МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Вычислительные машины, комплексы, системы и сети

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

на тему

Информационная система с архитектурой клиент-сервер

Студент (подпись, дата) 03.06.20 Мухаметов Т.Ш.

Руководитель (подпись, дата) 03.06.20 Карамышева Н.С,

Нормоконтролёр (подпись, дата) Кучин А.В.

Работа допущена к защите (протокол зас. кафедры от 03.06.2020 г. № 12)

Заведующий кафедрой (подпись) М.А. Митрохин

Работа защищена с отметкой хорошо (протокол заседания ГЭК от 17.06.20 № 11)

Секретарь ГЭК (подпись) Е.И. Гурин

Пенза 2020

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
_____ М.А. Митрохин
«30» апреля 2020г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

1. Студент Мухаметову Тимуру Шамильевичу группы 16ВВ1 факультета вычислительной техники.

2.Тема работы «Информационная система с архитектурой клиент-сервер».

Тема утверждена приказом ПГУ №__0335а/о___ от "_30__"_____апреля_____2020___г.

3. Руководитель бакалаврской работы к.т.н., доцент каф ВТ ПГУ Карамышева Н.С.

4. Задание на работу (назначение разработки, исходные данные и т.п.).
Назначение разработки- информационная система с архитектурой клиент-сервер

5. Перечень подлежащих разработке вопросов.
Требуется произвести анализ предметной области, определить архитектуру приложения, выбрать языки программирования для клиентского и серверного приложения, выбрать способ взаимодействия между клиентом и сервером, разработать протокол взаимодействия.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области	4	30.04.20-05.05.20	
2.	Выбор архитектуры приложения, языков программирования, способа взаимодействия	12	05.05.20-10.05.20	
3.	Разработка протокола взаимодействия	5	10.05.20-14.05.20	
4.	Разработка серверной части приложения	5	14.05.20-19.05.20	
5.	Разработка клиентской части приложения	8	19.05.20-23.05.20	
6.	Тестирование и отладка	6	24.05.20-29.05.20	
7.	Оформления выпускной работы	10	29.05.20-03.06.20	

Дата выдачи задания "30 "апреля 2020г.

Руководитель бакалаврской работы 30.04.20 Карамышева Н.С.
(подпись, дата)

Задание к исполнению принял студент 30.04.20 Мухаметов Т.Ш.
(подпись, дата)

Работу к защите допустить
Декан факультета _____ Фионова Л.Р.
(подпись, дата)

РЕФЕРАТ

Пояснительная записка содержит 79 листов, 18 рисунков, 5 таблиц, 16 источников, 4 приложения.

АРХИТЕКТУРА КЛИЕНТ СЕРВЕР, NETBEANS IDE, MYSQL COMMAND LINE, SQL, MAVEN, JAVA, SWING, SERVER SOCKET, JDBC.

Цель работы – разработка клиент-серверного приложения для организации работы персонала в рамках торговой сети, занимающейся непосредственной продажей товаров. Система должна обеспечивать взаимосвязь персонала торговой сети с базой данной на серверной стороне, позволять осуществление продажи определённого товара. Также система должна реализовывать добавление и удаление нового персонала или товара в базу данных.

Результатом разработки является проект, удовлетворяющий поставленной задаче. А именно, разработано, реализовано и протестировано приложение для организации работы персонала в рамках торговой сети.

Разработка производилась с использованием большого стека технологий, к которым относятся: язык программирования Java, язык структурированных запросов SQL, среда разработки NetBeans. Для хранения данных выбрана СУБД MySQL, а для её администрирования применялся инструмент MySQL Command Line.

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ ПРИНЯТЫХ СОКРАЩЕНИЙ 6
ВВЕДЕНИЕ 7
1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 9
1.1 Анализ архитектуры системы работы персонала торговой сети 9
1.2 Анализ функциональных блоков системы работы персонала торговой сети 10
2 ТРЕБОВАНИЯ К РАЗРАБОТКЕ ПРИЛОЖЕНИЯ ДЛЯ ОТСЛЕЖИВАНИЯ СИСТЕМЫ РАБОТЫ ПЕРСОНАЛА В ТОРГОВОЙ СЕТИ 13
3 ВЫБОР ТЕХНОЛОГИЙ, ЯЗЫКОВ И СРЕДСТВ ДЛЯ РАЗРАБОТКИ ПРИЛОЖЕНИЯ 15
3.1 Программные средства, используемые при разработке системы 15
3.2 NetBeans IDE 16
3.3 Java 17
3.4 SQL 18
3.5 MySQL Command Line 19
3.6 Maven 19
3.7 SWING 20
3.8 Server Socket 21
3.9 JDBC 23
3.10 Взаимодействие рассмотренных средств 24
4 РЕАЛИЗАЦИЯ 26
4.1 Алгоритм работы персонала 26
4.2 Описание структуры и таблиц БД 28
4.3 Основные функции доступные пользователям 29
4.4 Схема оформления заказа 31
5 МЕТОДЫ ИСПЫТАНИЙ 33
5.1 Добавление нового персонала 33
5.2 Удаление персонала 35
5.3 Добавление товара 37
5.4 Удаление товара 39
5.5 Авторизация 40
5.6 Продажа товара 42
ЗАКЛЮЧЕНИЕ 45
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 46
ПРИЛОЖЕНИЕ А 47
ПРИЛОЖЕНИЕ Б 51
ПРИЛОЖЕНИЕ В 76

ПЕРЕЧЕНЬ ПРИНЯТЫХ СОКРАЩЕНИЙ

БД - база данных

СУБД - система управления базой данных

ORM - (англ.Object-relational Mapping) - объектно-реляционное отображение

ПО - программное обеспечение

ОС - операционная система

ЯП - язык программирования

GUI - (англ. Graphical User Interface) - графический интерфейс пользователя

SQL - (англ. Structured Query Language) - язык структурированных запросов

URL - (англ. Uniform Resource Locator) - единый указатель ресурса

ВВЕДЕНИЕ

Сегодня информацию рассматривают как один из основных ресурсов развития общества, а информационные системы и технологии как средство повышения производительности и эффективности работы людей.

Создание современных электронных вычислительных машин позволило автоматизировать обработку данных во многих сферах человеческой деятельности, в том числе торговле. Без современных систем обработки данных трудно представить сегодня передовые производственные технологии, управление экономикой на всех ее уровнях, научные исследования, образование, издательское дело, функционирование средств массовой информации, проведение крупных спортивных состязаний. Значительно расширило сферу применения систем обработки данных появление персональных компьютеров.

Одним из наиболее распространенных классов систем обработки данных являются информационные системы.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

При разработке программного продукта необходимо провести анализ существующих вариантов решения задач. В данном разделе рассматриваются общие теоретические вопросы, связанные с проектированием и реализацией клиент-серверного приложения для организации работы персонала в рамках торговой сети.

Для того чтобы создать программный продукт с заданными параметрами, следует рассмотреть его архитектуру и функциональные блоки, без которых приложение не сможет корректно работать.

1.1 Анализ архитектуры системы работы персонала торговой сети

Архитектура системы работы персонала является трехзвенной архитектурой, которая состоит из трех основных частей:

1. Клиентская часть
2. Серверная часть
3. База данных

Рассмотрим конкретнее, что представляет каждая часть, какова роль каждой в проекте.

Клиентская часть - непосредственно та часть, с которой взаимодействует пользователь. Она представляет собой графический интерфейс, компоненты. Взаимодействие пользователя и программы происходит именно через клиентскую часть.

Серверная часть - программа, которая принимает и обрабатывает запросы от пользователей. Серверную часть можно реализовать с помощью различных технологий и языков программирования. Процесс взаимодействия с клиентской частью состоит из получения запроса от клиента, обработка его

на сервере и отправка результата данного запроса обратно на клиентскую сторону.

База данных (БД) - совокупность взаимосвязанных, хранящихся вместе данных при наличии такой минимальной избыточности, которая допускает их использование оптимальным образом для одного или нескольких приложений.

Система управления базами данных (СУБД) - программное обеспечение, необходимое для создания баз данных, поддержания их в актуальном состоянии и организации поиска в них необходимой информации. При принятии запроса сервером на взаимодействие с какими- либо данными из БД, он в свою очередь посылает SQL-запрос СУБД, где происходит соответствующая операция по добавлению, модификации, получению или удалению данных. После принятия сервером ответа от БД, он посылает результат запроса обратно на клиентскую часть.

1.2 Анализ функциональных блоков системы работы персонала торговой сети

Ни одна система продажи товаров не может обойтись без некоторых функциональных блоков, то есть они являются обязательными для любой программы. К таким важным частям относятся:

- блок авторизации или регистрации персонала;
- блок удаления персонала;
- блок по добавлению товара;
- блок по удалению товара;
- система поиска товаров;
- каталог общего списка товаров;
- блок продажи товара.

Функционал блока регистрации персонала служит для того, чтобы создать личную учетную запись в базе данных, с целью получения доступа к полному функционалу системы. Учетная запись хранит информацию о главных

характеристиках клиента, таких как: фамилия, имя, отчество, раздел к котором он работает и др. Данная информация необходима для того, чтобы осуществить идентификацию пользователя для обеспечения доступа к функции по продаже товаров. Данная информация хранится на стороне сервера.

Функционал удаления персонала является неотъемлемой частью по администрированию персоналом в торговой сети. Для удаления представителя персонала из системы администратору сервера предоставляется список ранее зарегистрированных участников системы, которых он в свою очередь имеет возможность удалить из системы и БД развёртываемой системы. После удаления записи о персонале, он также удаляется из БД на сервере.

Блок авторизации необходим для входа клиента в систему. Данный блок обеспечивает ввод клиентом необходимого перечня данных для его идентификации на серверной стороне и предоставление клиенту необходимых полномочий для работы в системе.

Блок по добавлению товара позволяет администратору сервера добавить новую запись о товаре или сделать пополнение уже ранее добавленного товара в БД. Для выполнения данной операции представителю персонала торговой сети предоставляется возможность указать весь перечень по описанию добавляемого товара. Запись в БД о новом товаре должна содержать в себе информацию о его названии, разделе в котором он располагается и количестве товара на складе. Добавление происходит на серверной стороне.

Функционал удаления товара - это важная и неотъемлемая часть по управлению товарооборотом в торговой сети. Для удаления товара серверному администратору предоставляется широкий список товаров, которые он может удалить из БД. После удаления записи о товаре, он также удаляется из БД на сервере. Изъятие товара из оборота происходит при помощи команды администратора сервера, после проделанной операции на

серверной стороне организуется подключение к БД, поиск товара с выбранными характеристиками и его последующее удаление из БД.

Система поиска товаров - система, которая служит для удобного и быстрого поиска товара по названию. Она также позволяет пользователю убедиться присутствует ли товар на складе текущей торговой сети. Является важной составной частью операции по продаже товара и располагается на стороне клиента.

Каталог товаров - список товаров, содержащий их полное описание. Он служит пользователю для удобного просмотра товаров, которые предоставляет данная торговая сеть, а также позволяет осуществить выбор отображаемых товаров для последующей продажи.

Функционал блока оформления продажи товара служит для осуществления непосредственной продажи товара. Выбор товара происходит из предоставляемого списка товаров. Список разворачивается на клиентской стороне. Перед продажей требуется выбор товара, также требуется указать количество товара для продажи и только после этого на серверную сторону от лица авторизировавшегося клиента запустится команда о продаже выделенного товара.

2 ТРЕБОВАНИЯ К РАЗРАБОТКЕ ПРИЛОЖЕНИЯ ДЛЯ СИСТЕМЫ РАБОТЫ ПЕРСОНАЛА В ТОРГОВОЙ СЕТИ

Система работы персонала в торговой сети должна выполнять список основных функций для осуществления процесса торговли, к ним относятся:

- со стороны сервера:
  - регистрация нового представителя персонала в сети, добавление перечня необходимой информации о нём и предоставление данных для входа в сеть.
  - добавление нового вида товаров в систему и в БД с заполнением всех необходимых полей.
  - отображение перечня персонала, который находится в сети в данный момент времени.
  - обработка поступающих команд о продаже товара от авторизированных пользователей системы.
- со стороны клиента:
  - функционал авторизации, для идентификации представителя персонала в сети и предоставление ему всего перечня прав по эксплуатации системы товарооборота.
  - осуществление функции просмотра всего списка товаров или списка с фильтром в качестве раздела, в котором осуществляет деятельность представитель персонала.
  - поиск товара в системе и последующая продажа найденного товара.
  - возможность выбора и приобретение товара, а также имеется возможность указать количество товара для продажи в текущей операции.
  - просмотр контактной информации о фирме.

При запуске клиентского приложения пользователю предоставляется упрощённый интерфейс. После выполнения операции авторизации, где пользователь указывает свои персональные данные для входа и его идентификации, он может войти в систему как зарегистрированный пользователь и получить статус авторизированный. С этого момента продавец получает все права по изучению списка товара, представленного на вкладках клиентского приложения, а также по использованию всех функций, представленных для персонала.

Работник магазина после авторизации имеет возможность работать над содержимым торговой сети магазина. Подразумевается, что работников в развёртываемой торговой сети будет несколько, так как в обслуживании такого ресурса необходимо много времени и сил.

Клиент-серверное приложение позволяет улучшить работу магазина за счет упрощения работы и взаимодействия с товарами, возможности работы из любого места, автоматизации процессов обработки заказов и продажи. Конечному потребителю используемая программа торговой сети позволяет экономить время при приобретении и покупке интересующего их товара. Приложение должно иметь простой интуитивно понятный интерфейс для работы, осуществлять правильную и быструю обработку запросов, хранить большое количество данных о товарах и пользователях в рамках одной базы данных.

3 ВЫБОР ТЕХНОЛОГИЙ, ЯЗЫКОВ И СРЕДСТВ ДЛЯ РАЗРАБОТКИ ПРИЛОЖЕНИЯ

3.1 Программные средства, используемые при разработке системы

Для реализации программного продукта «Иформационная система с архитектурой клиент-сервер» были рассмотрены следующие технологии, языки и архитектуры:

- NetBeans IDE – среда разработки
- SQL - декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.
- MySQL Command Line – инструмент для администрирования СУБД SQL;
- Maven – универсальный сборщик проектов для языка Java.
- Swing – библиотека для создания графического интерфейса для программ на языке Java.
- Server Socket – библиотека языка Java для осуществления обмена информации в сети клиент-сервер.
- JDBC – библиотека для осуществления подключения к БД для программ языка Java.
- Java - сильно типизированный объектно-ориентированный язык программирования;

Архитектура:

В разработанном приложении используется архитектура «Клиент - сервер».

Как правило компьютеры и программы, входящие в состав информационной системы, не являются равноправными. Некоторые из них владеют ресурсами (файловая система, база данных и т.д.), другие имеют возможность обращаться к этим ресурсам. Программу, управляющую ресурсом, называют сервером этого ресурса (файл-сервер, сервер базы данных, вычислительный сервер...). Клиент и сервер какого-либо ресурса могут находится как в рамках одной вычислительной системы, так и на различных компьютерах, связанных сетью.

Основной принцип архитектуры "клиент-сервер" заключается в разделении функций приложения на три группы:

- ввод и отображение данных (клиентская сторона);
- прикладные функции, характерные для данной предметной области;
- функции управления ресурсами (серверная сторона

Поэтому, в любом приложении выделяются следующие компоненты:

- компонент представления данных;
- прикладной компонент;
- компонент управления ресурсом.

Выделенные части приложения взаимодействуют друг с другом, обмениваясь сообщениями в заранее согласованном формате. В этом случае двухзвенная архитектура клиент-сервер становится трехзвенной, а к некоторых случаях, она может включать и больше звеньев.

3.2 NetBeans IDE

NetBeans IDE - бесплатная интегрированная среда разработки с открытым исходным кодом для разработчиков программного обеспечения. Среда предоставляет все средства, необходимые для создания профессиональных десктоп приложений, корпоративных, мобильных и веб-приложений на платформе Java, а также C/C++, PHP, JavaScript, Groovy и Ruby.

Возможна разработка Java десктоп приложений с профессиональными графическими интерфейсами пользователя. Используется визуальный редактор - Swing GUI Builder. Работа осуществляется путем перетаскивания элементов графического интерфейса из палитры на холст. Предварительное позиционирование элементов можно осуществлять с помощью указателя мыши. Панель свойств и инспектор компонентов предоставляют возможность тонкой настройки каждого компонента интерфейса.

Основные преимущества данной среды разработки являются:

- рабочая область среды IDE является полностью настраиваемой - существует возможность пользовательской настройки действий, выполняемых с помощью панели, назначения "горячих" клавиш и т.д.
- IDE имеет в своем составе расширенный многоязыковой редактор для различных языков программирования - Java, C/C++, Ruby, Groovy, PHP, JavaScript, CSS, XML, HTML, RHTML, JSP, документацию Javadoc. Существует возможность расширения функций редактора с целью поддержки любого другого языка.
- Производится проверка ошибок во время ввода, отображение вариантов для автозавершения кода и фрагментов документации по требуемому языку программирования.
- Редактор может генерировать и вставлять в исходный код стандартные фрагменты кода на Java или других языках.

3.3 Java

Java - в современном мире Java является одним из самых известных и используемых объектно-ориентированных языков программирования. Данный ЯП разрабатывался как универсальный для решения различного горизонта задач. Java прошел огромный путь развития, в текущий момент является уже не просто ЯП, а целой платформой, которая объединяет огромное количество различных технологий, библиотек и т.д. Все это позволяет вести разработку на Java серьезных enterprise приложений, а большое количество дополнительных модулей дает возможность подобрать конкретные средства для решения поставленной задачи. Многие разработчики выбирают Java, в результате чего растет количество различных решений для возникающих вопросов, а также увеличивается объем литературы, доступной для изучения.

К особенностям Java относится:

- Одной из главных особенностей языка Java является то, что его код сначала транслируется в специальный байт-код, независимый от платформы. Затем происходит выполнения данного кода специальной виртуальной машиной Java. Данный принцип отличается от других интерпретируемых языков, таких как PHP или Perl, с другой стороны Java не является чисто компилируемым языком как C или C++;

- архитектура языка обеспечивает кроссплатформенность и аппаратную переносимость программ на Java, из этого следует, что программам не нужно перекомпилироваться, чтобы быть используемыми на других платформах, таких как: Windows, Linux, Mac Os и других.

-Java является языком с Си-подобным синтаксисом и близок в этом отношении к C/C++ и C#. Данная особенность позволяет разработчикам, уже имеющих опыт работы с этими языками, быстро приступить к разработке приложений на Java.

- в Java присутствует автоматический сборщик мусора (англ. Garbage Collector), это позволяет разработчику не уделять времени на ручную

очистку памяти от ранее использованных объектов, что значительно упрощает и ускоряет работу.

- Java является объектно-ориентированным языком. Объектно-ориентированная модель содержит основные принципы: полиморфизм, наследование, инкапсуляцию. Это позволяет разработчикам на Java создавать сложные, но при этом гибкие, хорошо масштабируемые и расширяемые приложения.

3.4 SQL

SQL - структурированный язык запросов. Он используется для создания, выполнения каких - либо изменений и управления данными в реляционной БД, которая в свою очередь управляется посредством СУБД.

В разрабатываемой программе SQL является основным способом работы с базой данных и позволяет выполнять следующий набор операций:

- создание в базе данных новой таблицы;
- добавление в таблицу новых записей;
- изменение записей;
- удаление записей;
- выборка записей из одной или нескольких таблиц (в соответствии с заданным условием);
- изменение структур таблиц.

В первую очередь язык SQL является информационно-логическим языком, что означает, что он предназначен для описания, модификации и получения данных из реляционных БД. Вначале SQL был главным способом взаимодействия пользователя с БД и позволял ему выполнять такие операции как: создание в БД новых таблиц, добавление в таблицы данных, изменение существующих значений записей, удаление их, получение значений из таблиц в зависимости от условия запроса, модификация структур таблиц.

Со временем SQL стал сложнее и расширился новыми особенностями, поэтому стал приобретать черты, делающие его похожим на языки программирования. Несмотря на все изменения, затронувшие SQL, он является самым распространенным средством для взаимодействия прикладных программ с БД.

3.5 MySQL Command Line

MySQL Command Line – это инструмент для администрирования SQL сервера направленный на постоянную поддержку базы данных с целью поддержания и увеличения её эффективности. Критерий эффективности все больше зависит от безопасности баз данных, а корректное управление учетными записями пользователей гарантирует эффективное функционирование всей компании.

Для предоставления возможности надежного и эффективного управления базами данных, MySQL Command Line был оснащена мощным средством SQL администрирования. Данный инструмент полностью отвечает всем современным требованиям первоклассного SQL-администратора.

3.6 Maven

Мавен - это инструмент для сборки Java проекта: компиляции, создания jar, создания дистрибутива программы, генерации документации. Простые проекты можно собрать в командной строке. Если собирать большие проекты с командной строки, то команда для сборки будет очень длинной, поэтому её иногда записывают в bat/sh скрипт. Но такие скрипты зависят от платформы. Для того чтобы избавиться от этой зависимости и упростить написание скрипта используют инструменты для сборки проекта. Основным приемуществом Maven является то, что основные среды разработки на java легко открывают проекты, которые собираются с помощью maven. При этом зачастую проект настраивать не нужно - он сразу готов к дальнейшей разработке.

3.7 SWING

Технология Swing — это UI Java-платформы. Она выступает как программное обеспечение, управляющее всем взаимодействием пользователя и компьютера. По существу, она служит посредником между пользователем и внутренностями компьютера. Swing предоставляет механизмы для управления аспектами UI:

- Клавиатура: Swing предоставляет способ перехвата пользовательского ввода.
- Цвета: Swing предоставляет способ менять цвета, которые вы видите на экране.
- Текстовое поле для ввода: Swing предоставляет текстовые компоненты для обработки всех повседневных задач.

В любом случае Swing предоставляет вам все инструменты, необходимые для создания вашего собственного UI.

Базовым строительным блоком всей библиотеки визуальных компонентов Swing является JComponent. Это суперкласс каждого компонента. Он является абстрактным классом, поэтому в действительности вы не можете создать JComponent, но он содержит буквально сотни функций, которые каждый компонент Swing может использовать как результат иерархии классов.

JComponent — базовый класс не только для Swing-компонентов, но также и для пользовательских компонентов. Он знает, как обрабатывать все нажатия клавиш на клавиатуре. Подклассы, следовательно, должны только прослушивать определенные клавиши. Также, он содержит метод add(), который позволяет добавлять другие JComponent. Этим способом вы можете добавить любой Swing-компонент к любому другому Swing-компоненту для создания вложенных компонентов (например, JPanel, содержащую JButton, или даже более причудливые комбинации, например JMenu, содержащее JButton).

3.8 Server Socket

Библиотека Server Socket реализует сокеты сервера. Сокет сервера ожидает запросов, чтобы войти по сети. Это выполняет некоторую работу, основанную на том запросе, и затем возможно возвращает результат запрашивающей стороне.

Сокеты обеспечивают механизм связи между двумя компьютерами, использующими TCP. Клиентская программа создает сокет на своем конце связи и пытается подключить этот сокет к серверу.

Когда соединение установлено, сервер создает объект сокета на своем конце связи. Клиент и сервер теперь могут общаться, записывая и считывая данные из сокета.

Класс java.net.Socket представляет собой сокет, а класс java.net.ServerSocket предоставляет механизм для серверной программы для прослушивания клиентов и установления соединений с ними.

Следующие шаги выполняются при установлении соединения TCP между двумя компьютерами с использованием сокетов:

- сервер создает экземпляр объекта ServerSocket, определяющий, по какому номеру порта должна происходить связь;
- сервер вызывает метод accept () класса ServerSocket. Этот метод ожидает, пока клиент не подключится к серверу по указанному порту;
- после ожидания сервера клиент создает экземпляр объекта Socket, указывая имя сервера и номер порта для подключения;
- конструктор класса Socket пытается подключить клиента к указанному серверу и номеру порта. Если связь установлена, у клиента теперь есть объект Socket, способный связываться с сервером;
- на стороне сервера метод accept () возвращает ссылку на новый сокет на сервере, который подключен к клиентскому сокету.

Сокеты обеспечивают механизм связи между двумя компьютерами, использующими TCP. Клиентская программа создает сокет на своем конце связи и пытается подключить этот сокет к серверу.

Когда соединение установлено, сервер создает объект сокета на своем конце связи. Клиент и сервер теперь могут общаться, записывая и считывая данные из сокета.

Класс java.net.Socket представляет собой сокет, а класс java.net.ServerSocket предоставляет механизм для серверной программы для прослушивания клиентов и установления соединений с ними.

Следующие шаги выполняются при установлении соединения TCP между двумя компьютерами с использованием сокетов:

- сервер создает экземпляр объекта ServerSocket, определяющий, по какому номеру порта должна происходить связь;
- сервер вызывает метод accept () класса ServerSocket. Этот метод ожидает, пока клиент не подключится к серверу по указанному порту;
- после ожидания сервера клиент создает экземпляр объекта Socket, указывая имя сервера и номер порта для подключения;
- конструктор класса Socket пытается подключить клиента к указанному серверу и номеру порта. Если связь установлена, у клиента теперь есть объект Socket, способный связываться с сервером;
- на стороне сервера метод accept () возвращает ссылку на новый сокет на сервере, который подключен к клиентскому сокету.

Сервер создает экземпляр объекта ServerSocket, определяющий, по какому номеру порта должна происходить связь.

Сервер вызывает метод accept () класса ServerSocket. Этот метод ожидает, пока клиент не подключится к серверу по указанному порту.

После ожидания сервера клиент создает экземпляр объекта Socket,

указывая имя сервера и номер порта для подключения.

Конструктор класса Socket пытается подключить клиента к указанному серверу и номеру порта. Если связь установлена, у клиента теперь есть объект Socket, способный связываться с сервером.

На стороне сервера метод accept () возвращает ссылку на новый сокет на сервере, который подключен к клиентскому сокету.

После того, как соединения установлены, связь может происходить с использованием потоков ввода / вывода. Каждый сокет имеет и OutputStream, и InputStream. OutputStream клиента подключен к InputStream сервера, а InputStream клиента подключен к OutputStream сервера.

TCP является двусторонним протоколом связи, поэтому данные могут передаваться по обоим потокам одновременно. Ниже приведены полезные классы, предоставляющие полный набор методов для реализации сокетов.

3.9 JDBC

Java Database Connectivity – это стандартный API для независимого соединения языка программирования Java с различными базами данных.

JDBC решает следующие задачи:

- создание соединения с БД;
- создание SQL выражений;
- выполнение SQL – запросов;
- просмотр и модификация полученных записей.

Если говорить в целом, то JDBC – это библиотека, которая обеспечивает целый набор интерфейсов для доступа к различным БД.

Для доступа к каждой конкретной БД необходим специальный JDBC – драйвер, который является адаптером Java – приложения к БД.

JDBC API использует менеджер драйверов и специальные драйверы БД

для обеспечения подключения к различным базам данных.

JBDC Manager проверяет соответствие драйвера и конкретной БД. Он поддерживает возможность использования нескольких драйверов одновременно для одновременной работы с несколькими видами БД.

JDBC API состоит из следующих элементов:

- Менеджер драйверов управляет списком драйверов БД. Каждой запрос на соединение требует соответствующего драйвера. Первое совпадение даёт нам соединение;
- Драйвер (Driver). Этот элемент отвечает за связь с БД. Работать с ним нам приходится крайне редко. Вместо этого мы чаще используем объекты DriverManager, которые управляют объектами этого типа;
- Соединение (Connection). Этот интерфейс обеспечивает нас методами для работы с БД. Все взаимодействия с БД происходят исключительно через Connection;
- Выражение (Statement). Для подтверждения SQL-запросов мы используем объекты, созданные с использованием этого интерфейса;
- Результат (ResultSet). Экземпляры этого элемента содержат данные, которые были получены в результате выполнения SQL – запроса. Он работает как итератор и “пробегает” по полученным данным;
- Исключения (SQL Exception). Этот класс обрабатывает все ошибки, которые могут возникнуть при работе с БД.

3.10 Взаимодействие рассмотренных средств

Взаимодействие можно рассмотреть в рамках клиент-серверной модели, где основными блоками являются серверная часть, клиентская часть и БД. Для разработки серверной части, взаимодействующей с базой данных и частью клиента, а также обеспечивающей полного функционала торговой сети используются следующие средства. Все начинается со среды

разработки, так как без нее создание проектного продукта будет гораздо более громоздким и долгим, для этого мы воспользуемся NetBeans, вдобавок к ней идут средства для автоматизации сборки проектов maven. Следующим очень мощными инструментами являются библиотеки Server Socket и JDBC, именно они позволяет нам быстро передавать данные от БД к клиенту. Java и все выше сказанные средства позволяют описывать работу веб-сервера, те запросы, которые он должен обрабатывать и ответный результат, который он должен формировать. Без сервера приложения просто не будет существовать. Ни одна подобная система не обойдется без хранения и взаимодействия с данными.

4 РЕАЛИЗАЦИЯ

Прежде чем начинать непосредственную разработку всех частей приложения, необходимо еще раз проанализировать систему, какой функционал необходимо реализовывать, какие сущность должны находиться в базе данных, какие должны быть атрибуты у этих сущностей. Также необходимо рассмотреть, как именно будет взаимодействовать клиент, сервер и БД, какие для этого нужны будут механизмы и т.д.

4.1 Алгоритм работы персонала в торговой сети

Для того чтобы реализовать систему нужно подготовить клиента, сервера и СУБД.

Вся последовательность работы состоит из следующего алгоритма: пользователь своими действиями создает определенные события, которые обрабатываются клиентской частью, в зависимости от того что сделал клиент, серверу посылается запрос на обработку соответствующей задачи. Если во время выполнения запроса возникнет ошибка, то сервер вернет клиенту информацию об этой ошибке и не выполнит действие, если запрос пройдет успешно, то сервер обратится к СУБД для получения данных, запрашиваемых пользователем. Если при обращении к СУБД возникнет ошибка, то сервер соответствующим способом обработает ее и вернет сообщение об ошибке клиенту, в том же случае, если исключительных ситуаций не возникло, то запрос является выполненным и сервер вернет клиенту информацию об успешном выполнении запроса.

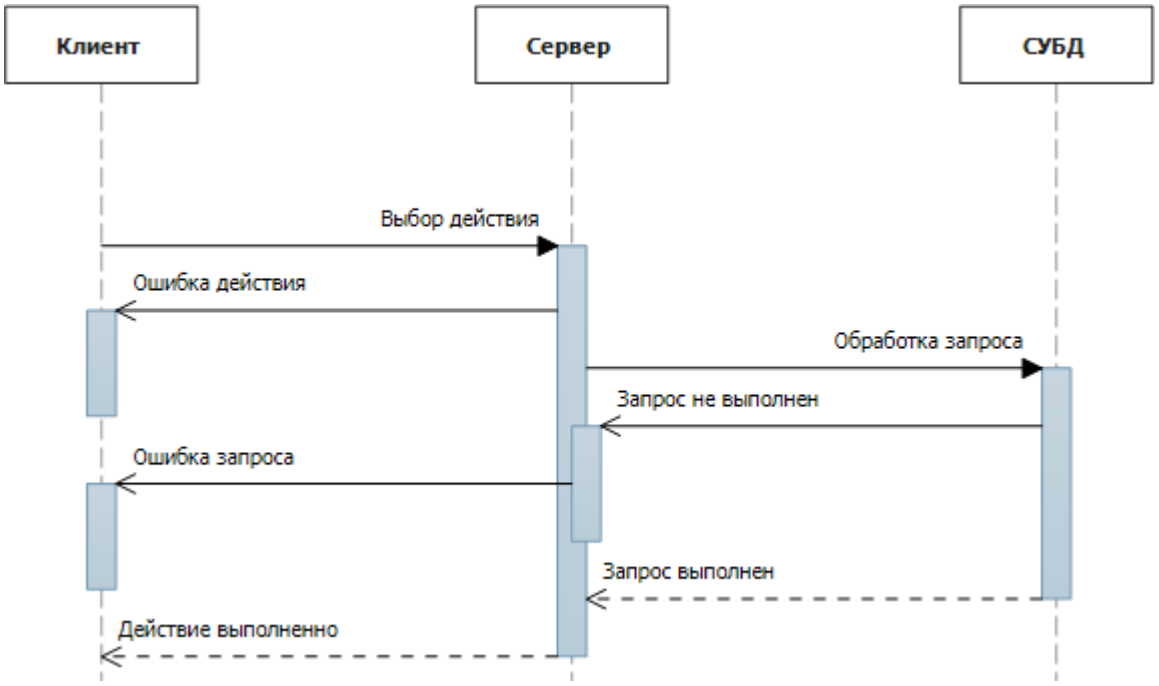


Рисунок 1 – Диаграмма последовательности работы системы

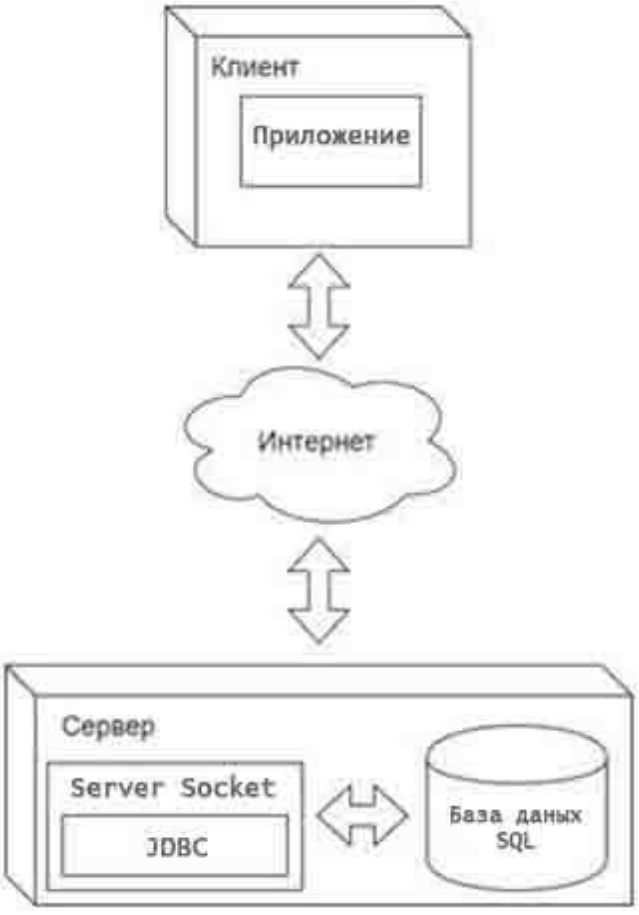


Рисунок 2 – Диаграмма развертывания системы работы персонала в торговой сети

4.2 Описание структуры и таблиц БД

Для разработки структуры БД, необходимо проанализировать какие сущности требуют рассмотрения, и какие атрибуты будут им принадлежать. Дополнительно необходимо выявить, как данные сущности будут взаимосвязаны. Все это является работой по созданию ERD (англ. Entity Relation Diagram), т.е. диаграммы взаимодействия сущностей.

Для нормальной работы системы электронной коммерции были разработаны следующие таблицы:

- sellers - таблица, содержащая список персонала и данные о каждом зарегистрированном представителе персонала;

- goods - таблица, которая содержит данные о всех товарах в развёртываемой системе торговой сети;

Таблица 1 - Атрибуты сущности «sellers» (продавцы).

Имя столбца	Описание	Тип данных	Пример
ID	Идентификатор пользователя	INTEGER	8437
NAME	ФИО продавца	VARCHAR(40)	Иванов Г.П.
DEPARTMENT	Раздел, в котором работает продавец	VARCHAR(15)	Молочный
LOGIN	Логин для входа в систему	VARCHAR(10)	ivan
PASSWORD	Пароль	VARCHAR(10)	ivan8milk1

Таблица 2 - Атрибуты сущности «goods» (товары).

Имя столбца	Описание	Тип данных	Пример
ID	Идентификатор товара	INTEGER	8437
NAME	Название товара	VARCHAR(40)	Сыр Российский
DEPARTMENT	Раздел, в котором располагается товар	VARCHAR(15)	Молочный

4.3 Основные функции доступные пользователям

Система работы персонала торговой сети организует доступ к функционалу только авторизированным пользователям. Пользователь в свою очередь может быть авторизован в системе или не авторизован. Распределение обязанностей сотрудников строится на договоренностях, построенных при личном взаимодействии, что означает, что программно все работники имеют одинаковый уровень доступа и взаимодействия со средой сайта.

Возможности, предоставленные категории идентифицированных пользователей сайта, можно рассмотреть в виде диаграммы использования. Данная диаграмма наглядно и полно показывает все функции, которыми они обладают.

Авторизированному пользователю доступен следующий функционал по использованию клиентского приложения:

- Авторизация
- Просмотр товаров

- Все товары
- Товары раздела
- Поиск товаров
- Продажа товаров

Диаграмма вариантов использования для авторизованного пользователя представлена на рисунке 3.

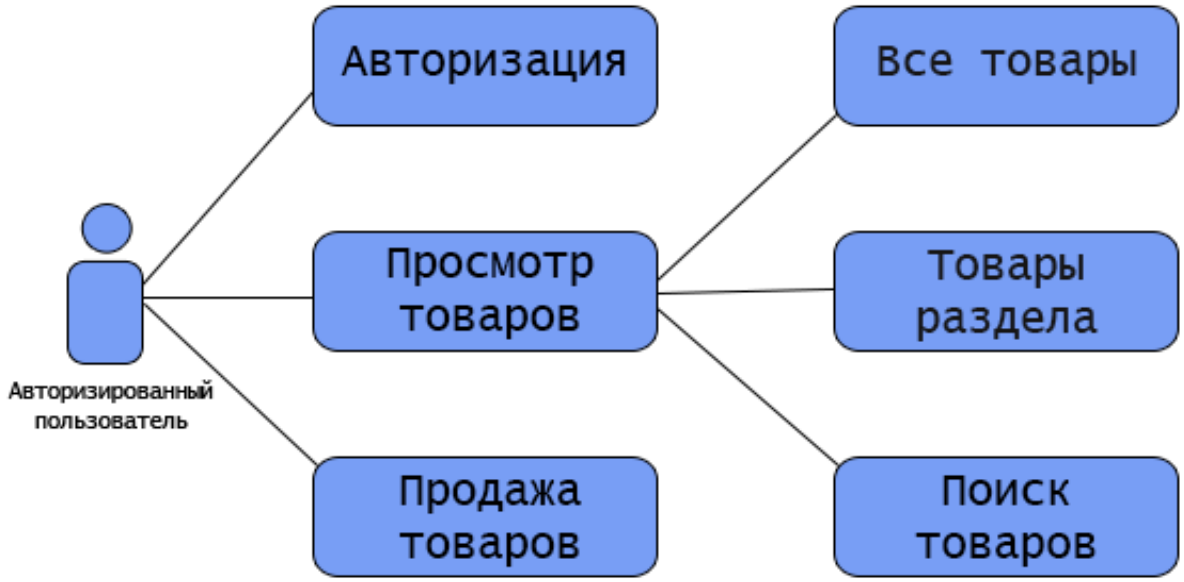


Рисунок 3 - Диаграмма вариантов использования авторизованного пользователя

4.4 Схема оформления заказа

Рассмотреть процесс осуществления операции по продаже можно с помощью диаграммы продажи товара. Он состоит из взаимодействия клиентской и серверной стороны.

Процесс начинается с того, что пользователь зашел в каталог товаров и выбрал товар. При выборе товара продавцу необходимо указать сумму продаваемого товара. После выбора товара на сервере происходит проверка на наличие выбранного товара. Контактные данные берутся с профиля пользователя и указываются при продаже. Запрос отправляется на сервер, где проверяется корректность введенных данных и дальнейшее продолжение процесса продажи. В том случае, когда данные указаны корректно, то система отправляет ответ об успешности выполнения продажи. На этом процесс продажи окончен. Диаграмма осуществления продажи представлена на рисунке 4.

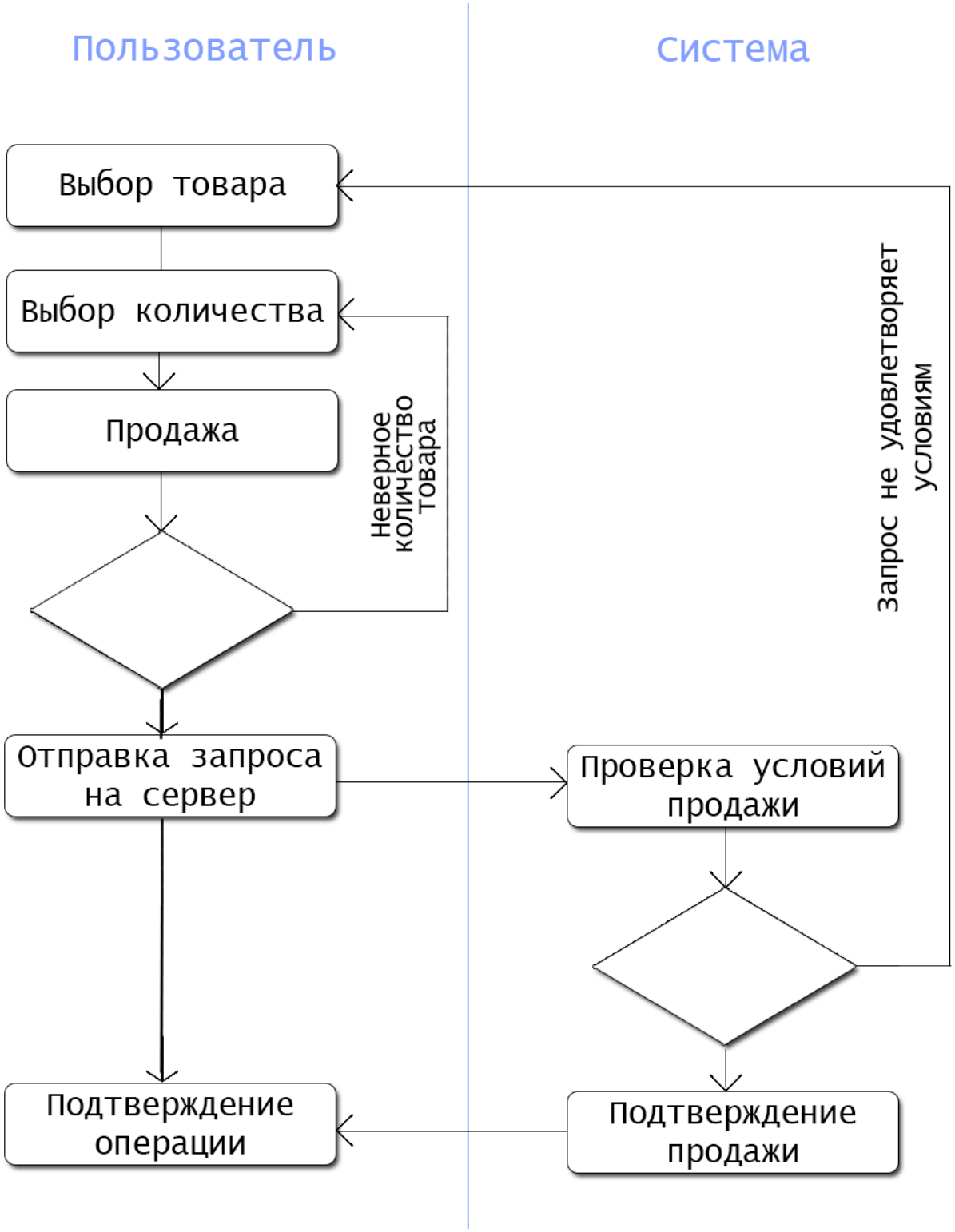


Рисунок 4 - Диаграмма осуществления продажи

5 МЕТОДЫ ИСПЫТАНИЙ

5.1 Добавление нового персонала

После выполнения работы, необходимо удостовериться в работоспособности программы, для чего необходимо провести действия по добавлению и удалению персонала, аналогичный функционал испытать с товарами, осуществить авторизацию и, в дальнейшем, осуществить продажу товара.

Для проверки работы регистрации необходимо выполнить следующие действия:

- Запустить серверное приложение, рисунок 5;

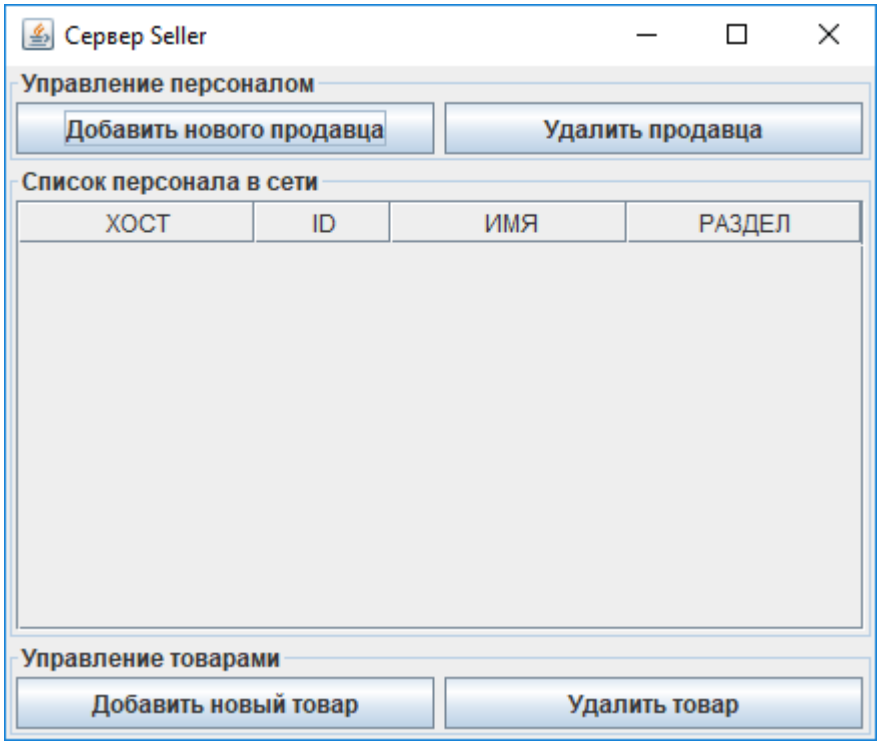


Рисунок 5 – Окно сервера

- Нажать на кнопку «Добавить нового продавца», после чего откроется окно добавления персонала, данное окно представлено на рисунке 6;

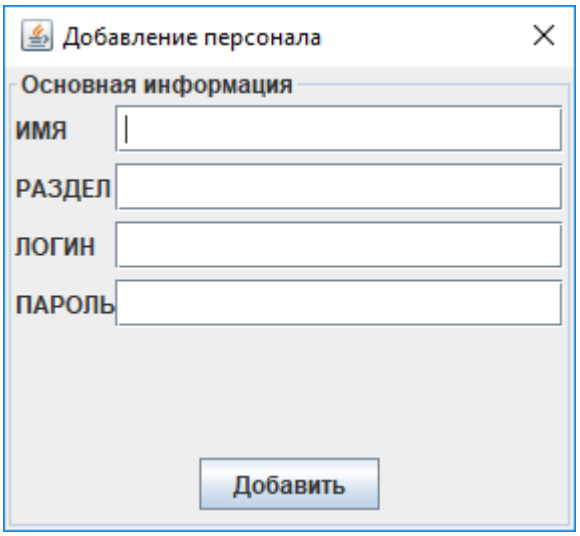


Рисунок 6 – Окно регистрации персонала

- Заполнить все необходимые поля информацией о новом продавце и нажать кнопку «Добавить», при соответствии введённой информации с условием ввода, произойдёт добавление нового члена торговой сети. Данный процесс отображён на рисунке 7.

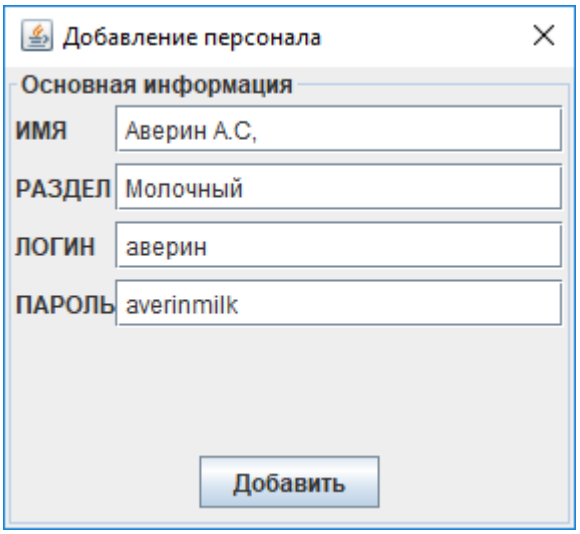


Рисунок 7 – Окно регистрации персонала

5.2 Удаление персонала

В процессе удаления представителя персонала происходит удаление всей информации о выбранном сотруднике. Весь процесс протекает на серверной стороне и выглядит следующим образом:

- Перед удалением необходимо открыть соответствующее окно. Перед нами отобразится список всех зарегистрированных в сети сотрудников. В данном окне администратор сервера имеет доступ к перечню информации каждого сотрудника, рисунок 8;

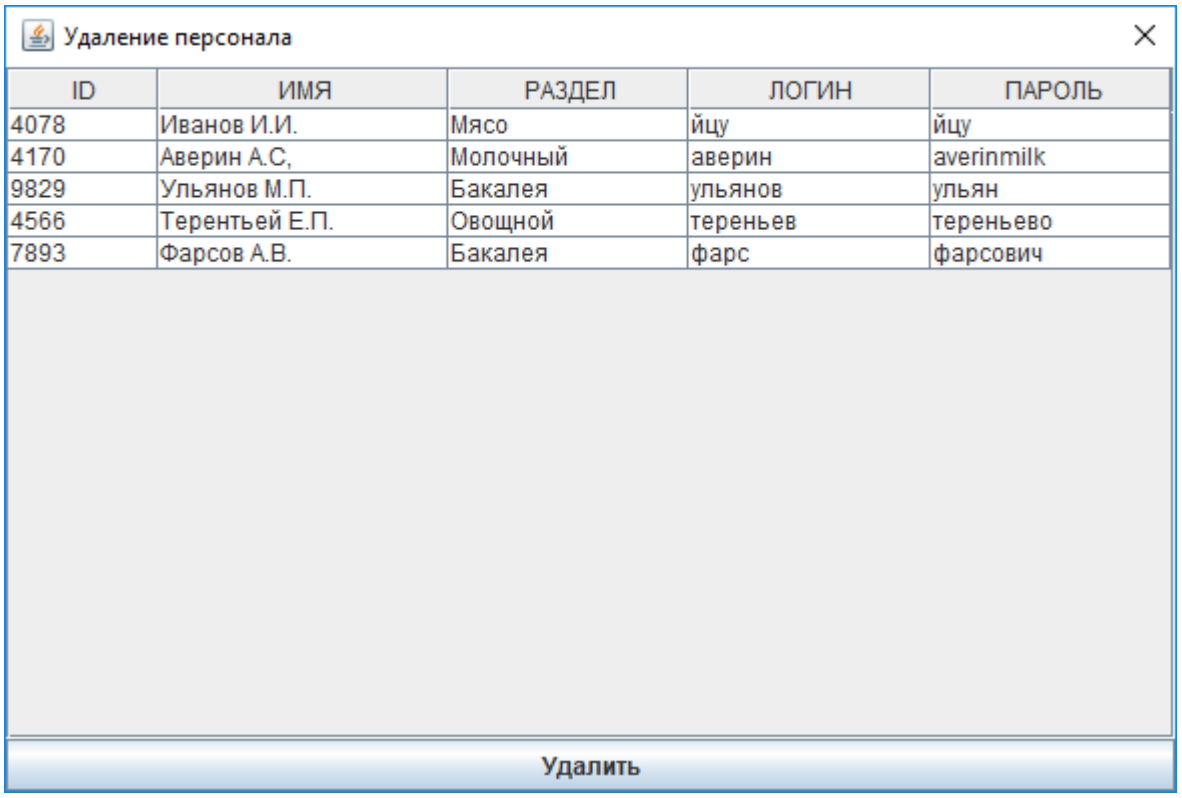


Рисунок 8 – Окно удаления персонала

- Для удаления требуется выбрать соответствующего сотрудника. После выбора сотрудника открывается возможность удалить выбранный персонал из системы, рисунок 9 и 10.

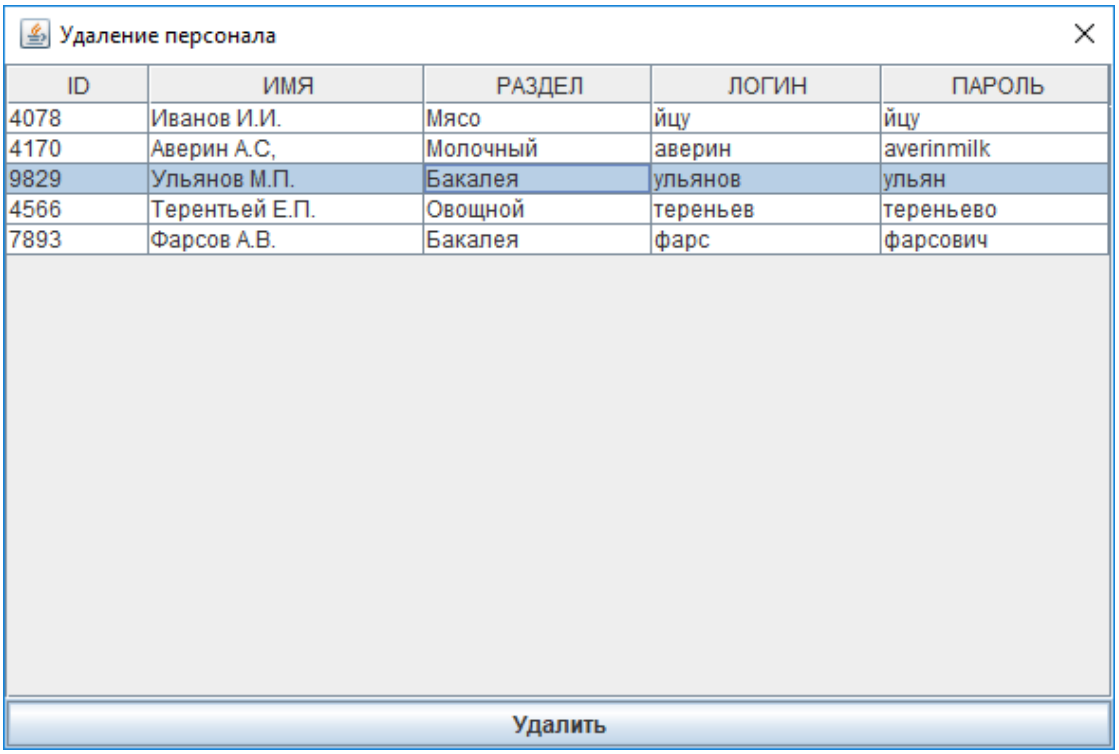


Рисунок 9 – Выбор персонала

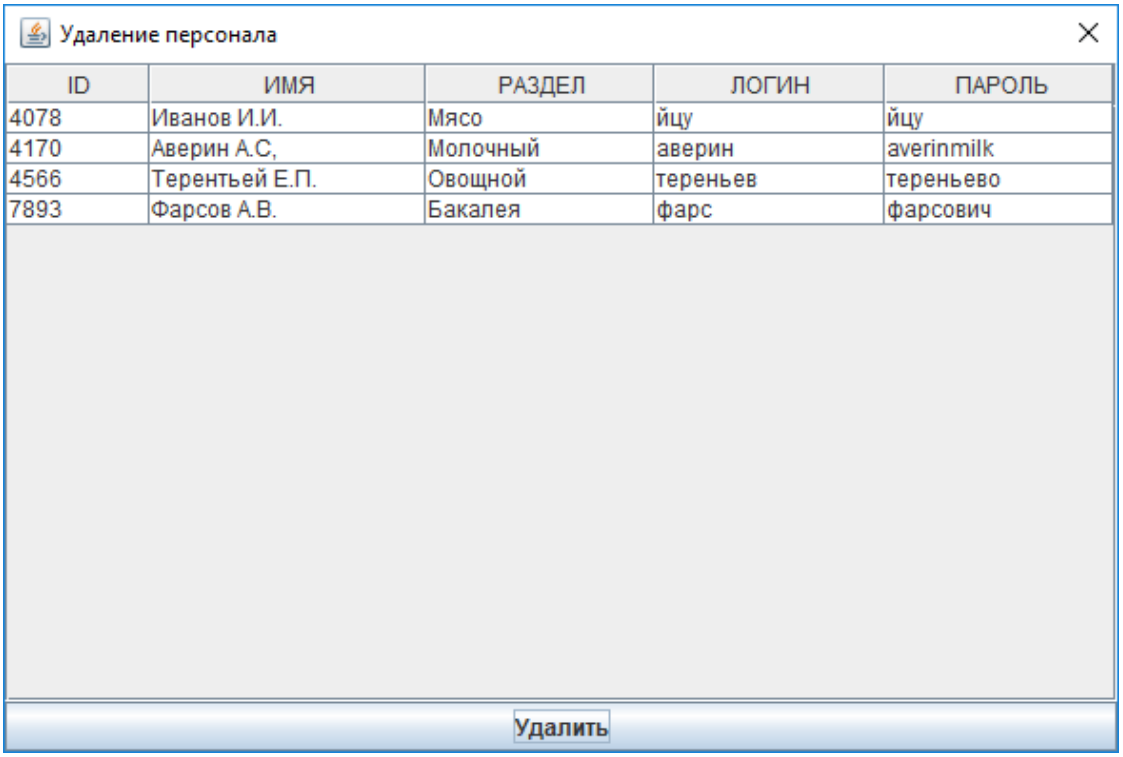


Рисунок 10 – Удаление персонала

5.3 Добавление товара

Процесс добавления товара протекает схожим методом как и при добавлении персонала:

- Перед добавлением необходимо открыть соответствующее окно на сервере. Отобразится окно с полями ввода, рисунок 11.

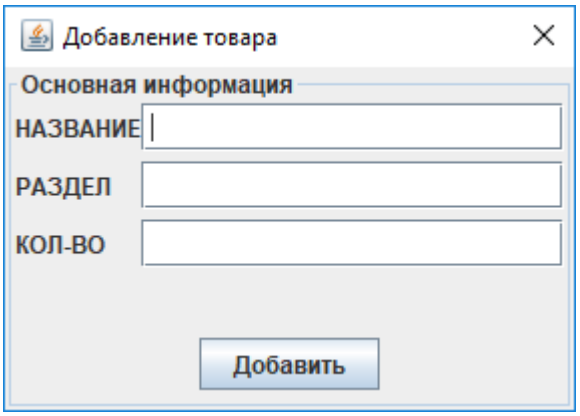


Рисунок 11 – Добавление товара

- Необходимо заполнить весь перечень требуемой информации, рисунок 12.

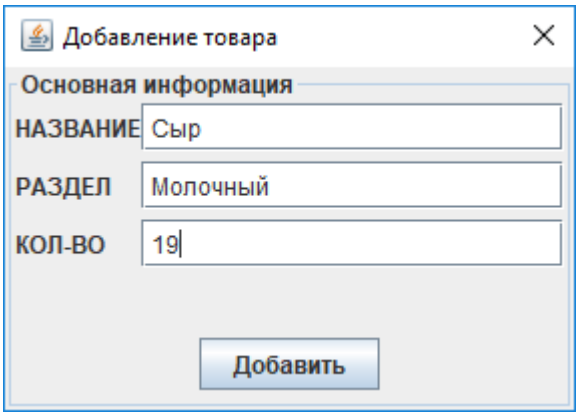


Рисунок 12 – Заполнение информации о товаре

При добавлении товара может возникнуть исключительная ситуация – когда добавляемый вид товара уже существует в базе данных и повторное добавление в БД и систему может негативно отобразиться на выполнении программы. В таком случае администратору предлагается возможность обновить количество добавляемого товара на складе.

На рисунках 13 и 14 отображается ранее описанная ситуация по добавлению товара.

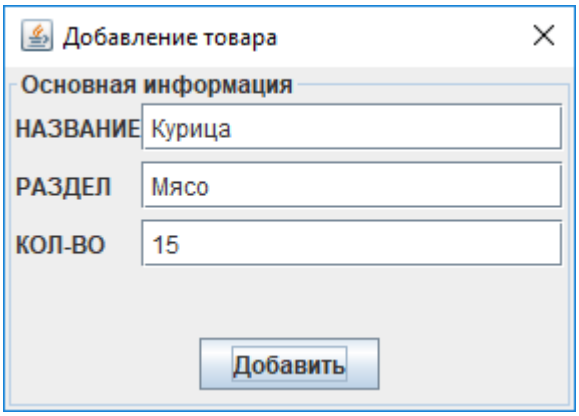


Рисунок 13 – Добавление существующего товара

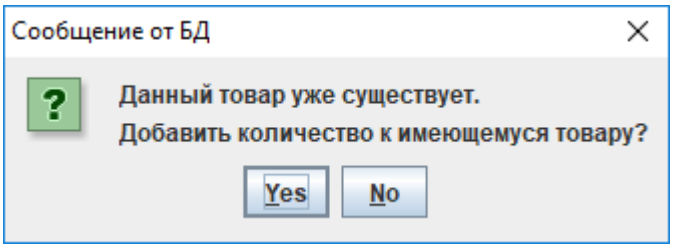


Рисунок 14 – Добавление количества товара

После подтверждения о добавлении количества товара к уже имеющемуся товару отсылается запрос на изменение данных к БД. Результатом выполненной работы будет сумма ранее добавленного товара и количества добавляемого товара.

5.4 Удаление товара

Удаление продукции протекает из окна «Удаление товара». Для его открытия необходимо на серверной стороне нажать кнопку «Удалить товар».

Перед нами откроется соответствующее окно, рисунок 15.

Рисунок 15 – Окно удаления товара

Выбор требуемого товара и его удаление происходит аналогично процессу удаления персонала. Данный процесс отображён на рисунке 16.

Рисунок 16 – Удаление товара

5.5 Авторизация

Перед входом в систему требуется запустить клиентское приложение. Следующим шагом после запуска является ввод данных для авторизации. У каждого пользователя системы имеется логин и пароль. Данные для входа указываются при регистрации сотрудника.

Перед процессом авторизации сотруднику необходимо указать его логин и пароль в соответствующих графах главного окно клиентского приложения, рисунок 17.

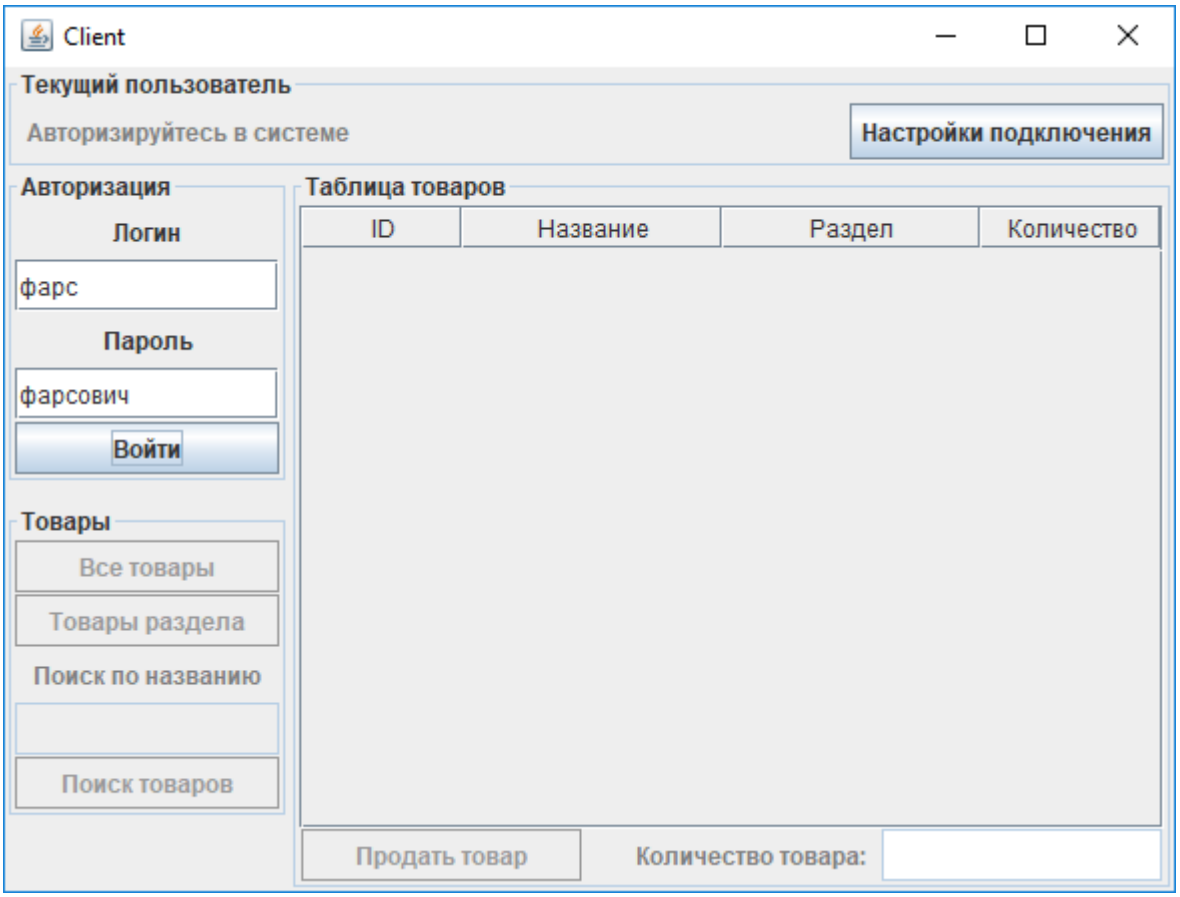


Рисунок 17 – Клиентское приложение

После нажатия кнопки «Вход» проверяется корректность введённых данных. На сторону сервера отсылается массив данных включающий введённый логин и пароль.

При удачной идентификации пользователя на серверной стороне, он оповещает клиентскую сторону о удачном входе и передаёт полномочия по использованию системы вошедшему сотруднику, процесс отображён на рисунке 18.

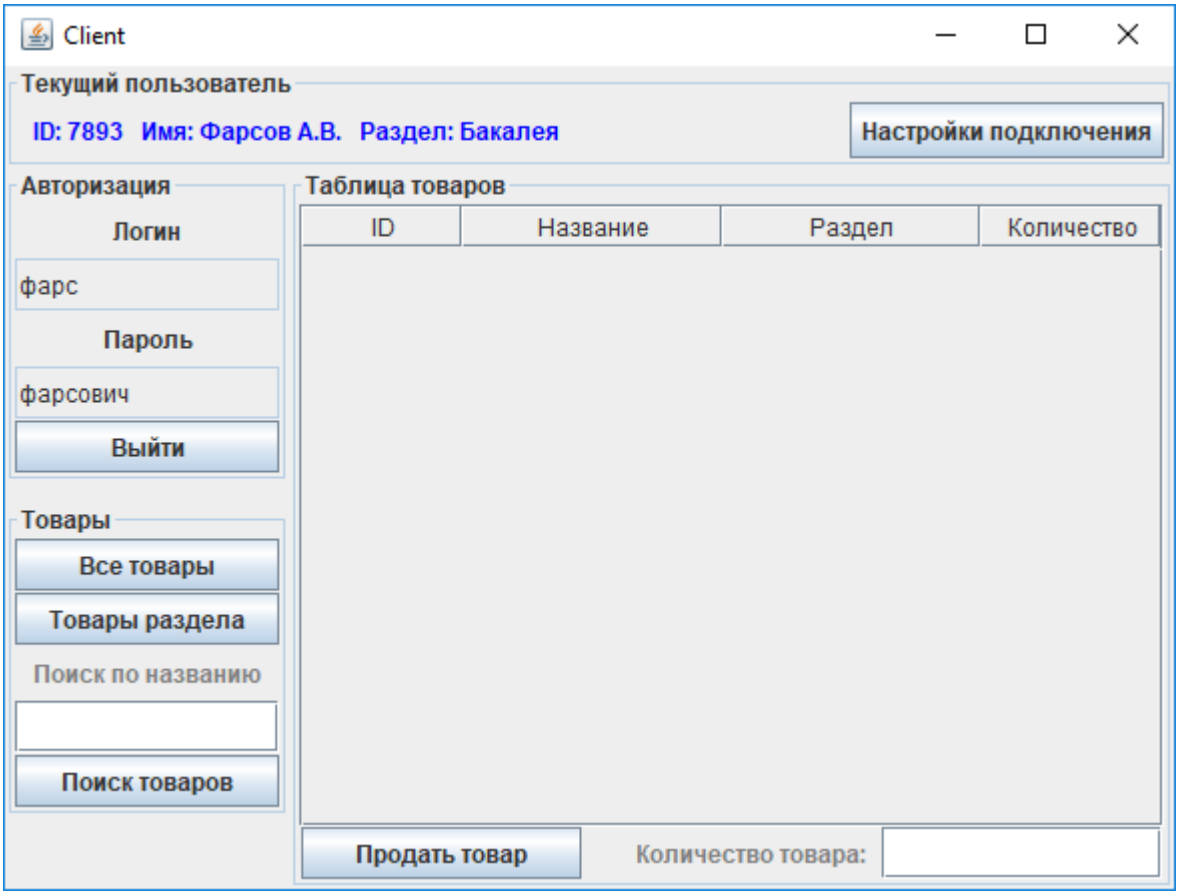


Рисунок 18 – Успешная идентификация пользователя

При успешной авторизации клиентское приложение получает информацию о вошедшем с этого приложения сотруднике. Также предоставляются права по эксплуатации системы.

В любое время текущий сотрудник имеет возможность отключиться от сервера и прекратить работу в системе.

5.6 Продажа товара

Продажа товаров – это наиболее важная функция из-за которого разрабатывалось текущее приложение. Возможность по продаже товаров имеют только зарегистрированные и авторизированные в системе сотрудники.

Перед продажей пользователю клиентского приложения, помимо авторизации, необходимо выбрать целевой товар, рисунок 19.

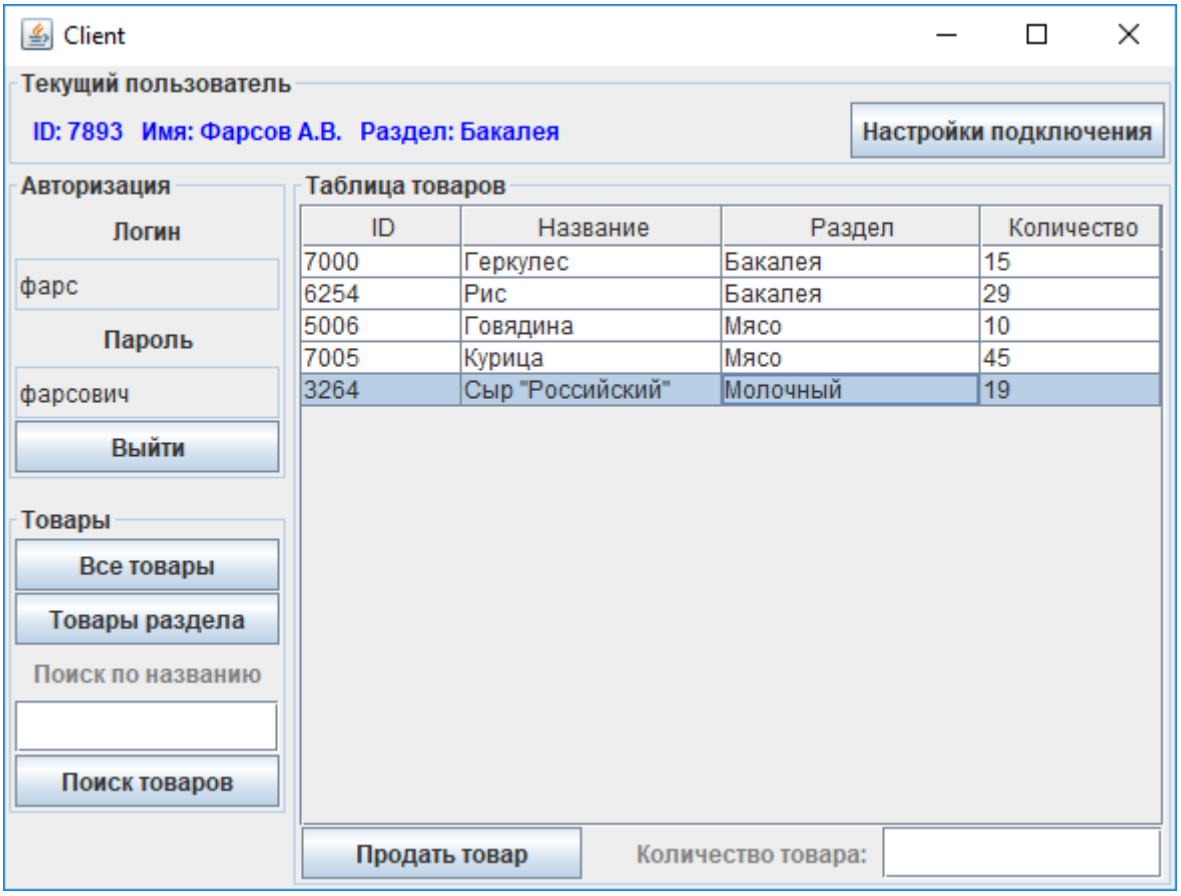


Рисунок 19 – Отображение и выбор товара

Следующим шагом следует определение количества товара, который требуется реализовать в данный момент времени. Последним этапом будет отправка команды серверу о реализации выбранного продукта и его сумма, рисунок 20.

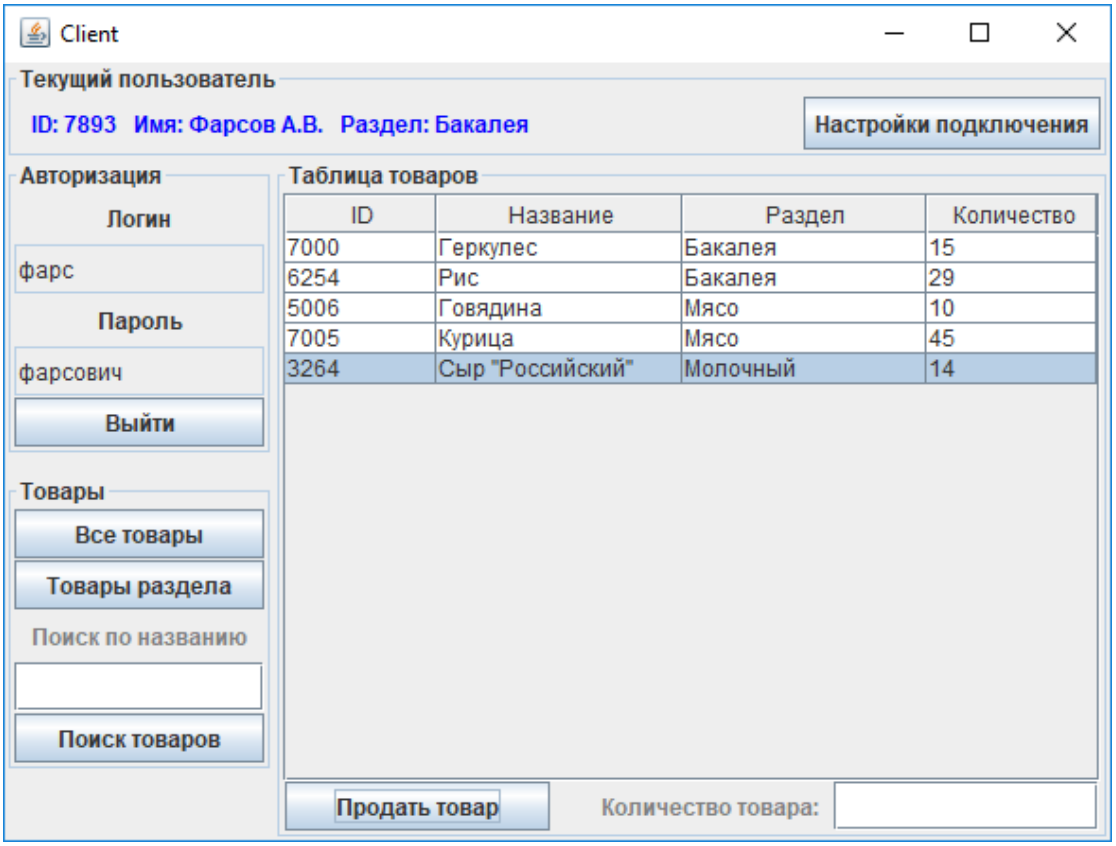


Рисунок 20 – Продажа указанного числа продукции

Наиболее распространённой проблемой по продаже может возникнуть то, что требуемого количества товара нет на складе. В разработанном продукте устраняется описанная проблема.

При возникновении похожей ситуации сервер оповещает пользователя о нехватки продукции на складе.

Описанная ситуация отображена на рисунке 21.

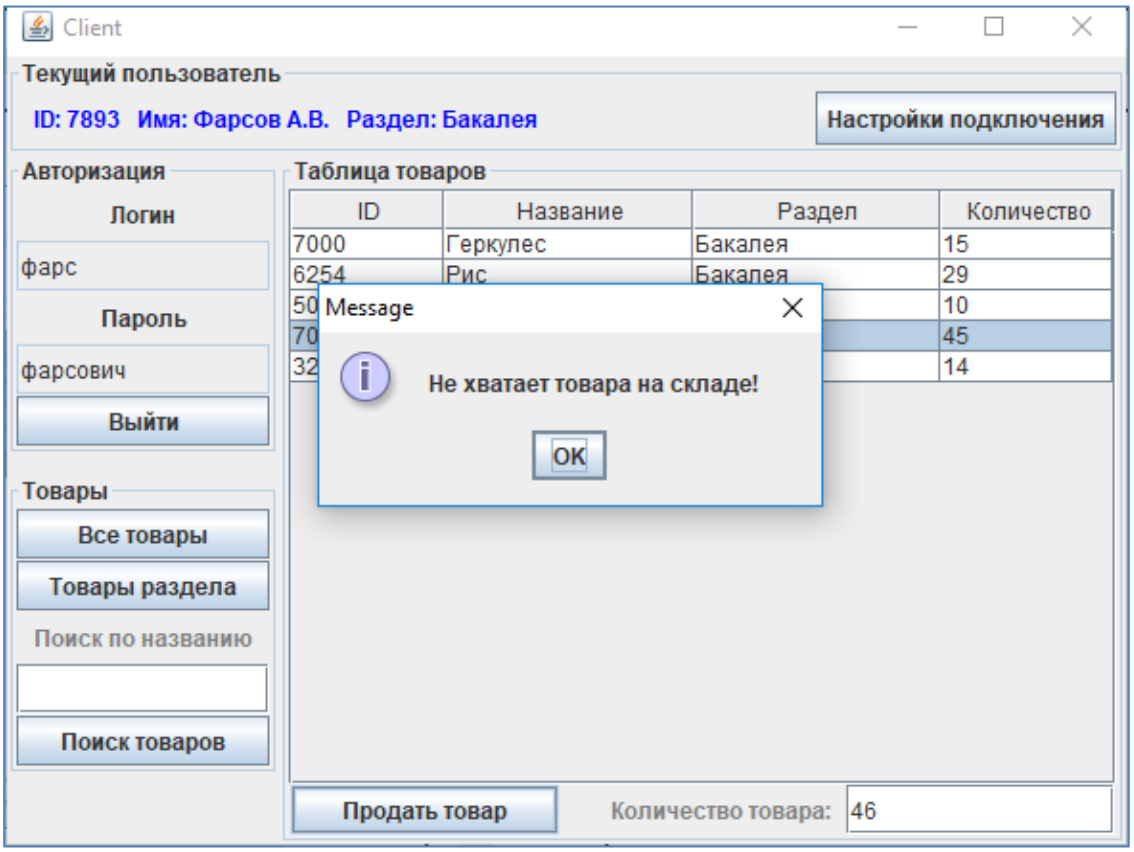


Рисунок 21 – Продажа указанного числа продукции

Текущее исключение возникает только в той ситуации, когда число продаваемого товара превышает число продукции расположенной на сладе. Обработка подобных ситуаций позволяет организовать более безопасную и стойкую систему по обороту продукции в торговой сети.

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной бакалаврской работы был разработан алгоритм решения задачи, создана и отложена программа, оформлена документация. Проведенные испытания показали, что клиент-серверное приложение для реализации товара в торговой сети соответствует всем поставленным требованиям и выполняет свою работу корректно.

За время работы с данным проектом было прочитано большое количество технической документации, изучены новые способы и механизмы программирования и проектирования систем. Значительно развит навык разработки программ на языке Java, администрирования баз данных.

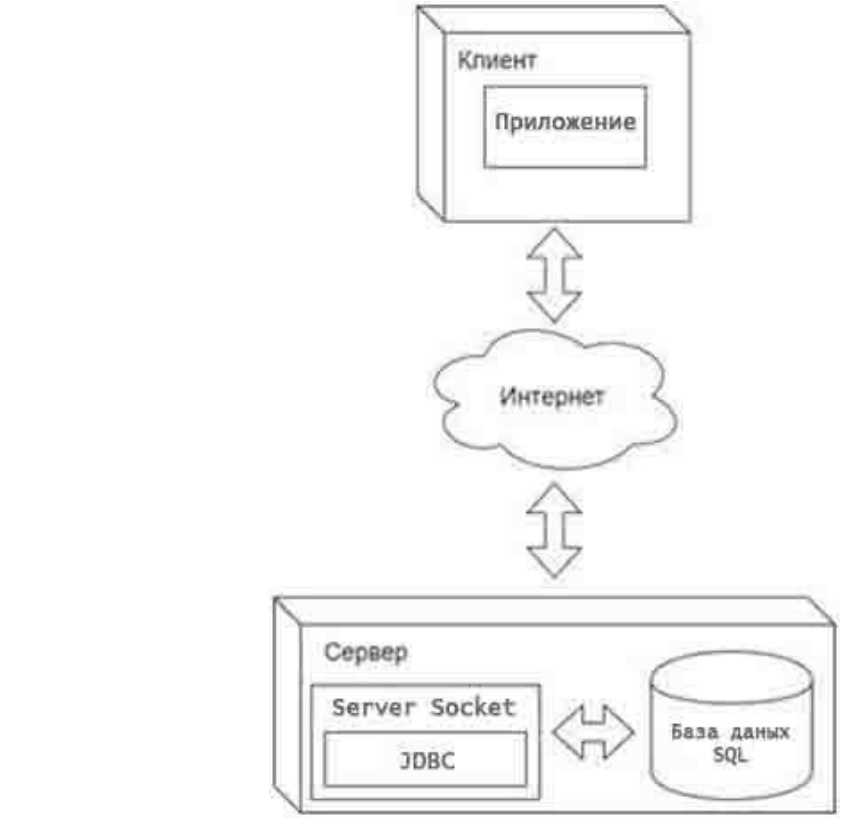
Опыт, полученный в ходе выполнения этого задания, неоспоримо поможет в дальнейшем трудоустройстве и развитии собственных навыков.

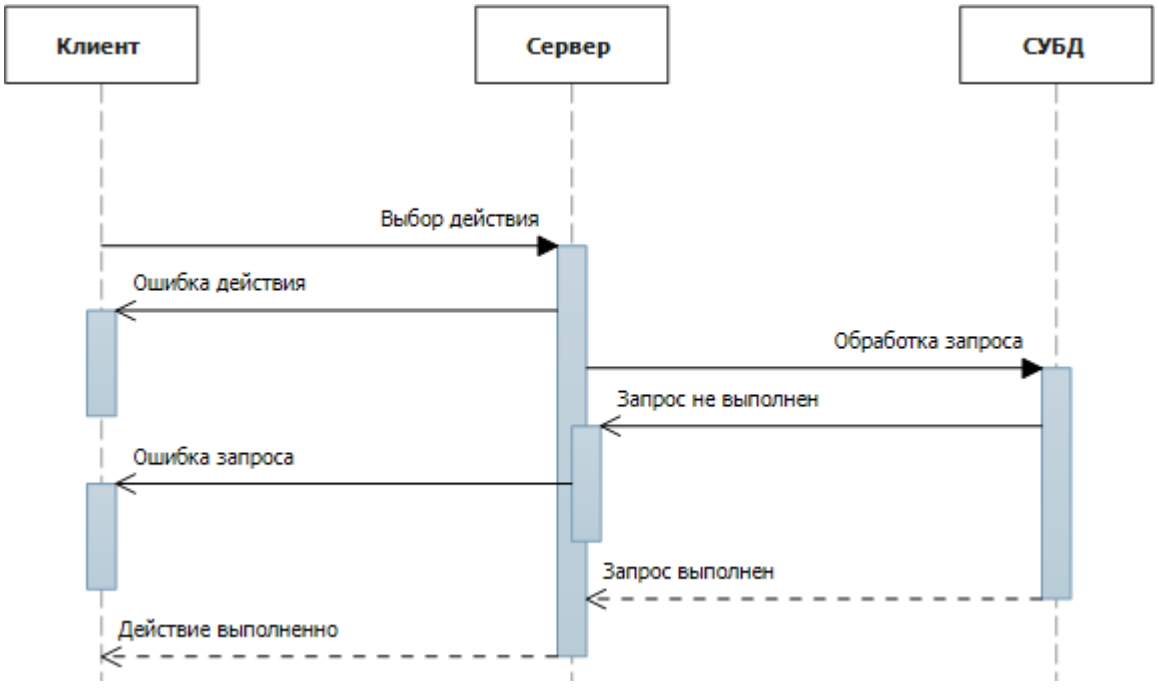
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Шилдт Герберт. Java 8: руководство для начинающих, 6-е изд. : Пер. с англ. - М.ООО "И.Д. Вильяме", 2015. - 720 с.: ил. - Парал. тит. англ.
2. Ржеуцкая С.Ю. Базы данных. Язык SQL: учеб. пособие / С.Ю. Ржеуцкая. – Вологда: ВоГТУ, 2010. – 159 с.
3. Бейли Л. Изучаем SQL. –СПб.: Питер, 2012. –592 с.
4. Clean Code: A Handbook of Agile Software Craftsmanship – Robert C. Martin
5. Алгоритмы на Java – Роберт Седжвик, Кевин Уэйн
6. Head First Object-Oriented Analysis and Design – Brett D. McLaughlin
7. Introduction to Programming in Java: An Interdisciplinary Approach - Robert Sedgewick
8. SCJP Sun Certified Programmer for Java 6 Exam 310-065 (Книга для подготовки к экзамену на сертификат Java программиста)
9. Интернет ресурс - http://study-java.ru/category/uroki-java/
10. Head First. Паттерны проектирования – Эрик Фримен, Элизабет Робсон
11. Интернет ресурс - https://javarush.ru/quests/lectures/questsyntax

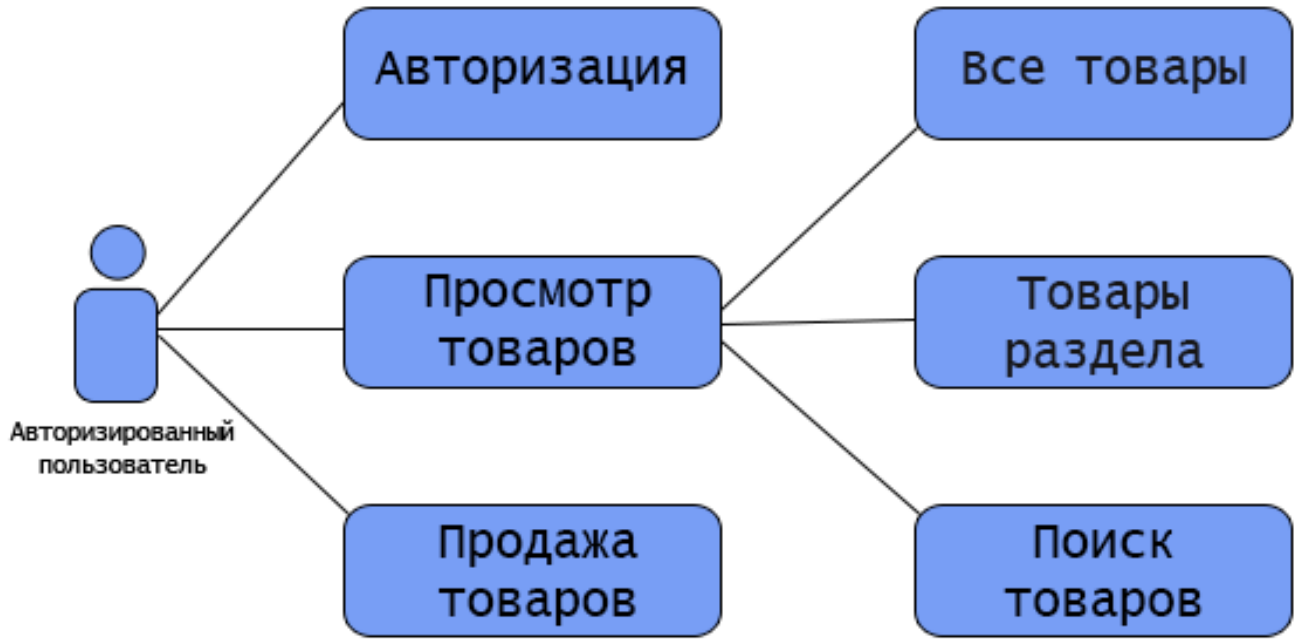
ПРИЛОЖЕНИЕ А

(Графическая часть)

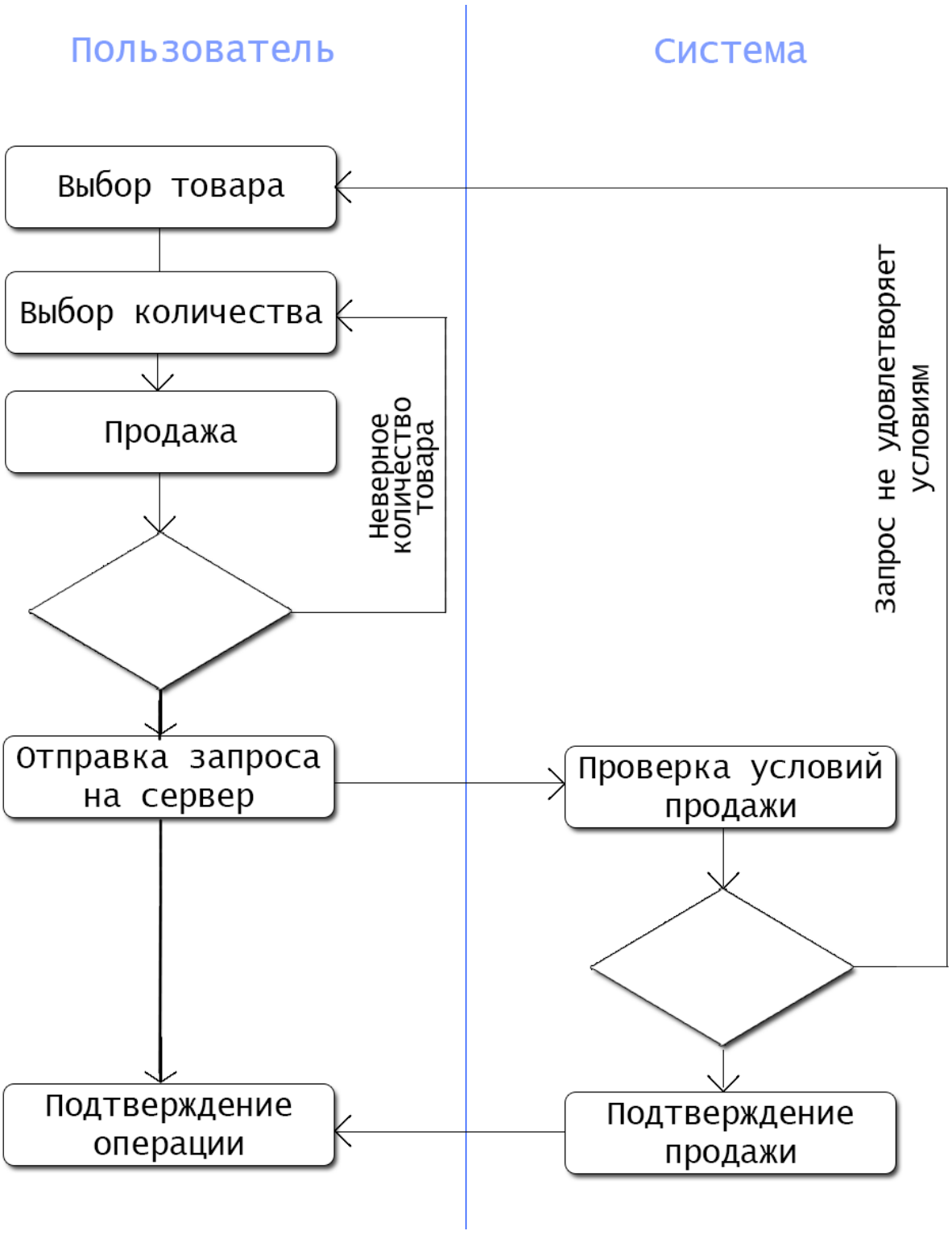




					ПГУ 2.09.03.01.27.002 Д1	
Изм.	Лист	№ докум.	Подпись	Дата		
Разраб.		Мухаметов Т.Ш.			Клиент-серверное приложение для организации работы персонала в торговой сети Структура взаимодействия компонентов системы	Лит. / Лист 46 / Листов 1
Провер.		Кармышева Н.С.				
Н. Контр.		Кучин А.В.				ФВТ гр. 16ВВ1
Утв.		Митрохин М.А.				



					ПГУ 2.09.03.01.27.006 Д4		
Изм.	Лист	№ докум.№	Подпись	Дата			
Разраб.		Мухаметов Т.Ш.			Клиент-серверное приложение для организации работы персонала в торговой сети Интерфейс пользователя UML диаграмма деятельности	Лит.	Лист 47 / Листов 1
Провер.		Кармышева Н.С.					
						ФВТ гр. 16ВВ1	
Н..Контр.		Кучин А.В.					
Утв.		Митрохин М.А.					



					ПГУ 2.09.03.01.27.006 Д4		
Изм.	Лист	№ докум.№	Подпись	Дата			
Разраб.		Мухаметов Т.Ш.			Клиент-серверное приложение для организации работы персонала в торговой сети Обработка продажи UML диаграмма деятельности	Лит.	Лист 48 / Листов 1
Провер.		Кармышева Н.С.					
Н..Контр.		Кучин А.В.				ФВТ гр. 16ВВ1	
Утв.		Митрохин М.А.					

ПРИЛОЖЕНИЕ Б

(Листинги программы)

Файл Server.java

```
package sellerserver;
import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.WindowConstants;
import javax.swing.table.DefaultTableModel;

public class Server extends JFrame {

    static final int PORT = 665;
    ArrayList<Handler> clients = new ArrayList<Handler>();
    JTable table;
    DefaultTableModel model;
    JButton bAddSeller, bDelSeller, bAddGood, bDelGood;
    int number = 0;

    public Server() {
        Socket clientSocket = null;
        ServerSocket serverSocket = null;

        try {
            serverSocket = new ServerSocket(PORT);
            System.out.println("> Сервер запущен...");

            revisionDir();
            createGUI();
            setVisible(true);
            tableListener();
            buttonsListener();

            while (true) {
                clientSocket = serverSocket.accept();
                Handler clientHandler = new Handler(clientSocket, this,
number);

                clients.add(clientHandler);
                number++;
                new Thread(clientHandler).start();
```

```
                }
            } catch (IOException ex) {
                ex.printStackTrace();
            } finally {
                try {
                    clientSocket.close();
                    System.out.println("Сервер остановлен");
                    serverSocket.close();
                } catch (IOException ex) {
                    ex.printStackTrace();
                }
            }
        }

        final void revisionDir() {
            try {
                if (!Files.exists(Paths.get("data")))
                    new File("data").mkdir();

                if (!Files.exists(Paths.get("data/settings.txt")))
                    new File("data/settings.txt").createNewFile();
            } catch(IOException ex) {
                ex.printStackTrace();
            }
        }

        final void createGUI() {
            // Настройки элементов на форме
            setBounds(50, 115, 450, 375);
            setTitle("Сервер Seller");
            setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

            // ЦЕНТРАЛЬНАЯ панель
            JPanel centerPanel = new JPanel(new GridLayout(1,5));
            centerPanel.setBorder(BorderFactory.createTitledBorder("Список
персонала в сети"));
            String[] columnNames = {"ХОСТ", "ID", "ИМЯ", "РАЗДЕЛ"};
            model = new DefaultTableModel(0, columnNames.length);
            model.setColumnIdentifiers(columnNames);
            table = new JTable(model) {
                @Override
                public boolean isCellEditable(int arg0, int arg1) {
                    return false;
                }
            };
            table.getColumnModel().getColumn(1).setPreferredWidth(25);
            centerPanel.add(new JScrollPane(table), BorderLayout.CENTER);
            add(centerPanel, BorderLayout.CENTER);

            // ВЕРХНЯЯ панель
            GridLayout grid = new GridLayout(1,2);
            grid.setHgap(5);
            JPanel topPanel = new JPanel(grid);
            topPanel.setBorder(BorderFactory.createTitledBorder("Управление
персоналом"));
        bAddSeller = new JButton("Добавить нового продавца");
```

```
        topPanel.add(bAddSeller);
                bDelSeller = new JButton("Удалить продавца");
                topPanel.add(bDelSeller);
                add(topPanel, BorderLayout.NORTH);

                // НИЖНЯЯ панель
                JPanel bottomPanel = new JPanel(grid);
                bottomPanel.setBorder(BorderFactory.createTitledBorder("Управление
товарами"));
                bAddGood = new JButton("Добавить новый товар");
                bottomPanel.add(bAddGood);
                bDelGood = new JButton("Удалить товар");
                bottomPanel.add(bDelGood);
                add(bottomPanel, BorderLayout.SOUTH);
            }

            void tableListener() {
                new Thread(new Runnable() {
                    @Override
                    public void run() {
                        int count = 0;
                        while (true) {
                            try {
                                if (count != clients.size()) {
                                    count = clients.size();
                                    model.setRowCount(0);

                                    // Заполнение таблицы клиентов а основе списка
                                    for (int i = 0; i < clients.size(); i++) {
                                        model.addRow(new Object[] {

clients.get(i).socket.getLocalAddress().toString()
                                                .substring(1) + "-" +
clients.get(i).number,
                                                clients.get(i).id,
                                                clients.get(i).name,
                                                clients.get(i).dep});
                                    }
                                }
                                model.fireTableDataChanged();
                                Thread.sleep(100);
                            } catch (InterruptedException ex) {

Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
                            }
                        }
                    }
                }).start();
            }

            void buttonsListener() {
                bAddSeller.addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        AddSeller dialog = openAddUserDialog();
                       dialog.setVisible(true);
```

```
                }
            });

            bDelSeller.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    DelSeller dialog = openDelUserDialog();
                    dialog.setVisible(true);
                }
            });

            bAddGood.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    AddGood dialog = openAddGoodDialog();
                    dialog.setVisible(true);
                }
            });

            bDelGood.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    DelGood dialog = openDelGoodDialog();
                    dialog.setVisible(true);
                }
            });
        }

        void setClientData(String HOST, String ID, String NAME, String DEP) {
            try {
                // Задержка потока для добавления строки в таблицу о новом
потоке
                Thread.sleep(600);
            } catch (InterruptedException ex) {
                Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null,
ex);
            }
            System.out.println("New client data from:" + HOST);
            for (int i = 0; i < model.getRowCount(); i++) {
                if (model.getValueAt(i, 0).toString().equals(HOST)) {
                    model.setValueAt(ID, i, 1);
                    model.setValueAt(NAME, i, 2);
                    model.setValueAt(DEP, i, 3);
                }
            }
        }


        AddSeller openAddUserDialog() {
            AddSeller dialog = new AddSeller(this);
            return dialog;
        }


        DelSeller openDelUserDialog() {
            DelSeller dialog = new DelSeller(this);
            return dialog;
```

```
 }


    AddGood openAddGoodDialog() {
        AddGood dialog = new AddGood(this);
        return dialog;
    }


    DelGood openDelGoodDialog() {
        DelGood dialog = new DelGood(this);
        return dialog;
    }

    // Удаление клиента
    public void removeClient(int id) {
        clients.remove(id);
    }
}
```

Файл Handler.java

```
package sellerserver;
import java.io.IOException;
import java.io.PrintWriter;
import java.io.UnsupportedEncodingException;
import java.net.Socket;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Properties;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;

public class Handler implements Runnable {

    // Переменные для работы с клиентом
    Server server;                  // экземпляр нашего сервера
    PrintWriter outMessage;         // исходящее сообщение
    Scanner inMessage;              // входящее собщение
    public Socket socket = null;
    boolean auth = false;
    public String host, id, name, dep;
    int number;

    // Переменные для работы с БД SQL
    String DB_URL = "jdbc:mysql://localhost:3306/seller";
    String DB_DRV = "com.mysql.jdbc.Driver";
    String DB_USER = "root";
    String DB_PASSWD = "root";
    String ENCOD = "cp1251";
    PreparedStatement pstmt = null;
Connection connection = null;
```

```
    public Handler(Socket socket, Server server, int number) {
            try {
                // Установка подключения
                this.server = server;
                this.socket = socket;
                this.outMessage = new PrintWriter(socket.getOutputStream());
                this.inMessage = new Scanner(socket.getInputStream());
                this.host = socket.getLocalAddress().toString().substring(1) +
"-" + number;
                this.number = number;
            } catch (IOException ex) {
                ex.printStackTrace();
            }
        }


        @Override
        public void run() {
            try {
                // Вход пользователя
                System.out.println("Присоединился " + host);
                sendCommand("get", "auth", "-");

                // Слушаем клиента
                while (true) {
                    if (inMessage.hasNext()) {
                        String[] msg = inMessage.nextLine().split("#");

                        if (msg[0].equals("exit")) {
                            System.out.println("Покинул сеть " + host);
                            endConnection();
                            break;
                        } else {
                            switch(msg[0]) {
                                case "get":
                                    handler_GetCommand(msg[1], msg[2]);
                                    break;
                                case "post":
                                    handler_PostCommand(msg[1], msg[2]);
                                    break;
                                default:
                                    System.out.println(host + ": " + msg[0] +
msg[1] + msg[2]);
                                    break;
                            }
                        }
                    }
                }
            } catch (Exception ex) {
                ex.printStackTrace();
            } finally {
                this.endConnection();
            }
    }
```

```
    void handler_GetCommand(String param, String data) {
            switch(param) {
                case "db":
                    try {
                        // Настройки подключения
                        Properties prop = new Properties();
                        prop.setProperty("user", "root");
                        prop.setProperty("password", "root");
                        prop.setProperty("useUnicode", "true");
                        prop.setProperty("characterEncoding", "cp1251");

                        // Подключение к БД
                        connection = DriverManager.getConnection(DB_URL, prop);
                        String sql = "SELECT * FROM goods";
                        pstmt = connection.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS);
                        pstmt.execute(sql);
                        final ResultSet res = pstmt.getResultSet();

                        while(res.next()) {
                            StringBuilder sb = new StringBuilder("");

                            // Заполнение строки
                            sb.append(res.getString(1))
                                    .append(";")
                                    .append(res.getString(2))
                                    .append(";")
                                    .append(res.getString(3))
                                    .append(";")
                                    .append(res.getString(4));

                            // При условии что строка не пуста
                            if (!sb.toString().isEmpty()) {
                                // Таблица пересылается построчно
                                // Отправка полученной записи клиенту
                                sendCommand("post", "db_row", sb.toString());
                                try {
                                    System.out.println(new
String(sb.toString().getBytes(), "cp1251"));
                                } catch (UnsupportedEncodingException ex) {

Logger.getLogger(Handler.class.getName()).log(Level.SEVERE, null, ex);
                                }
                            }

                            sb = null;
                        }
                    } catch(SQLException ex) {
                        ex.printStackTrace();
                    } finally {
                        try {
                            if (pstmt != null)
                            pstmt.close();
```

```
                    if (connection != null)
                        connection.close();
                } catch (SQLException ex) {
                    ex.printStackTrace();
                }
            }
            break;
        case "db_dep":
            try {
                // Настройки подключения
                Properties prop = new Properties();
                prop.setProperty("user", "root");
                prop.setProperty("password", "root");
                prop.setProperty("useUnicode", "true");
                prop.setProperty("characterEncoding", "cp1251");

                // Подключение к БД
                connection = DriverManager.getConnection(DB_URL, prop);
                String sql = "SELECT * FROM goods WHERE department='" + data +
"'";
                pstmt = connection.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS);
                pstmt.execute(sql);
                final ResultSet res = pstmt.getResultSet();

                while(res.next()) {
                    StringBuilder sb = new StringBuilder("");

                    // Заполнение строки
                    sb.append(res.getString(1))
                            .append(";")
                            .append(res.getString(2))
                            .append(";")
                            .append(res.getString(3))
                            .append(";")
                            .append(res.getString(4));

                    // При условии что строка не пуста
                    if (!sb.toString().isEmpty()) {
                        // Таблица пересылается построчно
                        // Отправка полученной записи клиенту
                        sendCommand("post", "db_dep_row", sb.toString());
                    }

                    sb = null;
                }
            } catch(SQLException ex) {
                ex.printStackTrace();
            } finally {
                try {
                    if (pstmt != null)
                        pstmt.close();

                    if (connection != null)
                        connection.close();
                } catch (SQLException ex) {
```

```
                    ex.printStackTrace();
                }
            }
            break;
        case "db_search":
            try {
                // Настройки подключения
                Properties prop = new Properties();
                prop.setProperty("user", "root");
                prop.setProperty("password", "root");
                prop.setProperty("useUnicode", "true");
                prop.setProperty("characterEncoding", "cp1251");

                // Подключение к БД
                connection = DriverManager.getConnection(DB_URL, prop);
                String sql = "SELECT * FROM goods WHERE name='" + data + "'";
                pstmt = connection.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS);
                pstmt.execute(sql);
                final ResultSet res = pstmt.getResultSet();

                while(res.next()) {
                    StringBuilder sb = new StringBuilder("");

                    // Заполнение строки
                    sb.append(res.getString(1))
                            .append(";")
                            .append(res.getString(2))
                            .append(";")
                            .append(res.getString(3))
                            .append(";")
                            .append(res.getString(4));

                    // При условии что строка не пуста
                    if (!sb.toString().isEmpty()) {
                        // Таблица пересылается построчно
                        // Отправка полученной записи клиенту
                        sendCommand("post", "db_search_row", sb.toString());
                    }

                    sb = null;
                }
            } catch(SQLException ex) {
                ex.printStackTrace();
            } finally {
                try {
                    if (pstmt != null)
                        pstmt.close();

                    if (connection != null)
                        connection.close();
                } catch (SQLException ex) {
                    ex.printStackTrace();
                }
```

```
                }
                break;
            default:
                // Нераспознанное входящее сообщение
                System.err.println("GET: "
                        .concat(param)
                        .concat("|")
                        .concat(data));
                break;
        }
    }


    void handler_PostCommand(String param, String data) {
        switch(param) {
            case "auth":
                String[] auth = data.split(";");
                setAuthorization(auth[0], auth[1]);
                break;
            case "sale":
                try {
                    connection =
DriverManager.getConnection(DB_URL,DB_USER,DB_PASSWD);
                    String[] count_id = data.split(";");
                    String sql = "UPDATE goods SET count='" + count_id[0] +
                            "' WHERE id=" + count_id[1] + "";
                    pstmt = connection.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS);
                    pstmt.executeUpdate();
                } catch(SQLException ex) {
                    ex.printStackTrace();
                } finally {
                    try {
                        if (pstmt != null)
                            pstmt.close();

                        if (connection != null)
                            connection.close();
                    } catch (SQLException ex) {
                        ex.printStackTrace();
                    }
                }
                break;
            default:
                // Нераспознанное входящее сообщение
                System.err.println("POST: "
                        .concat(param)
                        .concat("|")
                        .concat(data));
                break;
        }
    }


    void setAuthorization(String login, String password) {
        try {
```

```
            // Настройки подключения
            Properties prop = new Properties();
            prop.setProperty("user", "root");
            prop.setProperty("password", "root");
            prop.setProperty("useUnicode", "true");
            prop.setProperty("characterEncoding", "cp1251");

            // Подключение к БД
            connection = DriverManager.getConnection(DB_URL, prop);
            String SEE = "SELECT password FROM sellers WHERE login IN ('" + login +
"')";
            pstmt = connection.prepareStatement(SEE,
Statement.RETURN_GENERATED_KEYS);
            ResultSet rs = pstmt.executeQuery(SEE);

            if (rs.next() && rs.getString("password").equals(password)) {
                auth = true;
                sendCommand("post", "auth", "true");
                System.out.println(">>> Вход: " + login + "|" + password);

                // Установка информации о клиенте в таблицу клиентов
                String DATA = "SELECT id, name, department FROM sellers WHERE
login='" + login + "'";
                pstmt = connection.prepareStatement(DATA,
Statement.RETURN_GENERATED_KEYS);
                pstmt.execute(DATA);
                final ResultSet rsData = pstmt.getResultSet();
                rsData.next();
                id = rsData.getString(1);
                name = rsData.getString(2);
                dep = rsData.getString(3);

                sendCommand("post", "info", id.concat(";")
                        .concat(name).concat(";")
                        .concat(dep));

                server.setClientData(host, id, name, dep);
                connection.close();
                pstmt.close();
            } else {
                sendCommand("post", "auth", "false");
                System.err.println("Попытка входа: " + login + "|" + password);
                endConnection();
            }

            connection.close();
            pstmt.close();
        } catch(SQLException ex) {
            ex.printStackTrace();
        }
    }


    public void sendCommand(String type, String param, String data) {
        try {
```

```
                String message = type + "#" + param + "#" + data;
                outMessage.println(message);
                outMessage.flush();
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        }


        // клиент выходит из чата
        public void endConnection() {
            for (int i = 0; i < server.clients.size(); i++) {
                System.out.println("<<< Close handler with number: " +
this.number);
                if (server.clients.get(i).number == this.number) {
                    server.removeClient(i);
                    break;
                }
            }
        }

    }
```

Файл DelSeller.java

```
package sellerserver;
import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Properties;
import javax.swing.JButton;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;

public class DelSeller extends javax.swing.JDialog {
    static final String DB_URL = "jdbc:mysql://localhost:3306/seller";
    static final String DB_DRV = "com.mysql.jdbc.Driver";
    static final String DB_USER = "root";
    static final String DB_PASSWD = "root";

    JTable table;
    static DefaultTableModel model;
    JButton bDelUser;

    public DelSeller(java.awt.Frame parent) {
        super(parent, true);
       setSize(600,400);
```

```
        this.setLocation(parent.getLocation().x + 100, parent.getLocation().y +
100);

            createGUI();
            connectDB();
        }

        final void createGUI() {
            setTitle("Удаление персонала");
            GridLayout grid = new GridLayout(1,6);
            grid.setHgap(10);
            grid.setVgap(5);

            // центральная панель
            String[] columnNames = {"ID", "ИМЯ", "РАЗДЕЛ", "ЛОГИН", "ПАРОЛЬ"};
            model = new DefaultTableModel(0, columnNames.length);
            model.setColumnIdentifiers(columnNames);
            table = new JTable(model) {
                @Override
                public boolean isCellEditable(int arg0, int arg1) {
                    return false; // Запрет на редактирование таблицы
                }
            };
            table.getColumnModel().getColumn(0).setPreferredWidth(30);
            table.getColumnModel().getColumn(1).setPreferredWidth(100);
            add(new JScrollPane(table), BorderLayout.CENTER);

            // нижняя панель
            JPanel botPanel = new JPanel(new GridLayout(1,1));
            bDelUser = new JButton("Удалить");
            botPanel.add(bDelUser);
            add(botPanel, BorderLayout.SOUTH);
            bDelUser.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    int num_row = table.getSelectedRow();

                    if (num_row != -1) {
                        int id = Integer.parseInt(table.getValueAt(num_row,
0).toString());
                        deleteUser(id, num_row);
                    }
                }
            });
        }

        static void connectDB() {
            Connection connection = null;
            PreparedStatement pstmt = null;

            try {
                // Настройки подключения
                Properties prop = new Properties();
                prop.setProperty("user", "root");
                prop.setProperty("password", "root");
               prop.setProperty("useUnicode", "true");
```

```
                    prop.setProperty("characterEncoding", "cp1251");

                    // Подключение к БД
                    connection = DriverManager.getConnection(DB_URL, prop);
                    String sql = "SELECT * FROM sellers";
                    pstmt = connection.prepareStatement(sql,
    Statement.RETURN_GENERATED_KEYS);
                    pstmt.execute(sql);
                    final ResultSet rec = pstmt.getResultSet();

                    while(rec.next()) {
                        model.addRow(new Object[] {
                            rec.getString(1),
                            rec.getString(2),
                            rec.getString(3),
                            rec.getString(4),
                            rec.getString(5)});
                    }
                } catch(SQLException ex) {
                    ex.printStackTrace();
                } finally {
                    try {
                        if (pstmt != null) pstmt.close();
                        if (connection != null) connection.close();
                    } catch (SQLException ex) {

                    }
                }
            }

            void deleteUser(int id, int num_row) {
                try {
                    String sql = "DELETE FROM sellers WHERE id=" + id;
                    // Настройки подключения
                    Properties prop = new Properties();
                    prop.setProperty("user", "root");
                    prop.setProperty("password", "root");
                    prop.setProperty("useUnicode", "true");
                    prop.setProperty("characterEncoding", "cp1251");

                    // Подключение к БД
                    Connection connection = DriverManager.getConnection(DB_URL,
    prop);
                    PreparedStatement pstmt = connection.prepareStatement(sql,
    Statement.RETURN_GENERATED_KEYS);
                    pstmt.executeUpdate(sql);

                    model.removeRow(num_row);
                    if (pstmt != null) pstmt.close();
                    if (connection != null) connection.close();
                } catch(SQLException ex) {
                    ex.printStackTrace();
                }
            }
}
```

Файл DelGood.java

```
package sellerserver;
import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Properties;
import javax.swing.JButton;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;

public class DelGood extends javax.swing.JDialog {
    static final String DB_URL = "jdbc:mysql://localhost:3306/seller";
    static final String DB_DRV = "com.mysql.jdbc.Driver";
    static final String DB_USER = "root";
    static final String DB_PASSWD = "root";

    JTable table;
    static DefaultTableModel model;
    JButton bDelUser;

    public DelGood(java.awt.Frame parent) {
        super(parent, true);
        setSize(450,300);
        this.setLocation(parent.getLocation().x + 100,
parent.getLocation().y + 100);

        createGUI();
        connectDB();
    }

    final void createGUI() {
        setTitle("Удаление товара");

        // центральная панель
        String[] columnNames = {"ID", "НАЗВАНИЕ", "РАЗДЕЛ", "КОЛ-ВО"};
        model = new DefaultTableModel(0, columnNames.length);
        model.setColumnIdentifiers(columnNames);

        table = new JTable(model) {
            @Override
            public boolean isCellEditable(int arg0, int arg1) {
                return false; // Запрет на редактирование таблицы
            }
        };
        table.getColumnModel().getColumn(0).setPreferredWidth(30);
        table.getColumnModel().getColumn(1).setPreferredWidth(100);
    add(new JScrollPane(table), BorderLayout.CENTER);
```

```
        // нижняя панель
        JPanel botPanel = new JPanel(new GridLayout(1,1));
        bDelUser = new JButton("Удалить");
        botPanel.add(bDelUser);
        add(botPanel, BorderLayout.SOUTH);

        bDelUser.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                int num_row = table.getSelectedRow();

                // Если строка выбрана
                if (num_row != -1) {
                    int id = Integer.parseInt(table.getValueAt(num_row,
0).toString());
                    deleteGood(id, num_row);
                }
            }
        });
    }

    static void connectDB() {
        Connection connection = null;
        PreparedStatement pstmt = null;

        try {
            // Настройки подключения
            Properties prop = new Properties();
            prop.setProperty("user", "root");
            prop.setProperty("password", "root");
            prop.setProperty("useUnicode", "true");
            prop.setProperty("characterEncoding", "cp1251");

            // Подключение к БД
            connection = DriverManager.getConnection(DB_URL, prop);
            String sql = "SELECT * FROM goods";
            pstmt = connection.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS);
            pstmt.execute(sql);
            final ResultSet rec = pstmt.getResultSet();

            while(rec.next()) {
                model.addRow(new Object[] {
                    rec.getString(1),
                    rec.getString(2),
                    rec.getString(3),
                    rec.getString(4)});
            }
        } catch(SQLException ex) {
            ex.printStackTrace();
        } finally {
            try {
                if (pstmt != null) pstmt.close();
                if (connection != null) connection.close();
            } catch (SQLException ex) {
```

```
                }
            }
        }

        void deleteGood(int id, int num_row) {
            try {
                String sql = "DELETE FROM goods WHERE id=" + id;
                Connection connection = null;
                // Настройки подключения
                Properties prop = new Properties();
                prop.setProperty("user", "root");
                prop.setProperty("password", "root");
                prop.setProperty("useUnicode", "true");
                prop.setProperty("characterEncoding", "cp1251");

                // Подключение к БД
                connection = DriverManager.getConnection(DB_URL, prop);
                PreparedStatement pstmt = connection.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS);
                pstmt.executeUpdate(sql);

                model.removeRow(num_row);
                if (pstmt != null) pstmt.close();
                if (connection != null) connection.close();
            } catch(SQLException ex) {
                ex.printStackTrace();
            }
        }
    }
```

Файл AddSeller.java

```
    package sellerserver;
    import java.awt.BorderLayout;
    import java.awt.GridLayout;
    import java.awt.Insets;
    import java.awt.event.ActionEvent;
    import java.awt.event.ActionListener;
    import java.sql.Connection;
    import java.sql.DriverManager;
    import java.sql.PreparedStatement;
    import java.sql.ResultSet;
    import java.sql.SQLException;
    import java.sql.Statement;
    import java.util.Properties;
    import javax.swing.BorderFactory;
    import javax.swing.JButton;
    import javax.swing.JLabel;
    import javax.swing.JOptionPane;
    import javax.swing.JPanel;
    import javax.swing.JTextField;

    public class AddSeller extends javax.swing.JDialog {
        static final String DB_URL = "jdbc:mysql://localhost:3306/seller";
        static final String DB_DRV = "com.mysql.jdbc.Driver";
static final String DB_USER = "root";
```

```
    static final String DB_PASSWD = "root";

        static JTextField tName, tDep, tLog, tPas;
        JButton but;

        public AddSeller(java.awt.Frame parent) {
            super(parent, true);
            setSize(300,270);
            this.setLocation(parent.getLocation().x + 100,
parent.getLocation().y + 100);
            createGUI();
        }

        void createGUI() {
            setTitle("Добавление персонала");
            JPanel panel = new JPanel(new BorderLayout());
            panel.setBorder(BorderFactory.createTitledBorder("Основная
информация"));
            GridLayout grid = new GridLayout(6,1);
            grid.setHgap(5);
            grid.setVgap(5);

            JPanel pNorth = new JPanel(grid);
            JLabel lName = new JLabel("ИМЯ");
            pNorth.add(lName);
            JLabel lDep = new JLabel("РАЗДЕЛ");
            pNorth.add(lDep);
            JLabel lLog = new JLabel("ЛОГИН");
            pNorth.add(lLog);
            JLabel lPas = new JLabel("ПАРОЛЬ");
            pNorth.add(lPas);
            panel.add(pNorth, BorderLayout.WEST);

            JPanel pCenter = new JPanel(grid);
            tName = new JTextField("");
            tName.setMargin(new Insets(3, 3, 3, 3));
            pCenter.add(tName);
            tDep = new JTextField("");
            tDep.setMargin(new Insets(3, 3, 3, 3));
            pCenter.add(tDep);
            tLog = new JTextField("");
            tLog.setMargin(new Insets(3, 3, 3, 3));
            pCenter.add(tLog);
            tPas = new JTextField("");
            tPas.setMargin(new Insets(3, 3, 3, 3));
            pCenter.add(tPas);
            panel.add(pCenter, BorderLayout.CENTER);

            JPanel pSouth = new JPanel();
            but = new JButton("Добавить");
            pSouth.add(but);
            but.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    if (tLog.getText().length() != 0) {
                        char ch = tLog.getText().toCharArray()[0];
```

```
                        if (Character.isLetter(ch)) {
                            connectDB();
                        } else
                            JOptionPane.showMessageDialog(null,
                                "Логин должен начинаться с буквы!");
                    }
                }
            });
            panel.add(pSouth, BorderLayout.SOUTH);
            add(panel);
        }

        static void connectDB() {
            Connection connection = null;
            PreparedStatement pstmt = null;

            try {
                // Настройки подключения
                Properties prop = new Properties();
                prop.setProperty("user", "root");
                prop.setProperty("password", "root");
                prop.setProperty("useUnicode", "true");
                prop.setProperty("characterEncoding", "cp1251");

                // Подключение к БД
                connection = DriverManager.getConnection(DB_URL, prop);

                // Проверка на отсутствие аналогичного логина в БД
                String LOG = "SELECT login FROM sellers WHERE login='" +
tLog.getText() + "'";
                pstmt = connection.prepareStatement(LOG,
Statement.RETURN_GENERATED_KEYS);
                ResultSet rsLog = pstmt.executeQuery(LOG);
                if (rsLog.next()) {
                    JOptionPane.showMessageDialog(null, "Данный логин уже
занят.");
                } else if (tLog.getText().equals("users")) {
                    JOptionPane.showMessageDialog(null, "Ндопутимое значение в
поле логина.");
                } else {
                    // Генерация id для нового клиента
                    int new_id = 0;
                    boolean isGen = false;
                    while (!isGen) {
                        new_id = 1000 + (int)(Math.random()*9000);
                        String SEE = "SELECT id FROM sellers WHERE id=" +
new_id;
                        pstmt = connection.prepareStatement(SEE,
Statement.RETURN_GENERATED_KEYS);
                        ResultSet rs = pstmt.executeQuery(SEE);

                        boolean isFind = true;
                        while (rs.next())
                            isFind = false;

                        if (isFind)

                                                isGen = true;
```

```
                    }

                    // Добавление нового клиента в БД
                    String ADD = "INSERT INTO
sellers(id,name,department,login,password) VALUES(?,?,?,?,?)";
                    pstmt = connection.prepareStatement(ADD,
Statement.RETURN_GENERATED_KEYS);
                    String name = tName.getText(),
                            dep = tDep.getText(),
                            log = tLog.getText(),
                            pas = tPas.getText();

                    if (name.length() != 0 &&
                        dep.length() != 0 &&
                        log.length() != 0 &&
                        pas.length() != 0) {
                        pstmt.setInt(1, new_id);
                        pstmt.setString(2, name);
                        pstmt.setString(3, dep);
                        pstmt.setString(4, log);
                        pstmt.setString(5, pas);
                        pstmt.executeUpdate();
                    }

                    clearInput();
                }
            } catch(SQLException ex) {
                ex.printStackTrace();
            } finally {
                try {
                    if (pstmt != null) pstmt.close();
                    if (connection != null) connection.close();
                } catch (SQLException ex) {

                }
            }
        }

        static void clearInput() {
            tName.setText("");
            tDep.setText("");
            tLog.setText("");
            tPas.setText("");
        }
    }
```

Файл AddGood.java

```
package sellerserver;
import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.DriverManager;

                    import java.sql.PreparedStatement;
```

```
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Properties;
import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class AddGood extends javax.swing.JDialog {
    static final String DB_URL = "jdbc:mysql://localhost:3306/seller";
    static final String DB_DRV = "com.mysql.jdbc.Driver";
    static final String DB_USER = "root";
    static final String DB_PASSWD = "root";

    static JTextField tName, tDep, tCount;
    JButton but;

    public AddGood(java.awt.Frame parent) {
        super(parent, true);
        setSize(300,210);
        this.setLocation(parent.getLocation().x + 100,
parent.getLocation().y + 100);
        createGUI();
    }

    void createGUI() {
        setTitle("Добавление товара");
        JPanel panel = new JPanel(new BorderLayout());
        panel.setBorder(BorderFactory.createTitledBorder("Основная
информация"));

        GridLayout grid = new GridLayout(4,1);
        grid.setHgap(5);
        grid.setVgap(5);

        JPanel pNorth = new JPanel(grid);
        JLabel lName = new JLabel("НАЗВАНИЕ");
        pNorth.add(lName);
        JLabel lDep = new JLabel("РАЗДЕЛ");
        pNorth.add(lDep);
        JLabel lCount = new JLabel("КОЛ-ВО");
        pNorth.add(lCount);
        panel.add(pNorth, BorderLayout.WEST);

        JPanel pCenter = new JPanel(grid);
        tName = new JTextField("");
        tName.setMargin(new Insets(3, 3, 3, 3));
        pCenter.add(tName);
        tDep = new JTextField("");
        tDep.setMargin(new Insets(3, 3, 3, 3));
        pCenter.add(tDep);
        tCount = new JTextField("");

              tCount.setMargin(new Insets(3, 3, 3, 3));
```

```
        pCenter.add(tCount);
                panel.add(pCenter, BorderLayout.CENTER);

                JPanel pSouth = new JPanel();
                but = new JButton("Добавить");
                pSouth.add(but);
                but.addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        if (tName.getText().length() < 40 &&
                                tDep.getText().length() < 40 &&
                                tCount.getText().length() < 10) {
                            connectDB();
                        } else
                            JOptionPane.showMessageDialog(null,
                                    "Название и раздел должны быть короче 40
символов. "
                                            + "\nКоличество не длиньше 10
символов!");
                    }
                });
                panel.add(pSouth, BorderLayout.SOUTH);
                add(panel);
            }

            static void connectDB() {
                Connection connection = null;
                PreparedStatement pstmt = null;

                try {
                    // Настройки подключения
                    Properties prop = new Properties();
                    prop.setProperty("user", "root");
                    prop.setProperty("password", "root");
                    prop.setProperty("useUnicode", "true");
                    prop.setProperty("characterEncoding", "cp1251");

                    // Подключение к БД
                    connection = DriverManager.getConnection(DB_URL, prop);

                    // Проверка на отсутствие аналогичного логина в БД
                    String LOG = "SELECT id,count FROM goods WHERE name='" +
tName.getText() + "'";
                    pstmt = connection.prepareStatement(LOG,
Statement.RETURN_GENERATED_KEYS);
                    ResultSet rsLog = pstmt.executeQuery(LOG);

                    // 0 - отмена записи, 1 - обновление значения, -1 - новый товар
                    int confirm = -1;

                    if (rsLog.next()) {
                        // Если указанный товар уже имеется в БД
                        confirm = JOptionPane.showConfirmDialog(null,
                                "Данный товар уже существует."

                                    + "\nДобавить количество к имеющемуся товару?",
```

```
                                "Сообщение от БД",
                                JOptionPane.YES_NO_OPTION);

                        // Добавить количество введённого товара к существующему
                        if (confirm == 0) {
                            System.out.println("Добавление количества к товару: " +
tName.getText());
                            // Текущее количество товара
                            int cur_count = Integer.valueOf(rsLog.getString(2));
                            // Добавляемое количество товара
                            int add_count = Integer.valueOf(tCount.getText());
                            // Новое количество товара в БД
                            int count = cur_count + add_count;

                            String sql = "UPDATE goods SET count='" + count
                                    + "' WHERE id=" + rsLog.getString(1);
                            pstmt = connection.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS);
                            pstmt.executeUpdate();
                        } else
                            System.out.println("Отмена добавления. Код: " +
confirm);
                    }

                    if (confirm == -1) {
                        // Генерация id для нового товара
                        int new_id = 0;
                        boolean isGen = false;

                        while (!isGen) {
                            new_id = 1000 + (int)(Math.random()*9000);
                            String SEE = "SELECT id FROM goods WHERE id=" + new_id;
                            pstmt = connection.prepareStatement(SEE,
Statement.RETURN_GENERATED_KEYS);
                            ResultSet rs = pstmt.executeQuery(SEE);

                            boolean isFind = true;
                            while (rs.next())
                                isFind = false;

                            if (isFind)
                                isGen = true;
                        }

                        // Добавление нового клиента в БД
                        String ADD = "INSERT INTO goods(id,name,department,count)
VALUES(?,?,?,?)";
                        pstmt = connection.prepareStatement(ADD,
Statement.RETURN_GENERATED_KEYS);
                        String name = tName.getText(),
                                dep = tDep.getText(),
                                count = tCount.getText();

                        if (name.length() != 0 && dep.length() != 0) {
                            pstmt.setInt(1, new_id);
                            pstmt.setString(2, name);

                                        pstmt.setString(3, dep);
```

```
        pstmt.setString(4, count);
                            pstmt.executeUpdate();
                        }

                        clearInput();
                    }
                } catch(SQLException ex) {
                    ex.printStackTrace();
                    JOptionPane.showMessageDialog(null, "Введите корректные
данные!");
                } finally {
                    try {
                        if (pstmt != null) pstmt.close();
                        if (connection != null) connection.close();
                    } catch (SQLException ex) {

                    }
                }
            }

            static void clearInput() {
                tName.setText("");
                tDep.setText("");
                tCount.setText("");
            }
        }
```

ПРИЛОЖЕНИЕ В

(Презентация)

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ
КАФЕДРА «ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА»

Клиент-серверное приложение для организации работы персонала в торговой сети

Выполнил:
студент группы 16ВВ1
Мухаметов Т.Ш.
Руководитель:
к.т.н Карамышева Н.С.

Пенза 2020

Актуальность работы

- Технологии интернет жизни проникли во все сферы жизни, в том числе в экономическую. В связи с этим, для сохранения положения при котором компания способна конкурировать, ни одна торговая фирма не может обойтись без применения вспомогательных технологий. Из-за этого рост подобных технологических решений и спрос на подобное программное обеспечение растёт.

Цели и задачи

- Цель работы – создание клиент-серверного приложения для организации работы персонала в торговой сети, занимающейся непосредственной продажей товаров.
- Задачи:
  - Изучить предметную область;
  - Проанализировать предъявляемые требования;
  - Выбрать средства и технологии для разработки;
  - Разработать программные модули;
  - Протестировать и отладить программу.

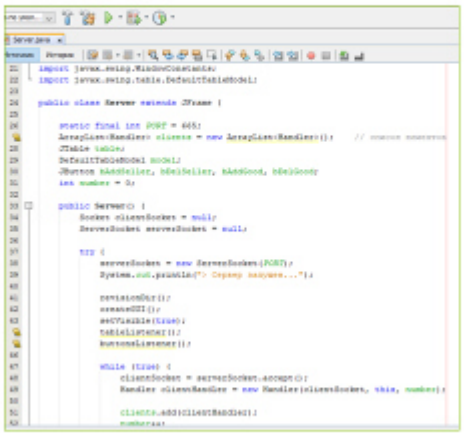
Анализ требований к системе

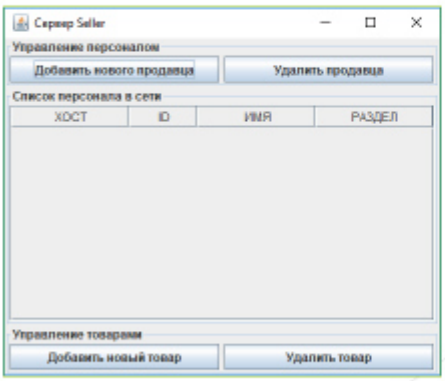
- Ведение списка товаров
- Продажа товара
- Авторизация в системе
- Хранение данных в БД

Выбор языков, средств и технологий

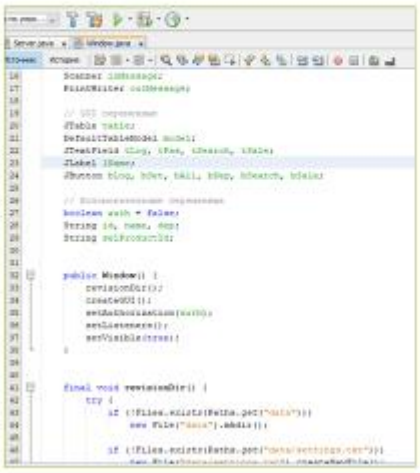
- Языки
  - Java
  - SQL
- Средства и технологии
  - NetBeans IDE
  - MySQL Command Line
  - Maven
  - Swing
  - Server Socket
  - JDBC

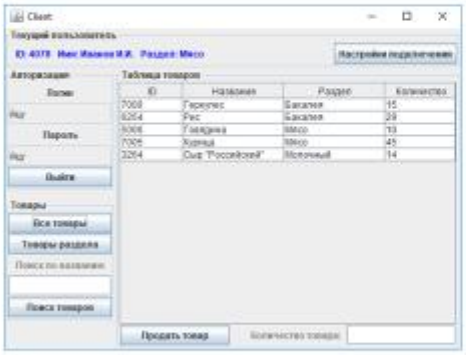
Разработка серверной части





Разработка клиентской части





Тестирование

- Тестирование отдельных модулей
- Комплексное тестирование взаимодействия модулей
- Исправление обнаруженных ошибок
- Повторная отладка

Выводы

- В результате проделанной работы:
  - Разработано и отлажено клиент-серверное приложение
  - Составлены и реализованы алгоритмы работы торговой системы
  - Оформлена соответствующая документация

Спасибо за внимание.