Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет инженерно-экономический

Кафедра экономики

Дисциплина «Распределённые системы обработки информации»

|  |  |
| --- | --- |
|  | «К ЗАЩИТЕ ДОПУСТИТЬ» |
|  | Руководитель курсового проекта  ассистент кафедры экономической информатики  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.П. Лыщик |
|  | \_\_\_.\_\_\_\_.2023 |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту

на тему:

**«система учёта и регистрации поступления товаров на склад логистической компании»**

|  |  |
| --- | --- |
|  | Выполнил студент группы 074002  Барковская Виктория Александровна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовой проект представлен на проверку \_\_\_.\_\_\_\_.2023  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  |  |

Минск 2023

Содержание

[Введение 5](#_Toc133886533)

[1 Организация деятельности логистической компании 7](#_Toc133886534)

[2 Постановка задачи на разработку программного средства логистической компании и обзор методов её решения 12](#_Toc133886535)

[3 Функциональное моделирование системы 17](#_Toc133886536)

[4 Информационная модель системы 23](#_Toc133886537)

[5 Проектирование системы логистической компании на основе UML 27](#_Toc133886538)

[5.1 Спецификация вариантов использования системы логистической компании 27](#_Toc133886539)

[5.2 Описание диаграммы последовательности создания отчёта «Отчёт о складах и товарах на них» 29](#_Toc133886540)

[5.3 Описание диаграммы состояний построения графика «Количество поставленных товаров за последние 6 месяцев» 32](#_Toc133886541)

[5.4 Диаграмма компонентов 34](#_Toc133886542)

[5.1 Диаграмма развёртывания 34](#_Toc133886543)

[5.5 Описание диаграммы классов 35](#_Toc133886544)

[6 Описание алгоритмов, реализующих бизнес-логику системы 38](#_Toc133886545)

[6.1 Схема алгоритма решения задачи методом ранга 38](#_Toc133886546)

[6.2 Схема алгоритма построения графика «Количество поставленных товаров за последние 6 месяцев» 39](#_Toc133886547)

[6.3 Схема алгоритма клиент-серверного взаимодействия 41](#_Toc133886548)

[7 Руководство по развёртыванию системы логистической компании 43](#_Toc133886549)

[8 Руководство пользователя 44](#_Toc133886550)

[9 Результаты тестирования разработанной программы 62](#_Toc133886551)

[Заключение 65](#_Toc133886552)

[Список использованных источников 67](#_Toc133886553)

[ПРИЛОЖЕНИЕ А 68](#_Toc133886554)

[ПРИЛОЖЕНИЕ Б 69](#_Toc133886555)

[ПРИЛОЖЕНИЕ В 73](#_Toc133886556)

[ПРИЛОЖЕНИЕ Г 75](#_Toc133886557)

Введение

Один из критериев эффективности и повышение уровня конкурентоспособности в современном бизнесе логистических компаний на сегодняшний период является автоматизация. Чем больше процессов можно доверить в управление компьютеру, тем меньше времени для решения задач остается у людей. WMS (Warehouse Management System, система управления складом) – это так называемое решение, которое в силу своих функциональных возможностей вполне способно содействовать предприятиям в аспекте повышения уровня автоматизации различных, необходимых для предприятия бизнес – процессов. Большинство из многих бизнес-процессов логистической компании, в том числе процессы принятия решений, можно сделать более производительными, если использовать информационные продукты и технологии.

Для логистических компаний автоматизация склада является первоочередной задачей. Для них система управления складом – это инструмент для предоставления каждому клиенту высочайшего уровня сервиса исходя из его индивидуальных требований, а также конкурентоспособных цен на услуги. Посредством автоматизации склада розничной сети можно добиться высокой оборачиваемости склада, осуществлять быструю комплектацию партий товара, отгрузку их потребителям. Для дистрибьюторской компании автоматизация склада позволяет осуществлять безукоризненное управление складом для достижения и поддержания конкурентных преимуществ в работе с розничными операторами и конечными пользователями.

Автоматизация склада с помощью системы управления складом WMS позволяет существенно сократить время выполнения операций, уменьшить их стоимость, сократить количество ошибок, улучшить качество обслуживания клиентов, повысить производительность работы персонала, уменьшить издержки хранения товаров, то есть осуществлять максимально эффективное управление складом.

Растущая популярность систем управления складом обусловлена тем, что профессиональные и адаптируемые решения, в зависимости от классификации и комплектации, обеспечивают оптимизацию всех складских процессов в сквозном режиме (приемка, размещение, хранение, комплектация грузов и многое другое), а также контроль работы персонала, техники, складского оборудования. Возможность контролировать операции, планировать их в автоматизированном режиме и осуществлять оперативное управление централизованно – серьезное преимущество, позволяющее сократить издержки и повысить качество работы, что определяет конкурентоспособность компании.

Объектом исследования курсового проекта является учёт и регистрация поступления товаров на склад логистической компании. Предметом исследования является процесс оптимизации учёта и регистрации поступления товаров на склад логистической компании.

Целью данного курсового проекта является повышение эффективности заполнения складов логистической компании и сокращение количества ошибок при работе персонала компании.

Поставленная цель потребовала решения следующих задач:

* ознакомление с особенностями организации деятельности логистической компании;
* исследование процесса заполнения склада товарами;
* моделирование представления разрабатываемого программного средства;
* разработка алгоритмов работы программного средства;
* разработка интерфейса программного средства;
* выполнение программной реализации программного средства;
* предоставление руководства пользователя для работы с программным средством;
* предоставление руководства по развёртыванию программного средства;
* тестирование работоспособности программного средства.

# Организация деятельности логистической компании

В настоящее время все большее внимание в среде информационных технологий уделяется решению экономических задач. Это могут быть задачи прогнозирования, анализа, но большую часть на данном этапе развития составляют задачи учета. Задача учета товаров на складе - одна из самых распространенных. Существует, конечно, множество решений, в том числе и на базе 1С. Но практичнее и надежнее представляется создание собственного программного обеспечения для этой цели.

Поэтому в рамках данной курсовой работы будет создана программная оболочка, которая, основываясь на базе данных, позволяет вести учёт поступающих на склад и выбывающих с него товаров.

Логистическая компания – это предприятие, оказывающее услуги по транспортировке, обработке и хранению грузов, содействуя своим клиентам в процессе продвижения товаров от производителя к потребителю.

Импортёры или оптовики поддерживают складские запасы и доставляют продукцию в магазины или крупным конечным покупателям. Они работают в большом географическом регионе, в то время как их контрагенты, магазины, работают в меньших регионах или зачастую в совсем небольшом районе.

WMS система (Warehouse Management System, система управления складом) – это система автоматизации складской отрасли, которая позволяет:

* оперативно, в реальном режиме времени управлять складом;
* получать информацию о состоянии склада в любых резервах;
* оптимизировать использование всех – вещественных, финансовых, трудовых ресурсов склада, посредством планирования, анализа загруженности складского пространства и оборачиваемости товаров.

Большинство современных WMS-систем реализуют следующие аспекты:

* управление основными складскими операциями (такими как приемка, размещение, инвентаризация, комплектация, оприходование, отгрузка и другие);
* ведение документооборота (как внутреннего, так и в аспекте взаимодействия с внешними структурами фирмы);
* эффективное управление трудовыми ресурсами.

В последнее время растет количество заказов на создание немногочисленных партий товаров или вообще изделий в единичном экземпляре. При этом производитель должен выполнить заказ качественно, в короткие сроки и предоставить гарантию на произведенный им продукт. Так же применение производственной логистики актуально во время организации производства сложных товаров, когда необходимым условием является кооперация. Например, в случае необходимости транспортировать продукт производства, операции по его перевозке могут выступать объектом производственной логистики, при условии использования собственного транспорта. В случае привлечения транспорта других организаций данные операции становятся объектом транспортной логистики, а не производственной. Для каждого из этих видов бизнеса существуют свои оптимальные модели его организации, такие как необходимая площадь помещений, расположение производства и точек продаж, схемы юридического оформления.

Термин «логистика» ассоциируется с наукой, учебной дисциплиной и специфической практической деятельностью. Само явление «логистика» встречается очень давно во многих областях человеческой жизни, но только относительно недавно она выделилась в самостоятельное научное и практическое направление.

Традиционными видами деятельности, в которых применяется логистика, являются: грузоперевозки, материально-техническое снабжение, управление запасами, складское хозяйство. Всплеск интереса к логистике пришел из развитых капиталистических стран, которые, добившись технологических вершин производства, ищут другие пути снижения издержек. Популярность логистики привлекла к себе многие отрасли экономики. Сегодня стали появляться новые направления: логистика в искусстве, медицине, спорте и т. д. Подобные словосочетания кажутся иногда несовместимыми и лженаучными, но судить об этом на современном этапе развития теоретических знаний о логистике трудно.

Эффективность функционирования предприятия, использующего логистику, достигается в основном за счет:

- резкого снижения себестоимости товара;

- повышения надежности и качества поставок.

Особенность логистики заключается в системном рассмотрении совокупности всех звеньев производственного процесса с позиций единой материалопроизводственной цепи, которая имеет название "логистическая система". Взаимодействие отдельных звеньев этой цепи осуществляется на техническом, технологическом, экономическом, финансовом, методологическом и других уровнях интеграции. Сокращение затрат на ресурсы и минимизация временных затрат достигается за счет оптимизации сквозного управления материальными и информационными потоками. Для каждого заказа необходимо оформить большое количество документов и просчитать все затраты. Данный курсовой проект должен решать данную проблему.

Новые принципы организации и управления, основанные на концептуальных подходах и методе мышления, объединяемых общим понятием "логистика", все в большей степени и с успехом применяются на практике наиболее эффективно функционирующими предприятиями, транспортными компаниями, фирмами и объединениями.

Методы логистики объединяют организацию и управление различными фазами, стадиями и аспектами экономической деятельности.

Логистика – это система наук управления продвижением материал потока, т.е. управление материалами и управление распределением. Целью логистики является обеспечение получения (доставки) продукции потребителю в нужное время и место при минимально возможных совокупных затратах трудовых, материальных, финансовых ресурсов.

Широкое применение логистики в практике хозяйственной деятельности объясняется необходимостью сокращения временных интервалов между приобретением сырья и поставкой товаров конечному потребителю. Логистика позволяет минимизировать товарные запасы, в ряде случаев вообще отказаться от их использования, существенно сократить время доставки товаров, ускоряет процесс получения информации, повышает уровень сервиса.

Выделяют шесть основных функциональных областей логистики:

* логистика снабжения и закупок;
* логистика производства;
* логистика распределения и сбыта;
* транспортная логистика;
* логистика складирования;
* информационная логистика.



Рисунок 1.1 – Функциональная область логистики

Каждая из этих областей подробно исследуется в рамках различных дисциплин логистического комплекса.

Логистика снабжения и закупок – первая подсистема в логистической системе предприятия по ходу движения материального потока. Основная цель логистики снабжения и закупок – удовлетворение потребностей производства в материально-технических ресурсах в нужном количестве, приемлемого качества, в установленное время, с минимальными затратами. В данной функциональной области логистики решаются задачи по оценке и выбору поставщика, заключению контрактов и контролю их исполнения, осуществляется оперативное планирование и организация закупочных процессов.

Логистика производства изучает процессы рационального движения материального потока в рамках технологического цикла изготовления продукции. Основная цель логистики производства – оптимизация потоков материальных ресурсов и незавершенного производства на пути их движения от склада снабжения до склада сбыта готовой продукции через качественное изменение характеристик в рамках производственного процесса.

Логистика распределения и сбыта оптимизирует материальные потоки в товаропроводящих цепях. Основная цель распределительной логистики – доведение готовой продукции предприятия до конечных пользователей напрямую или через посредников с минимальными затратами с обеспечением надлежащего уровня сервиса. Задачи логистики на данном этапе тесно переплетаются с задачами маркетинга и включают проектирование каналов распределения, организацию работы с участниками системы распределения (торговыми и логистическими посредниками), управление заказами, управление запасами готовой продукции, организацию обслуживания потребителей и др.

Основная цель транспортной логистики – оптимизация процессов транспортировки сырья, материалов и готовой продукции различными видами транспорта в сфере обращения. Среди задач транспортной логистики следует выделить задачи грузовладельцев и задачи транспортной организации. К задачам транспортной логистики для грузовладельца относится: выбор вида транспорта и транспортно-логистического посредника, выбор транспортной тары, принятие решения о партии поставки и др. В число логистических задач транспортной организации входит: выбор маршрутов перевозки, составление графиков работы транспорта, оптимизация порожних пробегов транспорта и др.

Функциональная сфера логистики складирования на предприятии соединяет подсистемы снабжения, производства и распределения. Основная цель складской логистики − оптимизация процессов складирования и хранения запасов, сырья, материалов, готовой продукции на складах предприятий, торговли посреднических организаций.

Цель информационной логистики – оптимизация информационных процессов в логистических системах, их синхронизация с процессами перемещения материальных, транспортных и других физических потоков.

# Постановка задачи на разработку программного средства логистической компании и обзор методов её решения

Повышение качества и эффективности управления – одна из главных задач большинства отечественных компаний, так как только качественное управление открывает путь к реальному повышению их конкурентоспособности и прибыльности, к привлечению инвестиций, а значит, к максимальным темпам роста стоимости бизнеса.

Для повышения качества управления менеджменту компании необходимы современные и многофункциональные инструменты, которые позволяют анализировать большое количество экономических параметров и принимать грамотные управленческие решения, адекватные сложившейся ситуации на рынке. Одним из таких инструментов являются информационные системы. Все чаще директора и владельцы компаний приходят к пониманию того, что информационные системы – это не дань моде, а необходимое условие для выживания и успешного развития.

Уровень популярности систем управления складом возрастает из года в год обуславливая тем, что профессиональные и адаптируемые WMS системы, в зависимости от классификации и комплектации, гарантируя оптимизацию всех складских процессов в сквозном режиме (приемка, размещение, хранение, комплектация товара и многое другое), а также контроль по выполнению работы персонала, технических единиц, складского оборудования. Вероятность, полноценно вести контроль над операциями, осуществлять планирование их в автоматизированном режиме и обеспечивать оперативное управление – является серьезным преимуществом, позволяющим сокращать издержки, тем самым увеличив уровень по качеству работы, что явно определяет конкурентоспособность предприятия. Для грамотной подготовки по внедрению WMS продуктов, необходимо провести точное планирование работ по автоматизации, которые отвечают индивидуальным особенностям, требованиям и критериям бизнеса.

Основными преимуществами систем управления складом являются: сокращение время выполнения операций, уменьшение их стоимости, сокращение количества ошибок, улучшение качества обслуживания клиентов, повышение производительности работы персонала, уменьшение издержек хранения товаров, то есть осуществление максимально эффективного управления складом.

Разработанная система должна облегчить работу фирмы. Цель автоматизации системы управления складами логистической компании – обеспечение быстрого и качественного выполнения операций учета и реализации поставок.

К преимуществам данной системы относится удобство компании в распределении товаров, ускорение процесса оформления заказа поставок, снижение вероятности допущения ошибки, снижение риска человеческого фактора, большая автономность системы. Администратор системы имеет полный контроль над ней. Он может вносить данные, рассчитанные аналитиками, в базу данных, добавлять и удалять пользователей, искать необходимые данные и так далее.

Недостатком является необходимость постоянно поддерживать данную систему в актуальном виде.

Решение основной задачи сводится к следующим подзадачам:

* хранение полного списка пользователей;
* хранение полного списка товаров, доступных для реализации;
* обеспечение оформления поставок на склад;
* обеспечение инкапсуляции вводимых данных для безопасности пользователей;
* вывод в удобном виде данных, необходимых для управления логистической компанией;
* предоставление удобной среды для управления складами.

Проанализировав деятельность логистической компании, можно уверенно сказать, что разработка данного программного продукта является весьма актуальной.

Приложение должно быть выполнено в архитектуре клиент-сервер с организацией взаимодействия с базой данных на объектно-ориентированном языке Java.

Java обладает большим количеством инструментов для создания приложений клиент-серверного типа с графическим пользовательским интерфейсом. Можно отметить, что немало важным достоинством выбранного языка программирования является надежность.

Суть клиент-серверной архитектуры заключается в следующем. Клиентские процессы посылают запросы серверному процессу, посылающему обратно результаты этих запросов. Взаимодействие между клиентским и серверным процессами представляет собой совместный транзакционный обмен, в котором активность исходит от клиента, а сервер реагирует на эту активность.

К преимуществам данной модели взаимодействия относится ее безопасность. Все данные хранятся на сервере, который, как правило, защищён гораздо лучше большинства клиентов. На сервере проще обеспечить контроль полномочий, чтобы разрешать доступ к данным только клиентам с соответствующими правами доступа. Также стоит упомянуть, что в приложениях клиент-сервер большое внимание уделяется созданию на клиентской стороне пользователю удобного и понятного интерфейса.

В данном программном средстве будет разрешена работа как администратору, так и обычному пользователю.

В возможности администратора будет входить управление учётными записями пользователей, а также добавление и редактирование данных о действиях пользователей.

Также для решения поставленных задач можно вывести следующие требования к приложению:

– многопоточность сервера для того, чтобы с приложением могли работать сразу несколько пользователей;

– приложение должно запускаться без использования интегрированных средств разработки;

– данные и интерфейс должны быть только на русском языке;

Необходимо отладить и протестировать разработанное приложение для того, чтобы была гарантия качественной работы в дальнейшем.

Для написания программы было выбрано IDE под названием «Intelij idea». IntelliJ IDEA – интегрированная среда разработки программного обеспечения для многих языков программирования, в частности Java, JavaScript, Python, разработанная компанией JetBrains.

К ее достоинствам относятся эффективные инструменты минимизации ошибок (проверка и оптимизация кода, наглядное выделение и отчеты об ошибках, поиск дублирующийся фрагментов кода, идентификация зависимостей и т.д.). Среда разработки также предоставляет функцию множественного выделения, позволяющую вносить в код правки в нескольких местах одновременно. Платформа предоставляет инструменты для работы с базами данных и SQL файлами, включая удобный клиент и редактор для схемы базы данных.

Создание UI проводилось с помощью инструмента JavaFX, который вместе с программой ScenceBuilder дает возможность написания красивого интерфейса. Макеты UI хранятся в формате FXML.

FXML – формат, служащий для описания интерфейса. Основан на XML, упрощает описание JavaFX.

SceneBuilder – приложение для создания UI на JavaFX, замена ручного написания каждой кнопки интерфейсом для создания оконного приложения.

Postgres pgAdmin – приложение для работы с языком SQL, полный набор взаимодействия, редактирования, и изменения SQL.

AllFusion Process Modeler 7 – программа для построения диаграмм IDEF. Данный инструмент содержит удобную структуру декомпозиций, которая помогает быстро создать нужные диаграммы.

Учитывая цель уменьшения трудозатрат на разработку сложного программного обеспечения необходимо использовать готовые унифицированные решения. Ведь шаблонность действий облегчает коммуникацию между разработчиками, позволяет ссылаться на известные конструкции, снижает количество ошибок.

В данном курсовом проекте я использовал паттерны Model-View-Controller, Singleton и Iterator.

MVC — это фундаментальный паттерн, который нашел применение во многих технологиях, дал развитие новым технологиям и каждый день облегчает жизнь разработчикам.

Одиночка ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Singleton) – [порождающий шаблон проектирования](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%80%D0%BE%D0%B6%D0%B4%D0%B0%D1%8E%D1%89%D0%B8%D0%B5_%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D1%8B_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), гарантирующий, что в однопоточном приложении будет единственный экземпляр некоторого класса, и предоставляющий глобальную точку доступа к этому экземпляру. У [класса](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) есть только один экземпляр, и он предоставляет к нему глобальную точку доступа. При попытке создания данного [объекта](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) он создаётся только в том случае, если ещё не существует, в противном случае возвращается ссылка на уже существующий экземпляр и нового выделения памяти не происходит. Существенно то, что можно пользоваться именно экземпляром класса, так как при этом во многих случаях становится доступной более широкая функциональность.

Iterator – [поведенческий](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5_%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D1%8B_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) [шаблон проектирования](https://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F). Представляет собой объект, позволяющий получить последовательный доступ к элементам [объекта-агрегата](https://ru.wikipedia.org/wiki/%D0%90%D0%B3%D1%80%D0%B5%D0%B3%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) без использования описаний каждого из агрегированных объектов.

Например, такие элементы как [дерево](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_(%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)), [связанный список](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D1%8F%D0%B7%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D1%81%D0%BF%D0%B8%D1%81%D0%BE%D0%BA), [хеш-таблица](https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D1%88-%D1%82%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D0%B0) и [массив](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D1%81%D0%B8%D0%B2_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) могут быть пролистаны (и модифицированы) с помощью объекта Итератор.

Перебор элементов выполняется объектом итератора, а не самой коллекцией. Это упрощает интерфейс и реализацию коллекции, а также способствует более логичному [разделению обязанностей](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BE%D1%82%D0%B2%D0%B5%D1%82%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D0%B8).

Особенностью полноценно реализованного итератора является то, что код, использующий итератор, может ничего не знать о типе итерируемого агрегата.

Задачи и подзадачи, описанные в данной главе, помогают четко понимать, что необходимо выполнять для разработки программы, удовлетворяющей всем требованиям. Разработанная программа упростит работу сотрудникам логистической компании по учёту и регистрации товаров логистической компании, тем самым повысив качество работы компании.

# Функциональное моделирование системы

Исходя из задания данного курсового проекта, целью моделирования системы является описание ее функциональности для дальнейшего использования созданной модели при разработке информационной модели.

Методология IDEF0 предписывает построение иерархической системы диаграмм – единичных описаний фрагментов системы. Сначала проводит­ся описание системы в целом и ее взаимодействия с окружающим миром (контекстная диаграмма), после чего проводится функциональная деком­позиция – система разбивается на подсистемы и каждая подсистема опи­сывается отдельно (диаграммы декомпозиции). Затем каждая подсистема разбивается на более мелкие и так далее до достижения нужной степени подробности.

Каждая IDEF0-диаграмм, а содержит блоки и дуги. Блоки изображают функции моделируемой системы. Дуги связывают блоки вместе и отобра­жают взаимодействия и взаимосвязи между ними.

Функциональные блоки (работы) на диаграммах изображаются прямоугольниками, означающими поименованные процессы, функции или задачи, которые происходят в течение определенного времени и имеют распознаваемые результаты. Имя работы должно быть выражено отглагольным существительным, обозначающим действие.

IDEF0 требует, чтобы в диаграмме было не менее трех и не более шести блоков. Эти ограничения поддерживают сложность диаграмм и модели на уровне, доступном для чтения, понимания и использования.

Каждая сторона блока имеет особое, вполне определенное назначение. Левая сторона блока предназначена для входов, верхняя – для управления, правая – для выходов, нижняя – для механизмов. Такое обозначение отражает определенные системные принципы: входы преобразуются в выходы управление ограничивает или предписывает условия выполнения преобразований, механизмы показывают, что и как выполняет функция.

Контекстная диаграмма представляет собой общее описание системы и пути её взаимодействия с пользователями.

Главным был выбран процесс «Выполнить заказ» (рисунок 1.3). В данном процессе будет происходить поставка и отгрузка товара, а также оформление документов заказа. Данную работу будут выполнять грузчики, заведующий складом и кладовщик. Для выполнения процесса выполнения заказа необходимы: продукция, транспортная накладная, договор на поставку продукции, путевой лист, счёт-фактура и договор на отгрузку товара. В результате проведённой работы для покупателя будет выданы подписанные документы заказа и отгруженная продукция. Для компании будут получены сопроводительные документы отгрузки и документы отчёта операций на складе. Весь процесс выполнения заказа будет происходить в соответствии с требованиями по охране труда при складских работах, а также с законом Республики Беларусь о бухгалтерском учёте и отчётности.

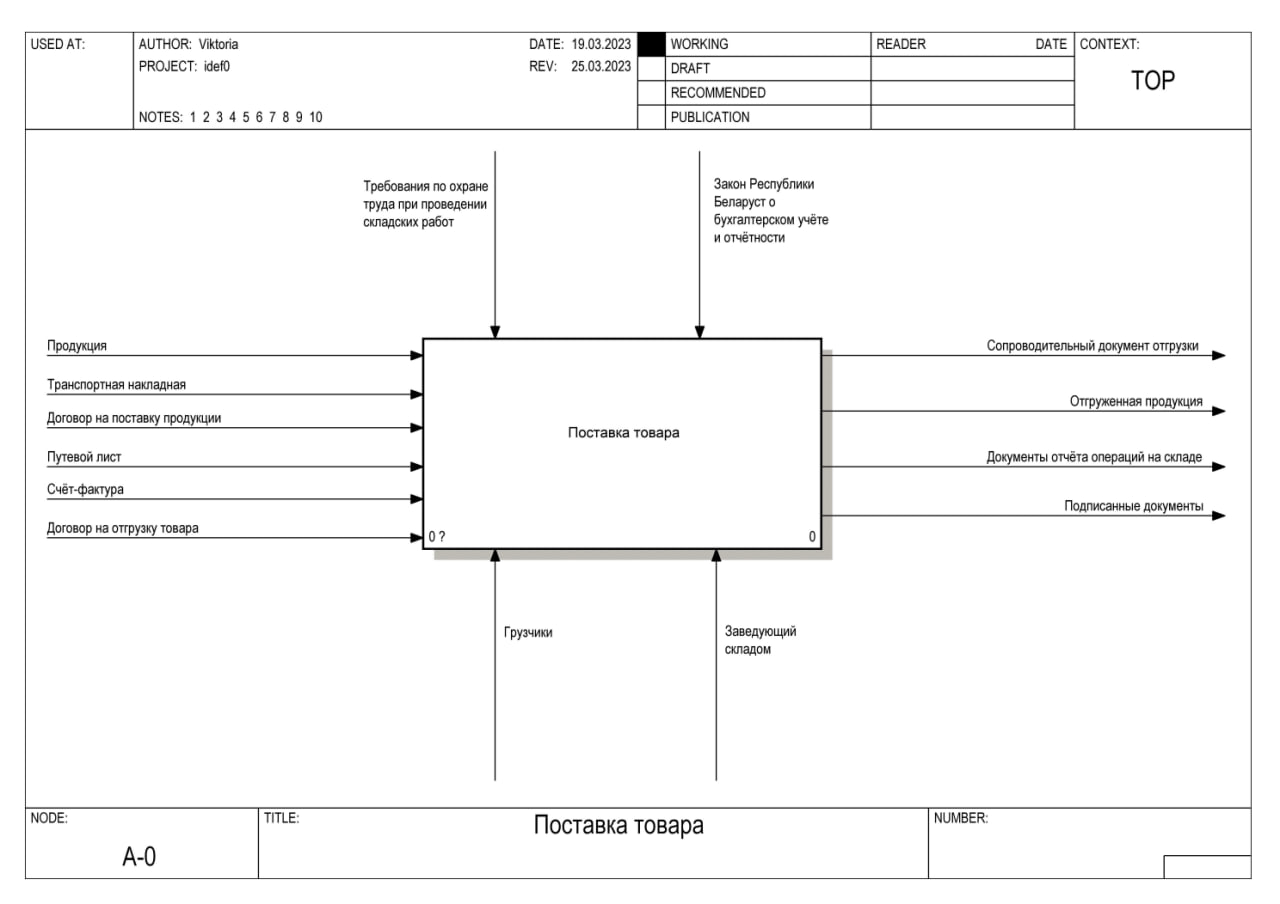


Рисунок 1.3 – Контекстная диаграмма верхнего уровня

Главный процесс разбивается на проверку корректности составления документов, составление документов, проверку поставленной продукции, занесение данных в БД, присвоение штрихкода продукции и размещение продукции на складе (рисунок 1.4).

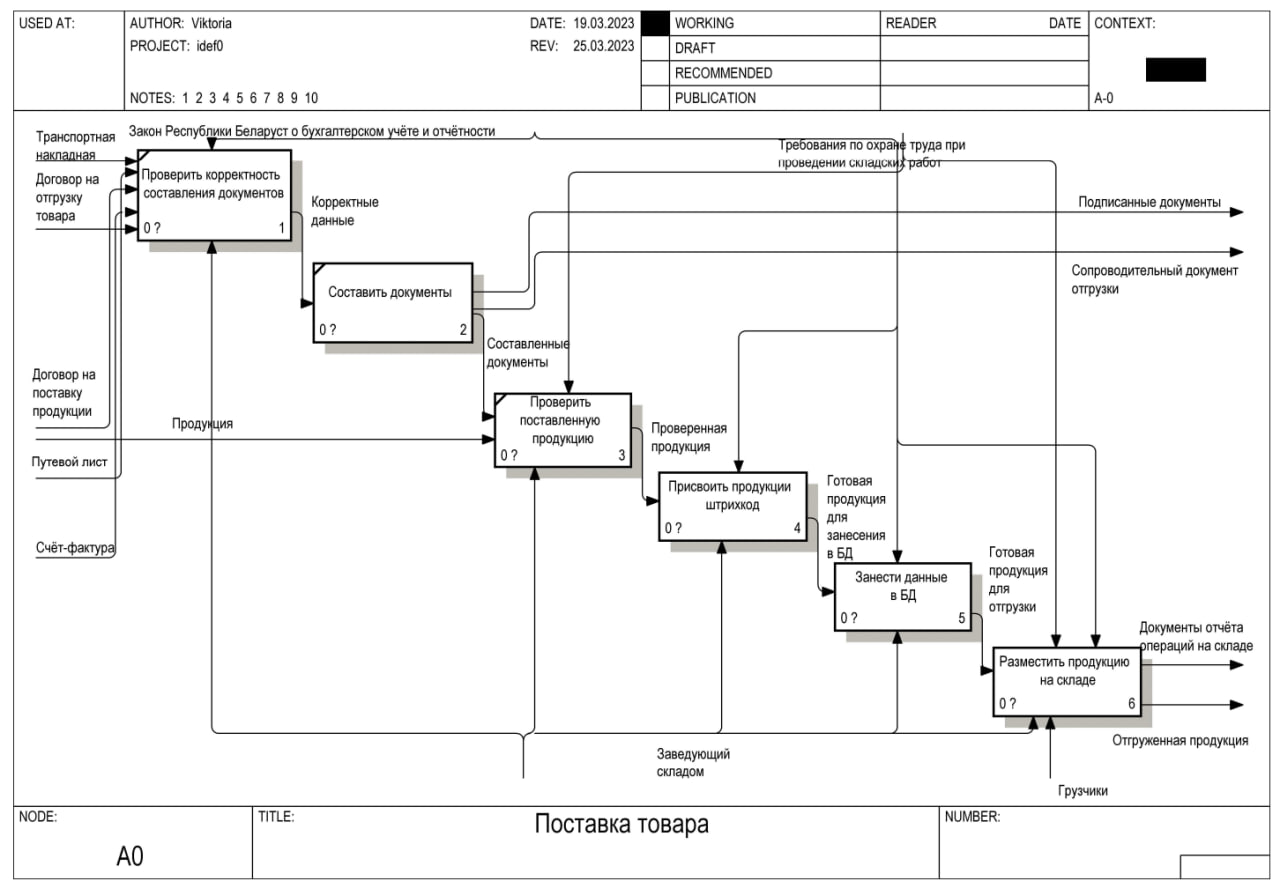


Рисунок 1.4 – Декомпозиция первого уровня

Далее рассмотрим процесс «Присвоить продукции штрихкод» (рисунок 1.5). Он состоит из блоков «Обновить данные в терминале», «Сканировать штрихкод», «Занести количество занесённого товара в терминал» и «Сформировать запрос на добавление данных в БД». Входными данными являются «Проверенная продукция». Управляет процессом «Закон Республики Беларусь о бухгалтерском учёте и отчётности». Участие в процессе принимает заведующий складом.

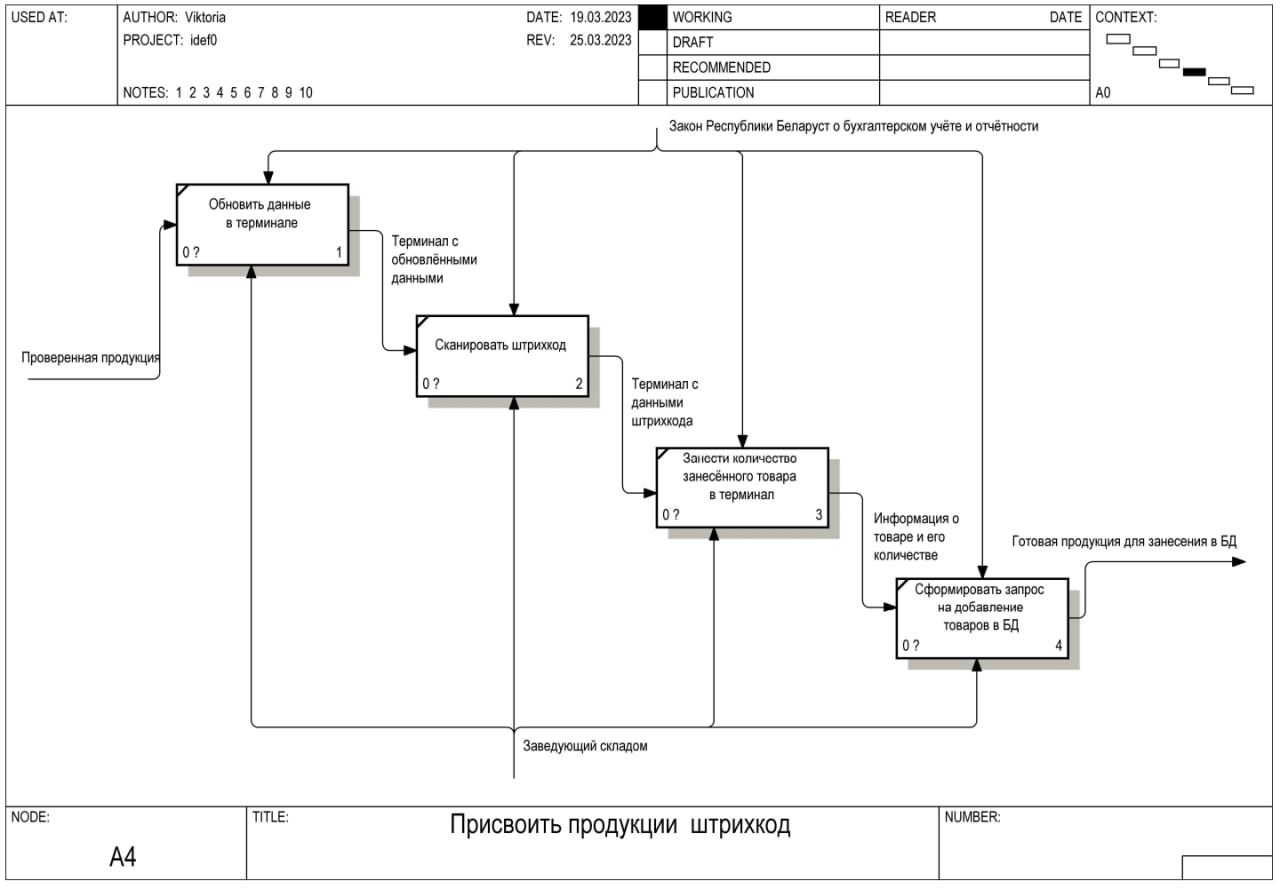


Рисунок 1.5 – Декомпозиция блока «Присвоить продукции штрихкод»

Рассмотрим процесс «Занести данные в БД» (рисунок 1.6). Данный процесс состоит из блоков «Проверить документы», «Найти необходимое местоположение продукции», «Сформировать список отгружаемых продуктов», «Занести информацию об отгруженных товарах в БД» и «Отправить продукцию на отгрузку». Входными данными является «Готовая продукция для занесения в БД». Управляют процессом «Закон Республики Беларусь о бухгалтерском учёте и отчётности». Участие в процессе принимает заведующий складом.

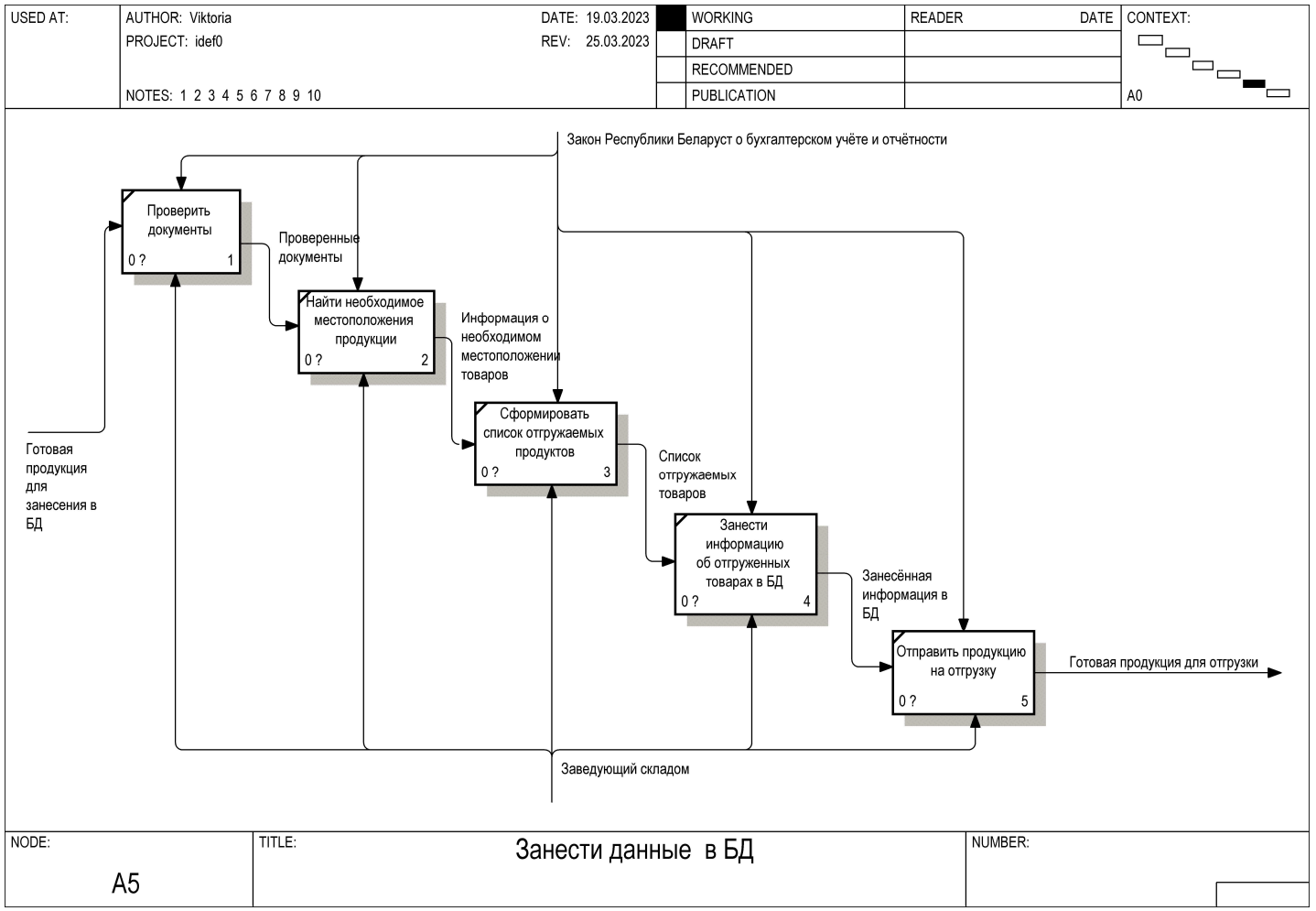


Рисунок 1.6 – Декомпозиция блока «Занести данные в БД»

Далее рассмотрим процесс «Разместить продукцию на складе» (рисунок 1.7). Он состоит из блоков «Обработать данные о принятый продукции», «Обработать данные о наличии товаров на складе», «Отгрузить товары на склад» и «Сформировать отчётную документацию о деятельности склада». Входными данными является «Готовая продукция для отгрузки». Управляют процессом «Закон Республики Беларусь о бухгалтерском учёте и отчётности» и «Требования по охране труда при проведении складских работ». Участие в процессе принимают заведующий складом и грузчики.

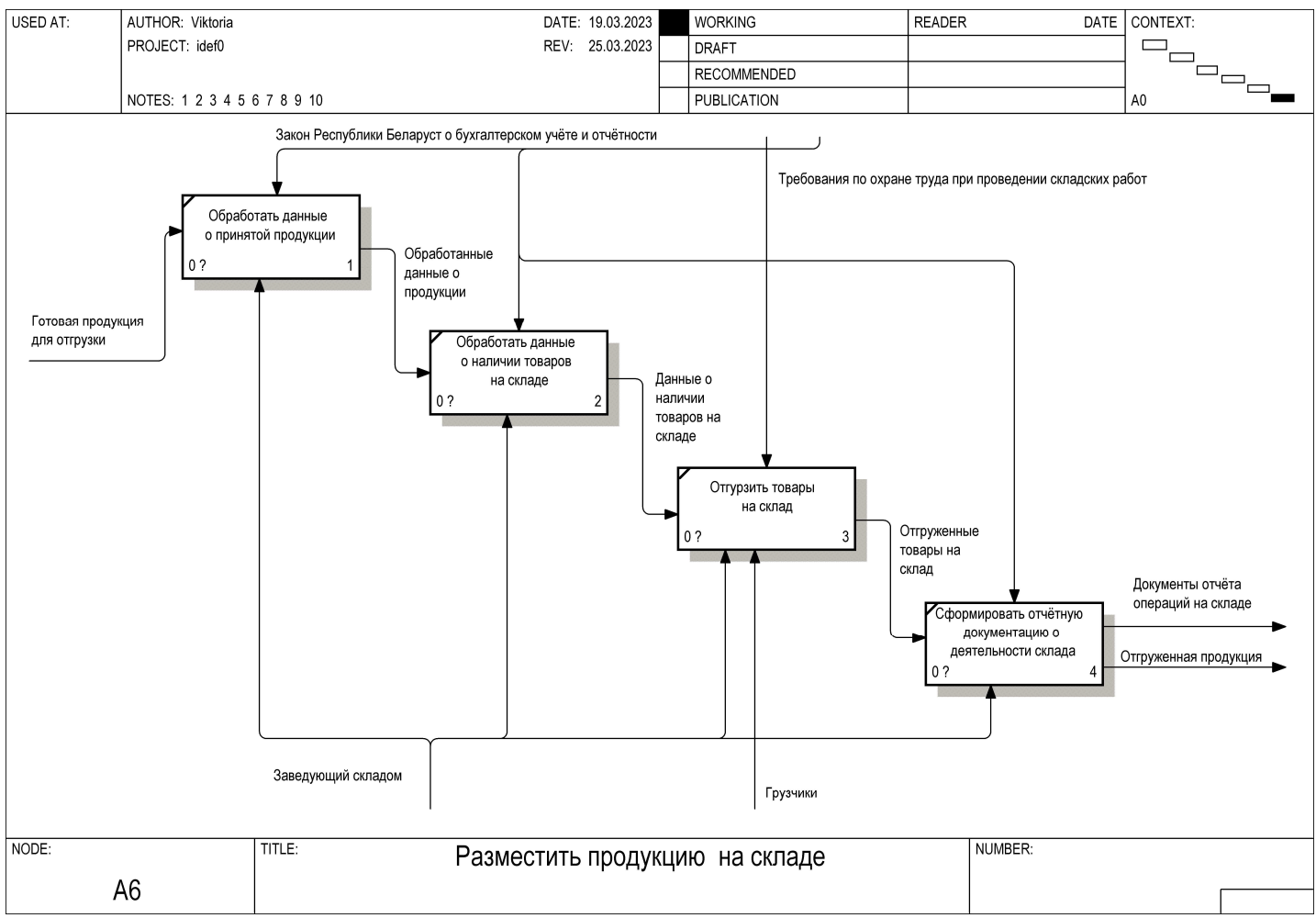


Рисунок 1.7 – Декомпозиция блока «Разместить продукцию на складе»

Прочитав и изучив все данные, описанные выше, можно понять, как происходит процесс учёта и регистрации товаров логистической компании, как необходимо организовать её деятельность и что обязательно должно присутствовать в ней, что способствует пониманию предметной области и поможет в дальнейшей разработке логики программы.

# Информационная модель системы

Информационная модель представляет объекты в виде таблиц. Она содержит массу информации, которая описывает свойства объекта, его связи и отношения с другими объектами моделирования.

Чтобы построить информационную модель необходимо проанализировать объект моделирования как сложную систему, а также определить цели моделирования.

В данном курсовом проекте для хранения, обработки и использования информации используется одна из наиболее популярных систем управления базами данных PostgreSQL.

Для представления структуры хранимых данных программного средства выбрана информационная методология IDEF1X.

IDEF1X является методом для разработки реляционных баз данных и использует условный синтаксис, специально разработанный для удобного построения концептуальной схемы. Использование метода IDEF1X наиболее целесообразно для построения логической структуры базы данных после того, как все информационные ресурсы исследованы и решение о внедрении реляционной базы данных, как части корпоративной информационной системы, было принято. Однако не стоит забывать, что средства моделирования IDEF1X специально разработаны для построения реляционных информационных систем, и если существует необходимость проектирования другой системы, скажем объектно-ориентированной, то лучше избрать другие методы моделированиях.

Основными компонентами в методике IDEF1X являются сущности, отношения и атрибуты.

Сущность определяют как множество объектов, обладающих общими свойствами. Конкретные элементы этого множества называют экземплярами сущности.

Отношения между сущностями в IDEF1X являются бинарными отношениями. Выделяют идентифицирующие отношения – связи типа родитель-потомок, в которых потомок (зависимая сущность) однозначно определяется своей связью с родителем, и неидентифицирующие отношения, означающие, что у связанного этим отношением экземпляра одной сущности может быть, а может и не быть соответствующего экземпляра второй сущности.

Идентифицирующее отношение изображают на IDEF1X-диаграмме сплошной линией между прямоугольниками связанных сущностей, неидентифицирующее отношение показывают пунктирной линией. На дочернем конце линии должно быть утолщение (жирная точка).

Различают также специфические и неспецифические отношения. Специфические отношения — это связи "один ко многим", а неспецифические — связи типа "многие ко многим".

Свойства сущностей называют атрибутами. Различают ключевые и неключевые атрибуты. Значение ключевого атрибута (ключа) однозначно идентифицирует экземпляр сущности. Ключевые атрибуты могут быть составными.

Внешний ключ — это атрибут, входящий в ключ родителя и наследуемый потомком. На IDEF1X-диаграммах ключи записывают в верхней части прямоугольника сущности, причем внешние ключи помечают меткой FK (Foreign Key), неключевые атрибуты помещают в нижнюю часть прямоугольников. В идентифицирующих отношениях все ключи родителя входят и в ключи потомка, в неидентифицирующих ключи родителя относятся к неключевым атрибутам потомка.

Информационная модель программного средства представлена на рисунке 4.5.

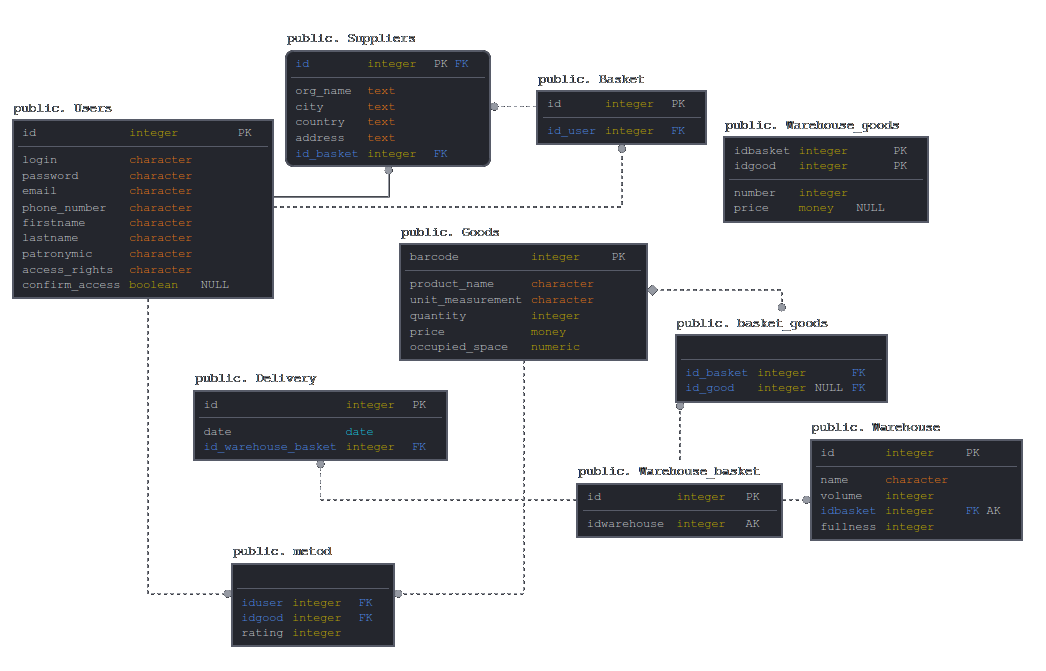


Рисунок 4.5 – Информационная модель программного средства

Информационная модель программного средства состоит из следующих сущностей: Users, Suppliers, Basket, Warehouse\_goods, Goods, basket\_goods, Delivery, Warehouse\_basket, Warehouse, metod.

Сущность «Users» предназначена для хранения данных о человеке, который пользуется системой. Данная сущность содержит поля: «id», «login», «password», «email», «phone\_number», «firstname», «lastname», «patronymic», «access\_rights» и «confirm\_access». Поле «id» является первичным ключом сущности. Поля «access\_rights» и «confirm\_access» предназначены для классификации и подтверждения пользователей к системе.

Сущность «Suppliers» предназначена для хранения информации о поставщиках. Данная сущность содержит поля: «id», «org\_name», «city», «country», «address» и «id\_basket». Поле «id» является первичным ключом сущности и имеет внешний ключ на «id» из таблицы «Users».

Сущность «Goods» предназначена для хранения информации о товарах. Данная сущность содержит поля: «barcode», «product\_name», «unit\_measurement», «quantity», «price» и «occupied\_space». Поле «barcode» является первичным ключом сущности.

Сущность «Basket» предназначена для хранения списка товаров, которые есть у поставщиков. Данная сущность содержит поля: «id» и «id\_user». Поле «id» предназначена для нумерации корзины и является первичным ключом сущности «Supplier». Поле «id\_user» имеет внешний ключ на таблицу поставщиков и хранит id поставщика.

Сущность «Warehouse\_basket» предназначена для хранения списка товаров на складе. Данная сущность содержит поля: «id» и «idwarehouse». Поле «id» является первичным ключом сущности.

Сущность «Delivery» предназначена для хранения информации о поставках на склады логистической компании. Данная сущность содержит поля: «id», «date» и «id\_warehouse\_basket». Поле «id» является первичным ключом сущности. Поле «id\_warehouse\_basket» имеет внешний ключ на поле «id» таблицы сущности «Warehouse\_basket».

Сущность «Warehouse» предназначена для хранения информации о складах логистической компании. Данная сущность содержит поля: «id», «name», «volume», «idbasket» и «fullness». Поле «id» является первичным ключом сущности. Поле «idbasket» имеет внешний ключ на поле «id» таблицы сущности «Warehouse\_basket».

Сущность «basket\_goods» предназначена для хранения списка товаров у каждого поставщика. Данная сущность содержит поля: «id\_basket» и «id\_good». Поля «id\_good» и «id\_basket» имеют внешние ключи на поле «id» таблицы сущности «Warehouse\_basket» и поле «barcode» таблицы сущности «Goods».

Сущность «metod» предназначена для хранения оценок экспертов. Данная сущность содержит поля: «iduser», «idgood» и «rating». Поля «iduser» и «idgood» имеют внешние ключи на поле «id» таблицы сущности «Users» и поле «barcode» таблицы сущности «Goods».

Сущность «Warehouse\_goods» предназначена для хранения списка товаров на каждом складе. Данная сущность содержит поля: «idbasket», «idgood», «number» и «price». Поля «idbasket» и «idgood» являются первичными ключами. Поля «number» и «price» содержат информацию о количестве товара и его цене реализации.

Нормализация – это систематический подход к разложению таблиц для устранения избыточности данных (повторения) и нежелательных характеристик, таких как аномалии вставки, обновления и удаления. Это многоэтапный процесс, который помещает данные в табличную форму, удаляя дублированные данные из таблиц отношений.

Если таблица не нормализована должным образом и имеет избыточность данных, она не только расходует дополнительный объем памяти, но также затрудняет обработку и обновление базы данных без потери данных. Аномалии вставки, обновления и удаления очень часты, если база данных не нормализована [7].

Правила нормализации выделяют следующие нормальные формы:

* первая нормальная форма;
* вторая нормальная форма;
* третья нормальная форма;
* нормальная форма Бойса-Кодда;
* четвертая нормальная форма.

Созданная модель находится в третьей нормальной форме, так как каждое поле, не входящее в первичный ключ, функционально полно зависит от первичного ключа, а также ни одно из полей, не являющихся ключом, не зависит функционально от любого другого не ключевого поля. И каждая строка содержит в своём поле не более одного значения. Таким образом, итоговая модель удовлетворяет условиям для разработки необходимой системы.

# Проектирование системы логистической компании на основе UML

Унифицированный язык моделирования (UML) является стандартным инструментом для создания «чертежей» информационных систем (ИС). С помощью UML можно визуализировать, специфицировать, конструировать и документировать элементы этих систем.

UML пригоден для моделирования любых систем: от информационных систем масштаба предприятия до распределенных Web-приложений и даже встроенных систем реального времени. Это очень выразительный язык, позволяющий рассмотреть систему со всех точек зрения, имеющих отношение к ее разработке и последующему развертыванию. Несмотря на обилие выразительных возможностей, этот язык прост для понимания и использования. Изучение UML удобнее всего начинать с его концептуальной модели, которая включает в себя три основных элемента: базовые строительные блоки, правила, определяющие, как эти блоки могут сочетаться между собой, и некоторые общие механизмы языка [5].

UML является одной из составляющих процесса разработки ИС. Хотя UML не зависит от моделируемой реальности, лучше всего применять его, когда процесс моделирования основан на рассмотрении прецедентов использования, является итеративным и пошаговым, а сама система имеет четко выраженную архитектуру.

UML – это язык для визуализации, специфицирования, конструирования и документирования элементов программных систем. Язык состоит из словаря и правил, позволяющих комбинировать входящие в него слова и получать осмысленные конструкции. В языке моделирования словарь и правила ориентированы на концептуальное и физическое представление системы. Язык моделирования, подобный UML, является стандартным средством для составления «чертежей» ИС [6].

Словарь UML включает три вида строительных блоков:

* Диаграммы;
* Сущности;
* Связи.

## **Спецификация вариантов использования системы логистической компании**

Диаграмма вариантов использования  – диаграмма, отражающая отношения между [актерами](https://ru.wikipedia.org/w/index.php?title=%D0%90%D0%BA%D1%82%D0%B5%D1%80_(UML)&action=edit&redlink=1) и [прецедентами](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D1%86%D0%B5%D0%B4%D0%B5%D0%BD%D1%82_(UML)) и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Прецедент – возможность моделируемой системы (часть её функциональности), благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат. Прецедент соответствует отдельному сервису системы, определяет один из вариантов её Ыиспользования и описывает типичный способ взаимодействия пользователя с системой.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества актеров, взаимодействующих с системой с помощью так называемых вариантов использования. При этом актером называется любой объект, субъект или система, взаимодействующая с моделируемой системой извне. В свою очередь вариант использования – это спецификация функций, которые система предоставляет актеру. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемых системой при взаимодействии с актером. При этом в модели никак не отражается то, каким образом будет реализован этот набор действий.

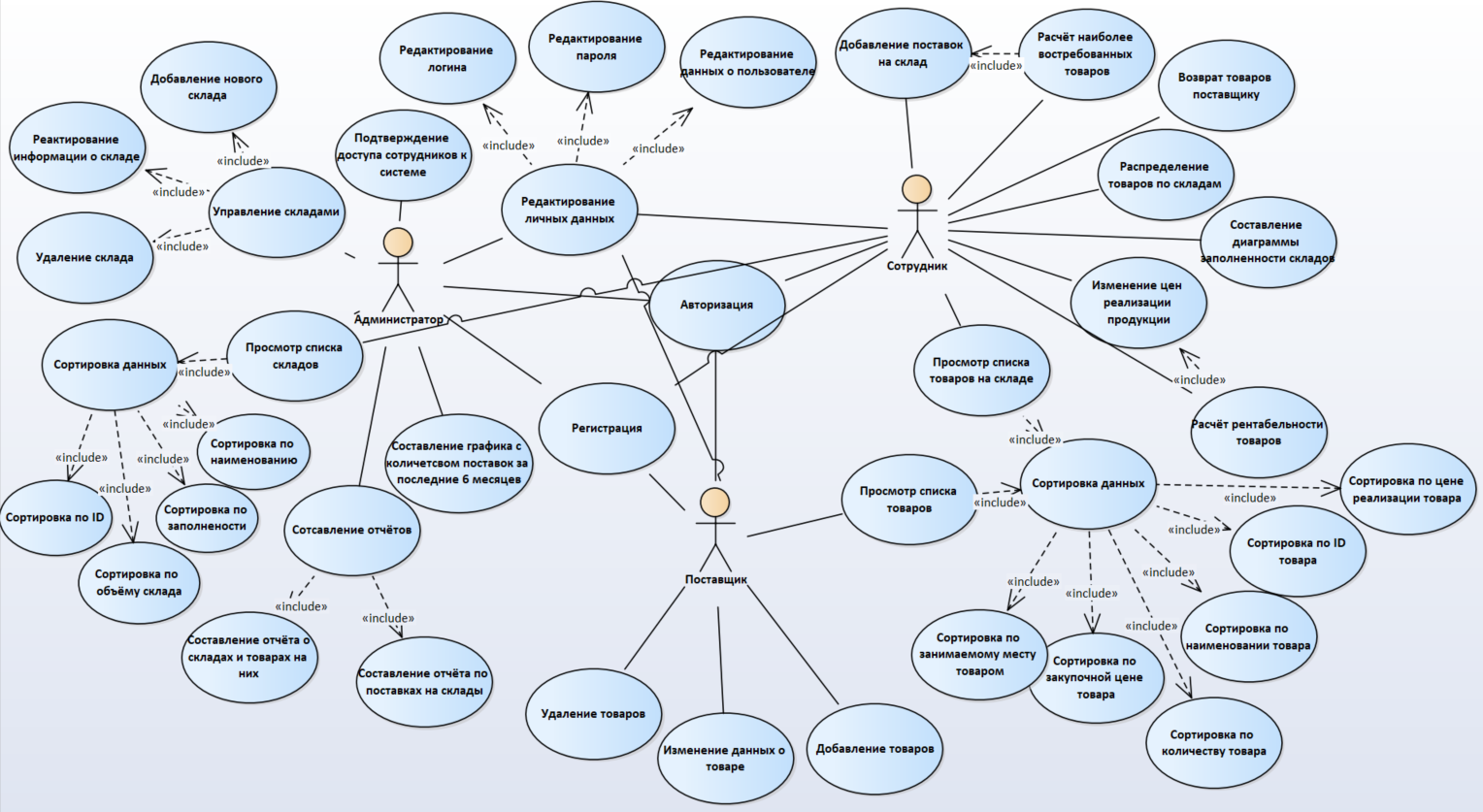
Диаграмма вариантов использования данной программы представлена на рисунке 5.1.

Рисунок 5.1 – Диаграмма вариантов использования

Сущностями (актёрами) в программном средстве системы управления складами логистической компании являются «Администратор», «Сотрудник» и «Поставщик».

Сущности «Администратор» доступны следующие варианты использования:

* управление складами;
* подтверждение доступа сотрудников к системе
* редактирование личных данных;
* авторизация;
* регистрация;
* составление отчётов;
* просмотр списка складов;
* составление графика с количеством поставок за последние 6 месяцев.

Сущности «Сотрудник» доступны следующие варианты использования:

* авторизация;
* регистрация;
* редактирование личных данных;
* добавление поставок на склад;
* расчёт наиболее востребованных товаров;
* возврат товаров поставщику;
* распределение товаров по складам;
* составление диаграммы заполненности складов;
* изменение цен реализации продукции;
* просмотр списка товаров на складе;
* расчёт рентабельности товаров.

Сущности «Поставщик» доступны следующие варианты использования:

* авторизация;
* регистрация;
* редактирование личных данных;
* просмотр списка товаров;
* добавление товаров;
* изменение данных о товарах;
* удаление товаров.

## **Описание диаграммы последовательности создания отчёта «Отчёт о складах и товарах на них»**

Диаграмма последовательностей относится к диаграммам взаимодействия UML, описывающим поведенческие моменты системы, взаимодействие объектов в течение определённого отрезка времени.

Диаграмма последовательности отражает поток событий, происходящих в рамках варианта использования. На этой диаграмме изображаются только те объекты, которые непосредственно участвуют во взаимодействии так как ключевым моментом является именно динамика взаимодействия объектов во времени и не используются возможные статистические ассоциации с другими.

Все действующие лица показаны в верхней части диаграммы. Стрелки соответствуют сообщениям, передаваемым между действующим лицом и объектом или между объектами для выполнения требуемых функций.

Как было сказано выше, взаимодействие между актёрами отображается при помощи специальных стрелок, передающих управление от отправителя (от кого идёт стрелка) к получателю (тот, к кому направлена стрелка). Стрелки демонстрируют ход сценария и те события, которые происходят во время анализируемого прецедента. Всего существует 5 видов стрелок:

* синхронное сообщение – актёр-отправитель передаёт ход управления актёру-получателю, которому необходимо провести в прецеденте некоторое действие. Пока проводимое актёром-получателем действие не будет завершено (соответственно, не будет получено ответное сообщение), актёр-отправитель теряет возможность производить какие-либо действия. Графически изображается как сплошная линия со стрелкой в виде закрашенного треугольника, после которой идёт прямоугольник, отражающий деятельность объекта, в конце которого находится ответное сообщение;
* ответное сообщение – данное сообщение является ответом на синхронное сообщение. Обычно, содержит какое-либо возвращаемое изначальному актёру-отправителю значение, также возвращающее ему управление (возможность действовать). Графически изображается пунктирной линией с открытой стрелкой;
* асинхронное сообщение – актёр-отправитель передаёт ход управления актёру-получателю, которому необходимо провести в прецеденте некоторое действие. Основное отличие от синхронного сообщения состоит в том, что актёр-отправитель не теряет возможности совершать другие действия. Графически изображается сплошной линией с открытой стрелкой.
* потерянное сообщение – сообщение без адресата (есть отправитель, нет получателя);
* найденное сообщение – сообщение без отправителя.

Пример сообщений на диаграмме последовательности предоставлен на рисунке 5.2.



Рисунок 5.2 – Пример сообщений на диаграмме последовательности

Диаграмма последовательности построение отчёта «Отчёт о складах и товарах на них» предоставлена на рисунке 5.3.

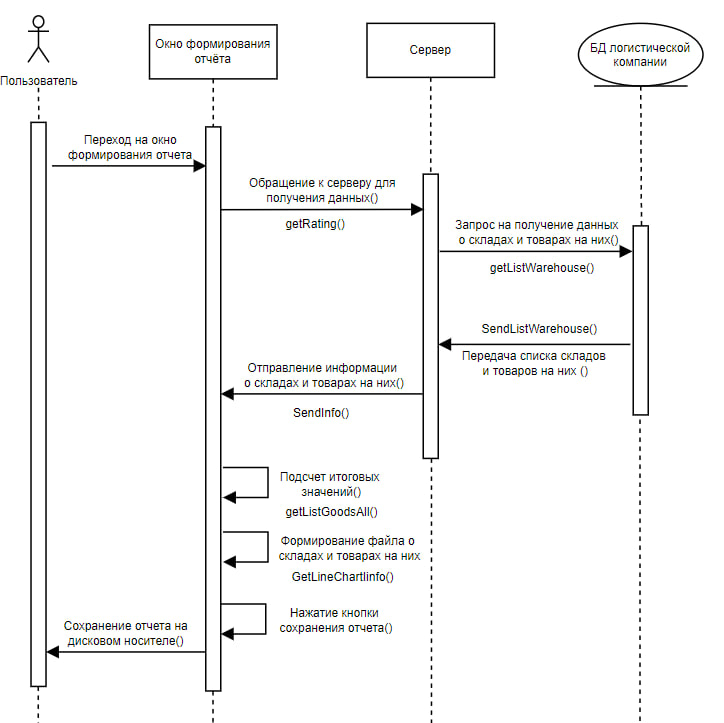


Рисунок 5.3 – Диаграмма последовательности создания отчёта «Отчёт о складах и товарах на них»

Диаграмма последовательности создания отчёта «Отчёт о складах и товарах на них» состоит из следующих объектов: пользователь, окно формирования отчёта, сервер и база данных логистической компании.

Пользователь нажимает кнопку формирования отчёта об складах и товарах, после чего серверу отправляется запрос на выборку данных об складах и товарах на них. Полученные данные сервер отправляет в приложение, в котором происходит подсчёт итоговых данных и формирование отчёта. Далее при нажатии на кнопку сохранения отчёт будет сохранён на дисковой носитель.

## **Описание диаграммы состояний построения графика «Количество поставленных товаров за последние 6 месяцев»**

Диаграмма состояний – это, по существу, [диаграмма состояний](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0_%D1%81%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D0%B9_(%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%BE%D0%B2)) из теории автоматов со стандартизированными условными обозначениями, которая может определять множество систем от компьютерных программ до [бизнес-процессов](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81). Используются следующие условные обозначения:

* круг, обозначающий начальное состояние;
* окружность с маленьким кругом внутри, обозначающая конечное состояние;
* скруглённый прямоугольник, обозначающий состояние. Верхушка прямоугольника содержит название состояния. В середине может быть горизонтальная линия, под которой записываются активности, происходящие в данном состоянии;
* стрелка, обозначающая переход. Название события, вызывающего переход, отмечается рядом со стрелкой. [Охраняющее выражение](https://ru.wikipedia.org/wiki/%D0%9E%D1%85%D1%80%D0%B0%D0%BD%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) может быть добавлено перед «/» и заключено в квадратные скобки, что значит, что это выражение должно быть истинным, чтобы переход имел место. Если при переходе производится какое-то действие, то оно добавляется после «/»;
* толстая горизонтальная линия с либо множеством входящих линий и одной выходящей, либо одной входящей линией и множеством выходящих. Это обозначает объединение и разветвление соответственно.

Диаграмма состояний построения графика «Количество поставленных товаров за последние 6 месяцев» представлена на рисунке 5.4.

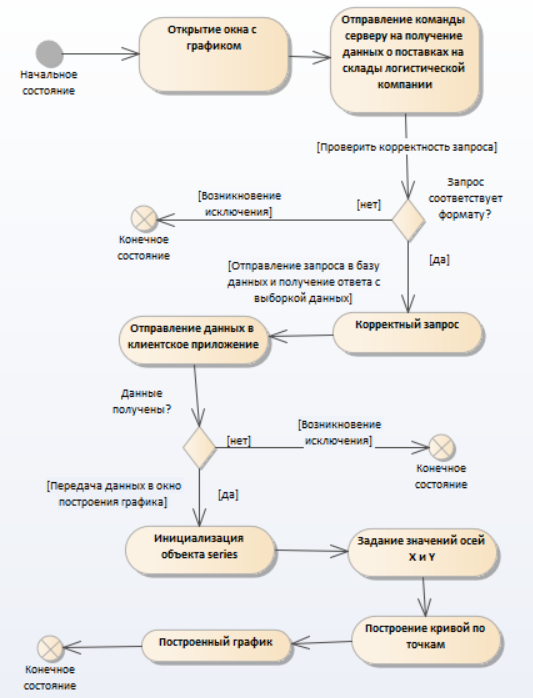


Рисунок 5.4 – Диаграмма состояний построения графика «Количество поставленных товаров за последние 6 месяцев»

Для построения графика «Количество поставленных товаров за последние 6 месяцев» необходимо перейти в окно построения графика, после чего выполняется запрос на сервер для получения данных о поставках логистической компании. В свою очередь сервер делает запрос в базу данных о поставках компании за последние 6 месяцев и отправляет полученные данные на клиентское приложение. В случае некорректности запроса или ошибки выборки из базы данных, возникнет исключение в приложении.

После получения клиентским приложением данных о поставках, выполняется инициализация объекта типа Series, именование осей X и Y и построение кривой линии по точкам. В результате чего получаем график количества поставленных товаров за последние 6 месяцев.

## **Диаграмма компонентов**

Диаграмма компонентов – это, по сути, диаграмма классов, которая фокусируется на компонентах системы, которые часто используются для моделирования представления статической реализации системы.

Грамотное использование диаграммы компонентов позволяет:

– представить физическую структуру компонентов;

– обратить внимание на компоненты системы и как они связаны;

– акцентировать внимание на поведение системы в части, касающейся интерфейса.

Диаграмма компонентов данного проекта представлена на рисунке 5.5.

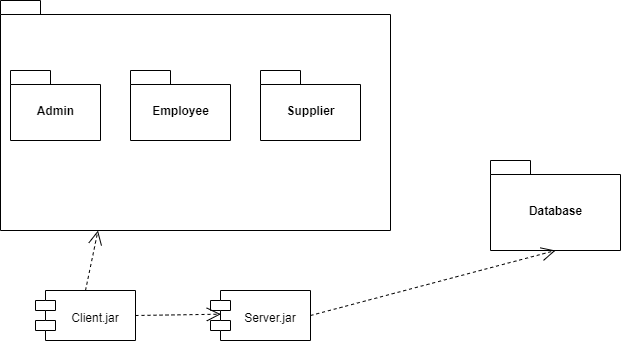


Рисунок 5.5 – Диаграмма компонентов

С помощью данной диаграммы можно определить архитектуру разрабатываемой системы, а также просмотреть зависимости между программными компонентами.

## **Диаграмма развёртывания**

В контексте унифицированного языка моделирования (UML) диаграмма развертывания относится к семейству структурных диаграмм, поскольку она описывает аспект самой системы. В этом случае схема развертывания описывает физическое развертывание информации, генерируемой программным обеспечением на аппаратных компонентах.

Диаграмма развертывания показывает физическое расположение системы по отношению к друг другу, показывая, на каком оборудовании оно запускается и способы взаимодействия частей системы. Данная диаграмма представлена на рисунке 5.6.

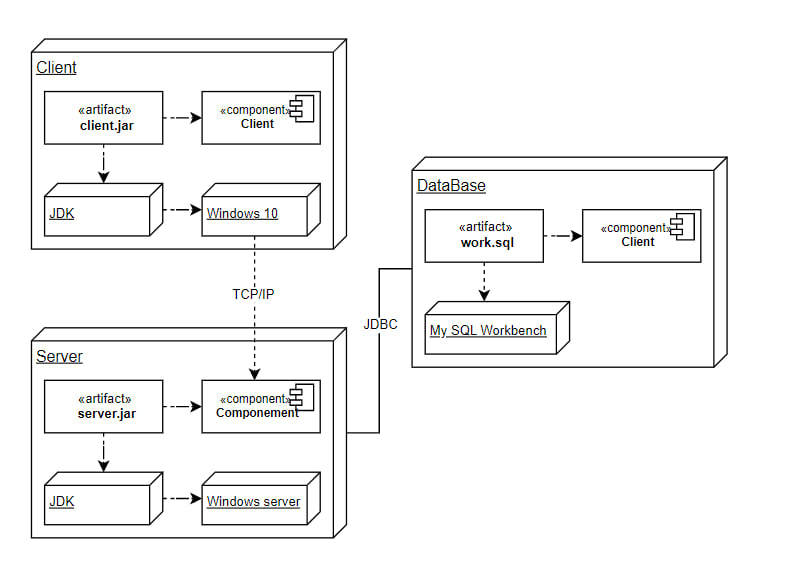


Рисунок 5.6 – Диаграмма развертывания

Диаграмма развертывания показывает все существующие аппаратные компоненты приложения, а именно сервер с базой данным на одном персональном компьютере, а клиент на другом. Также благодаря этой диаграмме можно увидеть, какие программные компоненты работают на каждом узле.

## **Описание диаграммы классов**

Диаграмма классов – структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их коопераций, атрибутов (полей), методов, интерфейсов и взаимосвязей между ними. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования.

Целью создания диаграммы классов является графическое представление статической структуры декларативных элементов системы (классов, [типов](https://ru.wikipedia.org/wiki/%D0%A2%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) и т. п.). Она содержит в себе также некоторые элементы поведения (например – операции), однако их динамика должна быть отражена на диаграммах других видов ([диаграммах коммуникации](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0_%D0%BA%D0%BE%D0%BC%D0%BC%D1%83%D0%BD%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D0%B8), диаграммах состояний).  При представлении сущностей реального мира разработчику требуется отразить их текущее состояние, их поведение и их взаимные отношения.

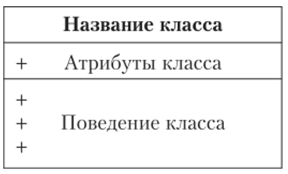


Рисунок 5.7 – Изображение класса в UML

Из рисунка 5.7 видно, что класс изображается в UML в виде прямоугольника, состоящего из трех разделов. Верхний раздел содержит название класса, средний – атрибуты класса, а нижний – варианты поведения класса.

Диаграмма классов серверного программного средства представлена на рисунке Б.1.

Класс ClientSocket используется для реализации многопоточности сервера. В нём выделяется сокет для работы с клиентом, который освобождается при отключении клиента. В данном классе реализоват метод sendData(), который определяет как отправлять сообщения на клиентское приложение.

Класс App содержит метод main(), который является точкой входа приложения, и метод run(), который выделяет сокет для работы сервера и создает объект класса ClinetSocket для реализации многопоточности и взаимодействие с клиентским приложением, а также определяет каким образом будет выполнятся обработка команд на сервере.

Класс DBConst содержит статические поля с наименованиями таблиц и полей таблиц и предназначен для удобства написания запросов в базу данных.

Класс DatabaseHandler хранит в себе поля с настройками подключения клиентских приложений к серверному, а также настройки для подключения к базе данных. Данный класс содержит поля для выполнения запросов в базу данных и получения результатов выборки данных.

Класс AnswerHelper содержит статические методы, которые предназначены для формирования ответов за запросы клиентских приложений.

Класс WarehouseWorker содержит методы для работы с данными о складах в базе данных.

Класс UserWorker содержит методы для работы с данными пользователей в базе данных.

Класс GoodWorker содержит методы для работы с данными товаров в базе данных.

Диаграмма классов клиентского программного средства представлена на рисунке Б.2 и Б.3.

Класс MainApp содержит метод main(), который является точкой входа приложения, и метод run(), который выделяет сокет для работы c сервером, а также отслеживает ответы сервера. Данный класс содержит метод sendData() для отправки запроса на серверное приложение.

Классы ResponceHandler и RequestHandler предназначены для обработки ответов и формирования запросов на серверное приложение.

Классы Supplier, Warehouse, User, Good и WarehouseGood предназначены для работы с данными, получаемыми с серверного приложения, и содержат методы для работы с ними.

Класс Controller содержит методы для перехода от одного окна приложения к другому, а также метод для вывода сообщения на экран.

Оставшиеся классы наследуются от класса Controller и предназначены для управления работой каждого окна, представленного в клиентском приложении.

В данном главе рассмотрен унифицированный язык UML и его применение и описано построение моделей представления системы. Разработанные диаграммы упрощают дальнейшую разработку приложения и самого кода программы. В данных диаграммах визуально отображены модели представления системы логистической компании, что помогает программисту при разработке приложения.

# Описание алгоритмов, реализующих бизнес-логику системы

## **Схема алгоритма решения задачи методом ранга**

Схема алгоритма решения задачи методом ранга представлена на рисунке 7.1.

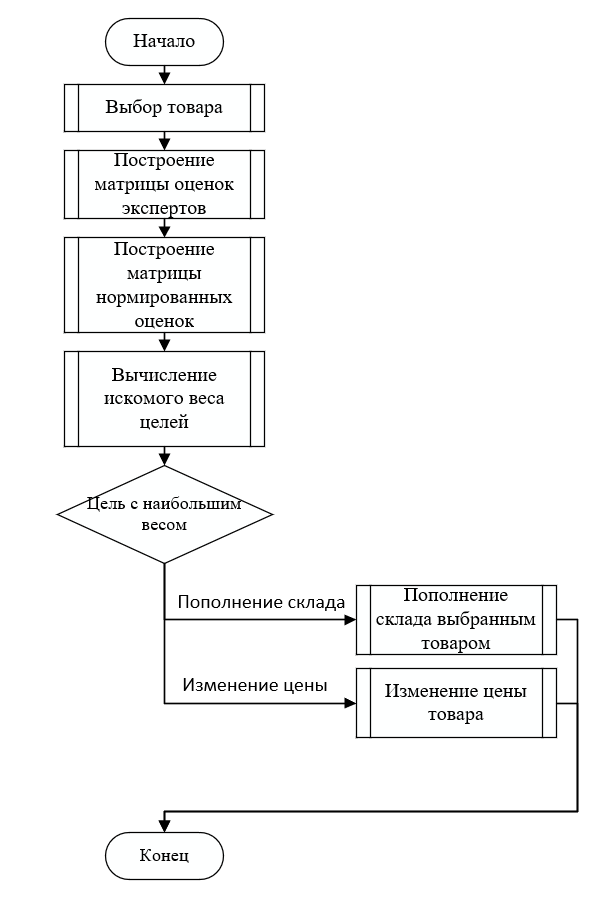


Рисунок 7.1 – Схема алгоритма решения задачи методом ранга

В начале алгоритма решения задачи сотрудник выбирает товар, над которым необходимо провести оценку. В программном средстве в табличной форме будут представлены матрицы оценок экспертов и нормированных оценок. Далее будет построена таблица с весами каждой цели и в зависимости от максимального веса будет выбрано действие, которое необходимо выполнить для улучшения показателей. Если максимальный вес будет иметь цель пополнения склада товаром, то можно перейти в меню поставок данного товара на склад. В противном случае доступен переход в меню для корректировки цены товара.

## **Схема алгоритма построения графика «Количество поставленных товаров за последние 6 месяцев»**

Для построения графика «Количество поставленных товаров за последние 6 месяцев» необходимо перейти в окно построения графика, после чего выполняется запрос на сервер для получения данных о поставках логистической компании. В свою очередь сервер делает запрос в базу данных о поставках компании за последние 6 месяцев и отправляет полученные данные на клиентское приложение. В случае некорректности запроса или ошибки выборки из базы данных, возникнет исключение в приложении.

После получения клиентским приложением данных о поставках, выполняется инициализация объекта типа Series, именование осей X и Y и построение кривой линии по точкам. В результате чего получаем график количества поставленных товаров за последние 6 месяцев.

Схема алгоритма построения графика «Количество поставленных товаров за последние 6 месяцев» представлена на рисунке 7.2.

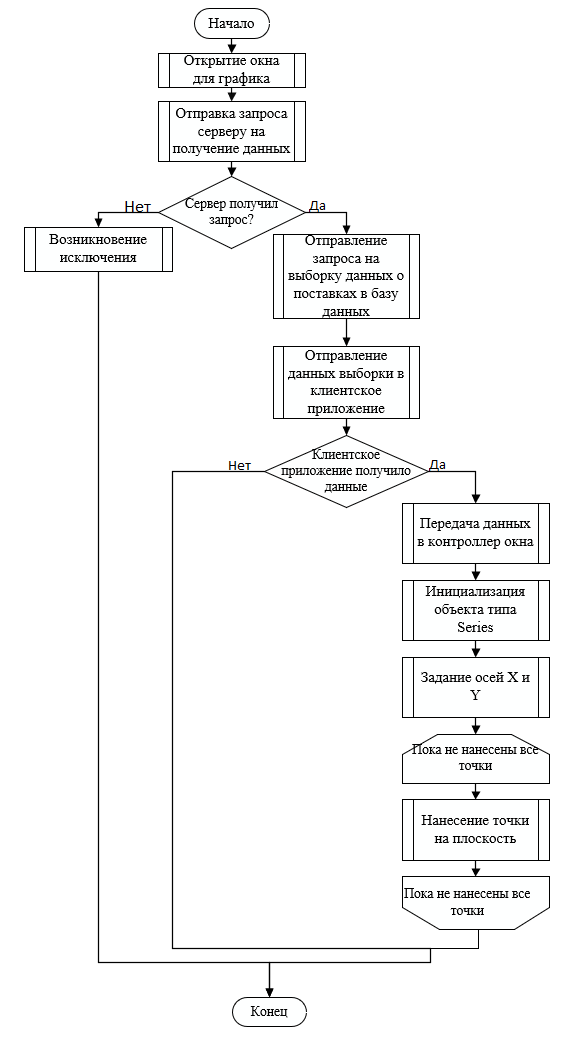


Рисунок 7.2 – Схема алгоритма построения графика «Количество поставленных товаров за последние 6 месяцев»

## **Схема алгоритма клиент-серверного взаимодействия**

Схема алгоритма клиент-серверного взаимодействия представлена на рисунке 7.3.

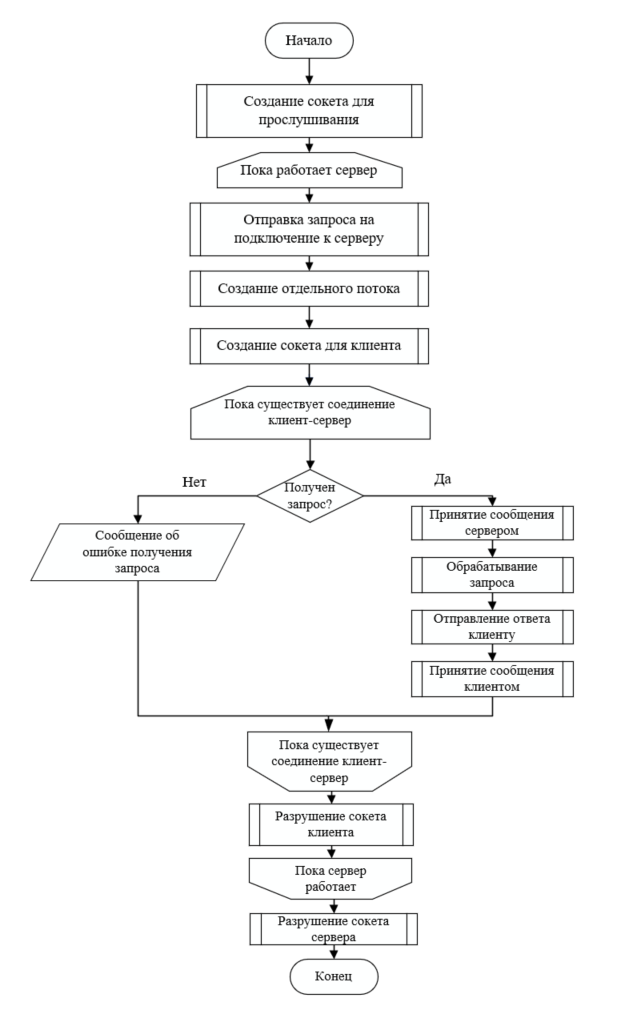


Рисунок 7.3 – Схема алгоритма клиент-серверного взаимодействия

В начале алгоритма клиент-серверного взаимодействия для клиента и сервера создаются сокеты (точки соединения). Для сокета сервера резервируется уникальная комбинация IP-адреса и порта, которая позволяет серверу обнаруживать запросы со стороны клиентов. При наличие такого запроса между сокетами клиента и сервера создаётся поток для работы с данным клиентов и устанавливается соединение, которое позволяет обмениваться данными. Сервер является многопоточным, поэтому доступно подключение нескольких клиентов. Данное соединение будет работать пока не будет прервано принудительно.

Разработанные схемы алгоритмов упрощают дальнейшую разработку приложения. В данной главе разработана схема алгоритмов принятия решения методом ранга, построения графика «Количество поставленных товаров за последние 6 месяцев», а также алгоритм клиент-серверного взаимодействия. В них простым языком описана логика работы приложения логистической компании, необходимая для реализации её деятельности.

# Руководство по развёртыванию системы логистической компании

Для возможности корректного использования системы требуется наличие на компьютере 32х или 64ч разрядной операционной системы Windows 7 и выше или Mac OS X 10.6 и выше. Данный программный продукт использует PostgreSQL для работы с базой данных. Также необходимо наличие на компьютере для клиента и сервера JDK версии 15 и выше.

Для запуска приложения необходимо последовательно запустить из консоли server.jar, а затем client.jar. При первом запуске приложения будет создана база данных, если до этого не была создана, то она сгенерируется автоматически с помощью SQL-скрипта, который находится в приложении А.

В результате описанных выше действий пользователь может корректно запустить данное программное приложение на любом удовлетворяющем требованиям устройстве.

# Руководство пользователя

После развёртывания системе на своём устройстве, описание которого описано в главе 8, пользователь видит главное меню авторизации, в котором будет предоставлен выбор, чтобы авторизовываться или зарегистрироваться.

Окно выбора действий в главном меню авторизации предоставлено на рисунке 9.1.

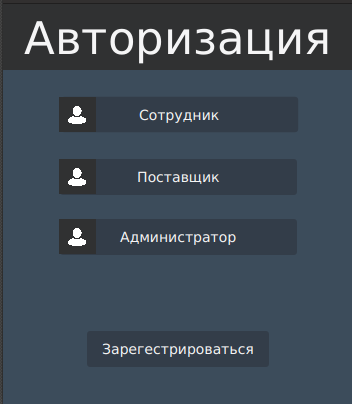


Рисунок 9.1 – Главное меню авторизации

При выборе пункта меню «Зарегистрироваться» пользователь перейдёт в меню регистрации, окно которого предоставлено на рисунке 9.2.

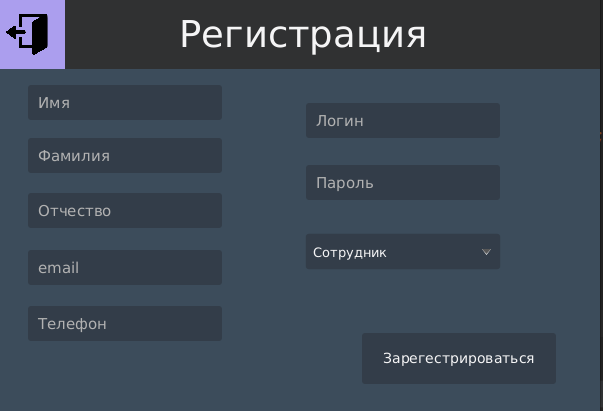


Рисунок 9.2 – Меню регистрации

В данном меню пользователю необходимо ввести личные данные регистрации, а также выбрать тип пользователя в системе. При нажатии на кнопку регистрации будет выведено сообщение об успешной регистрации или об ошибке. При выборе типа пользователя «Сотрудник» или «Администратор», то для входа в систему необходимо подтверждение доступа от существующего администратора.

После регистрации пользователь может авторизоваться в системе. Для авторизации необходимо ввести логин и пароль, которые были добавлены в систему при регистрации. Окно авторизации предоставлено на рисунке 9.3.

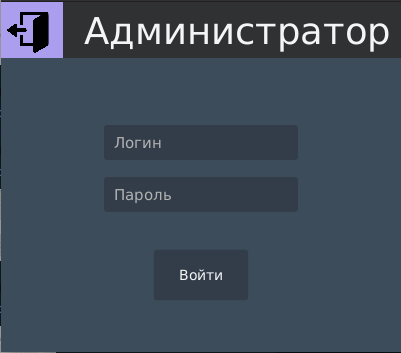


Рисунок 9.3 – Окно авторизации

При попытки авторизоваться неподтверждённого пользователя, будет выведено сообщение об ошибке, предоставленное на рисунке 9.4.

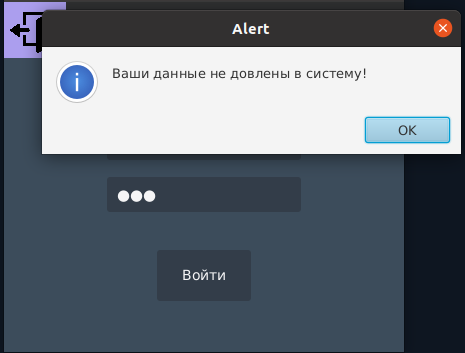


Рисунок 9.4 – Ошибка авторизации

* 1. Аккаунт администратора

При успешной авторизации администратора будет открыто главное меню администратора, предоставленное на рисунке 9.5.

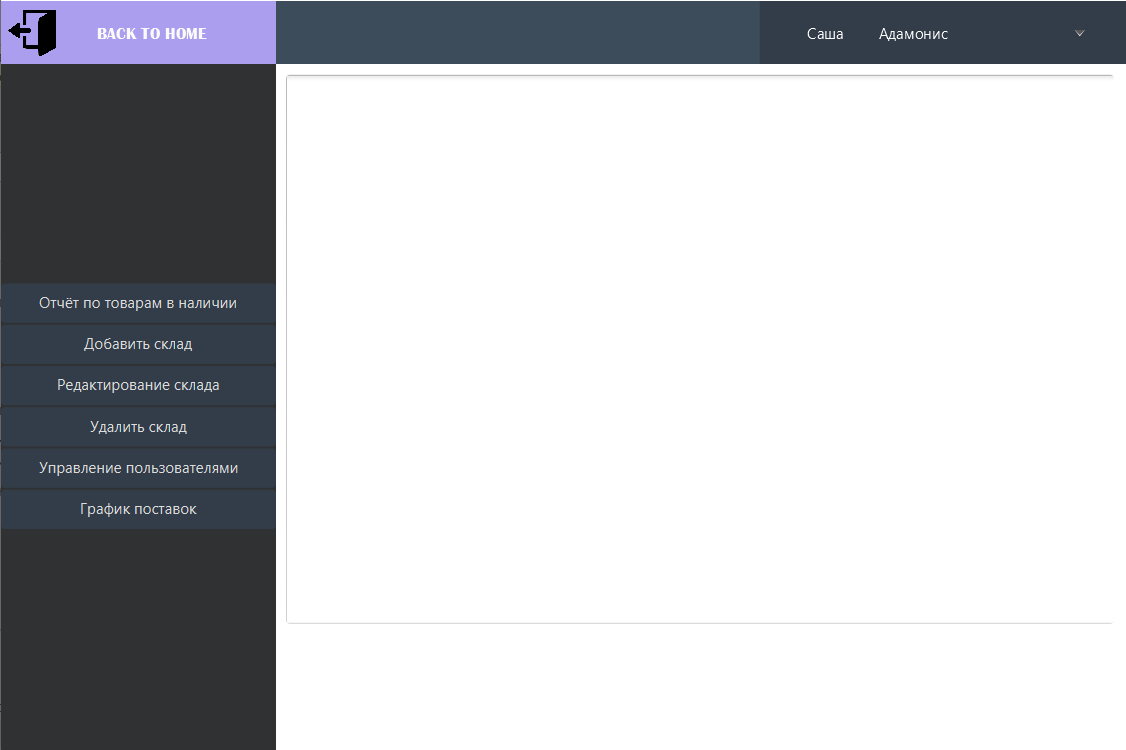


Рисунок 9.5 – Главное меню администратора

В главном меню администратора пользователю доступны функции получения отчёта по товарам на складах компании, добавления нового склада, редактирование данных складов, удаление складов, управление пользователями в системе, а также вывод графика поставок компании.

При выборе пункта меню с добавлением склада будет открыто оно добавления, в котором необходимо ввести данные склада. Окно добавления склада предоставлено на рисунке 9.6.

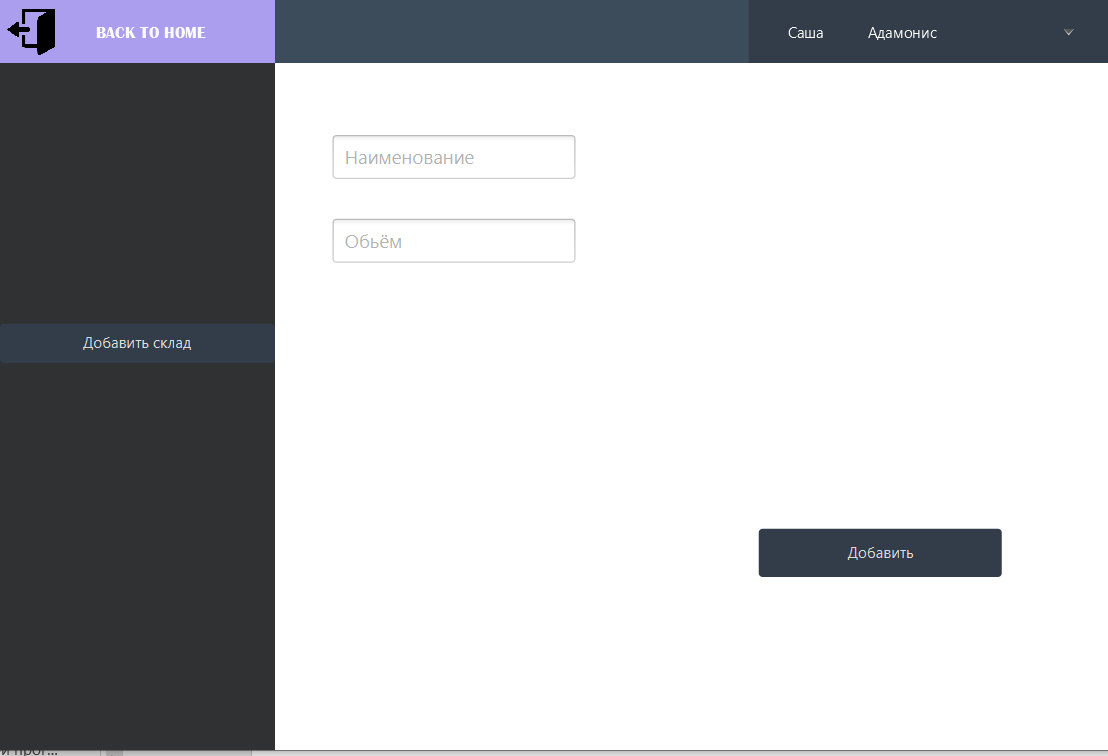


Рисунок 9.6 – Окно добавления склада

При успешном добавлении будет выведено соответствующее сообщение.

Выбрав пункт меню «Отчёт по товарам в наличии» администратор получит отчёт о товарах на складах, суммарное их количество и стоимость. Отчёт предоставлен на рисунке 9.7.

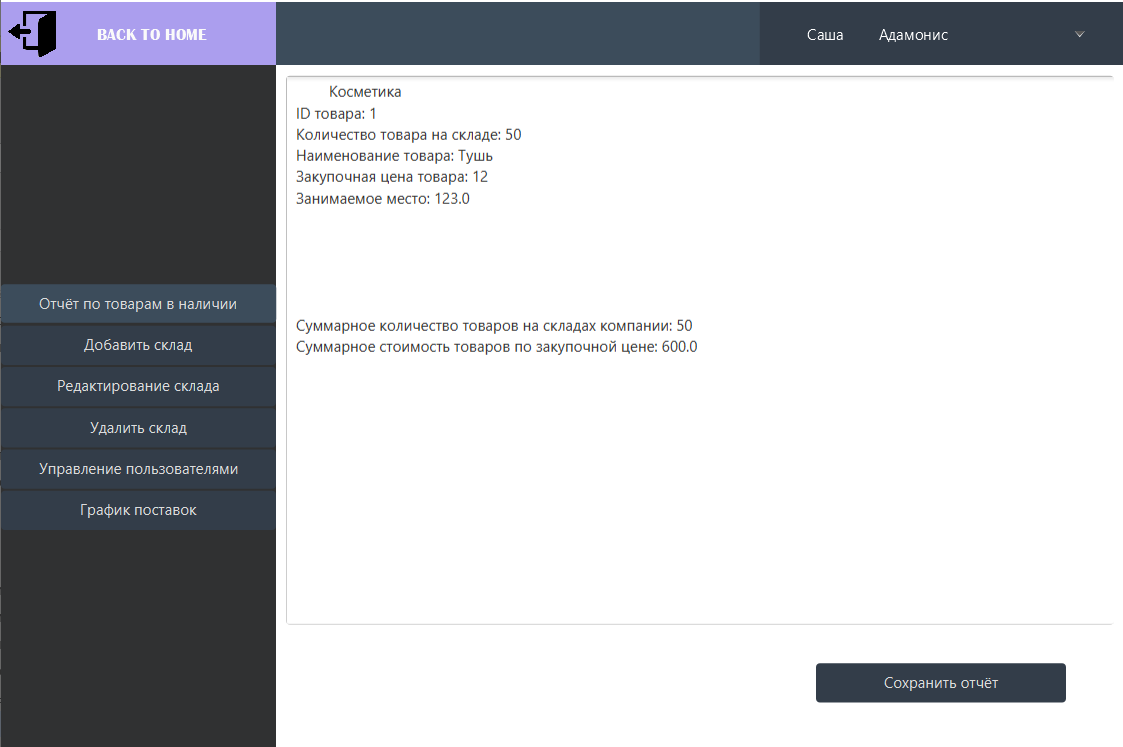


Рисунок 9.7 – Отчёт по товарам в наличии

После успешного сохранения отчёта на экран будет выведено соответствующе сообщение, предоставленное на рисунке 9.8.

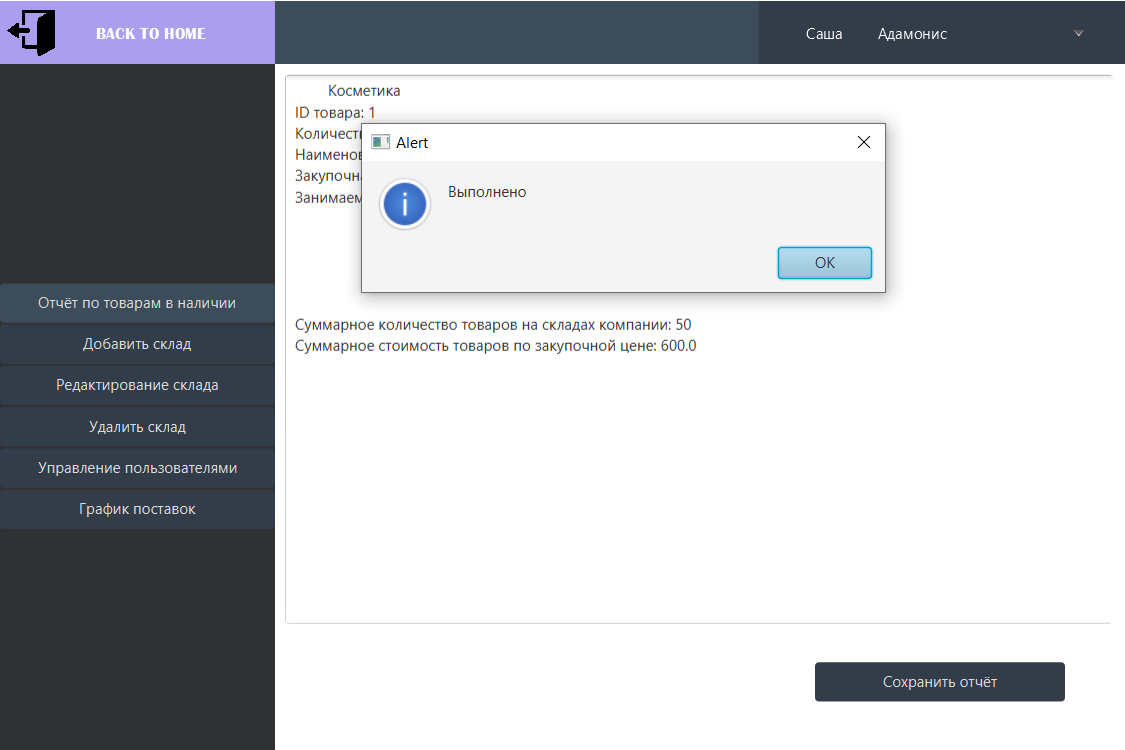


Рисунок 9.8 – Сообщение об успешном сохранении отчёта

При выборе пункта меню с редактированием информации о складах будет выведена таблица складов, в которой необходимо выбрать склад для редактирования. Вывод таблица выбора склада предоставлен на рисунке 9.9. После выбора склада будет открыто окно для редактирования, которое предоставлено на рисунке 9.10.

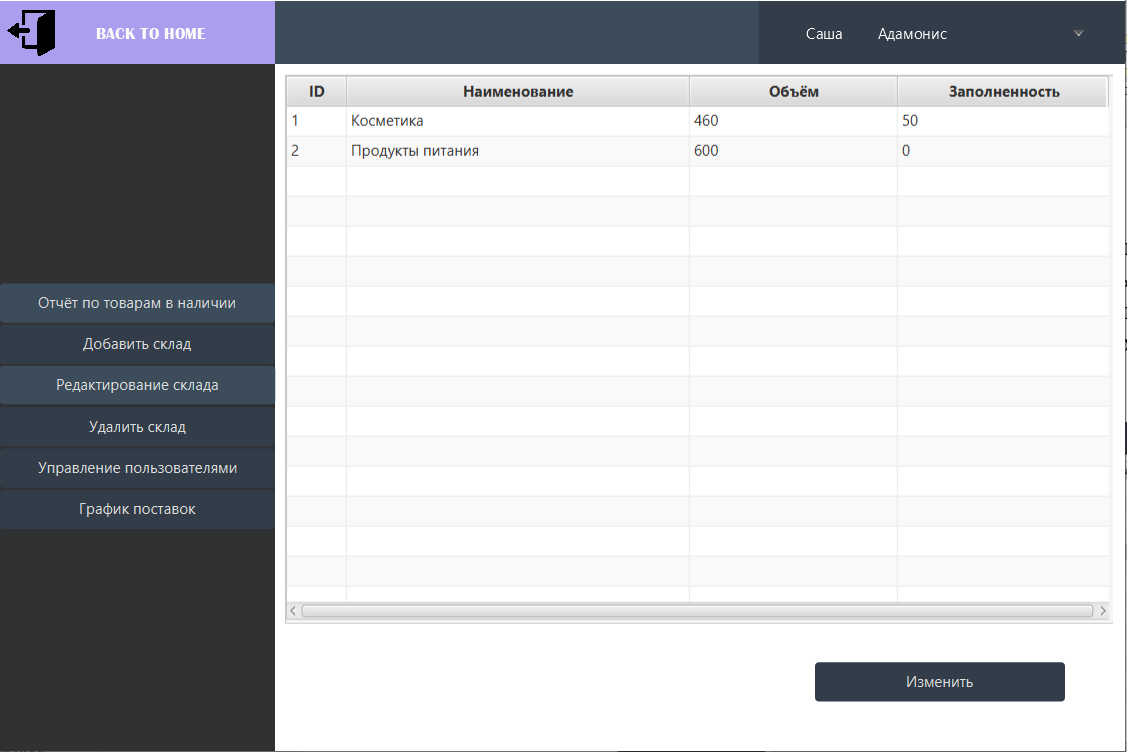


Рисунок 9.9 – Таблица выбора склада для редактирования

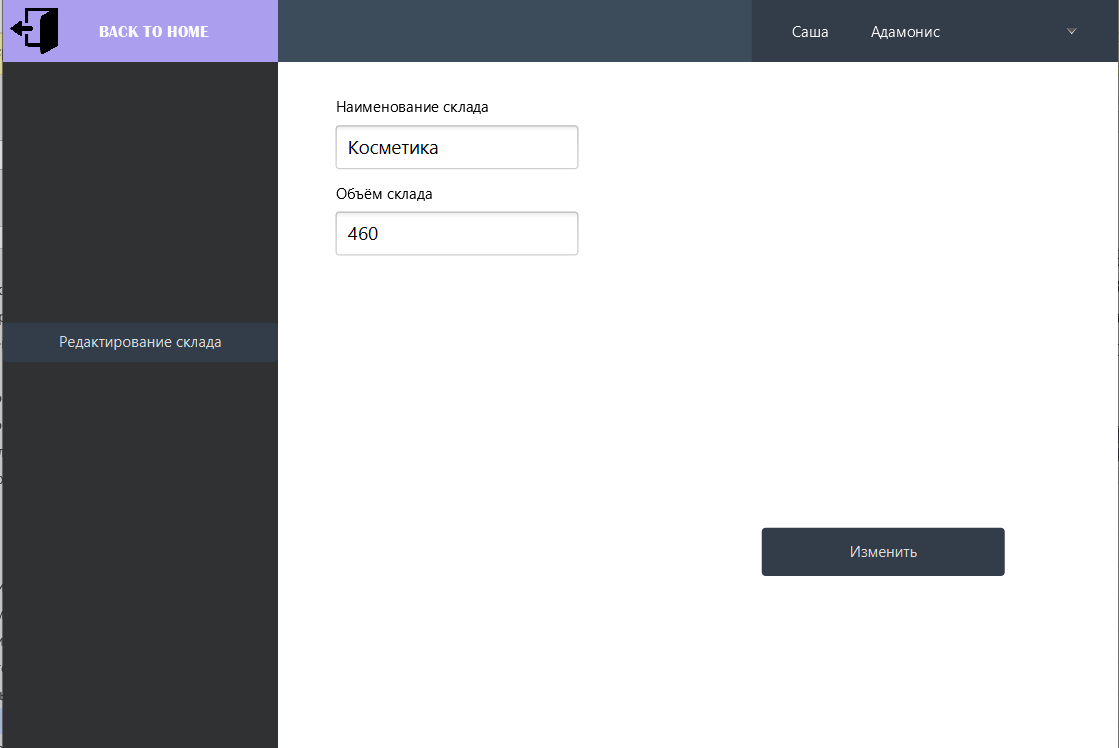


Рисунок 9.10 – Окно редактирования данных склада

Для управления данными пользователей администратор переходим в меню «Управление пользователями». Окно предоставлено на рисунке 9.11. При нажатии на кнопку управления пользователями будет открыто меню, в котором предоставлена возможность с удалением пользователей и подтверждения доступа пользователей к системе.

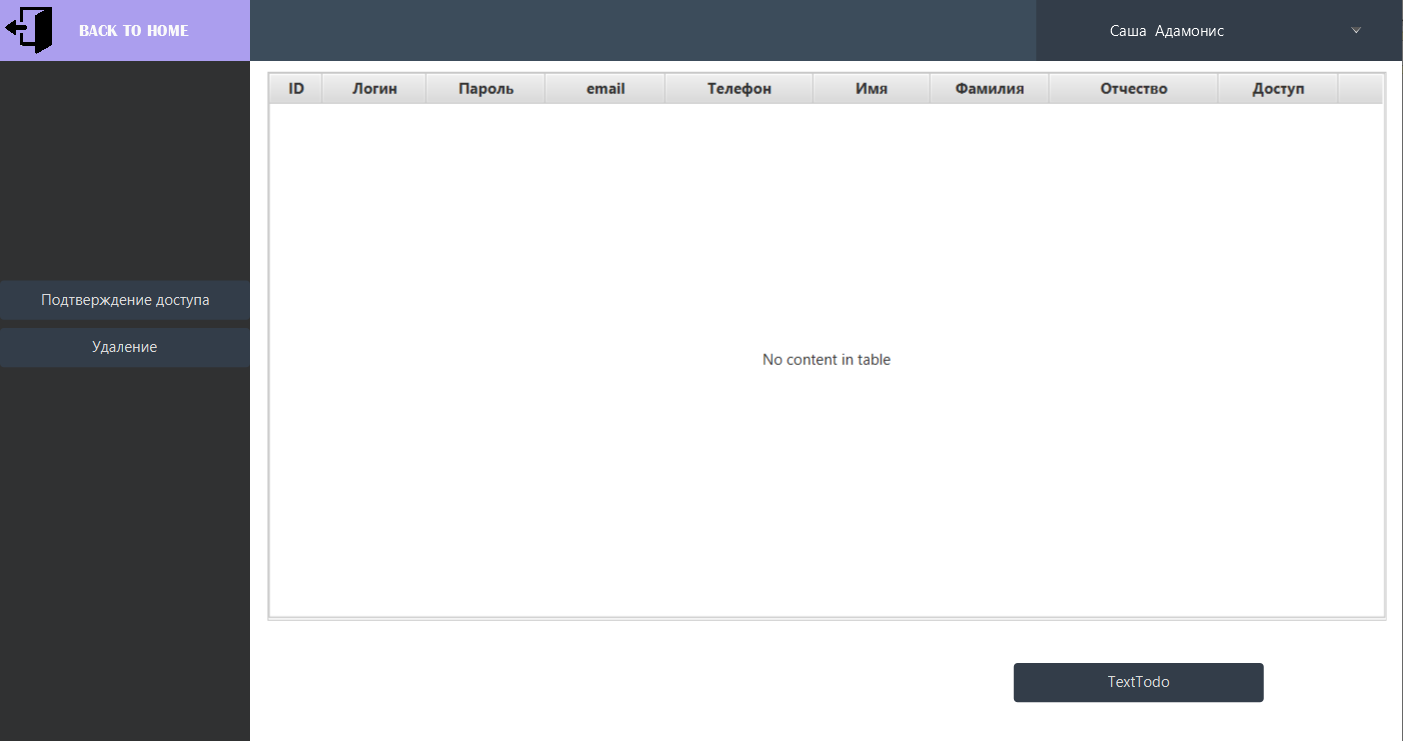


Рисунок 9.11 – Окно управления пользователями

При нажатии на кнопку управление пользователями будет выведена таблица со всеми пользователями в системе, в которой можно выбрать пользователя, которому необходимо предоставить доступ к системе. Таблица с пользователями в системе предоставлена на рисунке 9.12. При нажатии кнопки «Подтвердить» будет выведено сообщение об успешном подтверждении доступа.

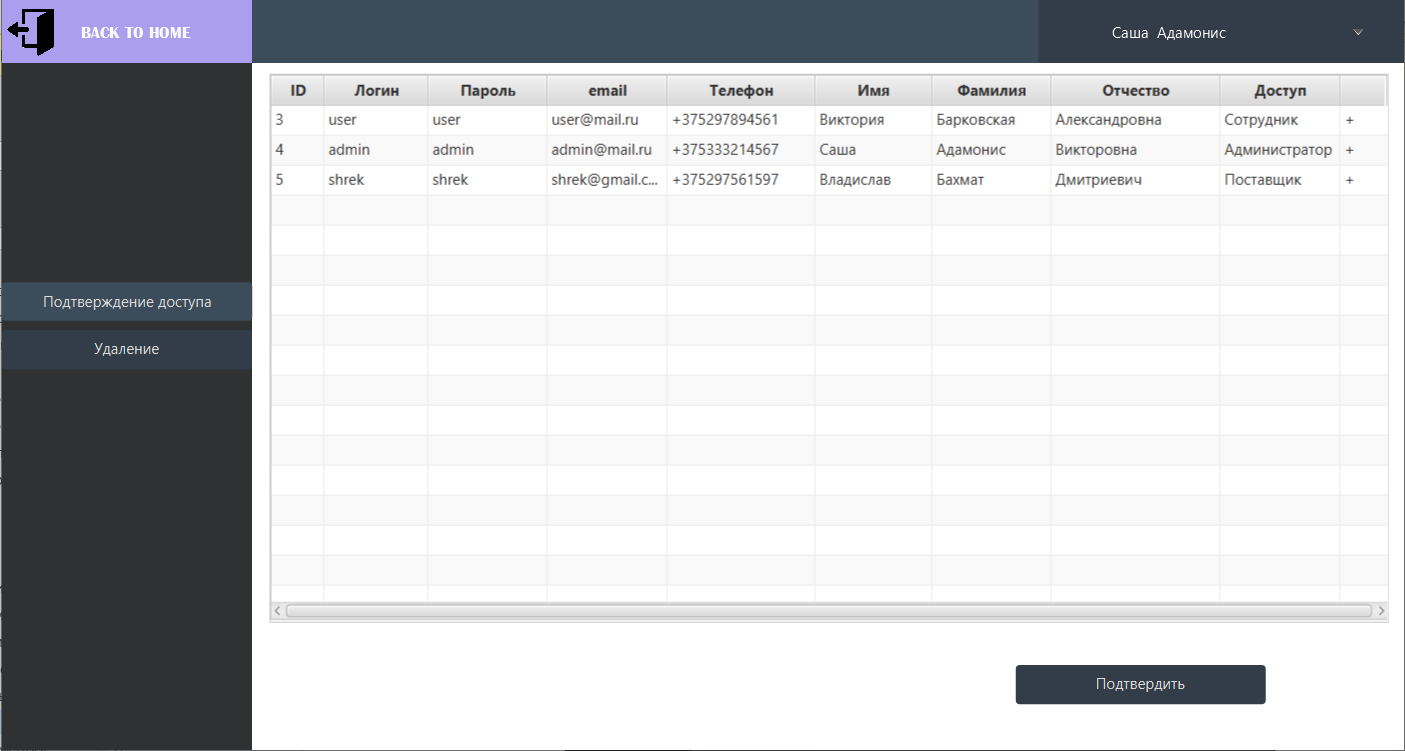


Рисунок 9.12 – Окно подтверждения доступа

При нажатии кнопки удаления пользователей будет выведена таблица пользователей. При выборе пользователя и нажатии кнопки удаления, выбранный пользователь будет удалён из системы. Окно удаления пользователей предоставлено на рисунке 9.13.

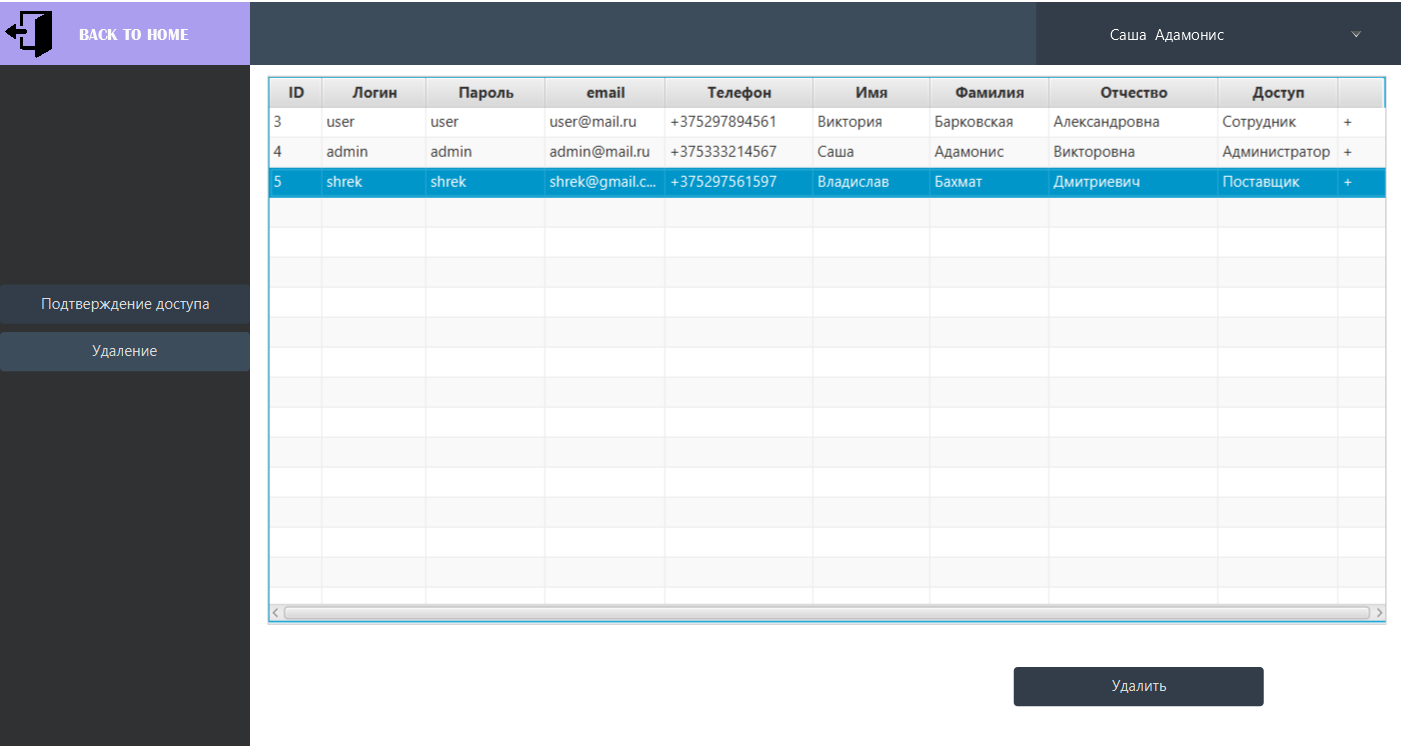


Рисунок 9.13 – Окно удаления пользователей

Выбрав пункт меню «График поставок», на экран будет выведен график количества поставленного товара за последние 6 месяцев. График предоставлен на рисунке 9.14.

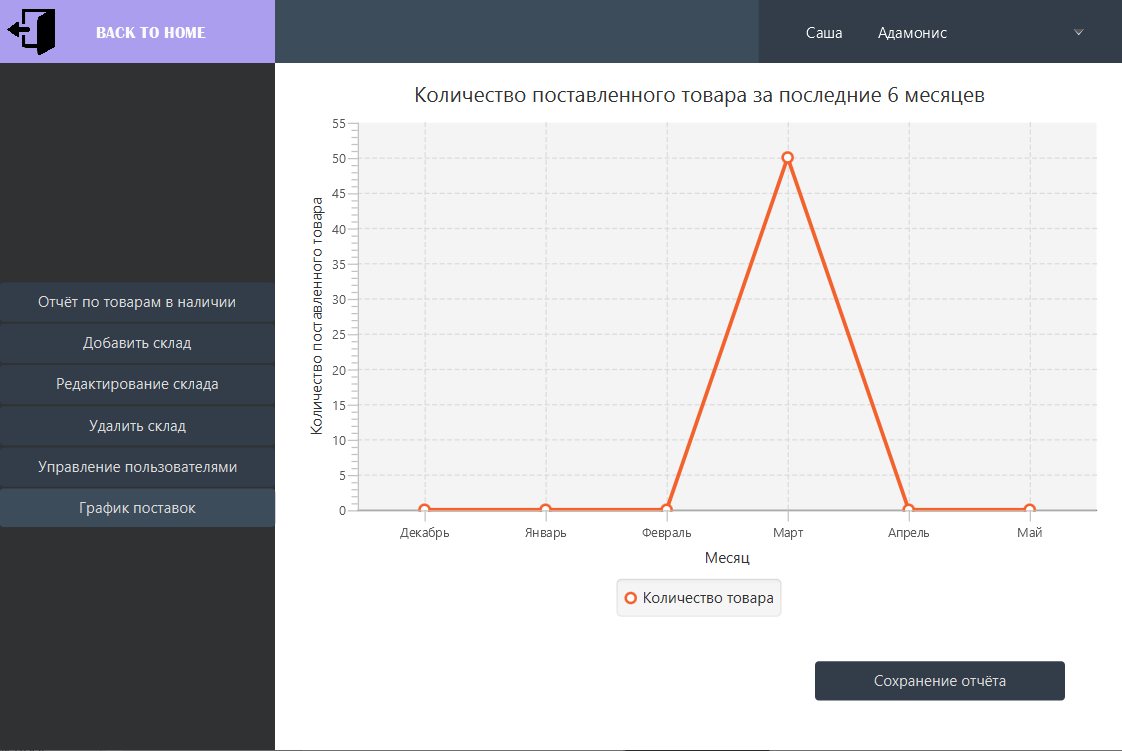


Рисунок 9.14 – График количества поставленного товара за последние 6 месяцев

При нажатии кнопки «Сохранить отчёт», отчёт о поставленных товарах будет сохранён на компьютере и будет выведено сообщение об успешном сохранении.

* 1. Аккаунт поставщика

При успешной авторизации поставщика будет открыто главное меню поставщика, предоставленное на рисунке 9.15. В главном меню поставщика предоставлены пункты меню для добавления товаров, редактирования товаров и удаление товаров.

При выборе пункта меню добавления товаров будет открыто оно для добавления, предоставленное на рисунке 9.16, в котором необходимо ввести информацию о добавляемом товаре.

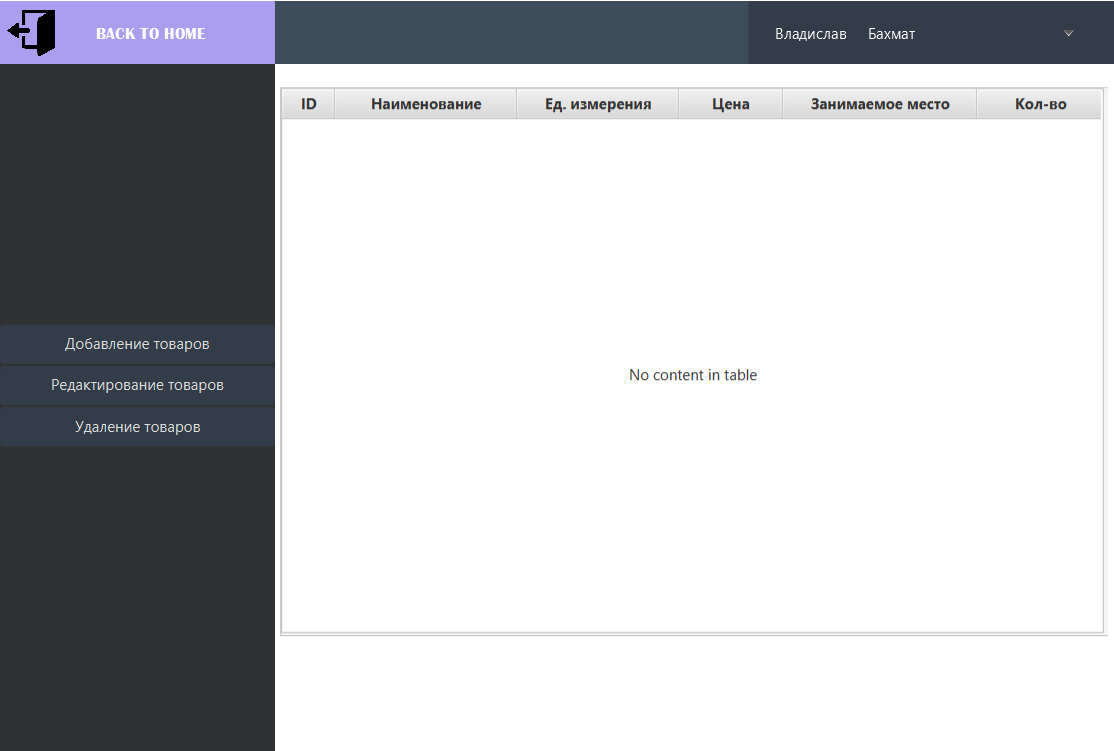


Рисунок 9.15 – Главное меню поставщика

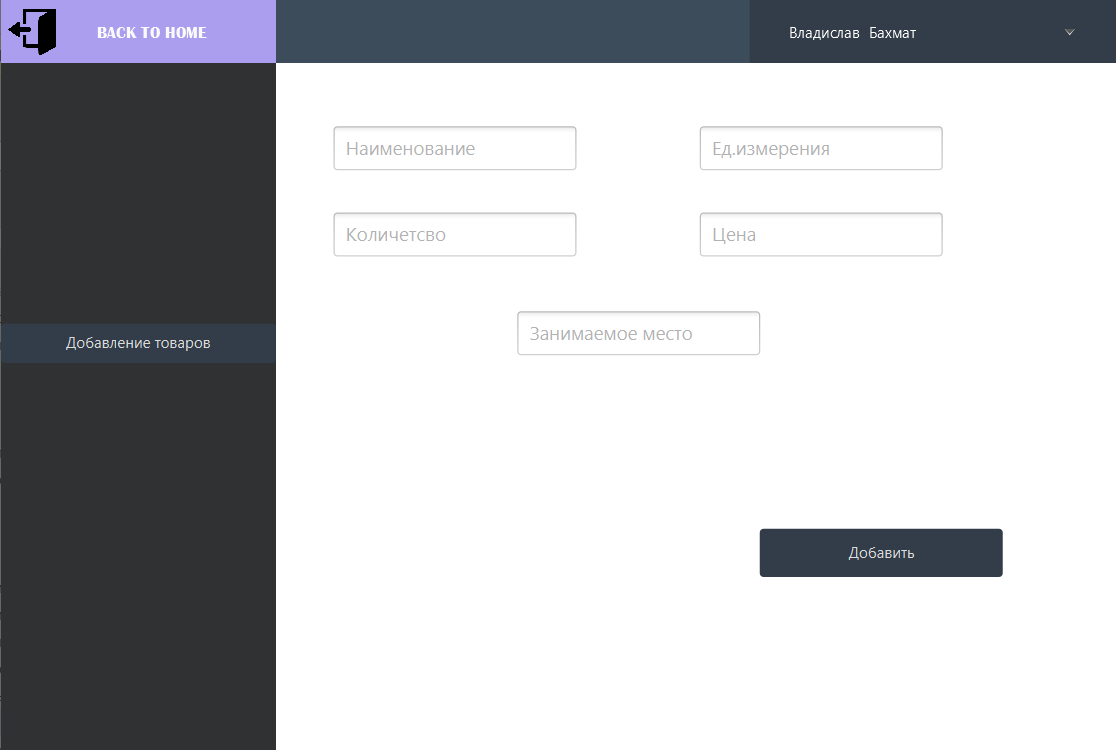


Рисунок 9.16 – Окно добавления товара

При успешном добавлении будет выведено соответствующее сообщение, предоставленное на рисунке 9.17.

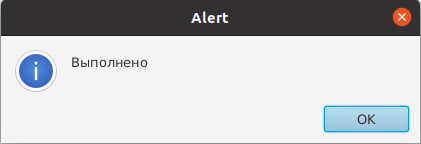


Рисунок 9.17 – Сообщение об успешном добавлении

При выборе пункта меню редактирования товаров будет выведена таблица существующих товаров, данные которых возможно изменить. Таблица выбора товаров предоставлена на рисунке 9.18.



Рисунок 9.18 – Главное меню поставщика

Выбрав данный пункт меню при отсутствии товаров у поставщика, то будет выведена соответствующее сообщение, предоставленное на рисунке 9.19.

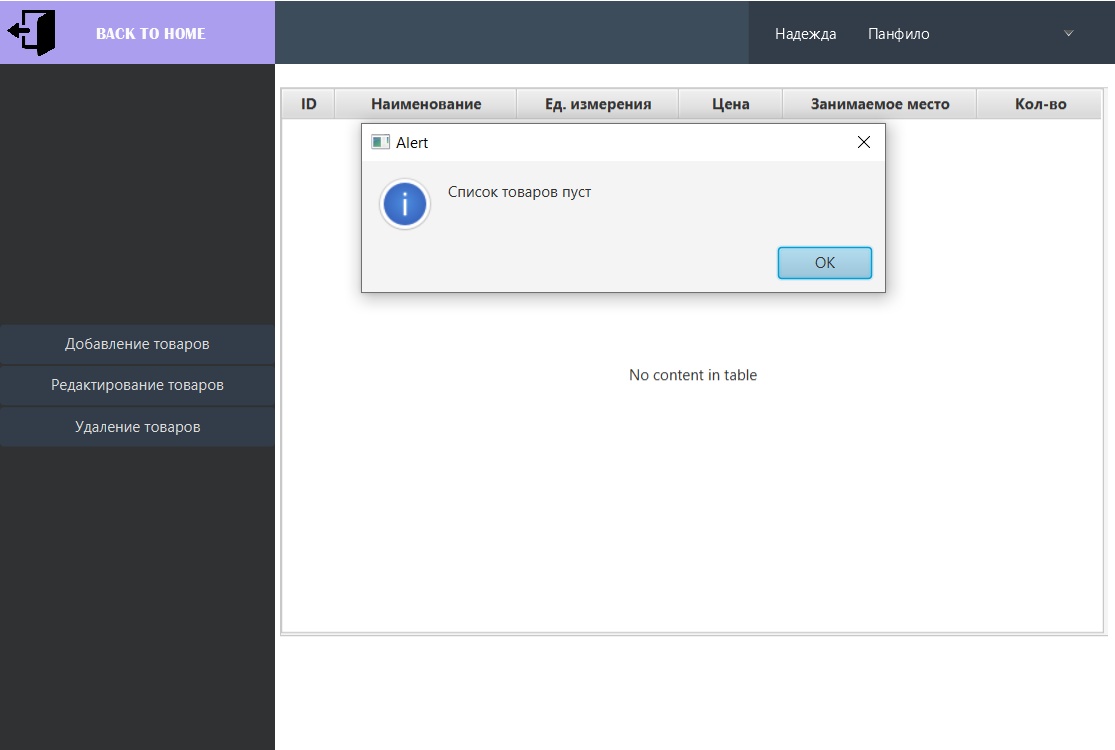


Рисунок 9.19 – Сообщение об ошибке вывода товаров в таблицу

При выборе товара и нажатии кнопки «Изменить» будет открыто окно для редактирования, предоставленное на рисунке 9.20. При успешном добавлении будет выведено соответствующее сообщение.

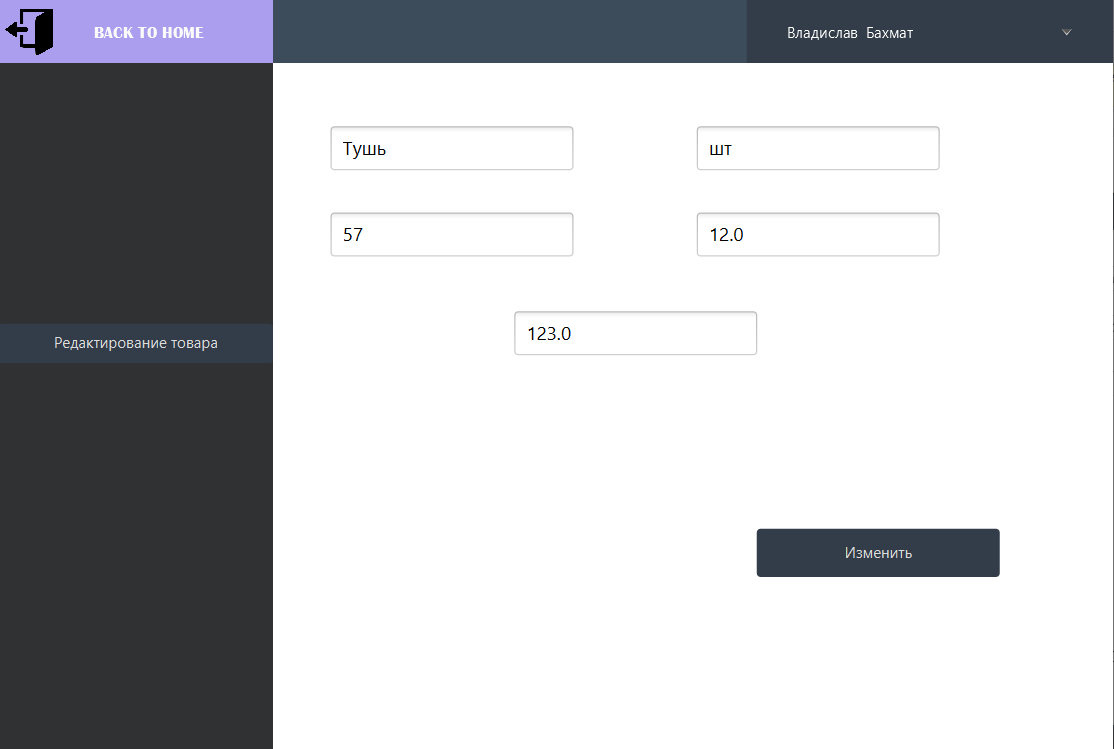


Рисунок 9.20 – Окно редактирования информации о товаре

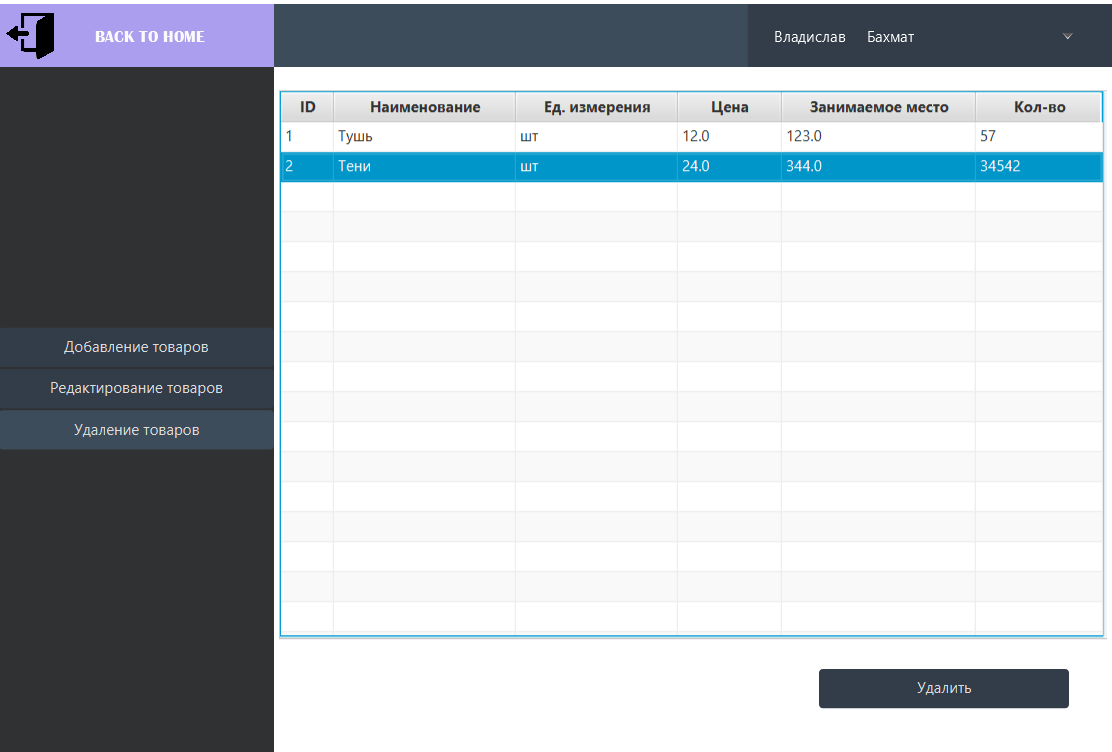


Рисунок 9.21 – Окно удаления товара у поставщика

При выборе в главном меню поставщика пункта меню удаления товаров, то будет выведена таблица с существующими товарами у поставщика и при нажатии на кнопку «Удалить» товар будет удалён и будет выведено соответствующее сообщение. Окно удаления товара предоставлено на рисунке 9.21.

* 1. Аккаунт сотрудника

После успешной авторизации в системе сотрудника будет открыто главное меню сотрудника, в котором предоставлены пункты меню для вывода графика заполненности складов товарами, добавление поставок, распределение товаров между складами, удаление товаров, оценка товаров, управление товарами и цены реализации. Окно главного меню сотрудника предоставлено на рисунке 9.22.

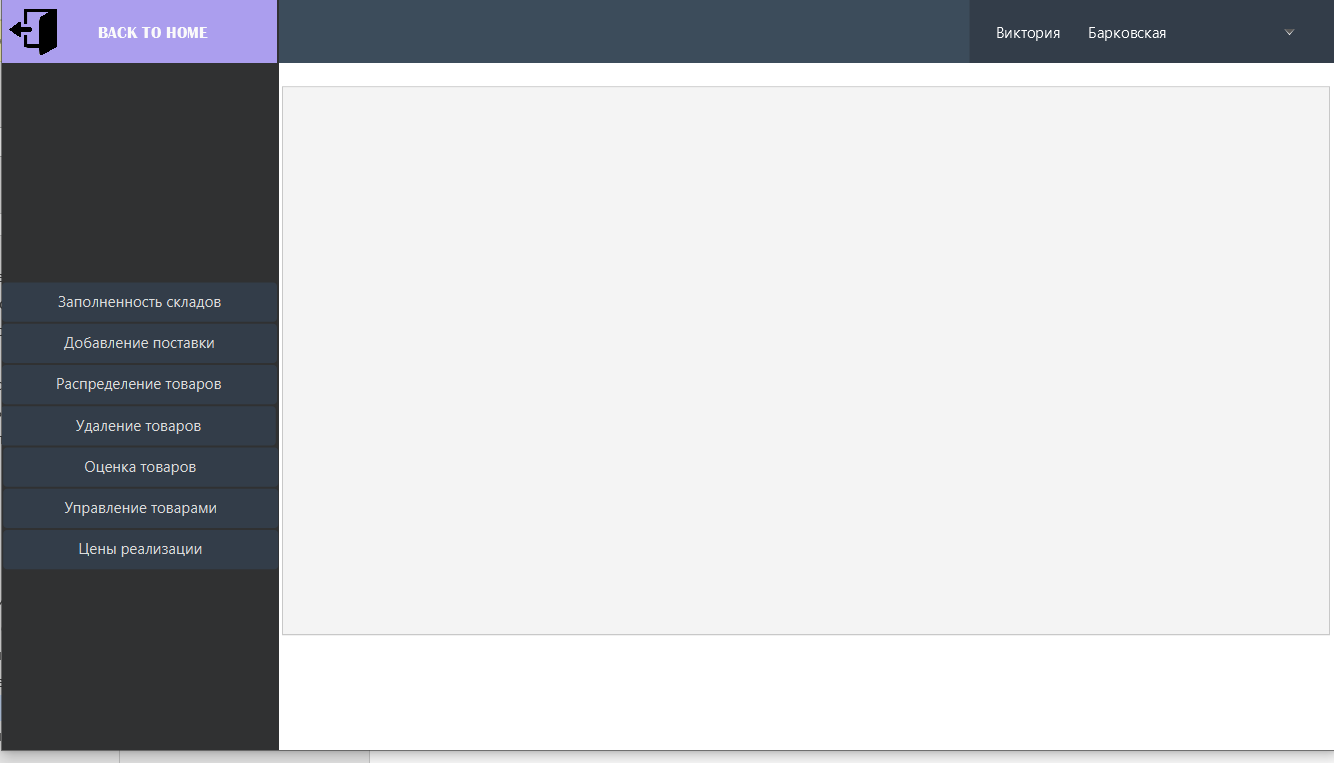


Рисунок 9.22 – Окно главного меню сотрудника

При выборе пункта меню для просмотра заполненности складов товарами будет выведен график отражающий заполненность складов в процентном соотношении. График заполненности складов предоставлен на рисунке 9.23.

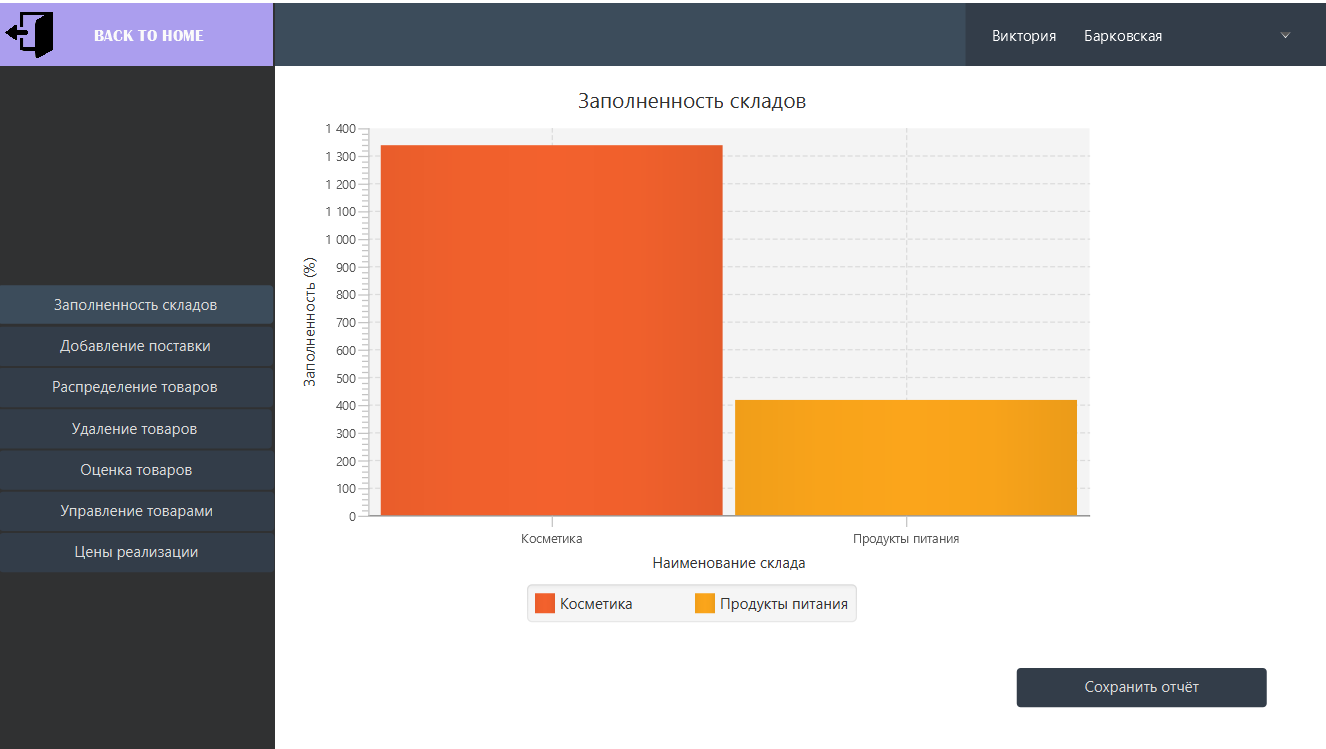


Рисунок 9.23 – График заполненности складов

При выборе пункта меню добавления поставки будет выведен график с доступными для поставки на склад товарами. После выбора товара для поставки, его количества и склад, на который необходимо поставить и нажатия кнопки добавить будет выведено сообщение о успешном добавлении товара на склад. Окно добавления поставки на склад представлено на рисунке 9.24.

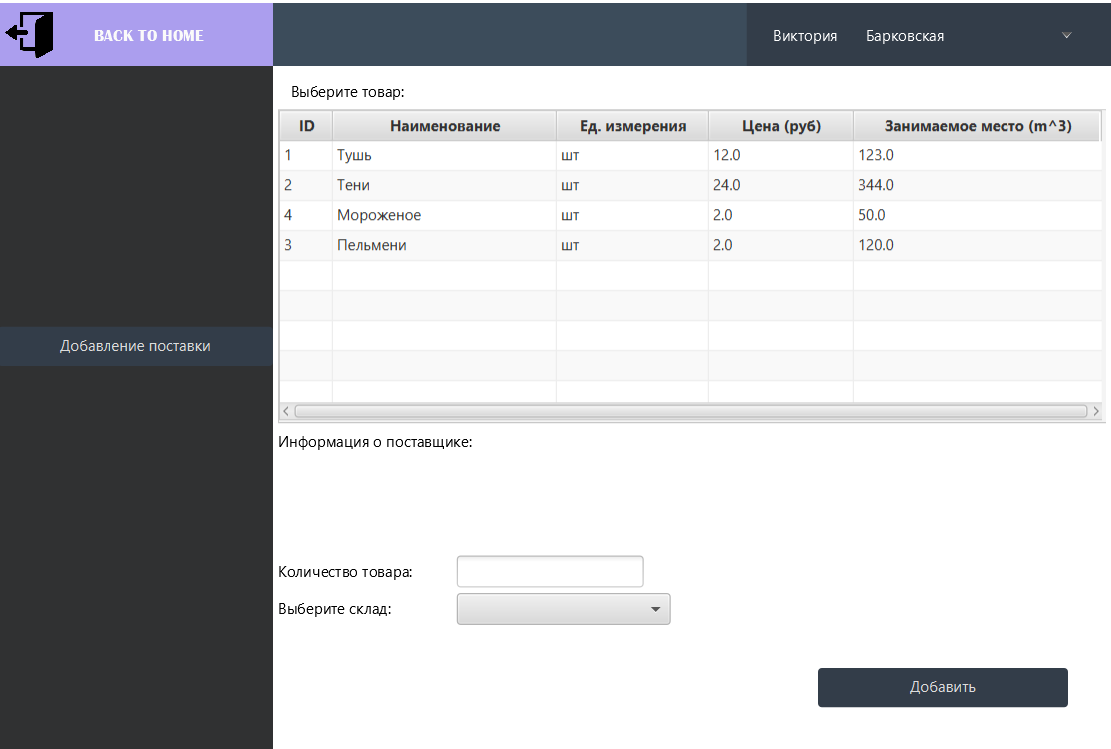


Рисунок 9.24 – Окно добавления поставки на склад

При выборе пункта меню с распределение товаров будет открыто окно, в котором необходимо выбрать склад, с которого необходимо вывезти товар. После выбора склада на экран будет выведен список товаров, которые находятся на нём. Далее необходимо выбрать склад, на который необходимо перевезти товар, и выбрать количество товара для перевозки. При нажатии кнопки «Перевезти» буде выведено сообщение, подтверждающее выполнения данного действия. Окна распределения товаров предоставлено на рисунке 9.25.

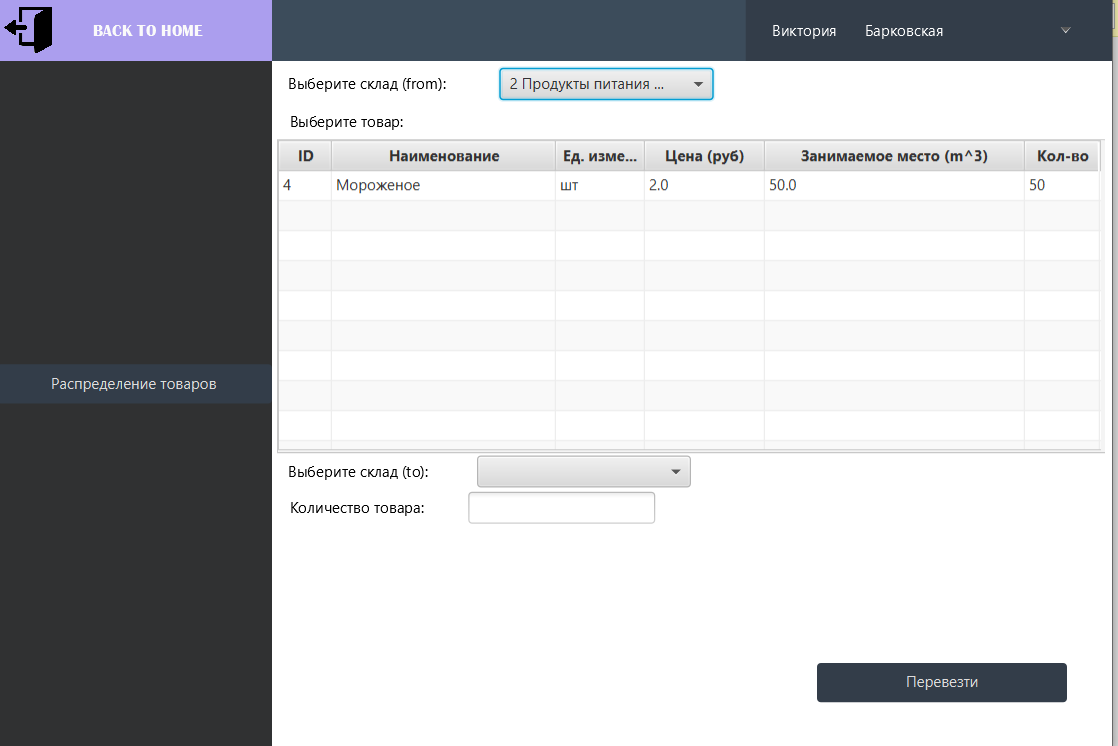


Рисунок 9.25 – Окно распределения товаров между складами

Выбрав пункт меню «Удаление товаров» сотрудник увидит список товаров, которые находятся на складах. Выбрать необходимый товар для удаления и нажав кнопку удаление, выбранный товар будет удалён и выведется сообщение об успешном удалении. Сообщение об успешном удалении предоставлено на рисунке 9.27. Окно выбора товара для удаления предоставлено на рисунке 9.27.

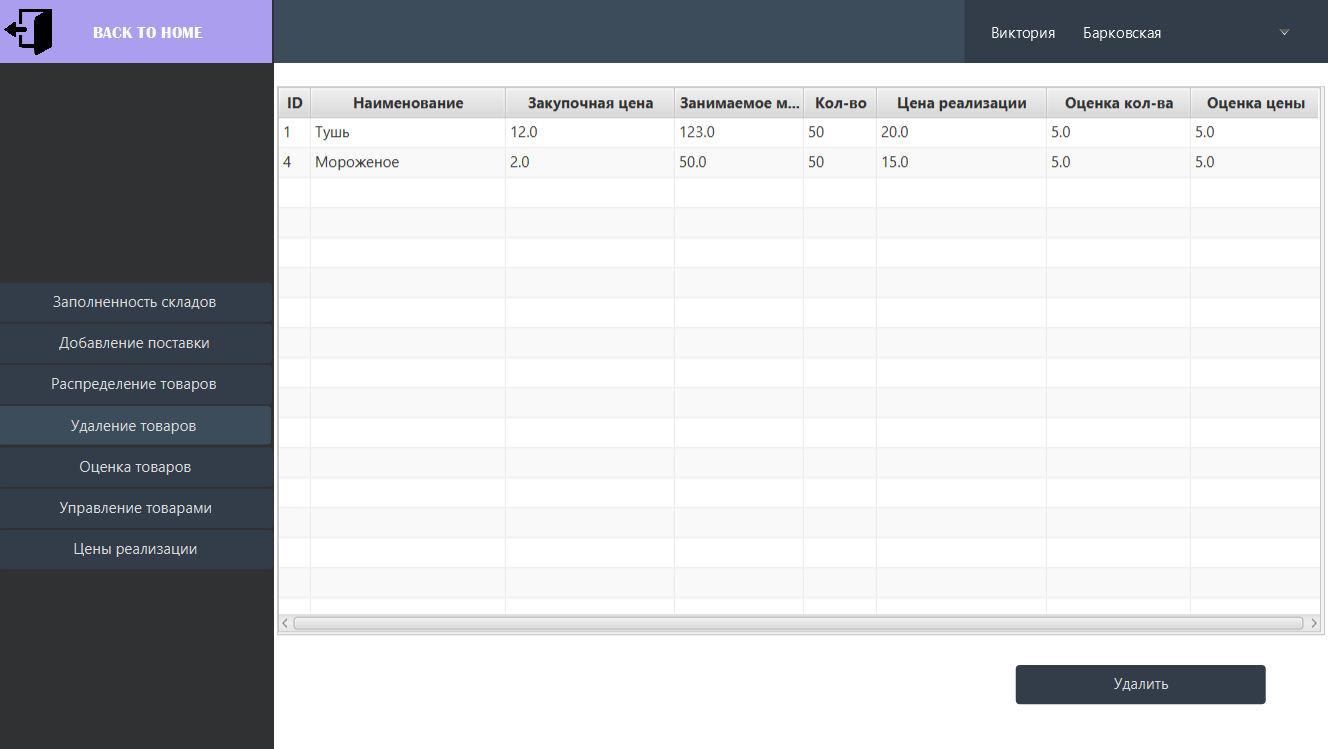


Рисунок 9.26 – Окно выбора товара для удаления

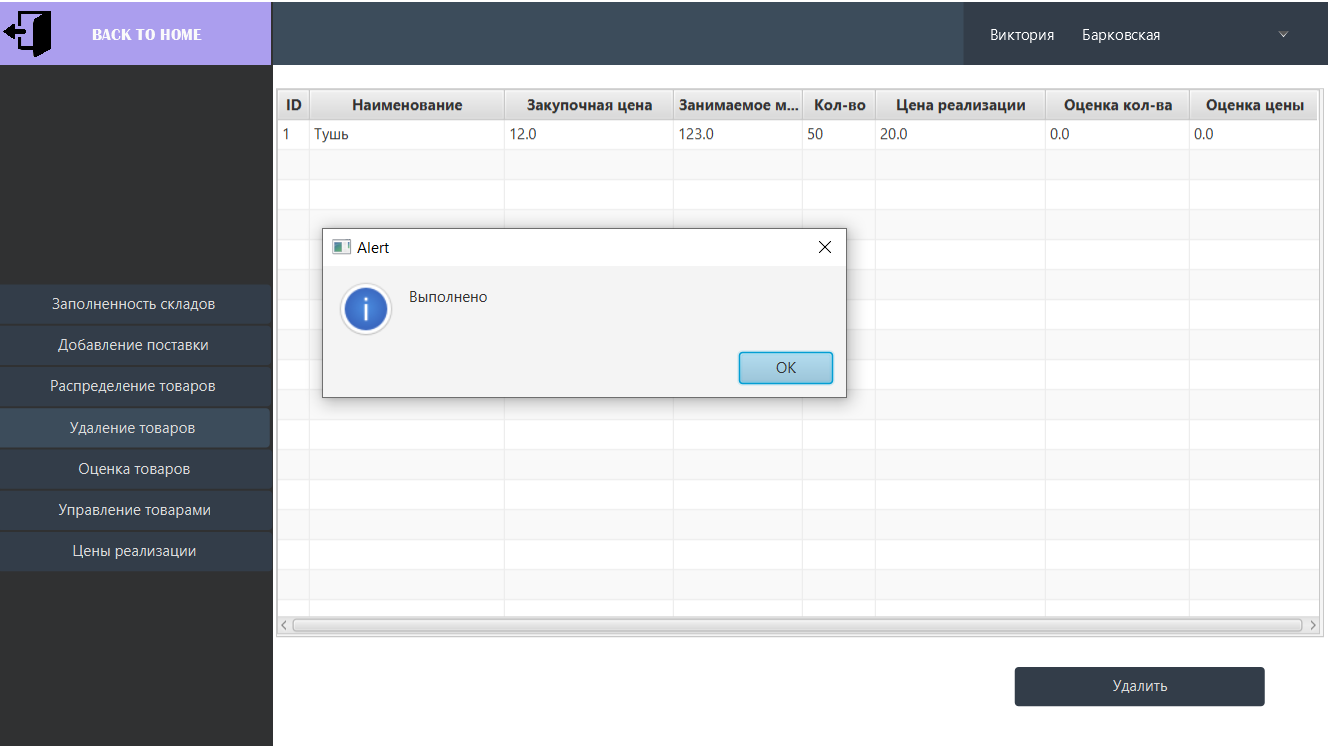


Рисунок 9.27 – Сообщение об успешном удалении

В пункте меню «Оценка товаров» сотрудник может выставить свои оценки на цели «Оценка количества» и «Оценка цены» для каждого товара. Оценки предназначены для дальнейшего анализа и выполнения задач для достижения данных целей. Окно выставления оценок предоставлено на рисунке 9.28.

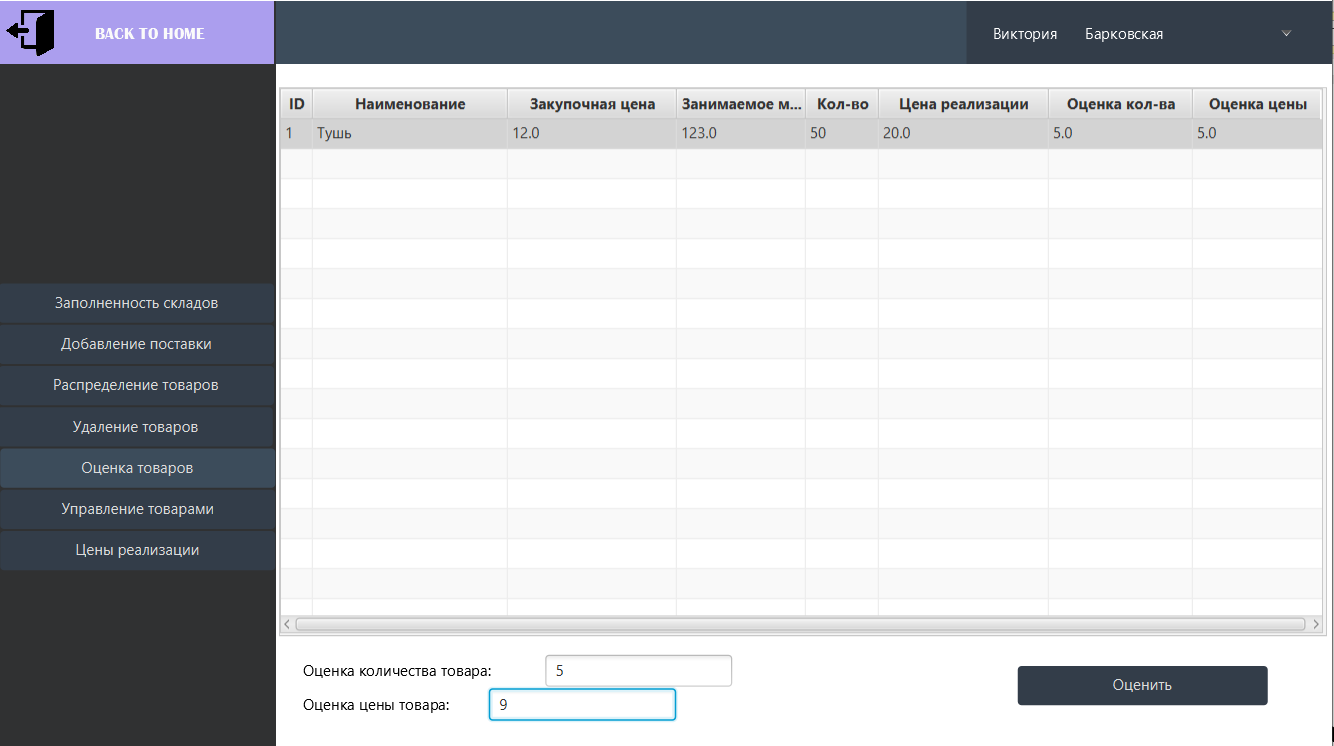


Рисунок 9.28 – Окно выставления оценок

При выборе пункта меню «Цены реализации» будет открыто окно со списком товаров и их ценами реализации. Сотрудник может задать цену реализации товара и при нажатии кнопки «Оценить», цена будет изменена и будет выведено соответствующее сообщение. Окно выставления цен реализации предоставлено на рисунке 9.29.

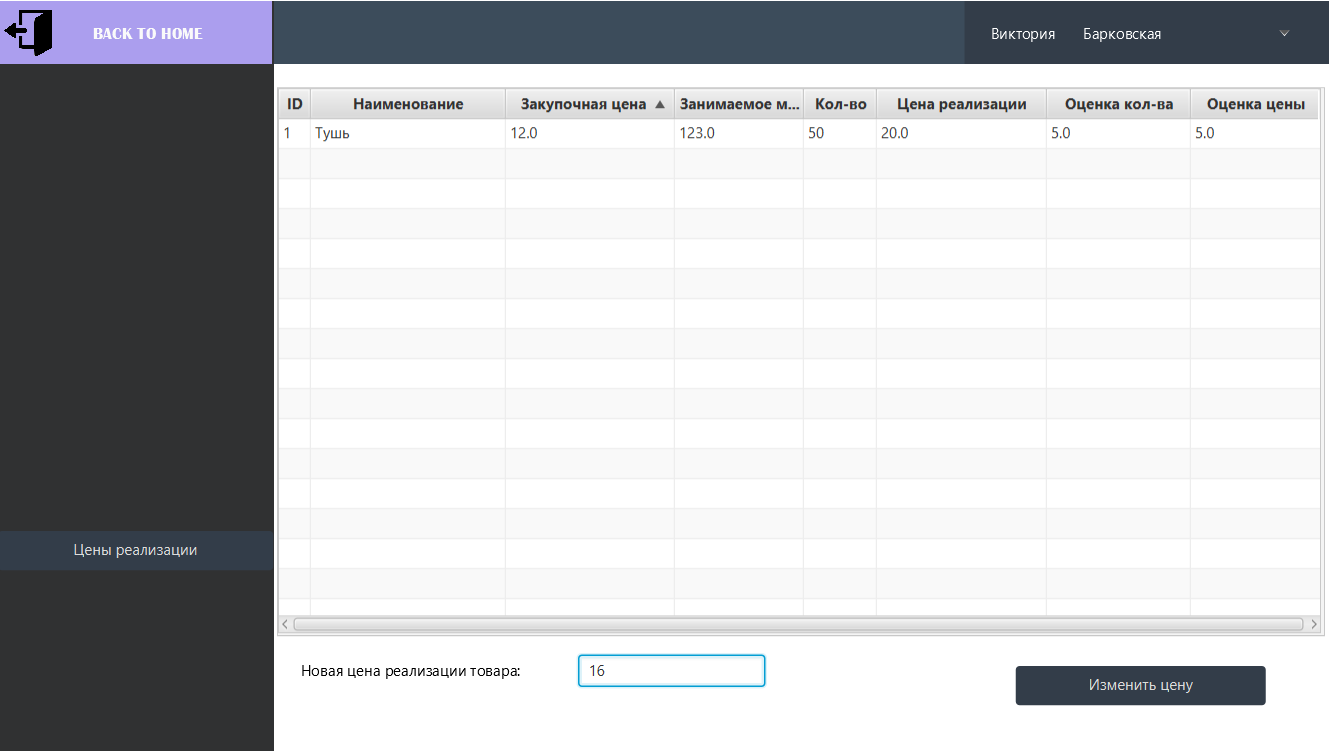


Рисунок 9.29 – Окно установления цен реализации товаров

При выборе пункта меню «Управление товарами» будет открыто окно со списком товаров, в котором необходимо выбрать товар для анализа. Окно выбора товара предоставлено на рисунке 9.30.

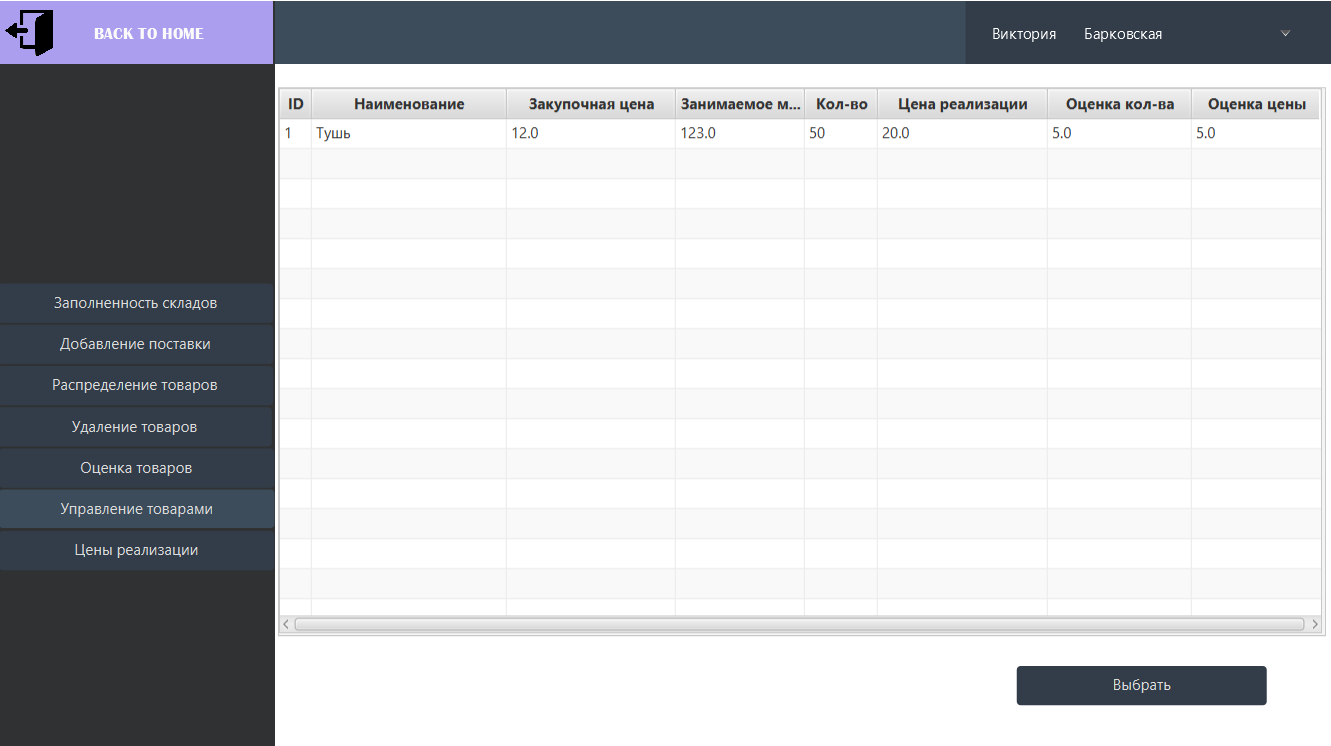


Рисунок 9.30 – Окно выбора товара для анализа

При нажатии на кнопку «Выбрать» будет открыто окно, в котором будет отображена матрица оценок экспертов, существующих в системе, для каждой цели, а также матрица экспертных оценок. На основе этих матриц будут рассчитаны веса каждой цели. Окно аналитических данных предоставлено на рисунке 9.31. При нажатии кнопки «Управление товаром» будут сравнены искомые веса целей и будет выведено сообщение о необходимой задаче для выбранного товара. Сообщение о необходимой задачи для выполнения предоставлено на рисунке 9.32.

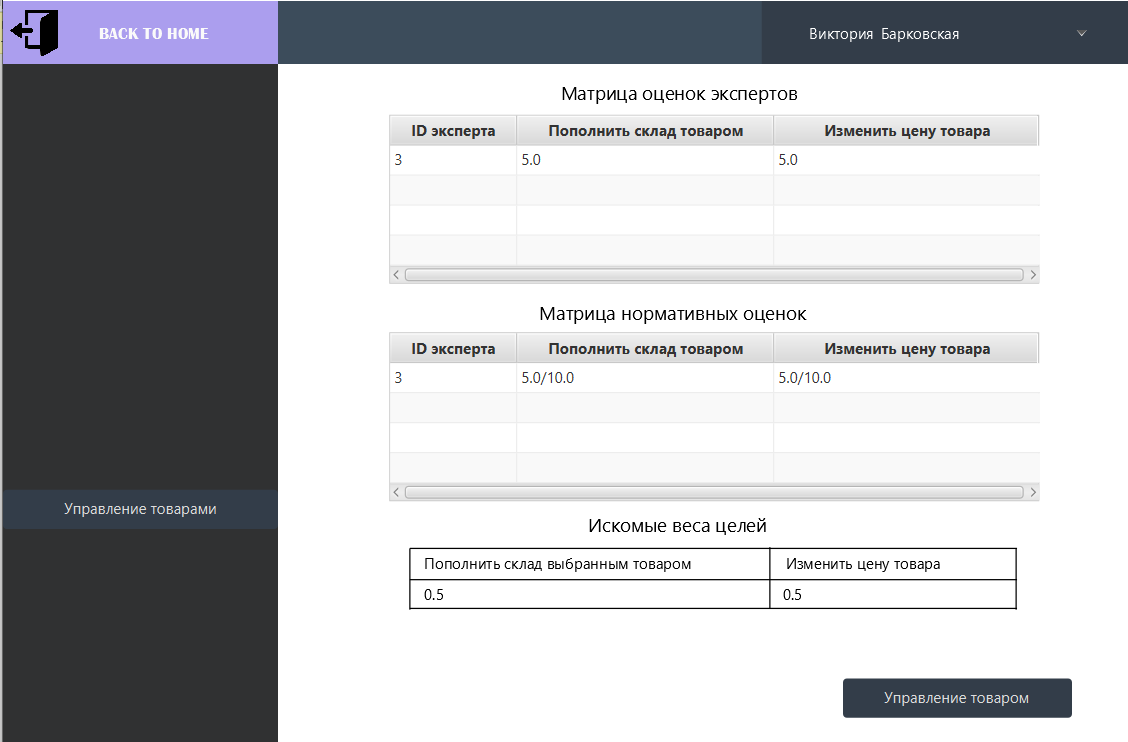


Рисунок 9.31 – Окно аналитических данных

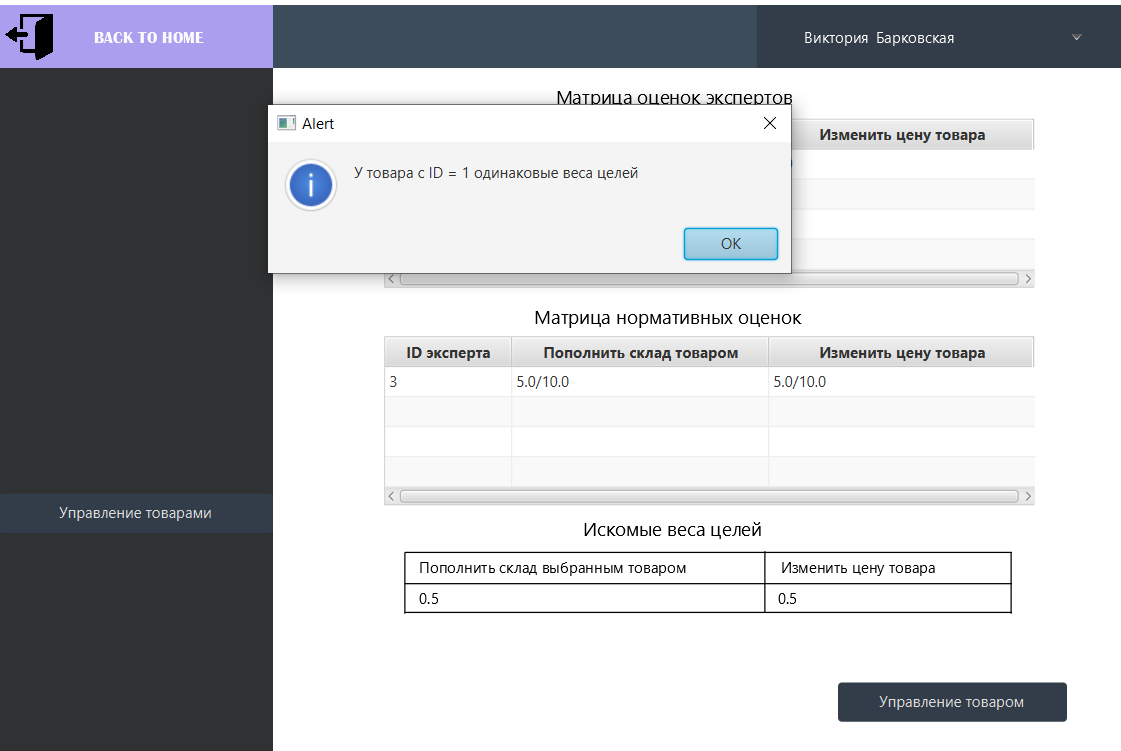


Рисунок 9.32 – Сообщение о необходимом действии

После закрытия сообщения будет открыто окно для выполнения цели. Если цель «Пополнить склад товаром» будет иметь больший вес, то будет открыто окно добавления поставки товаров на склад. В противном случае будет открыто окно изменение цены товара.

В данной главе описано руководство пользователя, в котором подробно описана специфика взаимодействия пользователей с системой. Описана работа программного средства со всех сторон: со стороны сотрудника компании, со стороны поставщика и со стороны администратора. В данной главе проиллюстрированы возможности программного средства системы управления складами.

# Результаты тестирования разработанной программы

Для ведения учета и регистрации товаров на складах логистической компании система должна успешно работать и быть защищенной от ошибок. Любая ошибка данной системы будет причиной потери денег компании. В случае ошибок ввода или некорректной работы с системой должны выводиться сообщения с предупреждениями.

Данная система не должна предоставлять доступ неподтверждённым пользователям. Если введенный логин и пароль есть в базе данных, то вход выполнен успешно, в противном случае выведется окно с сообщением (Рисунок 9.1).

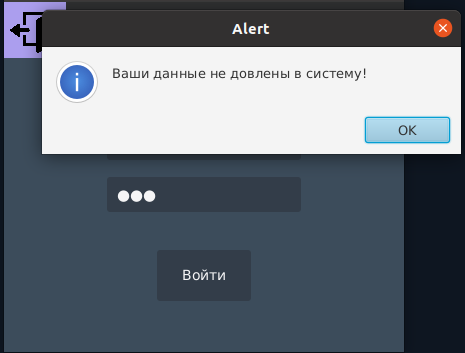


Рисунок 9.1 – Неправильно введенные данные

При регистрации все поля являются обязательными для заполнения. В случае некорректного ввода информации или её отсутствия в одном из полей невозможно обратиться к серверу, а значит и пройти регистрацию (рисунок 9.2).

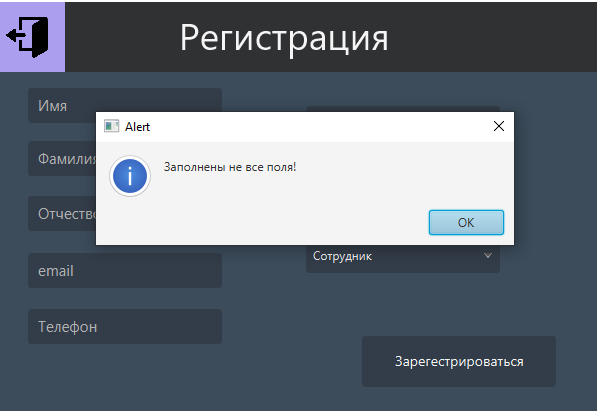


Рисунок 9.2 – Некорректный ввод информации для регистрации

При добавлении новых товаров следует запретить ввод некорректных данных и проверка на ввод существующего товара, чтобы система не выдавала ошибок при дальнейшем её использовании. В данном проекте предусмотрена подобная ошибка. При вводе информации о новом товаре существует проверка на совпадения с уже существующими и корректность введённых данных. Сообщение о выводе ошибки представлен на рисунке 9.3.

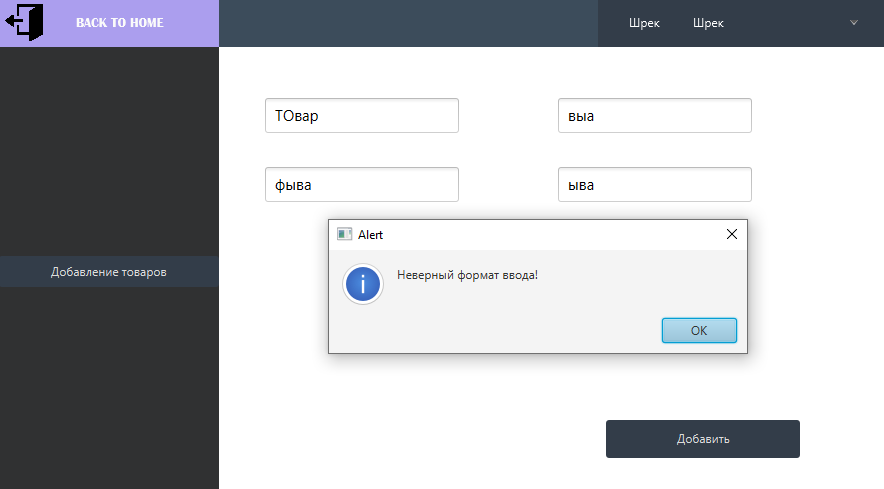


Рисунок 9.3 – Ошибка ввода некорректных данных

Важным фактом является при удалении данных из системы полностью исключить информацию о нем из всех таблиц. То есть при удалении поставщиков – удалить его товары. Так при отсутствии товаров у поставщика и желании редактировать, удалить или проанализировать данные о товарах действие не имеет смысла, поэтому будет выведена ошибка, изображенная на рисунке 9.4.

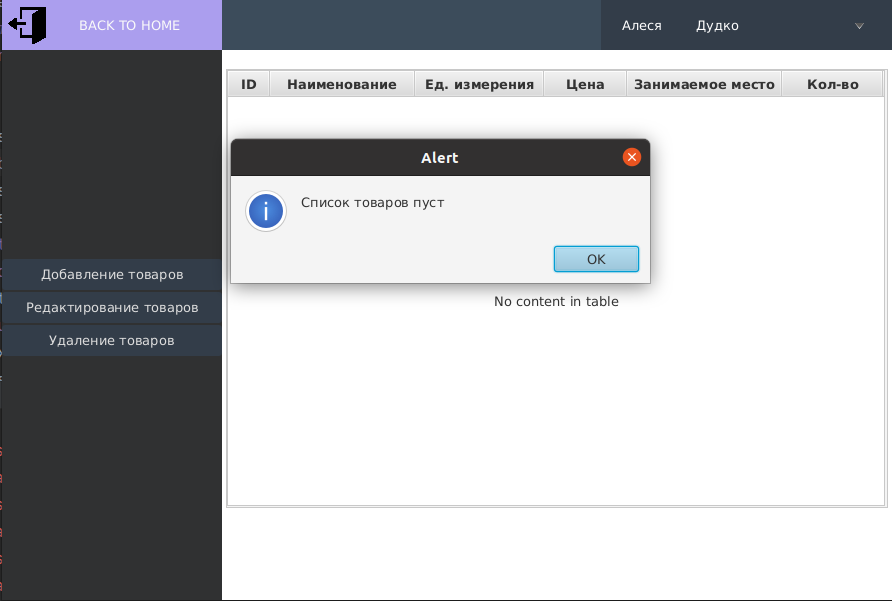


Рисунок 9.4 – Ошибка отсутствия данных

# Заключение

В курсовом проекте была разработана программа для распределения товаров по складам. Разработка подобных систем требует планирование и подробного изучения области для облегчения и уменьшения количества ошибок при разработке системы. При разработке программного приложения был проведен анализ предметной области, исследованы особенности организации деятельности логистической компании, а в особенности процесс заполнения склада логистической компании товарами. Изученные материалы помогли оценить предметную область для более детальной проработки системы. На основе данных материалов произведено моделирование программного средства и разработаны алгоритмы работы программы, что облегчает написание системы учёта и регистрации товаров на складах логистической компании. Для этого был выбран язык моделирования UML, т.к. данный язык пригоден для моделирования систем любых масштабов. Данный язык моделирования позволяет рассмотреть систему со всех точек зрения, имеющих отношение к её разработке и последующему развёртыванию. При моделирование были выбраны диаграмма использования, последовательности, компонентов, состояний, развёртывания и диаграмма классов. Данные диаграммы позволяют полностью смоделировать систему учёта и регистрации товаров на складе логистической компании, что благоприятно влияет на разработку корректной системы.

При разработке системы учёта и регистрации товаров на складе логистической компании был выбран объектно-ориентированный язык Java. Java обладает большим количеством инструментов для создания приложений клиент-серверного типа с графическим пользовательским интерфейсом. Можно отметить, что немало важным достоинством выбранного языка программирования является надежность.

Для хранения данных системы была выбрана реляционная база данных postgres. Данное решение предоставляет универсальность хранения данных, т.к. предоставляет моделирование сущностей реального мира в виде набора связанных таблиц, и простоту манипуляции данными с точки зрения конечного пользователя.

После разработки программного средства было произведено тестирование работоспособности программы, т.к. данное средство работает с денежными ресурсами и ошибки будут чреваты большими потерями логистической компании. Написано руководство пользователя для пользователей системы для того, чтобы сотрудники могли быстро освоить функционал и пользовательский интерфейс, что благоприятно влияет на использование данного продукта. Разработанное руководство развёртывания разработана для сотрудников компании, чтобы была возможность лёгкой установки системы незнающими людьми.

Итогом данного курсового проекта является функционирующее программное приложение, пригодное для использования логистическими компаниями для распределения продукции по складам. В результате разработанная программа оснащена простым и удобным для пользователей интерфейсом, несложной системой навигации и необходимыми функциями для основной задачи курсового проекта. Сотрудники смогут оптимизировать свою работу и выполнять её более качественно и быстро.

Таким образом, данная программа значительно могла бы облегчить работу в любой логистической компании. Безусловно, в будущем, данное приложение можно усовершенствовать. Например, путём доработки интерфейса или добавления новых функций. Иными словами, приложение имеет возможность изменения и расширения своей функциональности, а также возможные перспективы использования и развития. Следовательно, программа может стать еще более компактной, оптимизированной и многофункциональной.

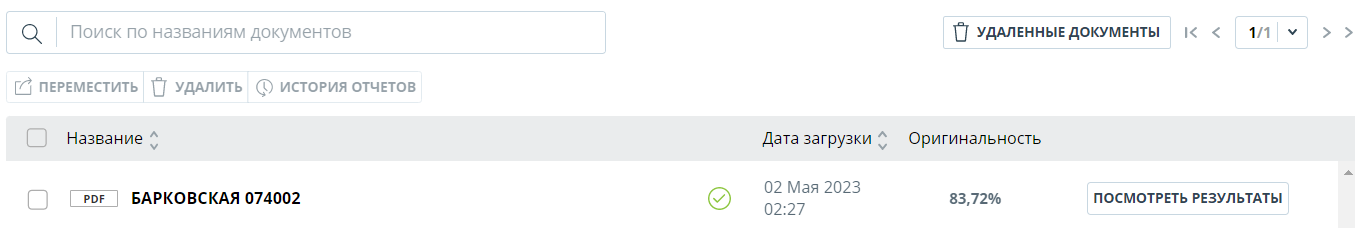
Список использованных источников

1. Петров Ю. А., Петрова Г. И., Терминологический словарь-справочник: экономика, маркетинг, менеджмент. - Микроэкономика: Учебник для вузов. 5-е изд.
2. Токмакова Н.О., Андриянова М.В., МЕНЕДЖМЕНТ-КОНСАЛТИНГ / учебно-методический комплекс. – М. : Изд. центр ЕАОИ, 2009. – 271 с.
3. Система управления складами [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://ru.wikipedia.org/wiki/Система_управления_складом>
4. Черемных С.В., Моделирование и анализ систем. IDEF-технологии. /М.: Финансы и статистика, 2006. – 188с.
5. П. Лузанов, Е. Рогов, И. Лёвшин, PostgreSQL для начинающих, 2017. – 166с.
6. Репин. В. В., Разработка архитектуры бизнес-процессов компании в Business Studio – М.: Манн, Иванов и Фербер, 2019. – 142с.
7. Кровчик Э., Сетевое программирование для профессионалов. -Изд. Центр ЛОРИ, 2005. – 417с.
8. Г. Шилд, Java. Полное руководство, 10-е издание, 2018. – 1500с.
9. Прохорёнок Н. А., JavaFX.). – СПБ.: БХВ-Петербург, 2020. – 768с.
10. Диаграммы состояний [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://skarlupka.ru/articles.php?id=71#:~:text=Диаграмма%20состояний%20(statechart%20diagram)%20—,следующий%20от%20состояния%20к%20состоянию>

# ПРИЛОЖЕНИЕ А

**(обязательное)**

**Отчет о проверке на заимствования в системе «Антиплагиат»**



# ПРИЛОЖЕНИЕ Б

**(обязательное)**

**Листинг SQL-скрипта**

SET statement\_timeout = 0;

SET lock\_timeout = 0;

SET idle\_in\_transaction\_session\_timeout = 0;

SET client\_encoding = 'UTF8';

SET standard\_conforming\_strings = on;

SELECT pg\_catalog.set\_config('search\_path', '', false);

SET check\_function\_bodies = false;

SET xmloption = content;

SET client\_min\_messages = warning;

SET row\_security = off;

CREATE EXTENSION IF NOT EXISTS adminpack WITH SCHEMA pg\_catalog;

COMMENT ON EXTENSION adminpack IS 'administrative functions for PostgreSQL';

SET default\_tablespace = '';

SET default\_table\_access\_method = heap;

CREATE TABLE public."Basket" (

id integer NOT NULL,

id\_user integer DEFAULT 0 NOT NULL

);

ALTER TABLE public."Basket" OWNER TO neaktub;

CREATE TABLE public."Delivery" (

id integer NOT NULL,

date date NOT NULL,

id\_warehouse\_basket integer NOT NULL

);

ALTER TABLE public."Delivery" OWNER TO neaktub;

CREATE TABLE public."Goods" (

barcode integer NOT NULL,

product\_name character varying(45) NOT NULL,

unit\_measurement character varying(20) NOT NULL,

quantity integer DEFAULT 0 NOT NULL,

price money NOT NULL,

occupied\_space numeric DEFAULT 0 NOT NULL

);

ALTER TABLE public."Goods" OWNER TO neaktub;

CREATE TABLE public."Suppliers" (

id integer NOT NULL,

org\_name text NOT NULL,

city text NOT NULL,

country text NOT NULL,

address text NOT NULL,

id\_basket integer NOT NULL

);

ALTER TABLE public."Suppliers" OWNER TO neaktub;

Продолжение приложения Б

CREATE SEQUENCE public."Suppliers\_id\_seq"

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER TABLE public."Suppliers\_id\_seq" OWNER TO neaktub;

ALTER SEQUENCE public."Suppliers\_id\_seq" OWNED BY public."Suppliers".id;

CREATE TABLE public."Users" (

id integer NOT NULL,

login character varying(45) NOT NULL,

password character varying(45) NOT NULL,

email character varying(45) NOT NULL,

phone\_number character varying(13) NOT NULL,

firstname character varying(45) NOT NULL,

lastname character varying(45) NOT NULL,

patronymic character varying(45) NOT NULL,

access\_rights character varying(20) NOT NULL,

confirm\_access boolean

);

ALTER TABLE public."Users" OWNER TO neaktub;

CREATE SEQUENCE public."Users\_id\_seq"

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER TABLE public."Users\_id\_seq" OWNER TO neaktub;

ALTER SEQUENCE public."Users\_id\_seq" OWNED BY public."Users".id;

CREATE TABLE public."Warehouse" (

id integer NOT NULL,

name character varying(45) NOT NULL,

volume integer DEFAULT 0 NOT NULL,

idbasket integer NOT NULL,

fullness integer NOT NULL

);

ALTER TABLE public."Warehouse" OWNER TO neaktub;

CREATE TABLE public."Warehouse\_basket" (

id integer NOT NULL,

idwarehouse integer NOT NULL

);

ALTER TABLE public."Warehouse\_basket" OWNER TO neaktub;

CREATE TABLE public."Warehouse\_goods" (

idbasket integer NOT NULL,

idgood integer NOT NULL,

Продолжение приложения Б

number integer NOT NULL,

price money

);

ALTER TABLE public."Warehouse\_goods" OWNER TO neaktub;

CREATE SEQUENCE public."Warehouse\_id\_seq"

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER TABLE public."Warehouse\_id\_seq" OWNER TO neaktub;

ALTER SEQUENCE public."Warehouse\_id\_seq" OWNED BY public."Warehouse".id;

CREATE TABLE public.basket\_goods (

id\_basket integer NOT NULL,

id\_good integer

);

ALTER TABLE public.basket\_goods OWNER TO neaktub;

CREATE TABLE public.metod (

iduser integer NOT NULL,

idgood integer NOT NULL,

rating integer NOT NULL

);

ALTER TABLE public.metod OWNER TO neaktub;

ALTER TABLE ONLY public."Suppliers" ALTER COLUMN id SET DEFAULT nextval('public."Suppliers\_id\_seq"'::regclass);

ALTER TABLE ONLY public."Users" ALTER COLUMN id SET DEFAULT nextval('public."Users\_id\_seq"'::regclass);

ALTER TABLE ONLY public."Warehouse" ALTER COLUMN id SET DEFAULT nextval('public."Warehouse\_id\_seq"'::regclass);

ALTER TABLE ONLY public."Delivery"

ADD CONSTRAINT "Delivery\_pkey" PRIMARY KEY (id);

ALTER TABLE ONLY public."Warehouse\_basket"

ADD CONSTRAINT "Warehouse\_basket\_pkey" PRIMARY KEY (id);

ALTER TABLE ONLY public."Warehouse\_goods"

ADD CONSTRAINT "Warehouse\_goods\_pkey" PRIMARY KEY (idbasket, idgood);

ALTER TABLE ONLY public."Basket"

ADD CONSTRAINT basket\_pk PRIMARY KEY (id);

ALTER TABLE ONLY public."Goods"

ADD CONSTRAINT goods\_pk PRIMARY KEY (barcode);

ALTER TABLE ONLY public."Suppliers"

Продолжение приложения Б

ADD CONSTRAINT suppliers\_pk PRIMARY KEY (id);

ALTER TABLE ONLY public."Users"

ADD CONSTRAINT users\_pk PRIMARY KEY (id);

ALTER TABLE ONLY public."Warehouse"

ADD CONSTRAINT warehouse\_pk PRIMARY KEY (id);

CREATE UNIQUE INDEX basket\_id\_uindex ON public."Basket" USING btree (id);

CREATE UNIQUE INDEX goods\_barcode\_uindex ON public."Goods" USING btree (barcode);

CREATE UNIQUE INDEX suppliers\_id\_uindex ON public."Suppliers" USING btree (id);

CREATE UNIQUE INDEX warehouse\_basket\_idwarehouse\_uindex ON public."Warehouse\_basket" USING btree (idwarehouse);

CREATE UNIQUE INDEX warehouse\_id\_uindex ON public."Warehouse" USING btree (id);

CREATE UNIQUE INDEX warehouse\_idgood\_uindex ON public."Warehouse" USING btree (idbasket);

ALTER TABLE ONLY public.basket\_goods

ADD CONSTRAINT "FK\_idGood" FOREIGN KEY (id\_good) REFERENCES public."Goods"(barcode);

ALTER TABLE ONLY public."Suppliers"

ADD CONSTRAINT "Suppliers\_basket\_fkey" FOREIGN KEY (id\_basket) REFERENCES public."Basket"(id) ON UPDATE CASCADE ON DELETE CASCADE;

ALTER TABLE ONLY public."Suppliers"

ADD CONSTRAINT "Suppliers\_id\_fkey" FOREIGN KEY (id) REFERENCES public."Users"(id) NOT VALID;

ALTER TABLE ONLY public."Warehouse"

ADD CONSTRAINT "idBasket" FOREIGN KEY (idbasket) REFERENCES public."Warehouse\_basket"(id) NOT VALID;

ALTER TABLE ONLY public.basket\_goods

ADD CONSTRAINT "idBasket" FOREIGN KEY (id\_basket) REFERENCES public."Warehouse\_basket"(id) NOT VALID;

ALTER TABLE ONLY public."Basket"

ADD CONSTRAINT "idUser" FOREIGN KEY (id\_user) REFERENCES public."Users"(id) NOT VALID;

ALTER TABLE ONLY public.metod

ADD CONSTRAINT id\_user FOREIGN KEY (iduser) REFERENCES public."Users"(id);

ALTER TABLE ONLY public."Delivery"

ADD CONSTRAINT idbasket FOREIGN KEY (id\_warehouse\_basket) REFERENCES public."Warehouse\_basket"(id);

ALTER TABLE ONLY public.metod

ADD CONSTRAINT idgood FOREIGN KEY (idgood) REFERENCES public."Goods"(barcode);

# ПРИЛОЖЕНИЕ В

**(обязательное)**

**Диаграмма классов**

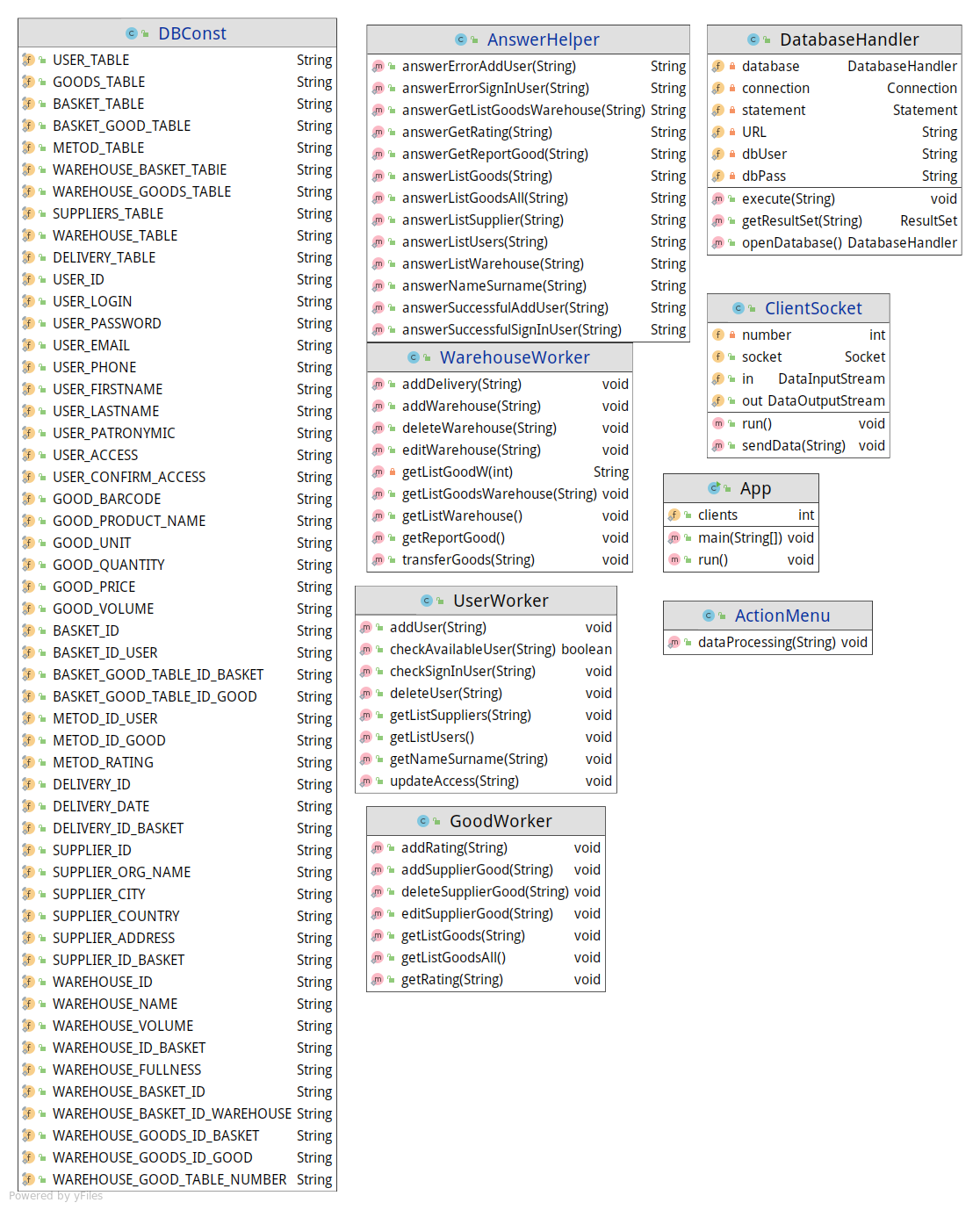


Рисунок Б.1 – Диаграмма классов серверного приложения

Продолжение приложения В

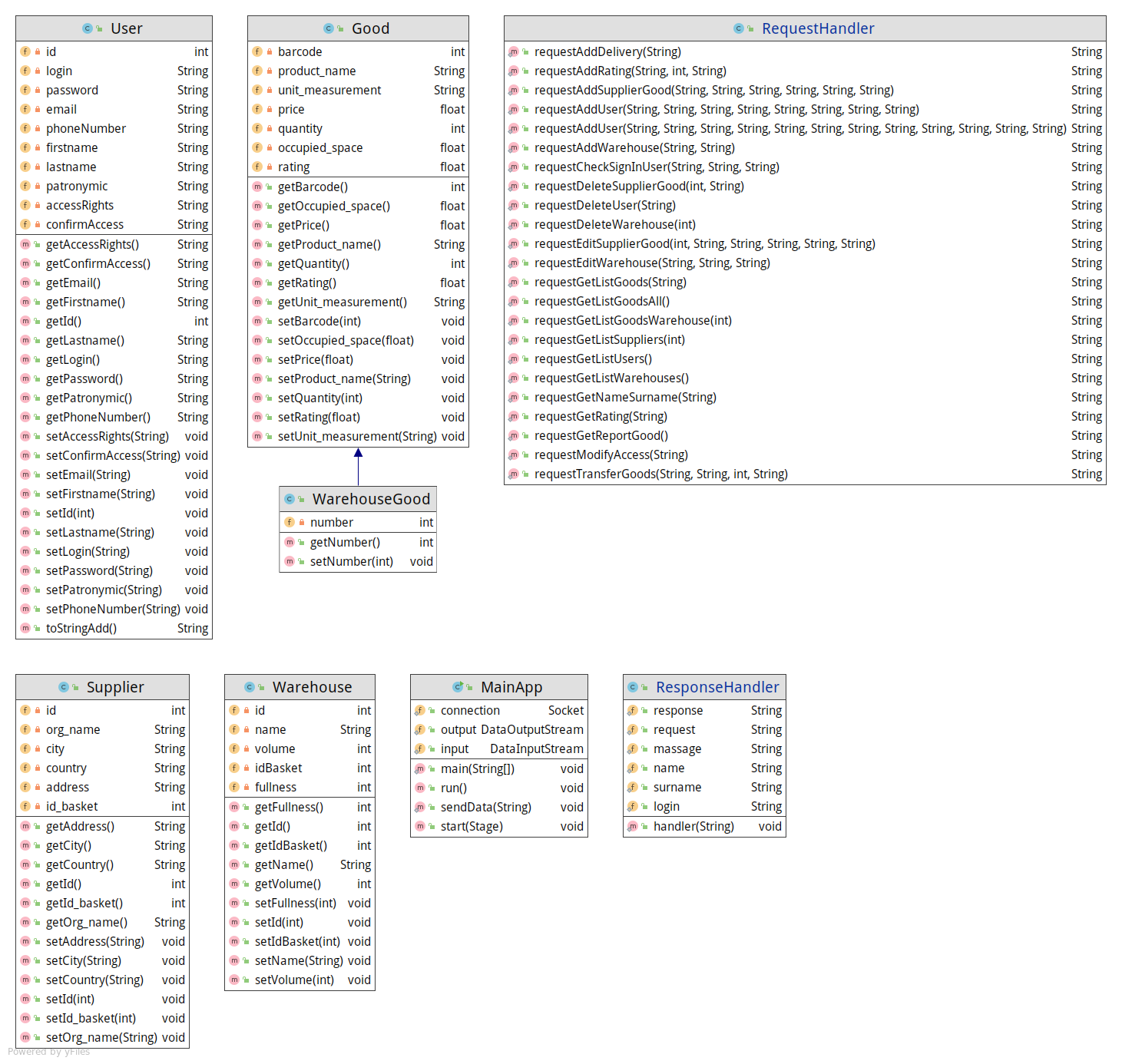


Рисунок Б.2 – Диаграмма классов клиентского приложения

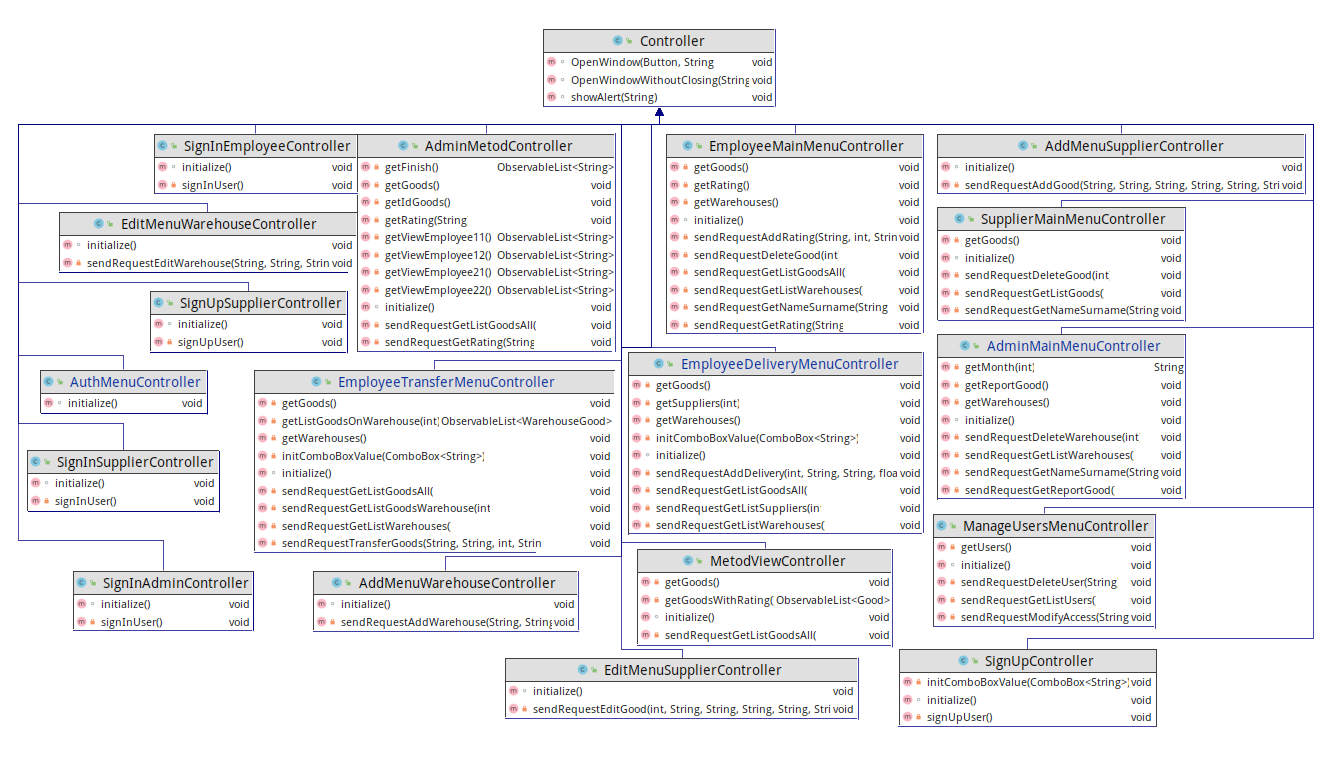
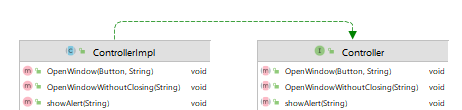


Рисунок Б.3 – Диаграмма классов клиентского приложения

# ПРИЛОЖЕНИЕ Г

**(обязательное)**

**Листинг кода**

**DatabaseHandler.java**

Начало формы

Конец формы

Начало формы

Конец формы

package databaseWork;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.sql.Statement;

import java.sql.ResultSet;

public class DatabaseHandler{

private static DatabaseHandler database;

private static Connection connection;

private static Statement statement;

private static String URL = "jdbc:postgresql://localhost:5432/postgres";

private static String dbUser = "neaktub";

private static String dbPass = "65129832";

public static DatabaseHandler openDatabase() throws SQLException

{

connection = DriverManager.getConnection(URL,/\*dbUser\*/"neaktub", dbPass);

return database;

}

public static ResultSet getResultSet(String str) {

ResultSet resultSet = null;

try {

statement = connection.createStatement();

resultSet = statement.executeQuery(str);

} catch (SQLException ex) {

ex.printStackTrace();

}

return resultSet;

}

public static void execute(String str)

{

try {

statement = connection.createStatement();

statement.execute(str);

} catch (SQLException ex) {

ex.printStackTrace();

}

}

}

**App.java**

package sample;

import java.io.IOException;

import java.net.ServerSocket;

import java.net.Socket;

public class App implements Runnable

Продолжение приложения Г

{

public static int clients = 1;

public static void main( String[] args )

{

new Thread(new App()).start();

}

@Override

public void run() {

ServerSocket s = null;

try {

System.out.println("Сервер запущен");

s = new ServerSocket(4567);

while (true) {

Socket socket = s.accept();

if (socket!=null)

{

System.out.println("Подключился " + clients + " пользователь. " + socket.getInetAddress());

new ClientSocket(socket, clients);

clients++;

}

}

}catch (Exception e) {

e.printStackTrace();

System.out.println("Ошибка сервера...");

}finally {

try {

System.out.println("Сервер отключён.");

assert s != null;

s.close();

}catch (IOException e) {

System.out.println("Ошибка завершения работы сервера...");

}

}

}

}

**ClientSocket.java**

package sample;

import connectionWork.ActionMenu;

import java.io.\*;

import java.net.\*;

import java.sql.SQLException;

public class ClientSocket extends Thread {

private int number;

public Socket socket;

public static DataInputStream in;

public static DataOutputStream out;

public ClientSocket(Socket s, int number) throws IOException {

this.number = number;

Продолжение приложения Г

socket = s;

in = new DataInputStream(socket.getInputStream());

out = new DataOutputStream(socket.getOutputStream());

start();

}

@Override

public void run() {

try {

while(true) {

ActionMenu.dataProcessing(in.readUTF());

}

}

catch (IOException | SQLException e) {

System.out.println("Пользователь " + number + " отключился.");

} finally {

try {

socket.close();

}

catch (IOException e) {

System.err.println("Сокет не закрыт");

}

}

}

public static void sendData(String ob1){

try{

out.flush();

out.writeUTF(ob1);

}

catch(IOException e){}

}

}

**WarehouseWorker.java**

package connectionWork;

import databaseWork.AnswerHelper;

import databaseWork.DBConst;

import databaseWork.DatabaseHandler;

import sample.App;

import sample.ClientSocket;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.text.DateFormat;

import java.text.SimpleDateFormat;

import java.util.Calendar;

import java.util.HashSet;

public class WarehouseWorker {

public static void addWarehouse(String str) throws SQLException {

String array[] = str.trim().split("%");

String sql = "SELECT MAX(" + DBConst.WAREHOUSE\_ID + ") FROM " + DBConst.WAREHOUSE\_TABLE;

DatabaseHandler.openDatabase();

Продолжение приложения Г

ResultSet res = DatabaseHandler.getResultSet(sql);

int id = 0;

if (res.next()) id = res.getInt(1);

sql = "SELECT COUNT(\*) FROM " + DBConst.WAREHOUSE\_BASKET\_TABlE;

res = DatabaseHandler.getResultSet(sql);

int count = 0;

if (res.next()) count = res.getInt(1);

int idBasket = 0;

if(count != 0){

sql = "SELECT MAX(" + DBConst.WAREHOUSE\_BASKET\_ID + ") FROM " + DBConst.WAREHOUSE\_BASKET\_TABlE;

res = DatabaseHandler.getResultSet(sql);

if (res.next()) idBasket = res.getInt(1);

}

sql = "INSERT INTO " + DBConst.WAREHOUSE\_BASKET\_TABlE + " VALUES (" + (idBasket + 1) + ", " + (id + 1) + ")";

DatabaseHandler.execute(sql);

sql = "INSERT INTO " + DBConst.WAREHOUSE\_TABLE

+ " VALUES ('" + (id + 1) + "'" + ", '" + array[0] + "', '" + array[1] + "', '" + (idBasket + 1)

+ "', '" + 0 + "')";

DatabaseHandler.execute(sql);

}

public static void getListWarehouse() throws SQLException{

String sql = "SELECT \* FROM " + DBConst.WAREHOUSE\_TABLE;

DatabaseHandler.openDatabase();

ResultSet res = DatabaseHandler.getResultSet(sql);

int rows = 0;

while(res.next()) {

rows++;

}

res = DatabaseHandler.getResultSet(sql);

String sentString = "";

int columns = res.getMetaData().getColumnCount();

for(int j = 0; j < rows; j++){

res.next();

for(int i = 1; i <= columns; i++){

if(j == (rows - 1) && i == columns){

sentString += res.getString(i);

}

else sentString += res.getString(i) + "%";

}

}

ClientSocket.sendData(AnswerHelper.answerListWarehouse(sentString));

}

public static void deleteWarehouse(String id) throws SQLException {

String sql = "SELECT " + DBConst.WAREHOUSE\_BASKET\_ID + " FROM " + DBConst.WAREHOUSE\_TABLE

+ " WHERE " + DBConst.WAREHOUSE\_ID + " = '" + id + "'";

DatabaseHandler.openDatabase();

ResultSet res = DatabaseHandler.getResultSet(sql);

Продолжение приложения Г

int idBasket = 0;

if (res.next()) idBasket = res.getInt(1);

sql = "DELETE FROM " + DBConst.WAREHOUSE\_TABLE + " WHERE " + DBConst.WAREHOUSE\_ID + " = '" + id + "'";

DatabaseHandler.execute(sql);

sql = "DELETE FROM " + DBConst.WAREHOUSE\_BASKET\_TABlE + " WHERE " + DBConst.WAREHOUSE\_BASKET\_ID

+ " = " + idBasket;

DatabaseHandler.execute(sql);

}

public static void editWarehouse(String str) throws SQLException {

String array[] = str.trim().split("%");

String sql = "UPDATE " + DBConst.WAREHOUSE\_TABLE + " SET "

+ DBConst.WAREHOUSE\_NAME + " = '" + array[1] + "', "

+ DBConst.WAREHOUSE\_VOLUME + " = '" + array[2] + "' "

+ "WHERE " + DBConst.WAREHOUSE\_ID + " = '" + array[0] + "'";

DatabaseHandler.openDatabase();

DatabaseHandler.execute(sql);

}

public static void addDelivery(String str) throws SQLException{

String array[] = str.trim().split("%");

String sql = "SELECT " + DBConst.WAREHOUSE\_ID\_BASKET + " FROM " + DBConst.WAREHOUSE\_TABLE

+ " WHERE " + DBConst.WAREHOUSE\_ID + " = " + array[1];

ResultSet res = DatabaseHandler.getResultSet(sql);

int idBasket = 0;

if (res.next()) idBasket = res.getInt(1);

sql = "SELECT COUNT(\*) FROM " + DBConst.WAREHOUSE\_GOODS\_TABLE + " WHERE "

+ DBConst.WAREHOUSE\_GOODS\_ID\_BASKET + " = " + idBasket + " AND "

+ DBConst.WAREHOUSE\_GOODS\_ID\_GOOD + " = " + array[0] + "";

DatabaseHandler.openDatabase();

res = DatabaseHandler.getResultSet(sql);

int count = 0;

if (res.next()) count = res.getInt(1);

if(count == 0) {

sql = "INSERT INTO " + DBConst.WAREHOUSE\_GOODS\_TABLE + " VALUES ("

+ idBasket + ", " + array[0] + ", " + array[2] + ")";

DatabaseHandler.execute(sql);

}

else{

sql = "SELECT " + DBConst.WAREHOUSE\_GOOD\_TABLE\_NUMBER + " FROM " + DBConst.WAREHOUSE\_GOODS\_TABLE

+ " WHERE "

+ DBConst.WAREHOUSE\_GOODS\_ID\_BASKET + " = " + idBasket + " AND "

+ DBConst.WAREHOUSE\_GOODS\_ID\_GOOD + " = " + array[0];

res = DatabaseHandler.getResultSet(sql);

int num = 0;

if (res.next()) num = res.getInt(1);

sql = "UPDATE " + DBConst.WAREHOUSE\_GOODS\_TABLE + " SET " + DBConst.WAREHOUSE\_GOOD\_TABLE\_NUMBER + " = "

Продолжение приложения Г

+ (num + Integer.valueOf(array[2]))

+ " WHERE " + DBConst.WAREHOUSE\_GOODS\_ID\_BASKET + " = " + array[1].split(" ")[0] + " AND "

+ DBConst.WAREHOUSE\_GOODS\_ID\_GOOD + " = " + array[0];

DatabaseHandler.execute(sql);

}

sql = "SELECT " + DBConst.WAREHOUSE\_FULLNESS + " FROM " + DBConst.WAREHOUSE\_TABLE + " WHERE "

+ DBConst.WAREHOUSE\_ID + " = " + array[1];

res = DatabaseHandler.getResultSet(sql);

int space = 0;

if (res.next()) space = res.getInt(1);

sql = "SELECT " + DBConst.GOOD\_VOLUME + " FROM " + DBConst.GOODS\_TABLE

+ " WHERE " + DBConst.GOOD\_BARCODE + " = " + array[0];

res = DatabaseHandler.getResultSet(sql);

float volume = 0;

if (res.next()) volume = res.getFloat(1);

float number = Float.valueOf(array[2])\*volume + space;

sql = "UPDATE " + DBConst.WAREHOUSE\_TABLE + " SET " + DBConst.WAREHOUSE\_FULLNESS + " = "

+ number

+ " WHERE " + DBConst.WAREHOUSE\_ID + " = " + array[1];

DatabaseHandler.execute(sql);

sql = "SELECT COUNT(\*) FROM " + DBConst.DELIVERY\_TABLE;

res = DatabaseHandler.getResultSet(sql);

int idDelivery = 0;

if (res.next()) idDelivery = res.getInt(1);

DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");

Calendar cal = Calendar.getInstance();

sql = "INSERT INTO " + DBConst.DELIVERY\_TABLE + " VALUES(" + (idDelivery + 1) + ", '"

+ dateFormat.format(cal.getTime()) + "', " + idBasket + ")";

DatabaseHandler.execute(sql);

sql = "SELECT " + DBConst.USER\_ID + " FROM " + DBConst.USER\_TABLE

+ " WHERE " + DBConst.USER\_ACCESS + " = '" + "Сотрудник'";

res = DatabaseHandler.getResultSet(sql);

int rows = 0;

while(res.next()) {

rows++;

}

int[] idUsers = new int[rows];

res = DatabaseHandler.getResultSet(sql);

for(int i = 0; i < rows; i++){

res.next();

idUsers[i] = res.getInt(1);

}

for(int i = 0; i < idUsers.length; i++){

sql = "SELECT COUNT(\*) FROM " + DBConst.METOD\_TABLE + " WHERE "

+ DBConst.METOD\_ID\_USER + " = " + idUsers[i] + " AND " + DBConst.METOD\_ID\_GOOD + " = " + array[0];

Продолжение приложения Г

res = DatabaseHandler.getResultSet(sql);

int countMetod = 0;

if(res.next()) countMetod = res.getInt(1);

if(countMetod == 0){

sql = "INSERT INTO " + DBConst.METOD\_TABLE + " VALUES(" + idUsers[i] + ", "

+ array[0] + ", " + 5 + ", " + 5 + ")";

DatabaseHandler.execute(sql);

}

}

updateFullness();

}

public static void getListGoodsWarehouse(String idBasket) throws SQLException {

String sql = "SELECT " + DBConst.WAREHOUSE\_GOODS\_ID\_GOOD + ", "

+ DBConst.WAREHOUSE\_GOOD\_TABLE\_NUMBER + " FROM " + DBConst.WAREHOUSE\_GOODS\_TABLE + " WHERE "

+ DBConst.WAREHOUSE\_GOODS\_ID\_BASKET + " = " + idBasket;

DatabaseHandler.openDatabase();

ResultSet res1 = DatabaseHandler.getResultSet(sql);

int rows = 0;

while(res1.next()) {

rows++;

}

String sendString = "";

res1 = DatabaseHandler.getResultSet(sql);

for(int i = 0; i < rows; i++){

res1.next();

if(i == (rows - 1)) {

sendString += res1.getInt(i+1) + "%";

sendString += res1.getString(i+2);

}

else {

sendString += res1.getInt(i+1) + "%";

sendString += res1.getString(i+2) + "%";

}

}

ClientSocket.sendData(AnswerHelper.answerGetListGoodsWarehouse(sendString));

}

public static void transferGoods(String str) throws SQLException {

String array[] = str.trim().split("%");

String sql = "SELECT " + DBConst.WAREHOUSE\_ID\_BASKET + " FROM " + DBConst.WAREHOUSE\_TABLE + " WHERE "

+ DBConst.WAREHOUSE\_ID + " = " + array[1];

DatabaseHandler.openDatabase();

ResultSet res = DatabaseHandler.getResultSet(sql);

int idToBasket = 0;

if (res.next()) idToBasket = res.getInt(1);

sql = "SELECT " + DBConst.GOOD\_VOLUME + " FROM " + DBConst.GOODS\_TABLE + " WHERE "

+ DBConst.GOOD\_BARCODE + " = " + array[2];

res = DatabaseHandler.getResultSet(sql);

Продолжение приложения Г

float space = 0;

if (res.next()) space = res.getFloat(1);

addDelivery(array[2] + "%" + array[1] + "%" + array[3] + "%" + space);

sql = "SELECT " + DBConst.WAREHOUSE\_ID\_BASKET + " FROM " + DBConst.WAREHOUSE\_TABLE + " WHERE "

+ DBConst.WAREHOUSE\_ID + " = " + array[0];

res = DatabaseHandler.getResultSet(sql);

int idFromBasket = 0;

if (res.next()) idFromBasket = res.getInt(1);

sql = "SELECT " + DBConst.WAREHOUSE\_GOOD\_TABLE\_NUMBER + " FROM " + DBConst.WAREHOUSE\_GOODS\_TABLE

+ " WHERE " + DBConst.WAREHOUSE\_GOODS\_ID\_BASKET + " = " + idFromBasket + " AND "

+ DBConst.WAREHOUSE\_GOODS\_ID\_GOOD + " = " + array[2];

res = DatabaseHandler.getResultSet(sql);

int number = 0;

if (res.next()) number = res.getInt(1);

if(number == Integer.valueOf(array[3])){

sql = "DELETE FROM " + DBConst.WAREHOUSE\_GOODS\_TABLE + " WHERE " + DBConst.WAREHOUSE\_GOODS\_ID\_BASKET

+ " = " + idFromBasket + " AND " + DBConst.WAREHOUSE\_GOODS\_ID\_GOOD + " = " + array[2];

DatabaseHandler.execute(sql);

sql = "UPDATE " + DBConst.WAREHOUSE\_TABLE + " SET " + DBConst.WAREHOUSE\_FULLNESS

+ " = " + 0 + " WHERE " + DBConst.WAREHOUSE\_ID

+ " = " + idFromBasket;

DatabaseHandler.execute(sql);

}

else {

sql = "SELECT " + DBConst.WAREHOUSE\_GOOD\_TABLE\_NUMBER + " FROM " + DBConst.WAREHOUSE\_GOODS\_TABLE

+ " WHERE " + DBConst.WAREHOUSE\_GOODS\_ID\_BASKET

+ " = " + idFromBasket + " AND " + DBConst.WAREHOUSE\_GOODS\_ID\_GOOD + " = " + array[2];

res = DatabaseHandler.getResultSet(sql);

int prevNumber = 0;

if (res.next()) prevNumber = res.getInt(1);

sql = "UPDATE " + DBConst.WAREHOUSE\_GOODS\_TABLE + " SET " + DBConst.WAREHOUSE\_GOOD\_TABLE\_NUMBER

+ " = " + (prevNumber - Integer.valueOf(array[3])) + " WHERE " + DBConst.WAREHOUSE\_GOODS\_ID\_BASKET

+ " = " + idFromBasket + " AND " + DBConst.WAREHOUSE\_GOODS\_ID\_GOOD + " = " + array[2];

DatabaseHandler.execute(sql);

sql = "SELECT " + DBConst.WAREHOUSE\_FULLNESS + " FROM " + DBConst.WAREHOUSE\_TABLE + " WHERE "

+ DBConst.WAREHOUSE\_ID + " = " + array[0];

res = DatabaseHandler.getResultSet(sql);

float spaceFrom = 0;

if (res.next()) spaceFrom = res.getFloat(1);

Продолжение приложения Г

float number1 = spaceFrom - Float.valueOf(array[3])\*space;

sql = "UPDATE " + DBConst.WAREHOUSE\_TABLE + " SET " + DBConst.WAREHOUSE\_FULLNESS + " = "

+ number1

+ " WHERE " + DBConst.WAREHOUSE\_ID + " = " + array[0];

DatabaseHandler.execute(sql);

}

}

public static void getReportGood() throws SQLException {

String sendString = "";

String sql = "SELECT " + DBConst.WAREHOUSE\_NAME + ", " + DBConst.WAREHOUSE\_BASKET\_ID

+ " FROM " + DBConst.WAREHOUSE\_TABLE;

DatabaseHandler.openDatabase();

ResultSet res = DatabaseHandler.getResultSet(sql);

int rows = 0;

while(res.next()) {

rows++;

}

res = DatabaseHandler.getResultSet(sql);

for(int i = 0; i < rows; i++) {

res.next();

int idBasket = res.getInt(2);

sendString += res.getString(1) + "%";

sendString += getListGoodWarehouse(idBasket);

if(i != (rows - 1)) sendString += "#";

}

ClientSocket.sendData(AnswerHelper.answerGetReportGood(sendString));

}

private static String getListGoodWarehouse(int idBasket) throws SQLException {

String sql = "SELECT " + DBConst.WAREHOUSE\_GOODS\_ID\_GOOD + ", " + DBConst.WAREHOUSE\_GOOD\_TABLE\_NUMBER

+ " FROM " + DBConst.WAREHOUSE\_GOODS\_TABLE + " WHERE "

+ DBConst.WAREHOUSE\_GOODS\_ID\_BASKET + " = " + idBasket;

DatabaseHandler.openDatabase();

ResultSet res1 = DatabaseHandler.getResultSet(sql);

int rows = 0;

while(res1.next()) {

rows++;

}

String sendString = "";

int idGood = 0;

res1 = DatabaseHandler.getResultSet(sql);

for(int i = 0; i < rows; i++){

res1.next();

if(i == (rows - 1)) {

idGood = res1.getInt(1);

sendString += res1.getInt(1) + "%";

sendString += res1.getString(2) + "%";

sql = "SELECT " + DBConst.GOOD\_PRODUCT\_NAME + ", " + DBConst.GOOD\_PRICE + ", " + DBConst.GOOD\_VOLUME

+ " FROM " + DBConst.GOODS\_TABLE + " WHERE " + DBConst.GOOD\_BARCODE + " = " + idGood;

ResultSet res2 = DatabaseHandler.getResultSet(sql);

Продолжение приложения Г

int rows1 = 0;

while(res2.next()) {

rows1++;

}

res2 = DatabaseHandler.getResultSet(sql);

for(int q = 0; q < rows1; q++){

res2.next();

if(q == (rows1 - 1)) {

sendString += res2.getString(1) + "%";

sendString += res2.getString(2) + "%";

sendString += res2.getFloat(3) + "%";

}

else {

sendString += res2.getString(1) + "%";

sendString += res2.getString(2) + "%";

sendString += res2.getFloat(3) + "%";

}

}

}

else {

idGood = res1.getInt(1);

sendString += res1.getInt(1) + "%";

sendString += res1.getString(2) + "%";

sql = "SELECT " + DBConst.GOOD\_PRODUCT\_NAME + ", " + DBConst.GOOD\_PRICE + ", " + DBConst.GOOD\_VOLUME

+ " FROM " + DBConst.GOODS\_TABLE + " WHERE " + DBConst.GOOD\_BARCODE + " = " + idGood;

ResultSet res2 = DatabaseHandler.getResultSet(sql);

int rows1 = 0;

while(res2.next()) {

rows1++;

}

res2 = DatabaseHandler.getResultSet(sql);

for(int q = 0; q < rows1; q++){

res2.next();

if(q == (rows1 - 1)) {

sendString += res2.getString(1) + "%";

sendString += res2.getString(2) + "%";

sendString += res2.getFloat(3) + "%";

}

else {

sendString += res2.getString(1) + "%";

sendString += res2.getString(2) + "%";

sendString += res2.getFloat(3) + "%";

}

}

}

}

return sendString;

}

private static String getListGoodW(int idBasket) throws SQLException {

String sql = "SELECT " + DBConst.WAREHOUSE\_GOODS\_ID\_GOOD + ", "

+ DBConst.WAREHOUSE\_GOOD\_TABLE\_NUMBER + " FROM " + DBConst.WAREHOUSE\_GOODS\_TABLE + " WHERE "

+ DBConst.WAREHOUSE\_GOODS\_ID\_BASKET + " = " + idBasket;

Продолжение приложения Г

DatabaseHandler.openDatabase();

ResultSet res1 = DatabaseHandler.getResultSet(sql);

int rows = 0;

while(res1.next()) {

rows++;

}

String sendString = "";

res1 = DatabaseHandler.getResultSet(sql);

for(int i = 0; i < rows; i++){

res1.next();

if(i == (rows - 1)) {

sendString += res1.getInt(1) + "%";

sendString += res1.getString(2);

}

else {

sendString += res1.getInt(1) + "%";

sendString += res1.getString(2) + "%";

}

}

return sendString;

}