

Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет
информатики и радиоэлектроники»

Факультет инженерно-экономический
Кафедра экономической информатики
Дисциплина «Программирование сетевых приложений»

«К ЗАЩИТЕ ДОПУСТИТЬ»
Руководитель курсового проекта
старший преподаватель
_____ Т.М. Унучек
_____._____.2021

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему:
**«АВТОМАТИЗАЦИЯ СКЛАДСКОГО УЧЕТА ТОВАРНО-
МАТЕРИАЛЬНЫХ ЦЕННОСТЕЙ»**

БГУИР КП 1-40 05 01-10 004 ПЗ

Выполнил студент группы 914301
КОНДРАШОВА Валерия Андреевна

(подпись студента)
Курсовой проект представлен на
проверку _____._____.2021

(подпись студента)

Минск 2021

РЕФЕРАТ

БГУИР КП 1-40 05 01-10 004 ПЗ

Кондрашова В. А. Автоматизация складского учета товарно-материальных ценностей: пояснительная записка к курсовому проекту / В.А. Кондрашова – Минск: БГУИР, 2021. – 65с.

Пояснительная записка 65с., 33 рис., 13 источников, 6 приложений

АВТОМАТИЗАЦИЯ СКЛАДСКОГО УЧЕТА, КЛИЕНТ-СЕРВЕР, УЧЕТ ТОВАРНО-МАТЕРИАЛЬНЫХ ЦЕННОСТЕЙ, ДОБАВЛЕНИЕ, БАЗА ДАННЫХ, РЕДАКТИРОВАНИЕ, ПОЛЬЗОВАТЕЛЬ, АДМИНИСТРАТОР, РАЗРАБОТКА ГРАФИЧЕСКОГО ИНТЕРФЕЙСА

Цель проектирования: разработка клиент-серверного приложения с использованием баз данных для повышения скорости и качества выполнения основных процессов на складе путем автоматизации деятельности сотрудников предприятия.

Методология проведения работы: в процессе постановки задачи был рассмотрен принцип установления соединения при помощи сетевого протокола ТСР/ІР, который ориентирован на обмен сообщениями между устройствами. Были использованы принципы объектно-ориентированного программирования при помощи языка Java.

Результаты работы: разработано многопоточное клиент-серверное приложение, спроектирован пользовательский графический интерфейс при помощи JavaFX. Разработан алгоритм взаимодействия клиента и сервера, где клиент посылает запросы на сервер. Сервер, в свою очередь, составляет в себе основную смысловую нагрузку, и соответственно, имеет прямой доступ к данным, хранящимся в базе данных. В результате проектирования базы данных, база была приведена к 3 нормальной форме. Для установления соединения сервера с базой данных был использован JDBC.

Область применения полученных результатов: в концепции сети передачи данных между физическими объектами, в автоматизации процессов предприятий, для взаимодействия техники и сетевых устройств в сети предприятий.

ЛИСТ ЗАДАНИЯ

ЛИСТ ЗАДАНИЯ
ПРИЛОЖЕНИЕ (!)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	6
1 ОСНОВНЫЕ ПОЛОЖЕНИЯ СИСТЕМЫ АВТОМАТИЗАЦИИ СКЛАДСКОГО УЧЕТА.....	8
2 ПОСТАНОВКА ЗАДАЧИ И ОБЗОР МЕТОДОВ ЕЕ РЕШЕНИЯ	13
2.1 Постановка задачи.....	13
2.2 Объектно-ориентированный язык Java	14
2.3 Технология JavaFX	15
2.4 Протокол TCP/IP.....	17
2.5 Объектно-реляционная система управления базами данных MySQL	19
2.6 Паттерн проектирования «Singleton»	20
2.7 Паттерн проектирования «Command».....	21
3 ФУНКЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ НА ОСНОВЕ СТАНДАРТА IDEF0	22
4 ИНФОРМАЦИОННАЯ МОДЕЛЬ СИСТЕМЫ СКЛАДСКОГО УЧЕТА И ЕГО ОПИСАНИЕ.....	27
5 ОПИСАНИЕ АЛГОРИТМА, РЕАЛИЗУЮЩЕГО БИЗНЕС-ЛОГИКУ СЕРВЕРНОЙ ЧАСТИ СИСТЕМЫ.....	33
6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ.....	35
6.1 Руководство менеджера.....	35
6.2 Руководство специалиста по доставке грузов.....	41
6.3 Руководство пользования для администратора	42
ЗАКЛЮЧЕНИЕ	47
ПРИЛОЖЕНИЕ А (обязательное) Листинг генерации базы данных.....	49
ПРИЛОЖЕНИЕ Б (обязательное) Листинг основных элементов.....	52
ПРИЛОЖЕНИЕ В (обязательное) Листинг алгоритма, реализующий бизнес- логику	56
ПРИЛОЖЕНИЕ Г (обязательное) Графический материал.....	60
ПРИЛОЖЕНИЕ Д (обязательное) Проверка на антиплагиат.....	64
ПРИЛОЖЕНИЕ Е (обязательное) Ведомость курсового проекта	65

ВВЕДЕНИЕ

Автоматизация работы склада готовой продукции – важный процесс, который стараются провести многие компании, чтобы упростить передвижение товаров перед продажей. Это упрощает их учет, помогает разобраться в остатках. Но в случае увеличения объемов и ассортимента стоит задуматься о переводе большего количества операций на автоматику.

Для успешной работы и завоевания уверенных позиций на рынке необходимы не только качественные товары, но и постоянное управление процессами, чёткий учёт товара, учёт продаж и поставок. Внедрение информационной системы позволяет построить весь процесс качественно.

Контроль материальных ценностей является основой прибыльного торгового бизнеса. Какими бы честными не были сотрудники, бесконтрольность порождает соблазн украсть или пренебречь обязанностями. Кроме того, знание остатков позволяет правильно оценить необходимость в сроках и ассортименте поставок очередной партии.

Для предприятия важным является конкурентоспособность. За любым развитием стоит увеличение нагрузки, ответственности и риска, а значит предприятию нужно постоянно двигаться вперед, искать новые методы оптимизации работы и автоматизации управления предприятием.

Совокупность работ, выполняемых на различных складах, примерно одинакова. Это объясняется тем, что в разных логистических процессах склады выполняют схожие функции: временное размещение и хранение материальных запасов; преобразование материальных потоков; предоставление информации о движении товаров.

Автоматизация складского учёта влияет на качество и скорость выполнения основных процессов на складе, приводит к совершенствованию систем управления и регулирования материальных и информационных потоков на складе. Это достигается путём внедрения современного программного обеспечения и компьютерного оборудования на предприятии.

Автоматизация складского учёта – способ оптимизации бизнес-процессов склада путём внедрения специализированных программных продуктов и оборудования.

Программное обеспечение позволяет не только учесть все, что хранится в складских помещениях. С его помощью можно выписывать задания для сотрудников. Они отправляются на устройства свободного человека, и он его выполняет.

Если все это будут делать только люди, это займет во много раз больше часов. Большие хранилища включают сотни тысяч наименований ассортимента, которые часто отличаются только цветом, незначительной деталью, одной цифрой в штрих-коде.

В крупных компаниях количество задач по движению может измеряться в тысячах ежедневно, поэтому подобное промедление способно обернуться дополнительными денежными затратами на простой и потерей клиентов. Не все готовы ждать слишком долго.

Каждому предприятию, которое занимается продажей готовой продукции необходимо вести учет продаваемых товарных единиц. После приобретения продукта товар попадает на склады предприятия. Когда на складе отсутствует автоматизированная система контроля за учетом товаров, размещением, предпродажной подготовкой, приходится сталкиваться с целым рядом проблем, связанных с возможностью оперативного получения информации. После автоматизации складского учета компания избавляется от фиксации всех операций вручную, что очень трудоёмко и не исключает большого количества ошибок из-за человеческого фактора.

Предметной областью является процесс складского учета на предприятии.

Целью выполнения курсового проекта является повышение скорости и качества выполнения основных процессов на складе путем автоматизации деятельности сотрудников предприятия.

Задачами данного курсового проекта являются:

- изучить предметную область;
- реализовать клиент-серверное приложение;
- связать сервер с базой данных;
- сделать приложение простым и понятным в использовании;
- предоставить возможность сохранения табличных результатов в файле;
- разработать базу данных;
- протестировать полученное приложение.

Преимуществами данной системы являются:

- быстрота обслуживания;
- экономия времени сотрудников склада;
- простота в использовании;
- быстрый доступ к постоянно меняющейся информации;
- доступ к информации в любое время суток из личного кабинета сотрудника.

1 ОСНОВНЫЕ ПОЛОЖЕНИЯ СИСТЕМЫ АВТОМАТИЗАЦИИ СКЛАДСКОГО УЧЕТА

Современный крупный склад – это сложное техническое сооружение, которое состоит из многочисленных взаимосвязанных элементов, имеет определенную структуру и выполняет ряд функций по преобразованию материальных потоков, а также накоплению, переработке и распределению грузов между потребителями.

При этом возможное многообразие параметров, технологических и объемно-планировочных решений, конструкций оборудования и характеристик разнообразной номенклатуры грузов, перерабатываемых на складах, относит склады к сложным системам. В то же время склад сам является всего лишь элементом системы более высокого уровня – логистической цепи, которая и формирует основные и технические требования к складской системе, устанавливает цели и критерии ее оптимального функционирования, диктует условия переработки груза.

Схематично склад предприятия может выглядеть следующим образом (см. рисунок 1.1):

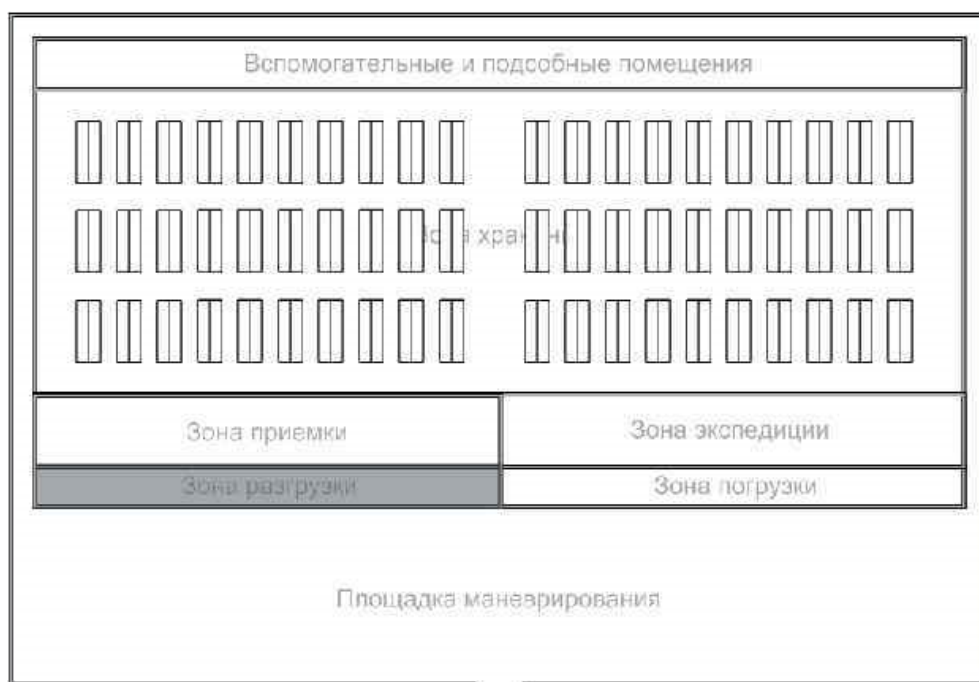


Рисунок 1.1 – Схема склада предприятия

Отсюда следует, что склад должен рассматриваться не изолированно, а как интегрированная составная часть логистической цепи. Только такой

подход позволит обеспечить успешное выполнение основных функций склада и достижение высокого уровня рентабельности.

В совокупности склад представляет собой помещение, в котором хранятся материальные ценности. Все они требуют учета. Для того, чтобы автоматизировать процесс складского учета созданы специальные программы, помогающие оперативно вести учет товара и автоматизировать финансовую деятельность.

Задачи склада:

- накопить нужный объем запасов;
- обеспечить груз подходящим уровнем безопасности;
- оптимизировать и ускорить погрузочно-разгрузочные процессы;
- рационально использовать помещения;
- организовать доставку;
- подготовить сырье к производству;
- отходы эффективно используются и перерабатываются;
- учитывать поступление и расходование, вести запись.

Но главной задачей складского учета все же является оптимизация бизнес-процессов и увеличение эффективности сотрудников. Таким образом работу учета можно разделить на следующие подзадачи:

- контролирование остатков и запасов. Для любого склада важно, чтобы учетное количество товаров совпадало с фактическим;
- объективное планирование закупок. Анализ остатков и движения товаров позволяет прогнозировать среднее количество продаж и планировать закупку таким образом, чтобы необходимых товаров всегда хватало, и они не оставались лежать на складе в виде неликвида;
- складской учет облегчает процесс инвентаризации и поиск необходимых товаров. Таким образом проводить пересчет и искать товары для выдачи клиентам становится гораздо легче;
- систематизация справочника товаров позволяет упорядочивать позиции, например, разделив их на категории и группы. Каждая категория хранится в отдельном месте, чтобы было проще комплектовать заказы и доставки;
- количество ошибок и краж заметно сократиться после внедрения системы для складского учета. Все оприходования, списания и перемещения будут зафиксированы документально и отображены в системе, которой вы пользуетесь. В итоге у вас станет меньше недостатков, излишков, пересортиц и краж, а также ошибок во время продаж и оформления заказов;

- с помощью оптимизации и зонирования территории вы сможете легко находить залежавшиеся единицы товаров, которые нужно списать или продать по акционной цене. Также можно обнаружить бракованные товары, для хранения которых стоит организовать отдельное место;

- эффективность сотрудников увеличится, так как им не придется тратить много времени на поиск товаров и комплектацию заказов.

За сложным термином автоматизация скрывается очень простая суть: автоматизация нужна для того, чтобы упорядочить работу склада и свести к минимуму участие человека. Все делает специальная программа для автоматизации учета. Высокого качества системы умеют следующее:

- контролировать складские остатки в режиме реального времени. Вы всегда точно знаете, сколько и какие товары есть в наличии. Ситуация, когда внезапно заканчивается определенная позиция, исключена;

- отслеживать все движения товаров. Отгрузка в магазин, миграция между складами, перемещения – все находится под контролем системы;

- сигнализировать о наличии товара на складе. Если что-то кончается, система вовремя предупредит об этом, и вы успеете сделать заказ у поставщика;

- формировать бухгалтерские отчеты и документы для поставщиков. Самостоятельно и в автоматическом режиме. Вам остается только их распечатать и подписать, если это необходимо;

- следить за сотрудниками. Она анализирует эффективность менеджеров, продавцов, кладовщиков и остальных лиц, участвующих в процессе;

- фиксировать продажи. Любая позиция списывается с остатков в тот момент, когда продавец пробивает кассовый чек;

- разрабатывать и внедрять программы лояльности для постоянных покупателей;

- контролировать продажи и формировать аналитику для руководителя;

- работать с несколькими складами, магазинами и торговыми точками.

Доступ к программе могут иметь несколько сотрудников одновременно;

- выгружать данные из других программ и форматов. Вбить вручную несколько сотен позиций – долго и сложно. Проще выгрузить информацию в систему их Excel-таблиц или других и баз данных.

Складские учетные системы повсеместно используются для проведения инвентаризаций на складах и осуществления контроля продукции. Применять эти программы можно также для контроля сервиса. Они помогут понять,

вовремя ли был отгружен товар для клиентов или вести учет по программам лояльности.

На рисунке 1.2 можно наглядно увидеть систему управления складом:



Рисунок 1.2 – Система управления складом

Преимущества складских учетных систем:

- программы для склада систематизируют материальные ценности, дают возможность отслеживать наличие товара на тот или иной момент времени;
- каждый продукт или товар имеет свою персональную карточку, в которой отражается его описание, характеристики, передвижение, количество и цена;
- система позволяет распечатывать все необходимые документы для первичного учета, например, накладные, расходные и приходные ордера, счета-фактуры;
- простой и удобный интерфейс позволяет спокойно работать с большим количеством данных;

- если организация планирует расширение, то в программе есть возможность добавлять дополнительные склады. После создания новых помещений можно настроить обмен данными между ними и другими подразделениями организации. Кроме этого, программа позволит следить за передвижением продукции и товаров между складами;

- есть возможность заносить и изменять информацию о сделках с поставщиками и покупателями;

- организации с упрощенной системой налогообложения «доходы за вычетом расходов» могут производить расчет для признания сумм расходов;

- прозрачный мониторинг позволяет в любой момент получить доступ к аналитическим данным программы;

- любые данные можно выгрузить в удобном вам формате и отправить по электронной почте;

Если программа складского учета имеет возможность обрабатывать заказы клиентов и выгружать итоговые отчеты, то вы сможете получать информацию о состоянии организации и оперативно исправлять недочеты.

Большинство конфигураций имеют возможность частичного или полного разграничения доступов. Например, полный доступ к данным может быть у руководителей, а частичный – рядовые сотрудники. Это позволяет сохранять конфиденциальность данных [1].

В разрабатываемом курсовом проекте присутствует разграничение ролей, где каждый сотрудник предприятия имеет доступ в ограниченному числу ресурсов, которые соответствуют занимаемой должности. Предусмотрены такие должности как: администратор, менеджер, специалист по доставке грузов. Администратор имеет полный доступ к базе сотрудников и может просматривать, добавлять, изменять информацию о сотруднике, а также удалять сотрудника в случае его увольнения. Роли администратора доступны и дополнительные функции, которые упростят и ускорят процесс взаимодействия с базой сотрудников. Менеджер имеет доступ к базе товаров, где менеджер может отслеживать состояние товаров, видоизменять базу данных в случае поступления или убытия нового товара, а также данная роль имеет дополнительные функции для комфортной работы с базой. Роль специалиста по доставке грузов заключается в просмотре листа задания, где любой специалист по работе с доставкой может выбрать любое задание на выполнение и тем самым удалить его из списка. Более подробно о функциональности программы, ограничения доступа к данным и дополнительным возможностям каждой из роли будет описано в следующих главах.

2 ПОСТАНОВКА ЗАДАЧИ И ОБЗОР МЕТОДОВ ЕЕ РЕШЕНИЯ

2.1 Постановка задачи

Задачей этого курсового проекта заключается в создании клиент-серверного приложения на основе многопоточного сервера, в котором реализовано взаимодействие с базой данных на объектно-ориентированном языке программирования Java.

Язык Java – это высокоуровневый объектно-ориентированный язык программирования, основанный на классах, который разработан таким образом, чтобы иметь как можно меньше зависимостей от реализации.

В отличие от серверного приложения, которое может быть реализовано, как в виде консольного, так и в виде оконного, клиентское приложение обязательно должно быть оконным и реализовано при помощи стандартных графических библиотек Java. Таких как:

- Swing;
- AWT;
- JavaFX.

Данный курсовой проект разрабатывался на языке Java версии SDK 17 с использованием набора инструментов для создания кроссплатформенных приложений JavaFX.

Разработанная система должна обладать следующей инфраструктурой:

- файлы должны работать в среде 32х разрядной ОС Windows 7 и выше;
- СУБД – Sybase SQL 11.0+, MS SQL Server 2008 R2+, MySQL 5.5+, PostgreSQL 9.0+, Java DB 10.x+;
- база данных должна генерироваться sql-скриптом;
- интерфейс программы и данные должны быть только на русском языке.

Связь между сервером и клиентом в приложении осуществляется при помощи TCP/IP. Серверная часть в виде консольного приложения, а клиентская будет иметь графический интерфейс.

При проектировании системы необходимо выполнить моделирование с использованием следующих стандартов:

- IDEF0 – функциональное моделирование процессов предметной области решаемой задачи (не менее чем 4 уровня). Используется CASE средство Allfusion Process Modeler;

- UML 2.0 – модели представления системы на основе UML;
- IDEF1.X – информационное моделирование.

Необходимо отладить и протестировать разработанное приложение для того, чтобы гарантировать его качественную работу в дальнейшем.

2.2 Объектно-ориентированный язык Java

Java – строго типизированный объектно-ориентированный язык программирования общего назначения, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle) [2].

Язык Java является объектно-ориентированным языком, а это означает, что писать программы на Java нужно с применением объектно-ориентированного стиля. И стиль этот основан на использовании в программе объектов и классов. Класс – это описание еще не созданного объекта, в некотором роде общий шаблон, состоящий из полей, методов и конструктора, а объект – экземпляр класса, созданный на основе этого описания [3].

Концепции ООП являются основополагающими элементами и составляют основу языка программирования Java. В рамках данного подхода выделяют следующие термины: абстракция, инкапсуляция, наследование и полиморфизм. Принцип ООП в Java позволяют программистам создавать компоненты, которые можно переиспользовать в различных частях программы не подвергая данные опасности.

Программы на Java транслируются в байт-код Java, выполняемый виртуальной машиной Java (JVM) – программой, обрабатывающей байтовый код и передающей инструкции оборудованию как интерпретатор.

Достоинством подобного способа выполнения программ является полная независимость байт-кода от операционной системы и оборудования, что позволяет выполнять Java-приложения на любом устройстве, для которого существует соответствующая виртуальная машина JVM (Java Virtual Machine).

Другой важной особенностью технологии Java является гибкая система безопасности, в рамках которой исполнение программы полностью контролируется виртуальной машиной. Любые операции, которые превышают установленные полномочия программы (например, попытка несанкционированного доступа к данным или соединения с другим компьютером), вызывают немедленное прерывание.

Подобная архитектура обеспечивает кроссплатформенность и аппаратную переносимость программ на Java, благодаря чему подобные программы без перекомпиляции могут выполняться на различных

платформах, а именно: Windows, Linux, Mac OS и т.д. Для каждой из платформ может быть своя реализация виртуальной машины JVM, но каждая из них может выполнять один и тот же код [4].

Часто к недостаткам концепции виртуальной машины относят снижение производительности. Ряд усовершенствований несколько увеличил скорость выполнения программ на Java:

- применение технологии трансляции байт-кода в машинный код непосредственно во время работы программы (JIT-технология) с возможностью сохранения версий класса в машинном коде;
- обширное использование платформенно-ориентированного кода (native-код) в стандартных библиотеках;
- аппаратные средства, обеспечивающие ускоренную обработку байт-кода (например, технология Jazelle, поддерживаемая некоторыми процессорами архитектуры ARM).

2.3 Технология JavaFX

JavaFX – это платформа на основе Java для создания приложений с насыщенным графическим интерфейсом. Может использоваться как для создания настольных приложений, запускаемых непосредственно из-под операционных систем, так и для интернет-приложений (RIA), работающих в браузерах, и для приложений на мобильных устройствах [5].

JavaFX позволяет создавать приложения с богатой насыщенной графикой благодаря использованию аппаратного ускорения графики и возможностей GPU.

JavaFX призвана заменить упоминавшую ранее библиотеку Swing. Платформа JavaFX конкурирует с Microsoft Silverlight, Adobe Flash и аналогичными системами.

С помощью JavaFX можно также создавать программы для различных операционных систем: Windows, MacOS, Linux и для самых различных устройств: десктопы, смартфоны, планшеты, встроенные устройства, ТВ. Приложение на JavaFX будет работать везде, где установлена исполняемая среда Java (JRE).

JavaFX предоставляет богатый набор графических и мультимедийных API-интерфейсов, а также использует современный графический процессор с помощью аппаратно-ускоренной графики. JavaFX также предоставляет интерфейсы, с помощью которых разработчики могут комбинировать графическую анимацию и управление пользовательским интерфейсом.

На рисунке 2.1 изображен результат проектирования оконного приложения посредством Java FX:

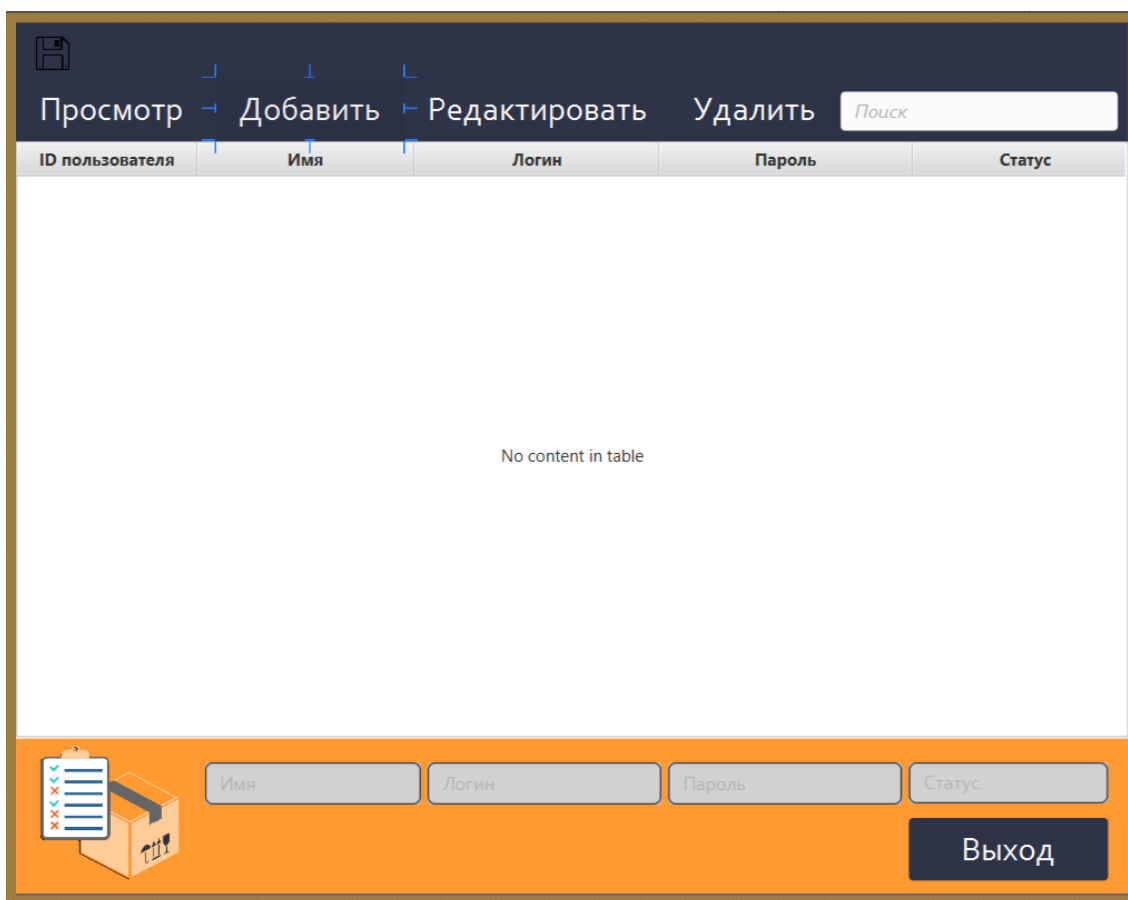


Рисунок 2.1 – Результат проектирования графического интерфейса с помощью JavaFX

JavaFX предоставляет большие возможности по сравнению с рядом других подобных платформ, в частности, по сравнению со Swing. Это и большой набор элементов управления, и возможности по работе с мультимедиа, двухмерной и трехмерной графикой, декларативный способ описания интерфейса с помощью языка разметки FXML, возможность стилизации интерфейса с помощью CSS, интеграция со Swing и многое другое.

Рассматриваемый набор инструментов для создания кроссплатформенных приложений JavaFX имеет ряд особенностей:

- изначально поставляется с большим набором частей графического интерфейса, таких как кнопки, текстовые поля, таблицы, деревья, меню, диаграммы и т.д.;
- часто использует стили CSS, и есть возможность использовать специальный формат FXML для создания GUI, а не делать это в коде Java. Это

облегчает быстрое размещение графического интерфейса пользователя или изменение внешнего вида или композиции;

- имеет готовые к использованию части диаграммы, поэтому нет необходимости писать их с нуля;
- дополнительно поставляется с поддержкой 3D графики, которая часто полезна при разработки каких-либо игр или подобных приложений.

2.4 Протокол TCP/IP

Протокол TCP/IP (Transmission Control Protocol/Internet Protocol) представляет собой стек сетевых протоколов, повсеместно используемый для Интернета и других подобных сетей.

TCP (Transmission Control Protocol) – один из основных сетевых протоколов Интернета, предназначенный для управления передачей данных в сетях и подсетях TCP/IP. Данный протокол обеспечивает надежную, последовательную передачу данных, а также отказоустойчивость при возникновении некоторых проблем. Процесс установки соединения по протоколу TCP изображен на рисунке 2.2.



Рисунок 2.2 – Установка соединения по протоколу TCP

Название TCP/IP произошло от двух наиболее важных протоколов:

– IP (интернет протокол) – отвечает за передачу пакета данных от узла к узлу. IP пересылает каждый пакет на основе четырехбайтного адреса назначения (IP-адрес).

– TCP (протокол управления передачей) – отвечает за проверку корректной доставки данных от клиента к серверу. Данные могут быть потеряны в промежуточной сети. В протоколе TCP добавлена возможность обнаружения ошибок или потерянных данных и, как следствие, возможность запросить повторную передачу, до тех пор, пока данные корректно и полностью не будут получены.

На сегодняшний день этот сетевой протокол используется как для связи компьютеров всемирной сети, так и в подавляющем большинстве корпоративных сетей. В наши дни используется версия протокола IP, известная как IPv4.

Протокол TCP/IP основан на OSI и так же, как предшественник, имеет несколько уровней, которые и составляют его архитектуру. Всего выделяют 4 уровня – канальный, межсетевой, транспортный и прикладной.

На рисунке 2.3 изображены уровни модели TCP/IP.

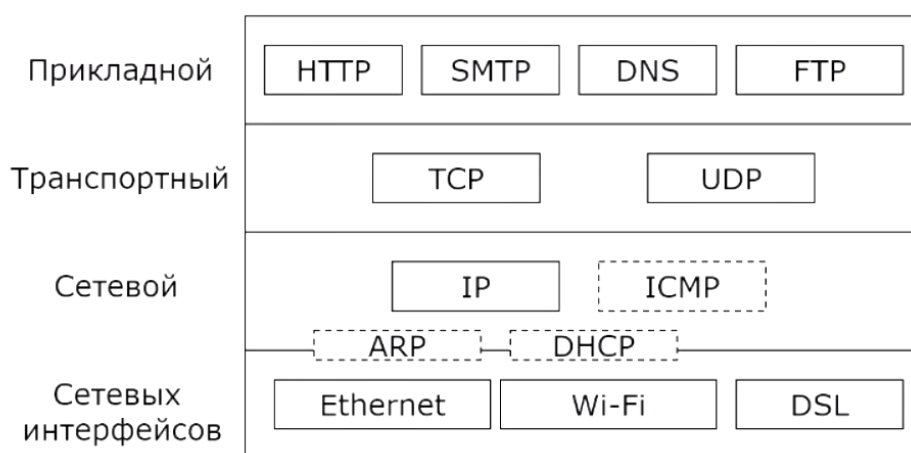


Рисунок 2.3 – Уровни модели TCP/IP

Протоколы этих уровней полностью реализуют функциональные возможности модели OSI. На стеке протоколов TCP/IP построено всё взаимодействие пользователей в IP-сетях. Стек является независимым от физической среды передачи данных [6].

Лидирующая роль стека TCP/IP объясняется следующими его свойствами:

- это наиболее завершённый стандартный и в то же время популярный стек сетевых протоколов, имеющий многолетнюю историю;
- почти все большие сети передают основную часть своего трафика с помощью протокола TCP/IP;
- это метод получения доступа к сети Internet;
- все современные операционные системы поддерживают стек TCP/IP;
- это гибкая технология для соединения разнородных систем как на уровне транспортных подсистем, так и на уровне прикладных сервисов;
- это устойчивая масштабируемая межплатформенная среда для приложений клиент-сервер.

Таким образом, стек протоколов TCP/IP получил большую популярность, и связан это прежде всего с распространением и развитием сетевых технологий internet/intranet, базирующихся на стеке протоколов TCP/IP, а также с более точным отражением процесса сетевого взаимодействия.

2.5 Объектно-реляционная система управления базами данных MySQL

MySQL – свободная реляционная система управления базами данных с открытым исходным кодом. На сегодняшний день это одна из самых популярных систем управления базами данных, которая основана на модели «клиент-сервер». Поэтому часто можно услышать словосочетание «MySQL-сервер». Это действительно сервер, только сервер базы данных. Он создан для обеспечения доступа к данным для других сервисов и приложений [7].

Рассматриваемая СУБД обладает кроссплатформенностью, имеются дистрибутивы под самые различные ОС, в том числе наиболее популярные версии Linux, Windows, MacOS.

Из преимуществ MySQL стоит отметить простоту использования, предоставление большого функционала, гибкость, низкую стоимость владения (относительно платных СУБД), масштабируемость, а также обеспечение хорошей скорости и производительности.

MySQL позволяет хранить целочисленные значения со знаком и беззнаковые, длиной в 1, 2, 3, 4 и 8 байтов, работает со строковыми и текстовыми данными фиксированной и переменной длины, позволяет осуществлять SQL-команды SELECT, DELETE, INSERT, REPLACE и UPDATE, обеспечивает полную поддержку операторов и функций

в SELECT- и WHERE- частях запросов, работает с GROUP BY и ORDER BY, поддерживает групповые функции COUNT(), AVG(), STD(), SUM(), MAX() и MIN(), позволяет использовать JOIN в запросах, в т.ч. LEFT OUTER JOIN и RIGHT OUTER JOIN, поддерживает репликацию, транзакции, работу с внешними ключами и каскадные изменения на их основе, а также обеспечивает многие другие функциональные возможности.

Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа MyISAM, поддерживающие полнотекстовый поиск, так и таблицы InnoDB, поддерживающие транзакции на уровне отдельных записей. Есть и другие типы таблиц, разработанные сообществом.

Как и любой программный продукт, система MySQL имеет определенные ограничения в своем функционале, что не позволяет использовать ее для работы с приложениями. Из недостатков можно выделить недостаточную надежность. В вопросах надежности некоторых процессов по работе с данными (например, связь, транзакции, аудит) MySQL уступает некоторым другим СУБД.

В данной работе использована версия MySQL 8.0.26. Для работы с MySQL в Java необходимо установить официальный драйвер MySQL Connector/J.

2.6 Паттерн проектирования «Singleton»

Одиночка (англ. Singleton) – это порождающий паттерн, который гарантирует существование только одного объекта определённого класса, а также позволяет получить доступ до этого объекта из любого места программы.

У класса есть только один экземпляр, и он предоставляет к нему глобальную точку доступа. При попытке создания данного объекта 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. Например, к описанным компонентам класса можно обращаться через интерфейс, если такая возможность поддерживается языком [8].

В объектно-ориентированном программировании существует правило хорошего тона – Single Responsibility Principle. Согласно этому правилу,

каждый класс должен отвечать лишь за один какой-то аспект. Совершенно очевидно, что любой Singleton-класс отвечает сразу за две вещи: за то, что класс имеет лишь один объект, и за реализацию того, для чего этот класс вообще был создан.

2.7 Паттерн проектирования «Command»

Паттерн Command (англ. «Команда») – это поведенческий паттерн проектирования, который превращает запросы в объекты, позволяя передавать их как аргументы при вызове методов, ставить запросы в очередь, логировать их, а также поддерживать отмену операций [9]. Цель данного паттерна заключается в создании структуры, в которой класс-отправитель и класс-получатель не зависят друг от друга напрямую. Организация обратного вызова к классу, который включает в себя класс-отправитель.

В объектно-ориентированном программировании шаблон проектирования Команда является поведенческим шаблоном, в котором объект используется для инкапсуляции всей информации, необходимой для выполнения действия или вызова события в более позднее время. Эта информация включает в себя имя метода, объект, который является владельцем метода и значения параметров метода.

Четыре термина всегда связаны с шаблоном Команда: команды (command), приёмник команд (receiver), вызывающий команды (invoker) и клиент (client). Объект Command знает о приёмнике и вызывает метод приемника. Значения параметров приёмника сохраняются в команде. Вызывающий объект (invoker) знает, как выполнить команду и, возможно, делает учёт и запись выполненных команд. Вызывающий объект (invoker) ничего не знает о конкретной команде, он знает только об интерфейсе. Оба объекта (вызывающий объект и несколько объектов команд) принадлежат объекту клиента (client). Клиент решает, какие команды выполнить и когда. Чтобы выполнить команду он передает объект команды вызывающему объекту (invoker). Использование командных объектов упрощает построение общих компонентов, которые необходимо делегировать или выполнять вызовы методов в любое время без необходимости знать методы класса или параметров метода. Использование вызывающего объекта (invoker) позволяет вести учёт выполненных команд без необходимости знать клиенту об этой модели учёта.

3 ФУНКЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ НА ОСНОВЕ СТАНДАРТА IDEF0

Методология функционального моделирования IDEF0 – это технология описания системы в целом как множества взаимозависимых действий или функций.

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 [10].

Описание системы с помощью IDEF0 называется функциональной моделью. Функциональная модель предназначена для описания существующих бизнес-процессов, в котором используются как естественный, так и графический языки. Для передачи информации о конкретной системе источником графического языка является сама методология IDEF0.

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.

Важно отметить функциональную направленность: IDEF0-функции системы исследуются независимо от объектов, которые обеспечивают их выполнение. «Функциональная» точка зрения позволяет четко разделить аспекты назначения системы от аспектов ее физической реализации.

Модель строится методом декомпозиции: от крупных составных структур к более мелким, простым. Элементы каждого уровня декомпозиции представляют собой действия по переработке информационных или материальных ресурсов при определенных условиях с использованием заданных механизмов. Каждое действие раскладывается на более мелкие операции по переработке определенной части информационных или материальных ресурсов при определенных условиях с использованием части заданных механизмов. Аналогично раскладываются операции. Последний шаг декомпозиции должен приводить к получению модели, степень детализации

которой удовлетворяет требованиям, заданным в самом начале процесса создания модели [11].

Стандарт IDEF0 представляет организацию как набор модулей, здесь существует правило – наиболее важная функция находится в верхнем левом углу, кроме того, существуют правила сторон:

- стрелка входа всегда приходит в левую кромку активности;
- стрелка управления – в верхнюю кромку;
- стрелка механизма – нижняя кромка;
- стрелка выхода – правая кромка.

Стрелки подписываются при помощи имен существительных (план, правила), а блоки – при помощи глаголов, т.е. в них описываются действия, которые производятся (авторизация, создание).

На рисунке 3.1 представлена контекстная диаграмма проекта верхнего уровня. Она определяет главный процесс. Входная информация: данные о готовой продукции, данные о сотрудниках склада. Информация обрабатывается и конвертируется в конечную цель: отгрузка продукции. Управляющий механизм: персонал склада, работники отдела продаж, сервер и БД программы. Механизм ограничения: нормативные документы (ЛНПА, ГОСТ, ТУ и пр. нормативные документы, Лицензия на занятие торговлей).

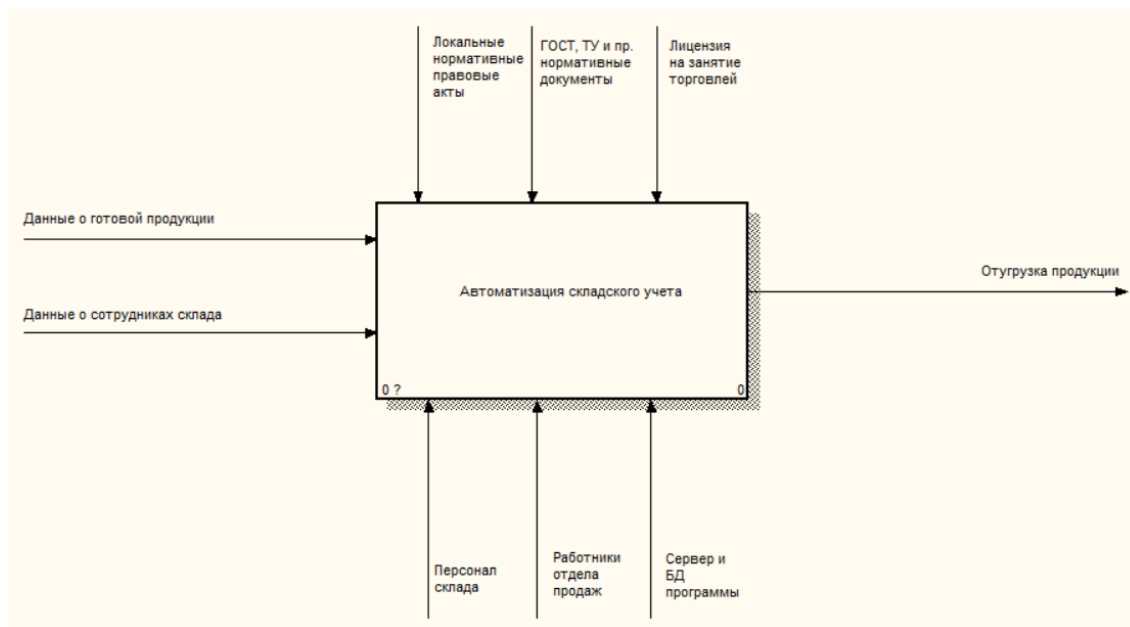


Рисунок 3.1 – Контекстная диаграмма автоматизированной системы складского учета

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

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.

Для того чтобы описать работу функционального блока более подробно, выполняется его декомпозиция, и моделируется диаграмма второго уровня, которая представлена на рисунке 3.2. На ней представлено пять функциональных блока декомпозиции.

Данный уровень включает в себя:

- авторизация;
- работа с продукцией;
- изменение объемов продукции на складе;
- создание текстового отчета;
- отправка задания специалисту по доставке грузов.

Конечная цель системы: отгрузка продукции.

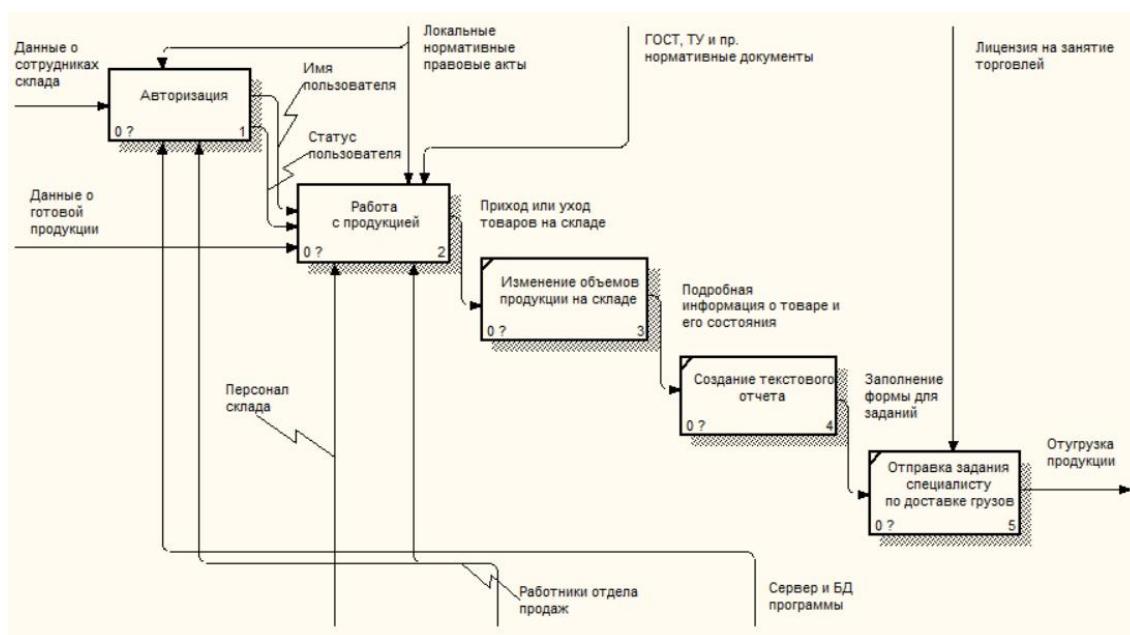


Рисунок 3.2 – Декомпозиция функционального блока

Далее следует декомпозиция процесса «Авторизация», результат которой показан на рисунке 3.3.

На данном уровне выделены следующие подпроцессы:

- ввести логин;

- ввести пароль
- проверить на наличие пользователя в системе;
- получить имя и статуса пользователя.

Итоговая цель: имя и статус пользователя.

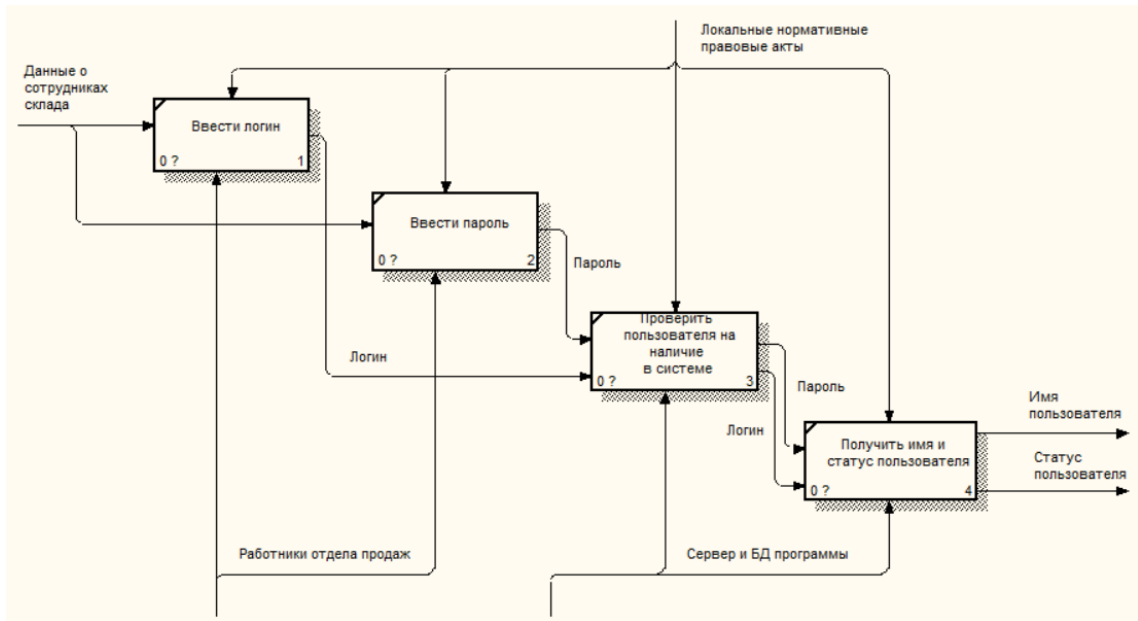


Рисунок 3.3 – Декомпозиция блока «Авторизация»

На следующем этапе разработки модели выполняется декомпозиция процесса «Работа с продукцией», показанная на рисунке 3.4.

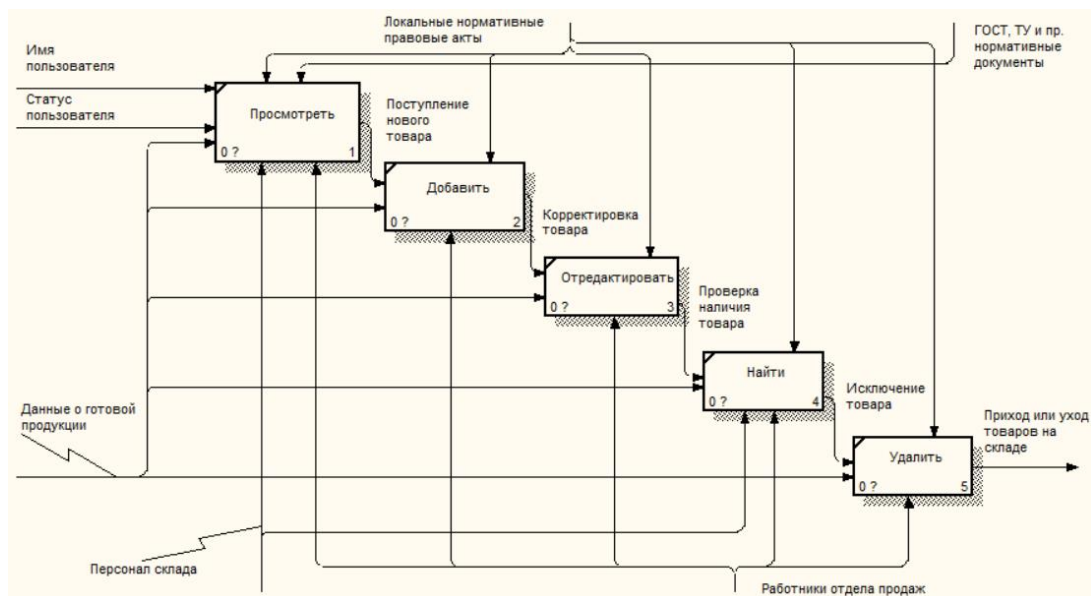


Рисунок 3.4 – Декомпозиция блока «Работа с продукцией»

В данном уровне существуют следующие подпроцессы:

- просмотреть;
- добавить;
- отредактировать;
- найти
- удалить.

Итоговая цель подпроцесса: приход и/или уход товаров на складе.

В управление рассматриваемого бизнес-процесса участвуют такие документы как: локальные нормативные правовые акты (ЛНПА), лицензия на занятие торговлей, ГОСТ, ТУ и прочие нормативные документы.

На каждом предприятии существуют свои внутренние правила, которые сотрудники должны знать и придерживаться соответствующим должностным пунктам. За это отвечают локальные нормативные правовые акты. ЛНПА является внутренним документом и разрабатывается для конкретного предприятия и действует только в рамках данной предприятия.

Также на уровне государства существуют правила и соответствующие нормативные документы, которыми должны руководствоваться предприятия в той или иной отрасли. Примером нормативных документов может стать ГОСТ или ТУ.

После соблюдения все правил и нормативных норм предприятие может приступить к полноценному функционированию.

В результате функционального моделирования бизнес-процесса на основе стандарта IDEF0 удалось построить логику функционирования предприятия, где можно сделать следующий вывод: после последовательного выполнения всех процессов происходит отгрузка продукции.

4 ИНФОРМАЦИОННАЯ МОДЕЛЬ СИСТЕМЫ СКЛАДСКОГО УЧЕТА И ЕГО ОПИСАНИЕ

Информационная система складского учета предназначена для учета и контроля движения материалов на складе, их приёмки и отпуска. Основными пользователями этой системы могут стать представители малого бизнеса, которым необходим электронный учет материалов. Он позволит существенно упростить и ускорить работу по учету, сделать этот процесс более наглядным, точным и удобным [12].

Объектами автоматизации информационной системы являются товары, хранящиеся на складе, учет операций по их движению, а также штат сотрудников склада. Все это будет храниться в электронной базе данных, сформированной на основе документации.

Информационная система складского учета позволит посмотреть и вносить все операции по приёмке и отпуску материалов, посмотреть все материалы, которые хранятся на складе, количество и подробную информацию о них.

Между объектами предметной области могут существовать связи, имеющие различный содержательный смысл. Эти связи могут быть обязательными или факультативными.

Если вновь порожденный объект оказывается по необходимости связанным с каким-либо объектом предметной области, то между этими двумя объектами существует обязательная связь. В противном случае связь является факультативной (необязательной).

Для того, чтобы обеспечить минимальную избыточность данных и физического объёма данных, а также ускоренный доступ, необходимо привести информационную модель к нормальной форме. При этом система называется нормализованной только в том случае, если она способна удовлетворять следующим требованиям: надежное хранение и обновление данных. Это поможет сократить риск потери данных или их искажения при внесении в БД.

Процесс построения информационной модели состоит из следующих шагов:

- определение сущностей;
- определение зависимостей между сущностями;
- задание первичных и внешних ключей;
- определение атрибутов сущностей;

- составление логической модели;
- переход к физическому описанию модели.

Логический уровень описывает схему базы данных в терминах выбранной модели данных. Это абстрактный взгляд на данные, на нем данные представляются так, как выглядят в реальном мире, и могут называться так, как они называются в реальном мире. Объекты модели, представляемые на логическом уровне, называются сущностями и атрибутами. Логическая модель данных может быть построена на основе другой логической модели, например на основе модели процессов. Логическая модель данных является универсальной и никак не связана с конкретной реализацией СУБД.

Спроектированная логическая модель изображена на рисунке 4.1:

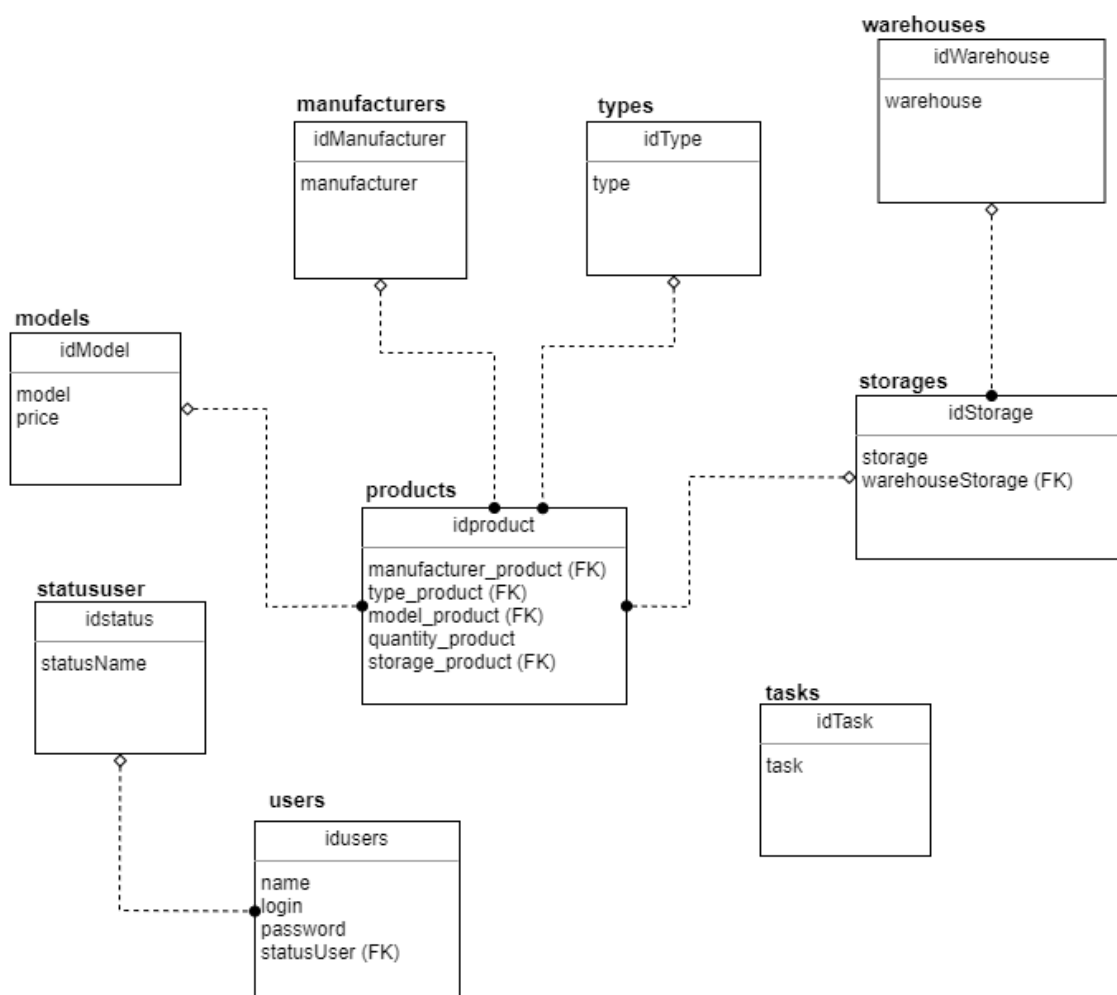


Рисунок 4.1 – Логическая модель системы

Физическая модель данных, напротив, зависит от конкретной СУБД, фактически являясь отображением системного каталога. В физической модели

содержится информация о всех объектах БД. Поскольку стандартов на объекты БД не существует, физическая модель зависит от конкретной реализации СУБД. Следовательно, одной и той же логической модели могут соответствовать несколько разных физических моделей. Если в логической модели не имеет значения, какой конкретно тип данных имеет атрибут, то в физической модели важно описать всю информацию о конкретных физических объектах – таблицах, колонках, индексах, процедурах и т. д. Разделение модели данных на логические и физические позволяет решить несколько важных задач.

На рисунке 4.2 изображена физическая модель системы складского учета:

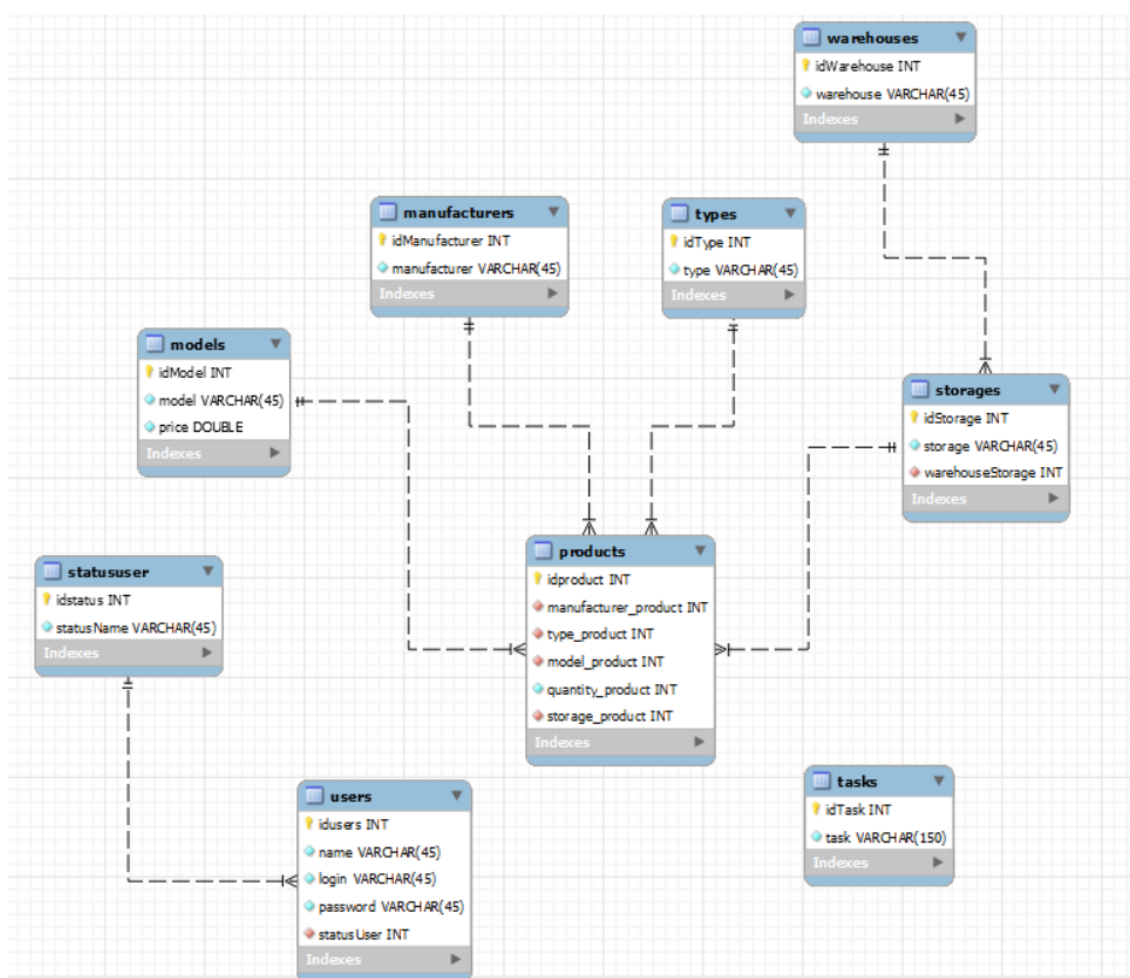


Рисунок 4.2 –Физическая модель системы

При графической иллюстрации можно заметить, что база данных состоит из 9 сущностей. Под сущностями следует понимать, что это объект предметной области, который исследуется и моделируется. Иными словами,

сущность – это любой различимый объект, о котором необходимо хранить информацию в базе данных [13].

Каждая сущность содержит атрибуты, которые определяют и отображают свойства и качества объекта. Рассмотрим каждую из сущностей:

1) users. Сущность, которая включает в себя информацию о пользователях, а именно: имя, логин, пароль, статус пользователя. В ячейке statusUser хранится айди статуса пользователя, по которому из сущности statususer вызывается роль каждого пользователя. Сущность users имеет следующие атрибуты:

- idusers (первичный ключ, тип int) – уникальный автоинкрементный номер, присваиваемый каждому пользователю.

- name (тип varchar (45)) – имя пользователя;

- login (тип varchar (45)) – логин пользователя;

- password (тип varchar (45)) – пароль пользователя;

- statusUser (внешний ключ, тип int) – статус пользователя.

2) statususer. В данной сущности хранятся все роли. Каждый авторизованный пользователь может иметь только одну роль, которая хранится в данной сущности. Сущность statususer включает в себя следующие атрибуты:

- idstatus (первичный ключ, тип int) – уникальный автоинкрементный код статуса;

- statusName (тип varchar (45)) – тип статуса.

3) products. Сущность, которая включает в себя информацию о готовой продукции, хранящейся на складе. Включает в себя следующие атрибуты:

- idproduct (первичный ключ, тип int) – уникальный автоинкрементный код продукта;

- manufacturer_product (внешний ключ, тип int) – хранится код производителя, который берется из сущности manufacturers;

- type_product (внешний ключ, тип int) – хранится код производителя, который берется из сущности types;

- model_product (внешний ключ, тип int) – хранится код модели, который берется из сущности models;

- quantity_product (тип int) – количество готовой продукции;

- storage_product (внешний ключ, тип int) – хранится код хранилища, который берется из сущности storages.

4) models. Сущность, которая включает в себя модели товаров, хранящиеся на складе, а также цену каждого товара. Сущность включает в себя следующие атрибуты:

- idModel (первичный ключ, тип int) – уникальный автоинкрементный код модели;
- model (тип varchar (45)) – модель продукта;
- price (тип double) – цена продукта.

5) manufacturers. В данной сущности хранятся производители готовой продукции, присутствующая на складе. Включает в себя следующие атрибуты:

- idManufacturer (первичный ключ, тип int) – уникальный автоинкрементный код производителя;
- manufacturer (тип varchar (45)) – производитель готовой продукции;

6) types. В сущности хранятся все виды товаров, присутствующие на складе. Включает в себя следующие атрибуты:

- idType (первичный ключ, тип int) – уникальный автоинкрементный код вида продукции;
- type (тип varchar (45)) – вид товара.

7) storages. Сущность, содержащая в себе номера хранилищ одного из склада. Включает в себя следующие атрибуты:

- idStorage (первичный ключ, тип int) – уникальный автоинкрементный код хранилища;
- storage (тип varchar (45)) – хранилище одного из склада предприятия;
- warehouseStorage (внешний ключ, тип int) – хранит в себе код склада, который берется из сущности warehouses.

8) warehouses. Сущность, содержащая номера складов предприятия. Включается в себя следующие атрибуты:

- idWarehouse (первичный ключ, тип int) – уникальный автоинкрементный код склада.
- warehouse (тип varchar (45)) – хранит в себе номер склада предприятия.

9) tasks. Сущность, в которой хранятся все задания, которые следует выполнить специалистам по доставке грузов. Включает в себя:

- idTask (первичный ключ, тип int) – уникальный автоинкрементный код задачи;
- task (тип varchar (150)) – задание для выполнения специалистом по доставке грузов.

Для некоторых атрибутов было принято решение объявить атрибут автоинкрементным. Это было сделано для того, чтобы при добавлении нового пользователя в базу или при поступлении нового товара на склад не указывать уникальный номер вручную.

Нормализация базы данных – это метод организации данных в базе данных.

Правила нормализации выделяют следующие нормальные формы:

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

Для того, чтобы база данных находилась в первой нормальной форме, она должна соответствовать следующим четырем правилам:

- 1) Она должна иметь только атомарные значения атрибутов.
- 2) Значения, хранящиеся в столбце, должны принадлежать одному домену.
- 3) Все столбцы таблицы должны иметь уникальные имена.
- 4) Порядок, в котором хранятся данные, не имеет значения.

Чтобы база данных находилась во второй нормальной форме, она должна отвечать следующим условиям:

- таблица должна быть в первой нормальной форме;
- таблица не должна иметь частичных зависимостей.

База данных находится в третьей нормальной форме, когда:

- таблица находится во второй нормальной форме;
- таблица не имеет транзитивных зависимостей.

Исходя из вышесказанного, спроектированная база данных, используемая в данном курсовом проекте, отвечает требованиям третьей нормальной формы.

5 ОПИСАНИЕ АЛГОРИТМА, РЕАЛИЗУЮЩЕГО БИЗНЕС-ЛОГИКУ СЕРВЕРНОЙ ЧАСТИ СИСТЕМЫ

Бизнес-логика в разработке информационных систем – совокупность правил, принципов, зависимостей поведения объектов предметной области (области человеческой деятельности, которую система поддерживает). Иначе можно сказать, что бизнес-логика – это реализация правил и ограничений автоматизируемых операций.

Бизнес-логика данного курсового проекта сосредоточена в серверной части.

В приложениях с бизнес-логикой на сервере используется возможность современных серверов БД исполнять хранимые SQL процедуры на сервере, куда и переносится максимально возможная часть бизнес-логики. Требования к серверу БД возрастают, однако резко понижаются требования к клиентским машинам (за счет выноса с них бизнес-логики) и к пропускной способности сети (клиенту передаются только данные, необходимые пользователю).

Одним из основных алгоритмов, реализующим бизнес-логику, является проверка наличия пользователя в системе. После запуска приложения появляется стартовое окно, предлагающее гостю авторизоваться и просит ввести логин и пароль. После получения логина и пароля система передает управление серверу, который сначала проверяет существование такого пользователя в сущности users. Если пользователя не существует, то система произведет анимацию в области заполнения логина и пароля. Анимация даст понять пользователю что такого аккаунта не существует, либо пользователь ввел данные некорректно. В случае если пользователь ввел данные и такой аккаунт существует, то сервер вернет то графическое окно, которое соответствует статусу только что введенных для инициализации значений. Если же пользователь ранее не был зарегистрирован в системе, то на стартовом окне также есть соответствующая кнопка, которая позволит зарегистрироваться новому пользователю.

После того как пользователь авторизовался или зарегистрировался, на экран появляется главное меню, где расположен основной функционал. Пользователю будут доступны такие функции как: просмотр товаров, добавление товара, редактирование товара, удаление товара, создание текстового отчета, поиск нужного товара по ключевым словам, а также возможность добавление задач специалистам по доставке грузов в случае поступления или убытия продаваемой продукции.

В данном курсовом проекте шла разработка системы по автоматизации складского учета, а это значит, что основным алгоритмом будет добавление поступившего товара в базу.

Алгоритм добавления товара в систему приведена на рисунке 5.1:

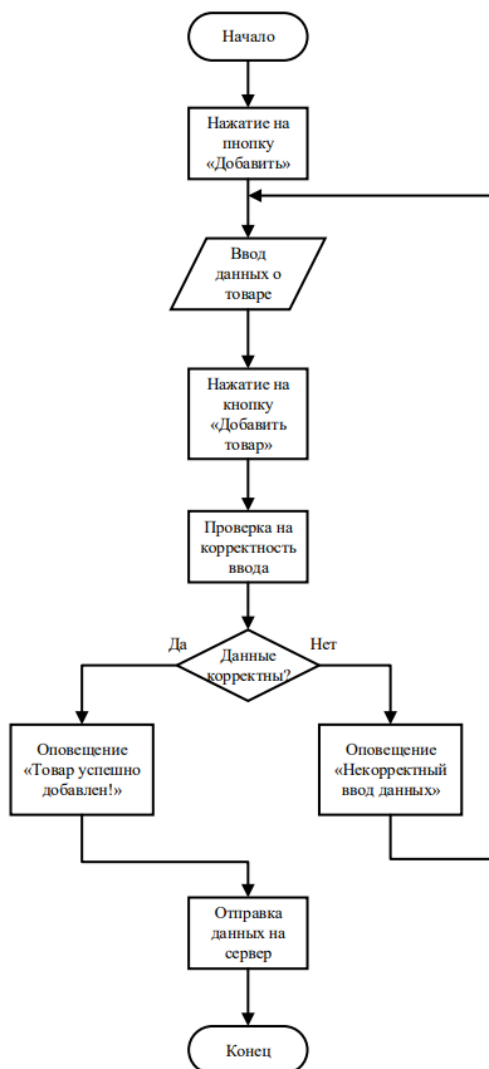


Рисунок 5.1 – Блок-схема алгоритма добавления товара

После нажатия кнопки «добавить товар», которая располагается на главном окне, открывается модальное окно, куда пользователь будет вводить информацию о поступившем товаре. Для минимизации некорректного ввода были предусмотрены выпадающие списки, где пользователь сможет выбор производителя, тип и номер хранилища. После ввода данных о товаре пользователю стоит нажать на кнопку добавить, после чего введенные данные отправляются на сервер. В случае некорректного ввода система оповестит об этом пользователя.

6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Для правильного использования программы и ознакомления со всеми возможностями программы следует изучить руководство пользователя. Руководство пользователя позволяет комфортно работать в системе и быстро получать необходимую информацию. В данном курсовом проекте предусмотрено несколько ролей, так как программный продукт будет введен в эксплуатацию на складе для автоматизации складского учета товарно-материальных ценностей, то есть данным продуктом будут пользоваться различные сотрудники предприятия. Каждый сотрудник имеет свои обязанности, следовательно, сотрудник должен иметь доступ к информации, которая соответствует занимаемой должности.

6.1 Руководство менеджера

В руководстве менеджера отражен основной функционал для комфортной работы с программой автоматизации складского учета.

После запуска программы открывается стартовая страница для входа в систему, изображенную на рисунке 6.1. Для авторизации следует ввести логин и пароль. В случае, если пользователь ввел некорректные данные, то в области ввода логина и пароль будет воспроизведена анимация.

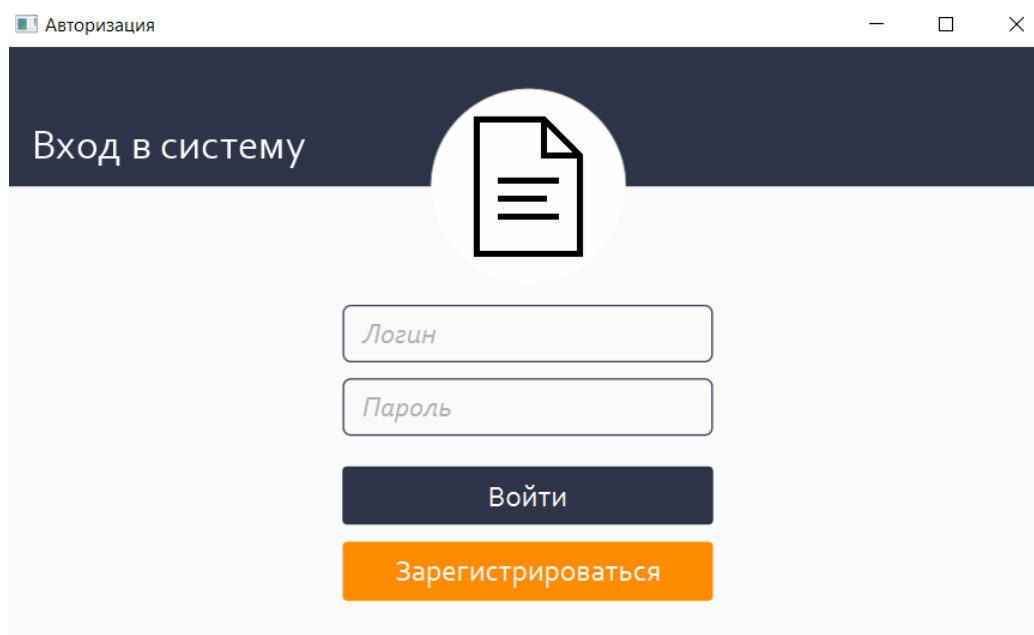



Рисунок 6.1 – Авторизация пользователя

Если пользователь впервые зашел в программу по складскому учету и хочет зарегистрироваться, то на стартовом окне присутствует кнопка «Зарегистрироваться», после чего пользователь будет перенесен на окно регистрации. Окно регистрации изображено на рисунке 6.2:

Регистрация

[← к авторизации](#)

Регистрация



Зарегистрироваться

Рисунок 6.2 – Регистрация пользователя

После успешного входа в систему в качестве менеджера открывается главное окно приложения. На рисунке 6.3 изображено окно главное окно пользователя.

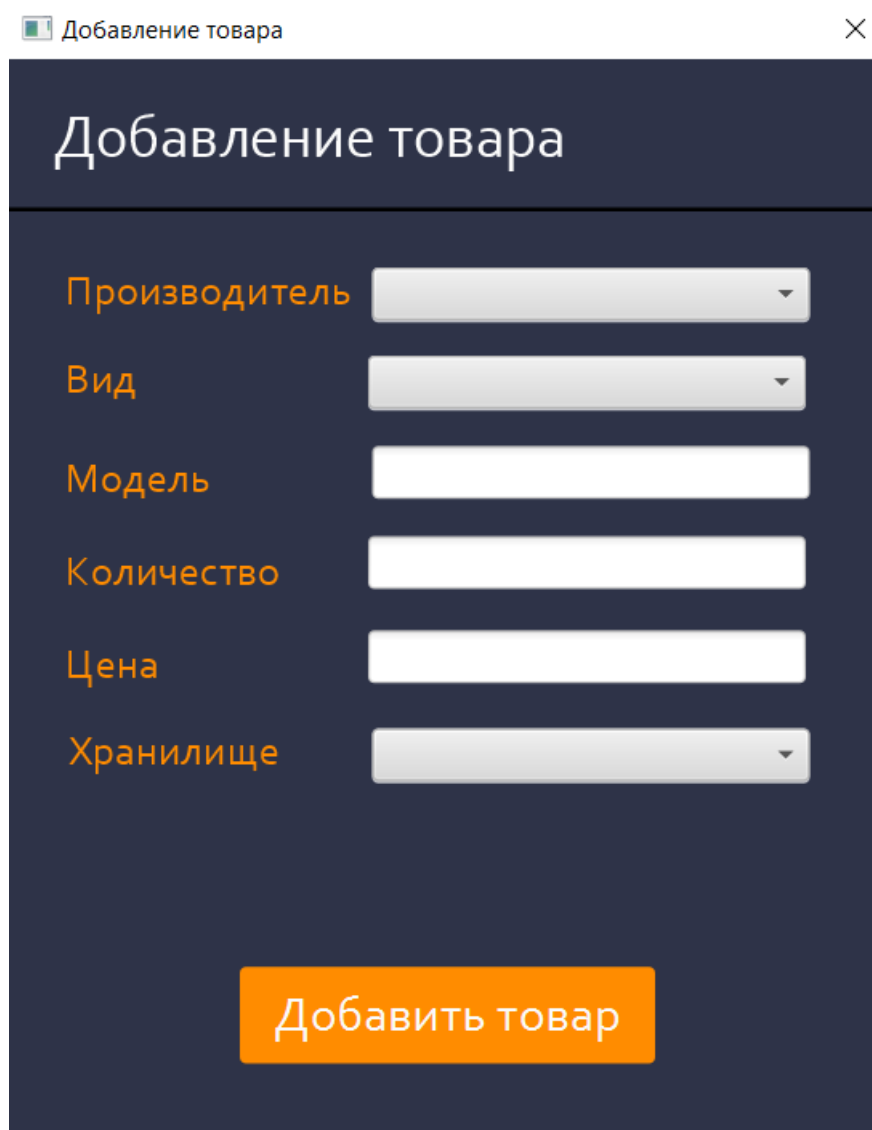
[illegible]

Рисунок 6.3 – Окно пользователя после авторизации

Менеджер имеет ряд функций:

- просматривать товары;
- добавлять товары;
- редактировать товары;
- удалять товары;
- поиск по ключевым словам;
- добавление задачи для специалиста по доставке грузов;
- создание текстового отчета.

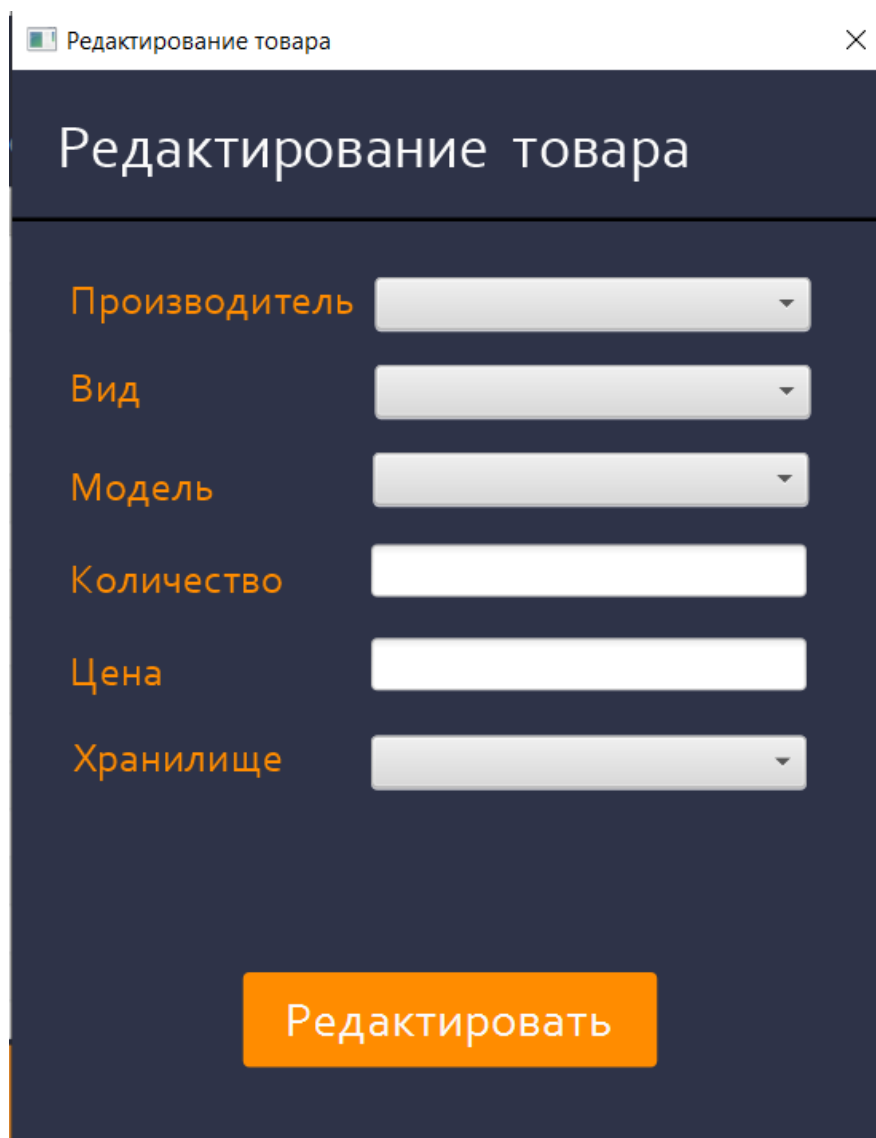
При нажатии на кнопку «Добавить» откроется новое окно для добавления нового товара в систему. Открываемое окно является модальным. Процесс добавления товара изображен на рисунке 6.4.



The image shows a modal window titled "Добавление товара" (Add Item) with a dark blue background. At the top, there is a title bar with a small icon and the text "Добавление товара", and a close button (X) on the right. The main content area contains six labels in orange text, each followed by a white input field: "Производитель" (Manufacturer) with a dropdown arrow, "Вид" (Type) with a dropdown arrow, "Модель" (Model) with a text input, "Количество" (Quantity) with a text input, "Цена" (Price) with a text input, and "Хранилище" (Storage) with a dropdown arrow. At the bottom center, there is a large orange button with the white text "Добавить товар" (Add Item).

Рисунок 6.4 – Добавление товара

Если пользователь хочет отредактировать существующий товар, то откроется соответствующее модальное окно:



Редактирование товара

Редактирование товара

Производитель

Вид

Модель

Количество

Цена

Хранилище

Редактировать

Рисунок 6.5 – Редактирование товара

В случае, если менеджер нацелен удалить товар, то ему следует выбрать товар в таблице, а затем нажать на кнопку «Удалить».

Зачастую в базе хранится крайне много товарных единиц и найти нужный товар для дальнейшего взаимодействия с ним крайне сложно. Поэтому, для упрощения поиска нужного товара в верхней части окна располагается поиск существующих товаров.

Поиск товара по ключевым словам изображен на рисунке 6.6:

ID Товара	Тип	Производитель	Модель товара	Количество	Цена	Склад	Хранилище
1	Ноутбук	Lenovo	IdeaPad S340-15API	3	1800.0	№1 г.Гом...	№1.1
30	Ноутбук	Lenovo	Lenovo IdeaPad3	6	2270.0	№1 г.Ми...	№2.1

Рисунок 6.6 – Поиск товаров

Также менеджер имеет возможность сохранить данные в формате текстового отчета. Создание текстового отчета продемонстрировано на рисунке 6.7:

Создание текстового отчета

Название файла:

Сохранить

Рисунок 6.7 – Создание текстового отчета

При создании пользователь самостоятельно задает название файла. В курсовом проекте была предусмотрена проверка на корректность ввода. В случае, если пользователь введет название документа с запрещенными символами (такими как /, *, :, <, >, |, ?), то на экране появится сообщение о том, что ввод названия документа оказался некорректным (см. рисунок 6.8).

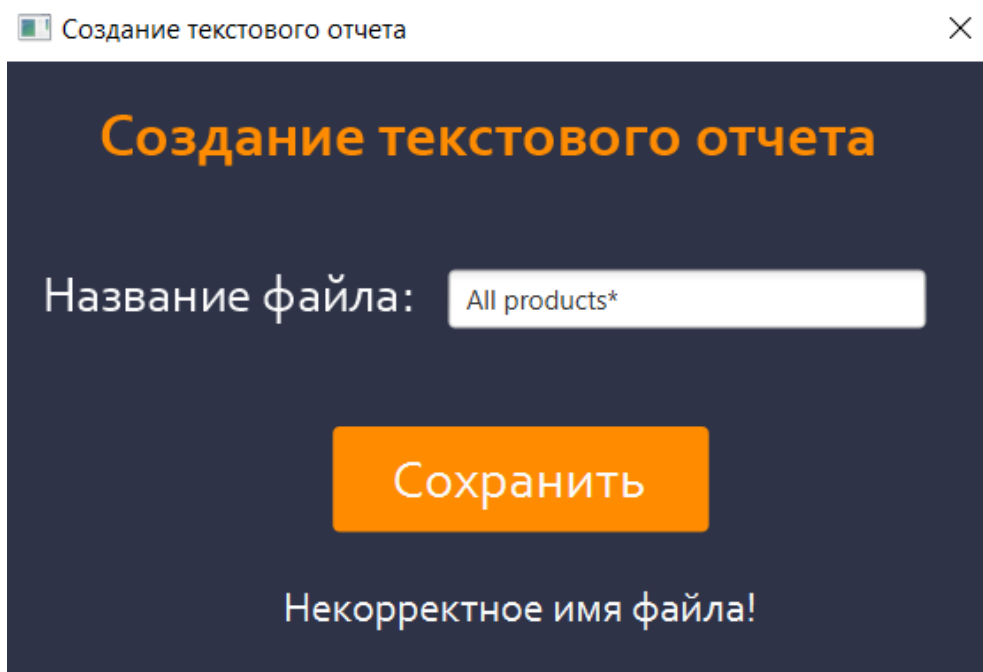


Рисунок 6.8 – Результат некорректного ввода названия документа

При поступлении новой продукции на склад ее необходимо перевести в нужное хранилище, а также добавить в базу. После добавления менеджером новых товаров, менеджер может добавить задание специалисту по доставке грузов. Для этого пользователю стоит нажать на кнопку «Добавить задачу».

Результат описанного действия изображен на рисунке 6.9:

добавления товара в базу менеджер оставляет задание для специалистов по доставке грузов. Общий список заданий находится у каждого специалиста в личном кабинете и выглядит это следующим образом (см. рисунок 6.10):

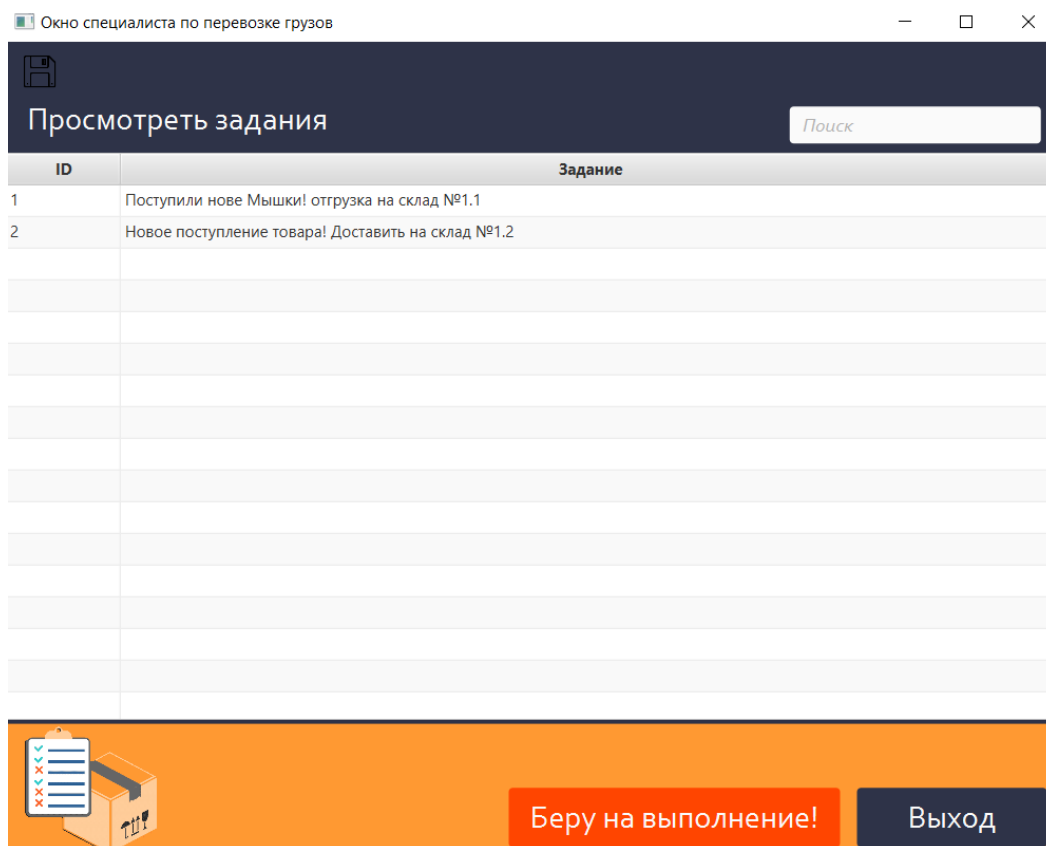


Рисунок 6.10 – Просмотр списка заданий

После того, как специалист нажимает на кнопку «Беру на выполнение!» выбранная задача удаляется из общего списка.

Также для удобства использования программы специалистом по доставке грузов был реализован поиск в верхней части окна, а также возможность сохранить доступные задания на данный момент в формате текстового отчета. После завершения работы специалистом он нажимает на кнопку «Выход».

6.3 Руководство пользования для администратора

После авторизации в качестве администратора появляется главное окно по управлению всеми пользователями системы.

Администратору доступны такие возможности как:

- просмотр пользователей системы;

- добавление пользователя;
- редактирование пользователя;
- удаление пользователя;
- поиск пользователя системы по имени, логину или паролю;
- создание текстового отчета со всеми пользователями системы.

Просмотр пользователей изображен на рисунке 6.11:

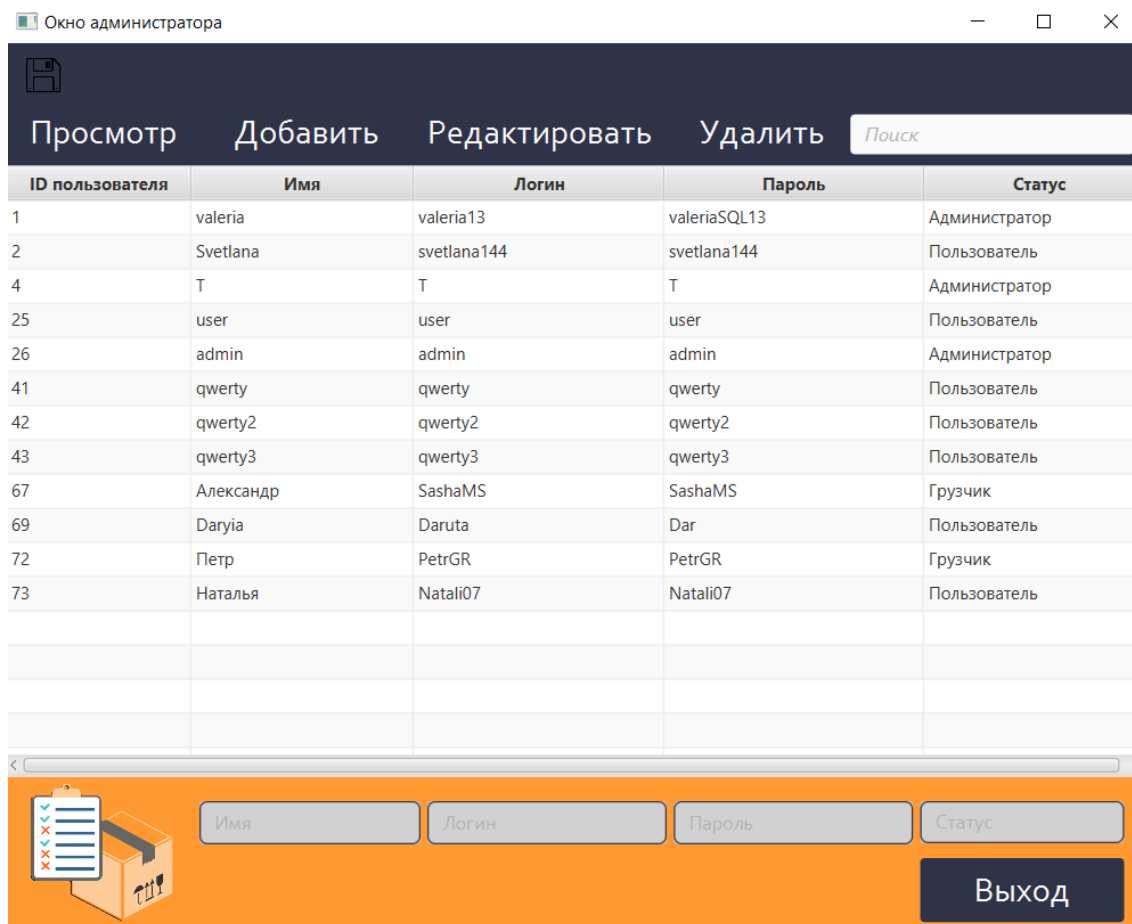


Рисунок 6.11 – Просмотр пользователей

Если администратор хочет добавить нового пользователя и присвоить статус, то следует в соответствующих полях внизу видимого окна ввести данные о новом пользователе и нажать на кнопку добавить. Процесс добавления нового пользователя в систему изображен на рисунке 6.12, а результат добавления изображен на рисунке 6.13.

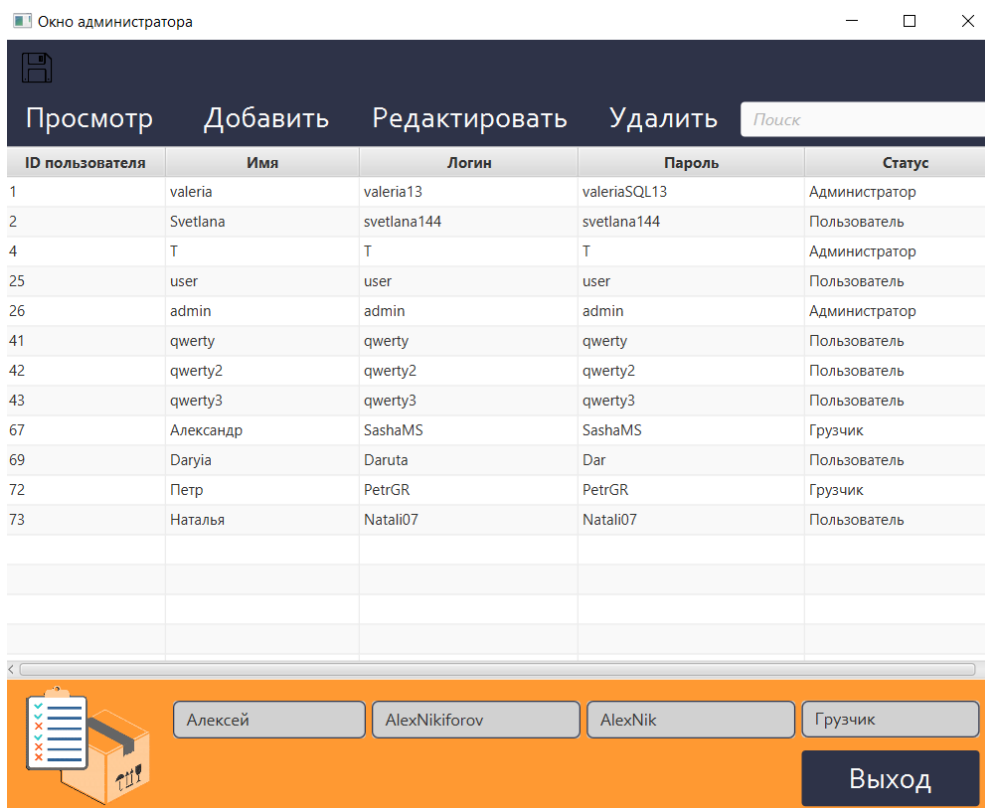


Рисунок 6.12 – Процесс добавления нового пользователя

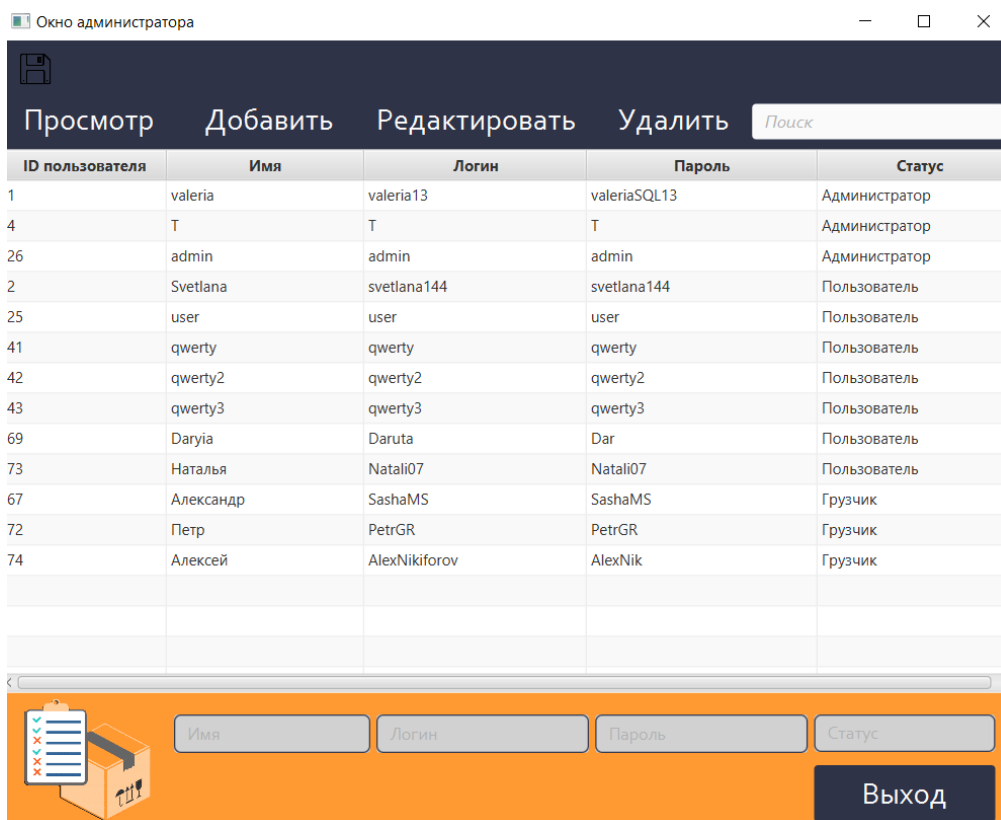


Рисунок 6.13 – Результат добавления нового пользователя в систему

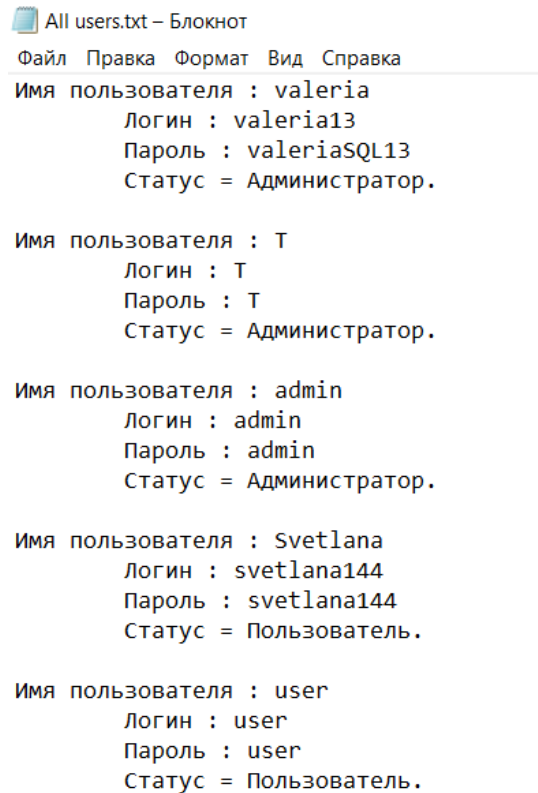
Зачастую на больших складах присутствует много рабочего персонала, поэтому, для удобства быстрого поиска нужного сотрудника в верхней части окна располагается поиск по ключевым словам. Результат поиска изображен на рисунке 6.14:

The screenshot shows a web application window titled 'Окно администратора'. At the top, there is a dark blue header bar with a save icon, four action buttons: 'Просмотр', 'Добавить', 'Редактировать', and 'Удалить', and a search input field containing the text 'svetlana'. Below the header is a table with five columns: 'ID пользователя', 'Имя', 'Логин', 'Пароль', and 'Статус'. The first row of the table contains the following data: ID 2, Name Svetlana, Login svetlana144, Password svetlana144, and Status Пользователь. Below the table is an orange footer bar containing a clipboard icon, four input fields labeled 'Имя', 'Логин', 'Пароль', and 'Статус', and a dark blue button labeled 'Выход'.

ID пользователя	Имя	Логин	Пароль	Статус
2	Svetlana	svetlana144	svetlana144	Пользователь

Рисунок 6.13 – Результат поиска пользователя системы

Для предоставления отчета о сотрудниках склада существует возможность создание текстового отчета. Результат создания текстового отчета изображен на рисунке 6.14:



```
All users.txt – Блокнот
Файл  Правка  Формат  Вид  Справка
Имя пользователя : valeria
    Логин : valeria13
    Пароль : valeriaSQL13
    Статус = Администратор.

Имя пользователя : T
    Логин : T
    Пароль : T
    Статус = Администратор.

Имя пользователя : admin
    Логин : admin
    Пароль : admin
    Статус = Администратор.

Имя пользователя : Svetlana
    Логин : svetlana144
    Пароль : svetlana144
    Статус = Пользователь.

Имя пользователя : user
    Логин : user
    Пароль : user
    Статус = Пользователь.
```

Рисунок 6.14 – Результат создания текстового отчета

После завершения работы над пользователями системы администратор по нажатию кнопки «Выход» завершит работу.

Таким образом, в данном курсовом проекте был реализован современный графический интерфейс с дополнительным функционалом для комфортной и быстрой работы с данными, а также предусмотрены отдельные графические окна для каждой из существующей роли. Графический интерфейс был спроектирован с помощью технологии для создания графических приложений JavaFX.

ЗАКЛЮЧЕНИЕ

Главной целью курсового проекта была разработка автоматизированной системы складского учета товарно-материальных ценностей. При решении поставленной задачи был выбран ряд соответствующих технологий и инструментов, которые позволили осуществить корректную работу данного приложения.

При функциональном моделировании было создано точное описание проектируемой системы, а также интерпретация полученного описания для определения оценочных знаний некоторых характеристик системы.

Информационная модель системы была создана для корректной работы с данными: хранение, обработка, удаление. При этом данная модель способна предоставлять доступ к информации, необходимой пользователю.

Проектирование UML-диаграмм позволило сформулировать основные требования к информационной системе, а также обеспечить объективность в выработке требований к проектированию системы. Данные модели описания требований к информационной системе преобразуются в систему моделей, описывающих концептуальный проект информационной системы. При этом сформировались модели архитектуры информационной системы, требования к программному обеспечению и информационному обеспечению.

В результате работы была разработана клиент-серверная GUI приложение, улучшающая процесс работы склада, где был предусмотрен индивидуальный функционал для каждой из существующей роли. Интерфейс данного программного продукта интуитивно понятен каждому пользователю, так как при проектировании графической части приложения были соблюдены общие правила расположения текстовой информации и элементов управления в окне.

В ходе написания программы по автоматизации складского учета товарно-материальных ценностей использовались основные средства языка программирования Java, позволяющие наиболее эффективно реализовать курсовой проект на базе соединения клиент-сервер и подойти к его разработке с точки зрения объектно-ориентированного программирования.

СПИСОК ИСТОЧНИКОВ

- [1] Автоматизация складского учета [Электронный ресурс]. – Режим доступа: <https://minsk.lcbit.ru/blog/programmy-dlya-skladskogo-ucheta/>.
- [2] Java: краткое руководство [Электронный ресурс]. – Режим доступа: <https://tproger.ru/translations/java-intro-for-beginners/>.
- [3] Принципы ООП на примере языка программирования Java [Электронный ресурс]. – Режим доступа: <https://topjava.ru/blog/oops-concepts-in-java>.
- [4] Введение в Java. Язык программирования Java [Электронный ресурс]. – Режим доступа: <https://metanit.com/java/tutorial/1.1.php>.
- [5] Введение в Java FX [Электронный ресурс]. – Режим доступа: <https://metanit.com/java/javafx/1.1.php>.
- [6] Что такое TCP/IP и как работает этот протокол [Электронный ресурс]. – Режим доступа: <https://timeweb.com/ru/community/articles/chto-takoe-tcp-ip>.
- [7] Что такое MySQL-сервер [Электронный ресурс]. – Режим доступа: <https://timeweb.com/ru/community/articles/chto-takoe-mysql-server>.
- [8] Паттерн одиночка на Java [Электронный ресурс]. – Режим доступа: <https://refactoring.guru/ru/design-patterns/singleton/java/example>.
- [9] Паттерн проектирования Command [Электронный ресурс]. – Режим доступа: <https://refactoring.guru/ru/design-patterns/command>.
- [10] Методология IDEF0 [Электронный ресурс]. – Режим доступа: <https://itteach.ru/bpwin/metodologiya-idef0>.
- [11] Черемных, С. Моделирование и анализ систем. IDEF-технологии / Черемных С.В., Семенов И.О., Ручкин В.С, Москва: Россия. – 2006. – 192 с.
- [12] Описание системы [Электронный ресурс]. – Режим доступа: https://vuzlit.ru/983027/opisanie_sistemy.
- [13] Батин, Н.В. Проектирование баз данных / Батин Н.В., Минск: Беларусь -2007. – 56 с.

ПРИЛОЖЕНИЕ А

(обязательное)

Листинг скрипта генерации базы данных

Файл WarehouseDB.sql

```
CREATE DATABASE `warehouse` /*!40100 DEFAULT CHARACTER SET
utf8mb4 COLLATE utf8mb4_0900_ai_ci */ /*!80016 DEFAULT
ENCRYPTION='N' */;
CREATE TABLE `manufacturers` (
  `idManufacturer` int NOT NULL AUTO_INCREMENT,
  `manufacturer` varchar(45) NOT NULL,
  PRIMARY KEY (`idManufacturer`)
) ENGINE=InnoDB AUTO_INCREMENT=8 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `models` (
  `idModel` int NOT NULL AUTO_INCREMENT,
  `model` varchar(45) NOT NULL,
  `price` double NOT NULL,
  PRIMARY KEY (`idModel`)
) ENGINE=InnoDB AUTO_INCREMENT=33 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `products` (
  `idproduct` int NOT NULL AUTO_INCREMENT,
  `manufacturer_product` int NOT NULL,
  `type_product` int NOT NULL,
  `model_product` int NOT NULL,
  `quantity_product` int NOT NULL,
  `storage_product` int NOT NULL,
  PRIMARY KEY (`idproduct`),
  KEY `products_models_FK` (`model_product`),
  KEY `products_manufacturers_FK` (`manufacturer_product`),
  KEY `products_types_FK` (`type_product`),
  KEY `products_storages_FK` (`storage_product`),
  CONSTRAINT `products_manufacturers_FK` FOREIGN KEY
(`manufacturer_product`) REFERENCES `manufacturers` (`idManufacturer`) ON
DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `products_models_FK` FOREIGN KEY (`model_product`)
REFERENCES `models` (`idModel`) ON DELETE CASCADE ON UPDATE
CASCADE,
```

```

        CONSTRAINT `products_storages_FK` FOREIGN KEY (`storage_product`)
REFERENCES `storages` (`idStorage`) ON DELETE CASCADE ON UPDATE
CASCADE,
        CONSTRAINT `products_types_FK` FOREIGN KEY (`type_product`)
REFERENCES `types` (`idType`) ON DELETE CASCADE ON UPDATE
CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=32 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

```

```

CREATE TABLE `statususer` (
    `idstatus` int NOT NULL,
    `statusName` varchar(45) NOT NULL,
    PRIMARY KEY (`idstatus`),
    UNIQUE KEY `idstatus_UNIQUE` (`idstatus`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

```

```

CREATE TABLE `storages` (
    `idStorage` int NOT NULL AUTO_INCREMENT,
    `storage` varchar(45) NOT NULL,
    `warehouseStorage` int NOT NULL,
    PRIMARY KEY (`idStorage`),
    KEY `storages_warehouses_FK` (`warehouseStorage`),
    CONSTRAINT `storages_warehouses_FK` FOREIGN KEY
(`warehouseStorage`) REFERENCES `warehouses` (`idWarehouse`) ON DELETE
CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

```

```

CREATE TABLE `tasks` (
    `idTask` int NOT NULL AUTO_INCREMENT,
    `task` varchar(150) NOT NULL,
    PRIMARY KEY (`idTask`)
) ENGINE=InnoDB AUTO_INCREMENT=17 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

```

```

CREATE TABLE `types` (
    `idType` int NOT NULL AUTO_INCREMENT,
    `type` varchar(45) NOT NULL,
    PRIMARY KEY (`idType`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

```

```

CREATE TABLE `users` (

```

```

`idusers` int NOT NULL AUTO_INCREMENT,
`name` varchar(45) NOT NULL,
`login` varchar(45) NOT NULL,
`password` varchar(45) NOT NULL,
`statusUser` int NOT NULL,
PRIMARY KEY (`idusers`),
KEY `users_statususer_FK` (`statusUser`),
CONSTRAINT `users_statususer_FK` FOREIGN KEY (`statusUser`)
REFERENCES `statususer` (`idstatus`) ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=75 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

```

```

CREATE TABLE `warehouses` (
  `idWarehouse` int NOT NULL AUTO_INCREMENT,
  `warehouse` varchar(45) NOT NULL,
  PRIMARY KEY (`idWarehouse`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

```

ПРИЛОЖЕНИЕ Б

(обязательное)

Листинг основных элементов

Файл ConnectionTCP.java

```
package mwarehouse.warehouse.common;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;

public class ConnectionTCP {
    private final Socket socket;
    private final ObjectInputStream inputStream;
    private final ObjectOutputStream outputStream;

    public ConnectionTCP(Socket socket) {
        this.socket = socket;
        try {
            this.outputStream = new
ObjectOutputStream(socket.getOutputStream());

            this.inputStream = new ObjectInputStream(socket.getInputStream());
        } catch (IOException e) {
            throw new RuntimeException("can't initialise", e);
        }
    }

    public void writeObject(Object object) {
        try {
            outputStream.writeObject(object);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }

    public Object readObject() {
        try {
            return inputStream.readObject();
        } catch (IOException | ClassNotFoundException e) {
            throw
                new RuntimeException(e);
        }
    }

    public void close() {
        try {
            inputStream.close();
            outputStream.close();
            socket.close();
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
```

Файл RequestHandler.java

```
package mwarehouse.warehouse.database;
import mwarehouse.warehouse.entity.Product;
import mwarehouse.warehouse.database.DatabaseHandler;
import mwarehouse.warehouse.common.ConnectionTCP;
import mwarehouse.warehouse.entity.Command;
import mwarehouse.warehouse.entity.Task;
import mwarehouse.warehouse.entity.User;
import mwarehouse.warehouse.singleton.ProgramLogger;
import java.io.IOException;
import java.net.Socket;
import java.util.List;

public class RequestHandler implements Runnable {
    private final ConnectionTCP connectionTCP; //создали объекта КЛАССА
    ConnectionTCP

    public RequestHandler(Socket socket) {
        connectionTCP = new ConnectionTCP(socket);
    } //connectionTCP - шнур
    @Override
    public void run() {
        DatabaseHandler userRepository = new DatabaseHandler();
        DatabaseHandler productRepository = new DatabaseHandler();

        while (true) {
            Command command = (Command) connectionTCP.readObject();
            System.out.println(command);
            switch (command) {
                case CREATE: {
                    User userr = (User) connectionTCP.readObject();
                    try {
                        userRepository.signUpUser(userr);
                        ProgramLogger.getProgramLogger().addLogInfo("Успешно!
Пользователь добавлен в БД (Table: users)!");
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
                break;
                case CREATE1: {
                    Product product = (Product) connectionTCP.readObject();
                    try {
                        userRepository.signUpProduct(product);
                        ProgramLogger.getProgramLogger().addLogInfo("Успешно!
Товар добавлен в БД (Table: products)!");
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
                break;
                case CREATETASK: {
                    Task task = (Task) connectionTCP.readObject();
                    try {
                        productRepository.addNewTask(task);
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
            }
        }
    }
}
```

```

    }
    break;
    case READ: {
        List<User> userr = null;
        try {
            userr = userRepository.getAllUsers();
            ProgramLogger.getProgramLogger().addLogInfo("Успешно!
Данные из БД о пользователях просмотрены!");
        } catch (IOException e) {
            e.printStackTrace();
        }
        connectionTCP.writeObject(userr);
    }
    break;
    case READ1: {
        List<Product> products = null;
        try {
            products = productRepository.getAllProducts();
            ProgramLogger.getProgramLogger().addLogInfo("Успешно!
Данные из БД о товарах просмотрены!");
        } catch (IOException e) {
            e.printStackTrace();
        }
        connectionTCP.writeObject(products); // с помощью
writeObject отправляем клиенту массив юзеров
    }
    break;
    case READTASKS: {
        List<Task> tasks = null;
        tasks = productRepository.gettAllTask();
        connectionTCP.writeObject(tasks);
    }
    break;
    case READMANUFACTURER: {
        List<String> manufacturers = null;
        manufacturers = productRepository.getAllManufacturer();
        connectionTCP.writeObject(manufacturers);
    }
    break;
    case READTYPE: {
        List<String> types = null;
        types = productRepository.getAllType();
        connectionTCP.writeObject(types);
    }
    break;
    case READSTORAGE: {
        List<String> storages = null;
        storages = productRepository.getAllStorage();
        connectionTCP.writeObject(storages);
    }
    break;
    case READMODEL: {
        List<String> models = null;
        models = productRepository.getAllModel();
        connectionTCP.writeObject(models);
    }
}

```

```

break;
case UPDATE: {
    User userr = (User) connectionTCP.readObject();

    try {
        userRepository.updateUser(userr);
        ProgramLogger.getProgramLogger().addLogInfo("Успешно!
Данные пользователя отредактированы!");
    } catch (IOException e) {
        e.printStackTrace();
    }
}
break;
case UPDATE1: {
    Product product = (Product) connectionTCP.readObject();

    try {
        productRepository.updateProduct(product);
        ProgramLogger.getProgramLogger().addLogInfo("Успешно!
Данные пользователя отредактированы!");
    } catch (IOException e) {
        e.printStackTrace();
    }
}
break;
case DELETE: {
    Integer id = (Integer) connectionTCP.readObject();
    try {
        userRepository.deleteUserByID(id);
        ProgramLogger.getProgramLogger().addLogInfo("Успешно!
Пользователь удален!");
    } catch (IOException e) {
        e.printStackTrace();
    }
}
break;
case DELETE1: {
    Integer id = (Integer) connectionTCP.readObject();
    try {
        productRepository.deleteProductByID(id);
        ProgramLogger.getProgramLogger().addLogInfo("Успешно!
Товар удален!");
    } catch (IOException e) {
        e.printStackTrace();
    }
}
break;
case DELETETASK: {
    Integer id = (Integer) connectionTCP.readObject();
    try {
        productRepository.deleteTaskByID(id);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
break;
case EXIT: {
    connectionTCP.close();
    return;
} } } } }

```

ПРИЛОЖЕНИЕ В

(обязательное)

Листинг алгоритма, реализующий бизнес-логику

Файл AddProductController.java

```
package mwarehouse.warehouse;

import java.io.IOException;
import java.net.URL;
import java.util.List;
import java.util.ResourceBundle;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.stage.Stage;
import mwarehouse.warehouse.common.ConnectionTCP;
import mwarehouse.warehouse.entity.Command;

public class AddProductController {
    ConnectionTCP connectionTCP;

    @FXML
    private ResourceBundle resources;

    @FXML
    private URL location;

    @FXML
    private Button add_add;

    @FXML
    private ComboBox manufacturer_ComboBox;

    @FXML
    private ComboBox type_ComboBox;

    @FXML
    private ComboBox storage_ComboBox;

    @FXML
    private TextField model_add;

    @FXML
    private TextField quantity_add;

    @FXML
```



```

private TextField price_add;

@FXML
private TextField qw_field;

String manufacturer, type, storage;
String model, quantity, price;

@FXML
void SelectManuf(ActionEvent event) {
    manufacturer =
manufacturer_ComboBox.getSelectionModel().getSelectedItem().toString();
}

@FXML
void SelectType(ActionEvent event) {
    type = type_ComboBox.getSelectionModel().getSelectedItem().toString();
}

@FXML
void SelectStorage(ActionEvent event) {
    storage = storage_ComboBox.getSelectionModel().getSelectedItem().toString();
}

@FXML
void Add(ActionEvent event) throws IOException {

    model = model_add.getText();

    quantity = quantity_add.getText();

    price = price_add.getText();

    if(manufacturer.isEmpty() || type.isEmpty() || model.isEmpty() ||
quantity.isEmpty() || price.isEmpty() || storage.isEmpty()){
        qw_field.setText("Присутствуют пустые поля!");
        qw_field.setStyle("-fx-background-color: #2E3348; -fx-text-fill:
#fafafa;");
    }
    else {
        System.out.println(manufacturer + " " + type + " " + model + " " +
quantity + " " + price + " " + storage);

        MainUserController mainUserController = new MainUserController();
        mainUserController.displayNameAdd(manufacturer,
            type,
            model,
            quantity,
            price,
            storage);
        qw_field.setText("Товар успешно добавлен!");
        qw_field.setStyle("-fx-background-color: #2E3348; -fx-text-fill:
#fafafa;");
    }

}

```

```

@FXML
void initialize() throws IOException {

    manufacturer="";
    type="";
    storage = "";

    connectionTCP = ConnectionTCP.getInstance();

    connectionTCP.writeObject(Command.READMANUFACTURER);
    List<String> allProducts = (List<String>) connectionTCP.readObject();
    ObservableList allPr = FXCollections.observableArrayList(allProducts);
    manufacturer_ComboBox.setItems(allPr);

    connectionTCP.writeObject(Command.READTYPE);
    List<String> allTypes = (List<String>) connectionTCP.readObject();
    ObservableList allTy = FXCollections.observableArrayList(allTypes);
    type_ComboBox.setItems(allTy);

    connectionTCP.writeObject(Command.READSTORAGE);
    List<String> allStorages = (List<String>) connectionTCP.readObject();
    ObservableList allSt = FXCollections.observableArrayList(allStorages);
    storage_ComboBox.setItems(allSt);

}
}

```

Файл MainUserController.java

```

public class MainUserController {
    private ConnectionTCP connectionTCP;
    private final ObservableList<ProductProperty> tableProductProperties =
FXCollections.observableArrayList();// вызовет конструктор 0

@FXML
    public void ShowDialogForAddTask (ActionEvent event){
        try {
            Stage stage = new Stage();
            Parent root =
FXMLLoader.load(getClass().getResource("AddTaskByUser.fxml"));
            stage.setTitle("Добавление задачи");
            stage.setResizable(false);
            stage.setScene(new Scene(root));
            stage.initModality(Modality.WINDOW_MODAL);
            stage.initOwner( ((Node)event.getSource()).getScene().getWindow() );
            stage.show();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void displayNameAdd(String manufacturer, String type, String model, String
quantity, String price, String storage){
        //получение данных о товаре с модального окна (Добавление товара) для
дальнейшего добавления в таблицу
        String manufacturer_product = manufacturer;
        String type_product = type;
        String model_product = model;
        Integer quantity_product = Integer.parseInt(quantity);
    }
}

```

```

        Double price_product = Double.parseDouble(price);
        String storage_product = storage;

        System.out.println(manufacturer_product + " " + type_product + " " +
model_product + " " + quantity_product + " " + price_product + " " +
storage_product);

        // добавление товара в БД
        try {
            connectionTCP = ConnectionTCP.getInstance();
        } catch (IOException e) {
            // System.out.println("Не нашел клиента! :(");
            e.printStackTrace();
            System.exit(-1);
        }
        try {
            Product product = new Product(manufacturer_product,
                type_product,
                model_product,
                quantity_product,
                price_product,
                storage_product);
            System.out.println("product = " + product);
            System.out.println("product manufacturer_product = " +
product.getManufacturer());

            connectionTCP.writeObject(Command.CREATE1);
            connectionTCP.writeObject(product);

        } catch (RuntimeException e) {
            e.printStackTrace();
        }
    }
}

```

ПРИЛОЖЕНИЕ Г **(обязательное)** **Графический материал**

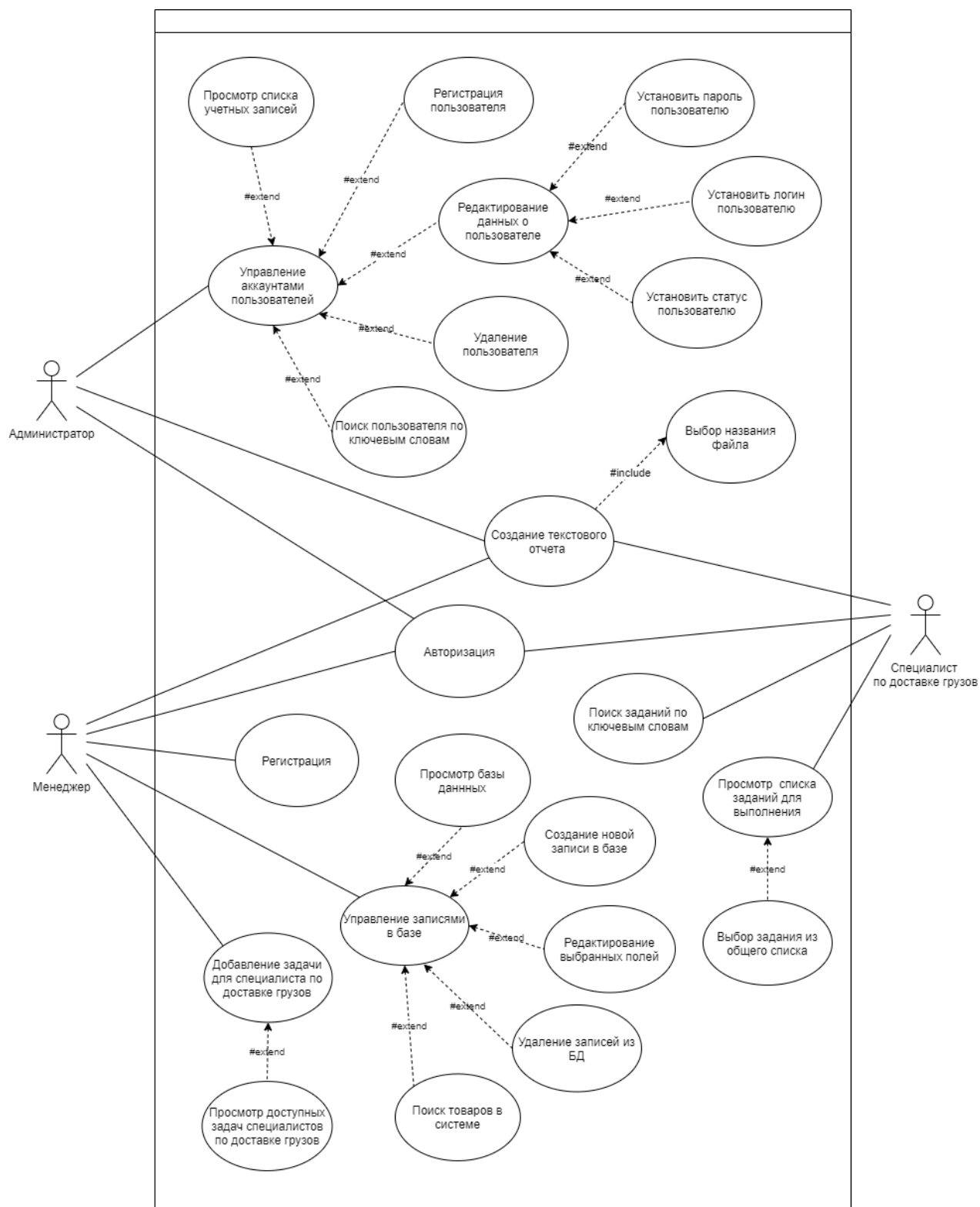


Рисунок Г.1 – Диаграмма вариантов использования

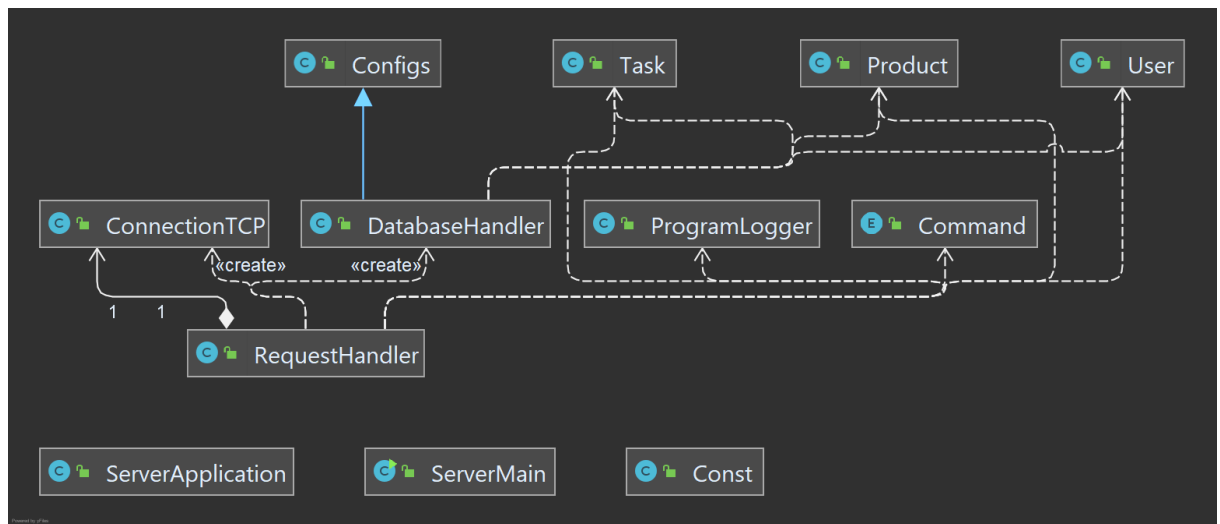


Рисунок Г.2 – Диаграмма классов серверной части

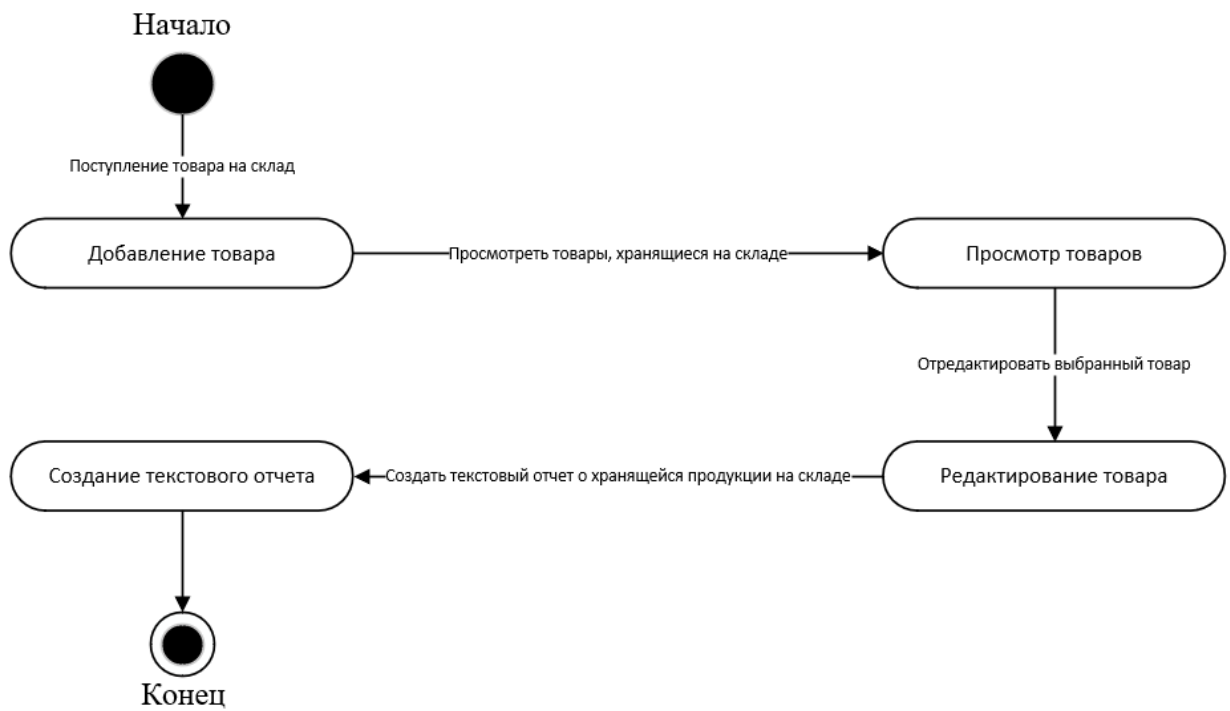


Рисунок Г.3 – Диаграмма состояния

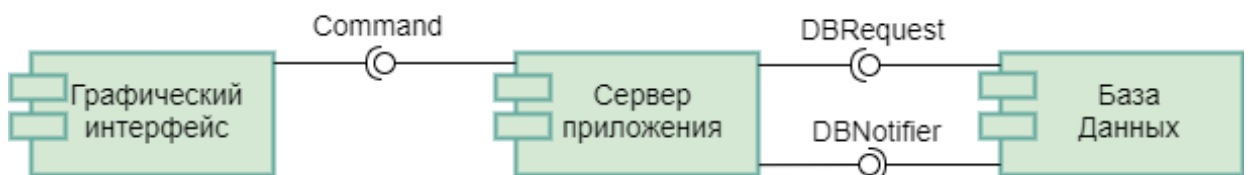


Рисунок Г.4 – Диаграмма компонентов

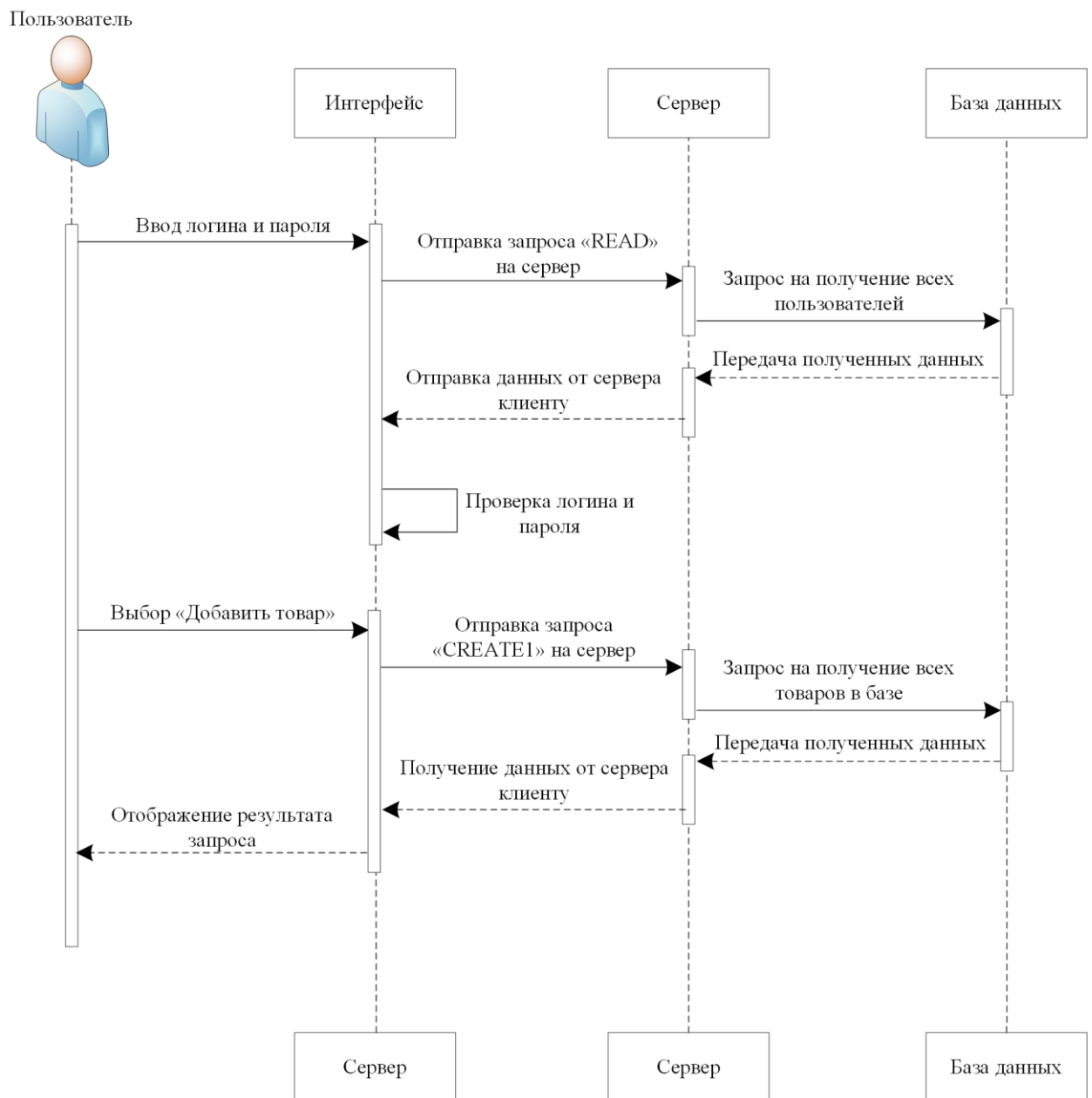


Рисунок Г.5 – Диаграмма последовательности

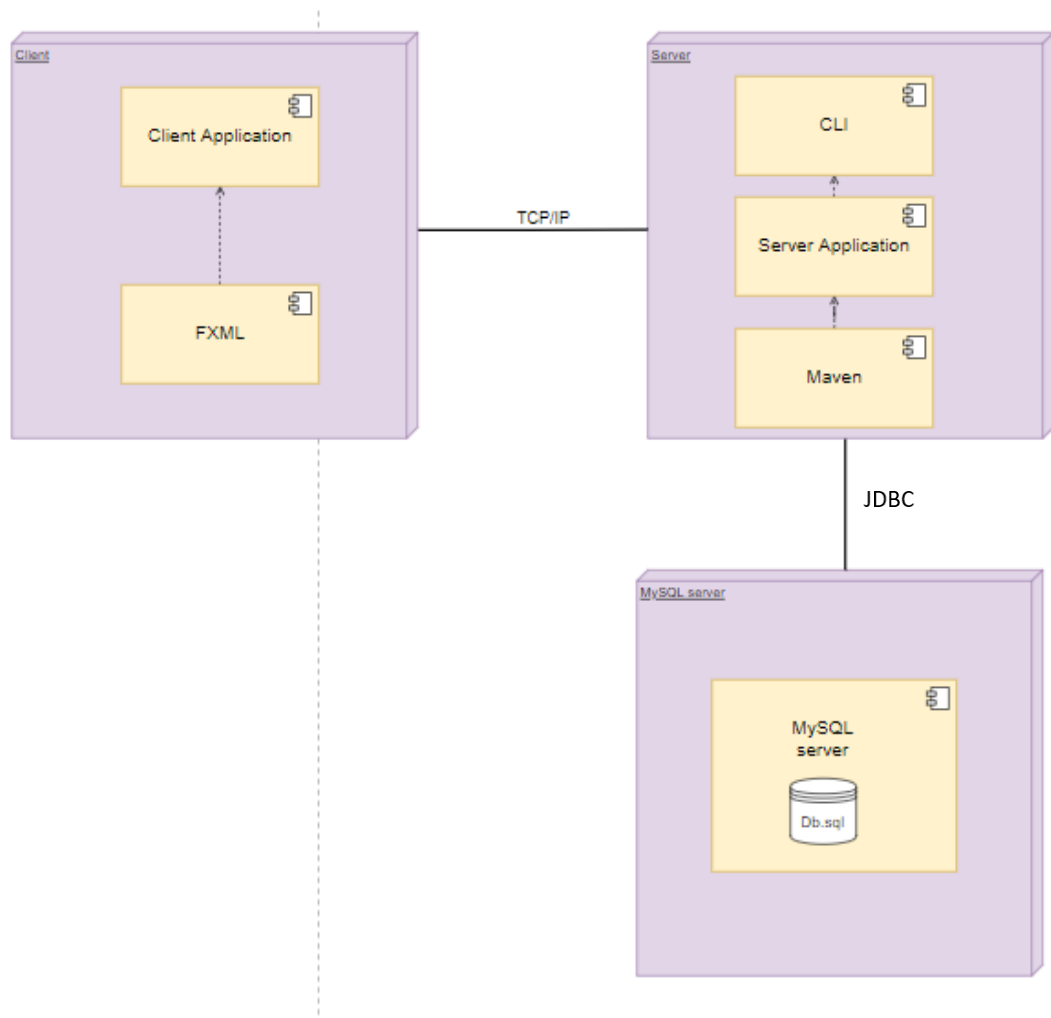


Рисунок Г.6 – Диаграмма развертывания

ПРИЛОЖЕНИЕ Д

(обязательное)

Проверка на антиплагиат



Отчет о проверке на заимствования №1



Автор: kamei2002@mail.ru / ID: 9160859

Проверяющий: (kamei2002@mail.ru / ID: 9160859)

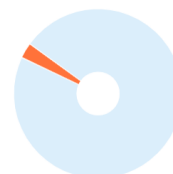
Отчет предоставлен сервисом «Антиплагиат» - users.antiplagiat.ru

ИНФОРМАЦИЯ О ДОКУМЕНТЕ

№ документа: 5
Начало загрузки: 15.12.2021 06:33:29
Длительность загрузки: 00:00:01
Имя исходного файла:
Пояснительная_записка_Warehouse.pdf
Название документа:
Пояснительная_записка_Warehouse
Размер текста: 69 кБ
Символов в тексте: 70414
Слов в тексте: 8465
Число предложений: 601

ИНФОРМАЦИЯ ОБ ОТЧЕТЕ

Начало проверки: 15.12.2021 06:33:30
Длительность проверки: 00:00:10
Комментарии: не указано
Модули поиска: Интернет Free



ЗАИМСТВОВАНИЯ

2,61%

САМОЦИТИРОВАНИЯ

0%

ЦИТИРОВАНИЯ

0%

ОРИГИНАЛЬНОСТЬ

97,39%

Рисунок Б.1 – Проверка на антиплагиат

ПРИЛОЖЕНИЕ Е
(обязательное)
Ведомость курсового проекта