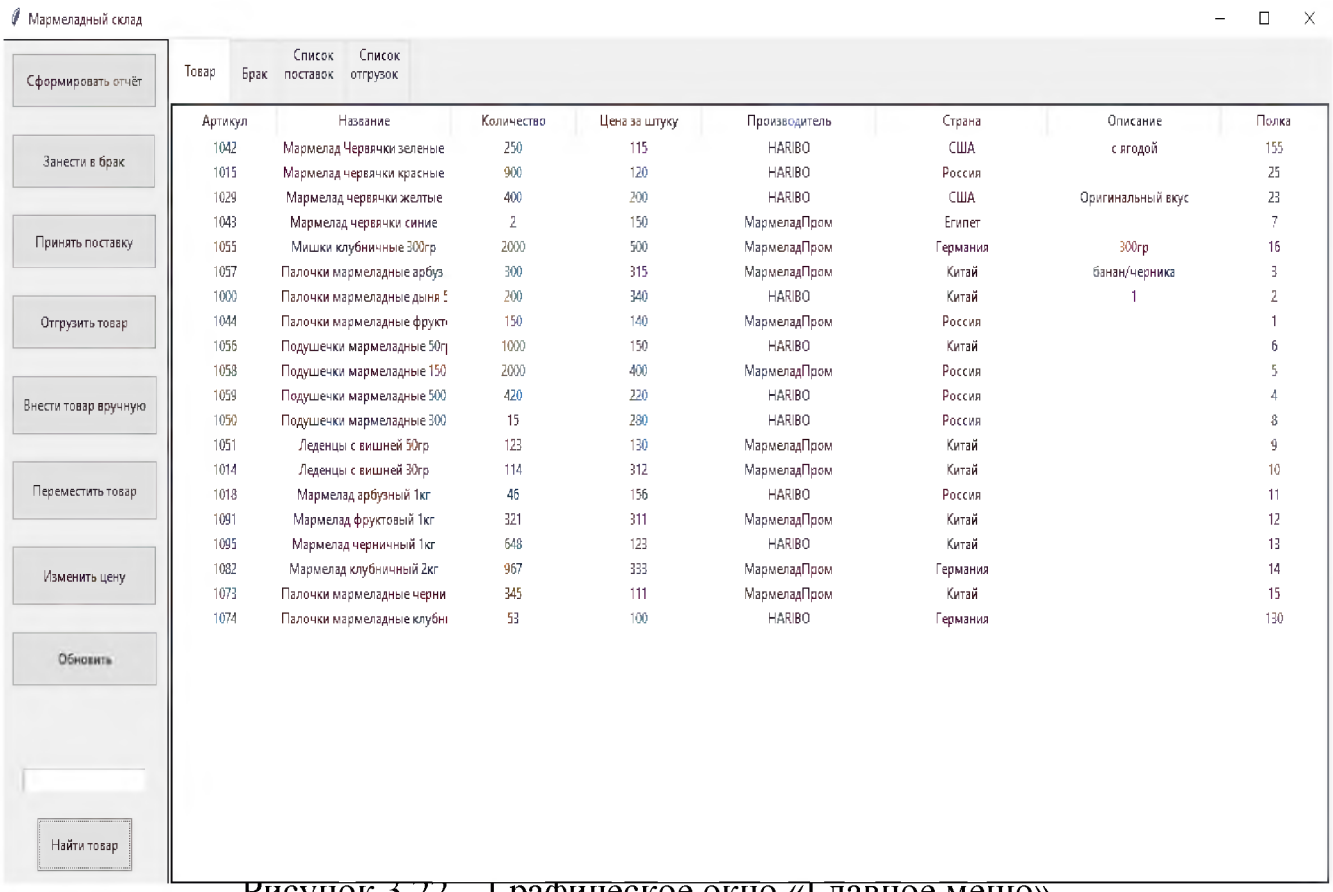


Рисунок 3.22 – Графическое окно «Главное меню»

Главное меню (рисунок 3.22) представляет собой интерфейс программы для учета товаров на складе. Оно содержит различные кнопки и поля, которые позволяют пользователю выполнять операции, связанные с учетом и управлением товаром на складе.

Кнопка - «Сформировать отчёт»:

Эта функция позволяет пользователю создавать различные виды отчетов.

Кнопка - «Занести в брак»:

Используется для фиксации некачественных или поврежденных товаров.

Кнопка - «Принять поставку»:

С этой функцией пользователь может вносить информацию о новых товарах, поступающих на склад, обновляя общий запас.

Кнопка - «Отгрузить товар»:

Позволяет учесть товары, которые были отправлены со склада, уменьшая их количество в общем учете.

Кнопка - «Внести товар вручную»:

Используется для добавления информации о товарах вручную.

Кнопка - «Переместить товар»:

Эта функция позволяет менять местоположение товаров на складе, обновляя данные о положении товара для эффективного занятия места и быстрого нахождения.

Кнопка - «Изменить цену»:

С помощью этой функции можно обновлять цены на товары в связи с изменением рыночных условий.

Кнопка - «Обновить»:

Это функция для обновления текущего вида или данных на экране, чтобы отобразить последние изменения в базе данных.

Поле для ввода - «поиск информации»:

Это текстовое поле позволяет вводить ключевые слова или фразы для поиска информации в базе данных товаров.

Названия столбцов определяют данные, которые будут отображаться в таблице складского учета:

«Артикул»: Уникальный идентификатор товара.

«Название»: Официальное наименование товара.

«Количество»: Текущий запас товара на складе.

«Цена за штуку»: Розничная цена одной единицы товара.

«Производитель»: Компания, производящая товар.

«Страна»: Страна, в которой товар был изготовлен.

«Описание»: Дополнительная информация о товаре, такая как характеристики, особенности или инструкции по использованию.

«Полка»: Место расположения товара в складском помещении для упрощения процесса нахождения и инвентаризации.

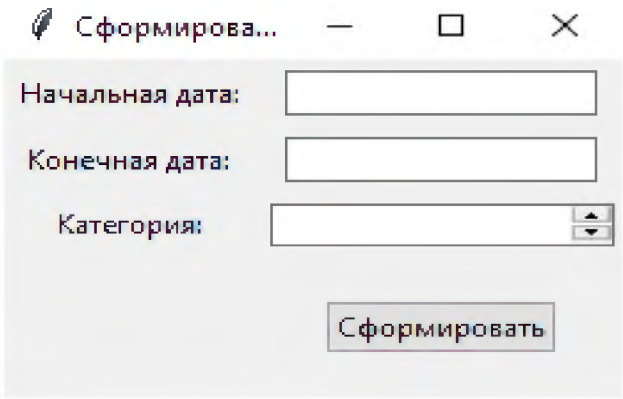


Рисунок 3.23 – Графическое окно «Сформировать отчёт»

Графическое окно «Сформировать отчёт»(рисунок 3.23) представляет собой интерфейс, который предоставляет пользователям возможность создавать отчёты с определёнными параметрами.

В окне есть три поля для ввода информации. Первое поле - «Начальная дата», предназначено для указания начальной даты, с которой будет происходить анализ данных для отчёта. Второе поле - «Конечная дата», используется для указания конечной даты, на которую ограничивается анализ данных в отчёте. Таким образом, отчёт будет включать только данные, собранные в указанный временной промежуток [29].

Третье поленазывается «Категория», и представлено в виде списка. В этом поле пользователь должен выбрать одну из трёх опций: «Поставка», «Отгрузка» или «Брак». Категория определяет, какая именно информация будет учтена в отчёте. Например, если выбрана категория «Поставка», отчёт будет содержать данные о всех поставках, сделанных в указанном временном промежутке.

После заполнения всех трёх полей, пользователь может нажать кнопку «Сформировать». Эта кнопка запускает процесс создания отчёта на основе заданных параметров. Отчёт может быть показан на экране или сохранён в файле [24].

Таким образом, графическое окно «Сформировать отчёт» предоставляет пользователю удобный инструмент для создания отчётов с различными параметрами и категориями данных. Это помогает пользователям

анализировать и оценивать нужную информацию, собранную в определённый период времени.

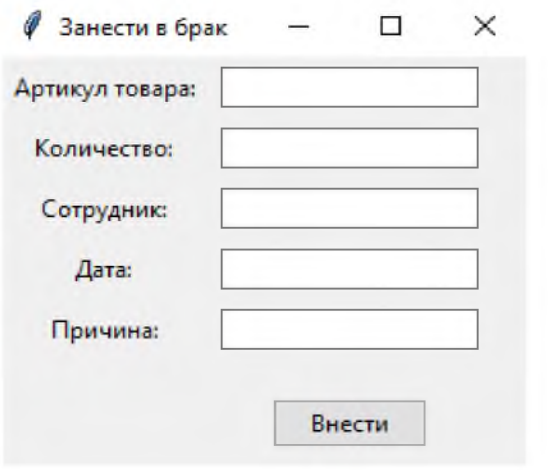


Рисунок 3.24 - Графическое окно «Занести в брак»

Графическое окно «Занести в брак» (рисунок 3.24) представляет собой интерфейсную форму, предназначенную для ввода информации о товаре, который необходимо занести в брак.

В окне есть следующие поля для ввода данных:

Поле для ввода «Артикул» - здесь указывается уникальный номер товара, с помощью которого можно идентифицировать конкретный товар.

Поле для ввода «Количество» - здесь указывается число единиц товара, которые нужно занести в брак.

Поле для ввода «Сотрудник « - здесь вводится имя того сотрудника, который ответственен за брак.

Поле для ввода «Дата» - здесь указывается дата, когда товар был занесён в брак.

Поле для ввода «Причина» - в это поле вносится информация о причине, по которой товар должен быть занесен в брак.

Также на графическом окне присутствует кнопка «Внести», которая выполняет функцию отправки введенной информации для обработки программой.

Графическое окно «Занести в брак» создано для удобного ввода информации о товаре, который требуется занести в брак. Окно обеспечивает пользователю

удобный интерфейс с полями для ввода всех необходимых данных, а также кнопкой для подтверждения введенной информации. Такой графический интерфейс позволяет упростить и ускорить процесс занесения товара в брак, а также делает информацию более структурированной и доступной для последующего анализа.

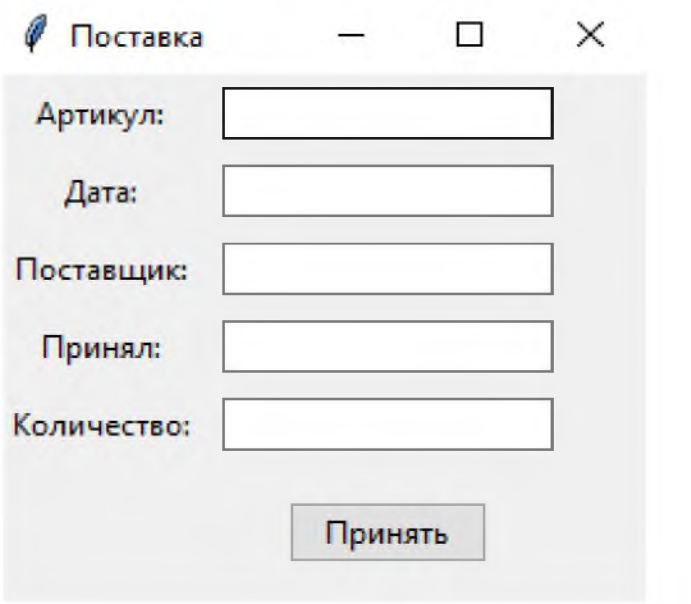


Рисунок 3.25 -Графическое окно «Поставка»

Окно «Поставка»(рисунок 3.25) является графическим интерфейсом, предназначенным для ввода информации о поставках товаров. В окне есть несколько полей для ввода, которые позволяют указать основные детали о поставке.

Поле «Артикул» предназначено для указания уникального идентификатора товара, который поставляется.

Поле «Дата» предназначено для указания даты, когда товар был принят на склад.

Поле «Поставщик» предназначено для указания фирмы или лица, которые поставляют товар.

Поле «Сотрудник « предназначено для указания имени сотрудника, который принимает товар.

Поле «Количество» предназначено для указания количества поставляемого товара.

После заполнения всех полей в окне, можно нажать кнопку «Принять», чтобы внести информацию о поставке в систему. Это позволяет сохранить данные о поставке для дальнейшего учета и анализа.

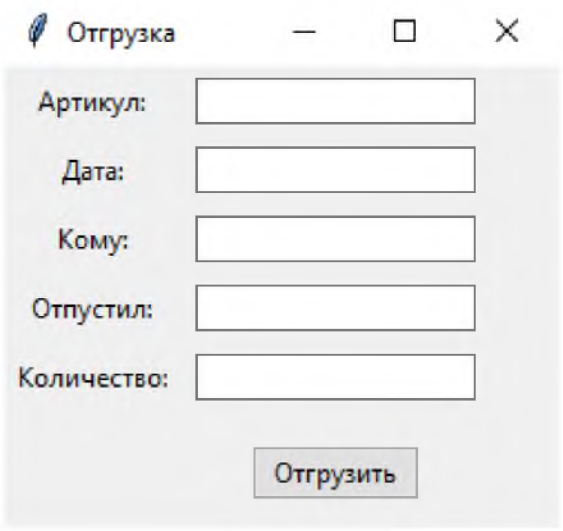


Рисунок 3.26 –Графическое окно «Отгрузка»

Это графическое окно «Отгрузка» (рисунок 3.26), которое предназначено для внесения информации о процессе отгрузки товара. В окне есть несколько полей для ввода данных и кнопка для внесения информации.

Поле для ввода «Артикул» предназначено для указания уникального кода товара.

Поле для ввода «Дата» предназначено для указания даты, когда товар был отгружен или будет отгружен.

Поле для ввода «Кому» предназначено для указания имени или названия организации, которой будет отгружен товар.

Поле для ввода «Сотрудник» предназначено для указания имени сотрудника, ответственного за отгрузку товара.

Поле для ввода «Количество» предназначено для указания количества единиц товара, которые были отгружены.

Кнопка «Отгрузить» позволяет внести введенную информацию о процессе отгрузки товара и сохранить ее.

После внесения всех необходимых данных и нажатия кнопки «Отгрузить», информация о процессе отгрузки будет сохранена, и вы сможете

использовать ее для учета, анализа или других целей, связанных с процессом отгрузки товаров.

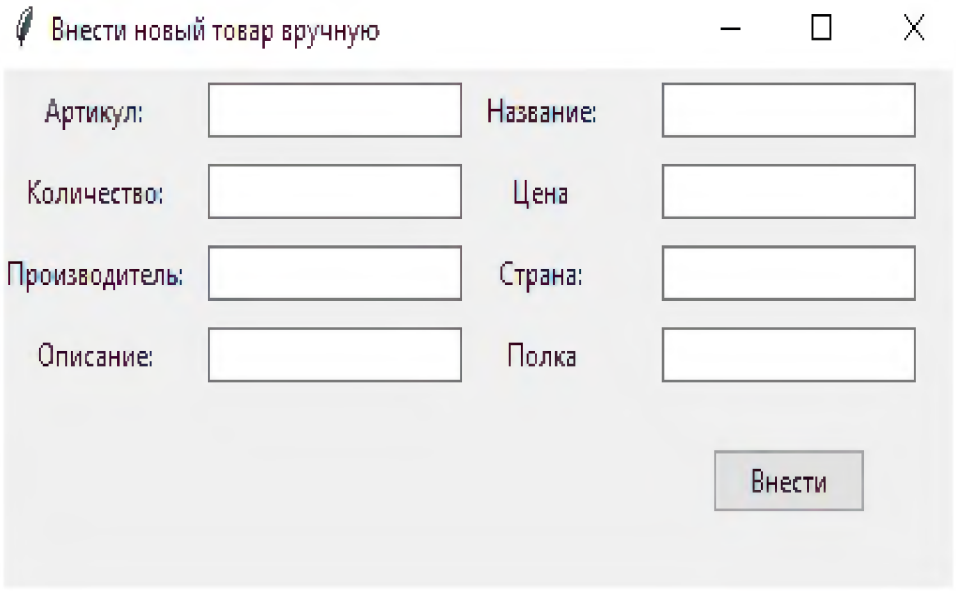


Рисунок 3.27 -Графическое окно «Внести новый товар вручную»

Графическое окно «Внести новый товар вручную» (рисунок 3.27) предоставляет пользователю возможность добавить информацию о новом товаре в базу данных. В окне имеются следующие поля для ввода:

Артикул: поле, предназначенное для указания уникального идентификатора товара. Артикул помогает уникально идентифицировать товар в системе.

Количество: поле для указания количества единиц товара.

Производитель: поле для указания производителя, ответственного за производство данного товара.

Описание: поле, где можно указать подробное описание товара, включая его основные характеристики, особенности и применение.

Название: поле для указания наименования товара. Это название будет использоваться для его идентификации в системе.

Цена: поле для указания стоимости товара. Здесь можно указать цену за одну единицу товара.

Страна: поле, где указывается страна, в которой произведен данный товар.

Полка: поле для указания полки, куда будет помещен товар.

Кнопка «Внести» предназначена для сохранения информации о новом товаре в системе.

Это окно предоставляет удобный и простой способ добавить новый товар с необходимой информацией, что позволяет легко управлять и обновлять данные о товарах в системе.

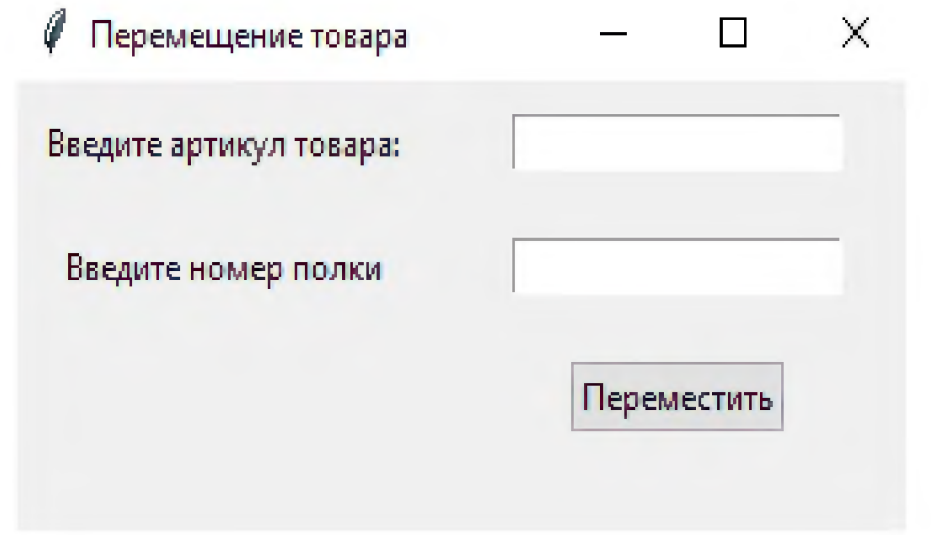


Рисунок 3.28 -Графическое окно «Перемещение товара»

Графическое окно «Перемещение товара»(рисунок 3.28) предназначено для удобного и эффективного перемещения товаров на полках. С помощью этого окна можно указать артикул товара и номер полки, на которую нужно переместить данный товар.

Окно имеет два поля для ввода информации. Первое поле предназначено для ввода артикула товара, по которому можно однозначно идентифицировать конкретный продукт. Второе поле предназначено для указания номера полки, на которую нужно переместить товар [30].

После ввода необходимой информации, пользователь может нажать на кнопку «Переместить», что активирует процесс перемещения товара. При нажатии кнопки система будет проверять, доступна ли указанная полка и находится ли товар на другой полке. Если все условия выполнены, система осуществит перемещение товара на указанную полку.

Таким образом, графическое окно «Перемещение товара» упрощает и

автоматизирует процесс перемещения товаров на определенные полки, улучшая организацию и эффективность работы с товаром.

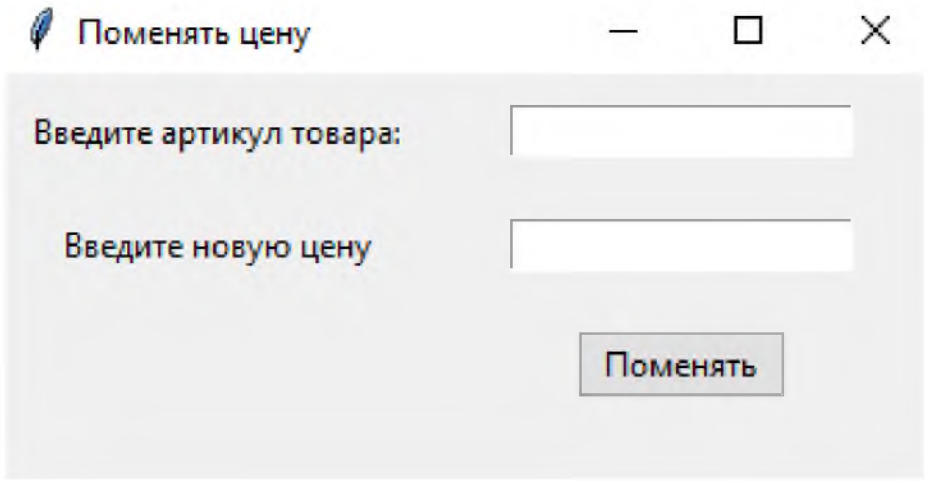


Рисунок 3.29 -Графическое окно «Поменять цену»

Графическое окно «Поменять цену»(рисунок 3.29) предоставляет пользователям возможность быстро и удобно изменять цену товаров. В окне присутствуют два поля для ввода информации - «Введите артикул товара» и «Введите новую цену».

В поле «Введите артикул товара» пользователь должен ввести уникальный идентификатор товара. Это позволяет точно идентифицировать товар, цену которого нужно изменить.

В поле «Введите новую цену» пользователь должен ввести желаемую новую цену для выбранного товара. Здесь можно указать любое числовое значение, отражающее стоимость товара.

После заполнения полей, пользователь нажимает на кнопку «Поменять», чтобы внести изменения. Кнопка выполняет операцию изменения цены товара согласно введенной информации.

Графическое окно «Поменять цену» обеспечивает удобный интерфейс для быстрого изменения цены товаров и позволяет точно определить, на какой товар будет произведено изменение.

Заключение

В результате разработки информационной системы учёта товаров была создана эффективная платформа, которая значительно улучшит процессы учёта и управления товарами в организации. Система предоставляет множество возможностей для автоматизации и оптимизации бизнес-процессов, что поможет улучшить эффективность работы, уменьшить затраты и повысить уровень обслуживания клиентов.

Одной из ключевых особенностей системы является её функциональность. Система позволяет вести учёт товаров с точностью до мельчайших деталей, начиная от ввода идентификационных данных товаров, хранения информации о поставках и продажах, и заканчивая определением остатков и составлением отчётов. Большинство процессов автоматизированы, что минимизирует ручной труд и снижает риск ошибок.

Важной частью системы является её удобный и интуитивно понятный интерфейс. Он разработан с учётом потребностей и привычек пользователей, что позволяет быстро освоить систему без особых усилий.

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).