Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет инженерно-экономический

Кафедра экономической информатики

Дисциплина «Распределённые системы обработки информации»

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

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту

на тему:

**«Автоматизированная система расчёта прибыли от продаж на примере салона мебели»**

|  |  |
| --- | --- |
|  | Выполнил студент группы 074002  Гайдук Марина Алексеевна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовой проект представлен на проверку \_\_\_.\_\_\_\_.2023  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2023

**РЕФЕРАТ**

**Гайдук, М. А.** Автоматизированная система расчёта прибыли от продаж на примере салона мебели / М. А. Гайдук. – Минск: БГУИР, 2023 – 76 с.

Пояснительная записка 59 с., 50 рис., 10 источников, 3 приложения.

*Ключевые слова:* система, автоматизированная, расчёт, прибыль, мебель, магазин, салон, разработка, информационная, модель, моделирование, диаграмма, анализ, тестирование, пользователь, ассистент, администратор.

*Объект исследования:* салон мебели.

*Предмет исследования:* способы и пути повышения качества расчёта прибыли от продаж на примере салона мебели.

*Цель курсового проекта:* совершенствование расчёта прибыли от продаж путём создания автоматизированной системы анализа продаж.

*Методология проведения работы:* в процессе разработки автоматизированной системы использованы методы анализа, систематизации, классификации, обобщения данных, качественно-количественные методы обработки данных, функциональный анализ процессов, принципа построения баз данных, моделирование системы с помощью UML-диаграмм.

*Результаты работы:* проведен теоретический анализ, а также моделирование деятельности по осуществлению расчёта прибыли от продаж. Выполнено проектирование и конструирование автоматизированной системы расчёта прибыли от продаж. Осуществлено тестирование и проверка работоспособности системы. Предоставлена инструкция по развертыванию автоматизированной системы, а также руководство пользователя.

*Область применения результатов:* магазин или предприятие, где необходимо рассчитывать прибыль от продаж, а также проводить анализ продаж.

СОДЕРЖАНИЕ

[Перечень условных обозначений, символов и терминов 5](#_Toc134540865)

[Введение 6](#_Toc134540866)

[1 Анализ и моделирование предметной области программного средства 8](#_Toc134540867)

[1.1 Описание предметной области 8](#_Toc134540868)

[1.2. Разработка функциональной модели предметной области 10](#_Toc134540869)

[1.3. Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований 15](#_Toc134540870)

[1.4. Разработка информационной модели предметной области 19](#_Toc134540871)

[1.5. UML-модели представления программного средства и их описание 22](#_Toc134540872)

[2 Проектирование и конструирование программного средства 29](#_Toc134540873)

[2.1. Постановка задачи 29](#_Toc134540874)

[2.2. Обоснование выбора компонентов и технологий для реализации программного средства 29](#_Toc134540875)

[2.3. Архитектурные решения 31](#_Toc134540876)

[2.4. Описание алгоритмов, реализующих ключевую бизнес-логику разрабатываемого программного средства 32](#_Toc134540877)

[2.5. Проектирование пользовательского интерфейса 35](#_Toc134540878)

[2.6. Методы и средства, используемые для обеспечения безопасности данных 37](#_Toc134540879)

[3 Тестирование и проверка работоспособности программного средства 38](#_Toc134540880)

[4 Руководство по развёртыванию и ипользованию программного средства 43](#_Toc134540881)

[4.1 Руководство по установке (развёртыванию) программного средства 43](#_Toc134540882)

[4.2 Руководство пользователя 43](#_Toc134540883)

[Заключение 59](#_Toc134540884)

[Список использованных источников 60](#_Toc134540885)

[Приложение А 61](#_Toc134540886)

[Приложение Б 62](#_Toc134540887)

[Приложение В 68](#_Toc134540888)

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ, СИМВОЛОВ И ТЕРМИНОВ

СУБД – Система управления базами данных.

SQL – *Structured Query Language –* декларативный язык взаимодействия с реляционной базой данных.

UML – *Unified Modeling Language –* графический язык визуализации, специфицирования, конструирования систем.

IDEF0 – *Integrated DEFinition 0* – методология функционального моделирования.

IDEF1X – *Integrated DEFinition 1 Extended* – методология моделирования описания данных.

JDBC – *Java DataBase Connectivity ­–* платформенно независимый промышленный стандарт взаимодействия Java-приложений с различными СУБД.

DAO – *Data Access Object* ­– абстрактный интерфейс к какому-либо типу базы данных или механизму хранения.

FTP – *File Transfer Protocol* – протокол передачи файлов в сети.

HTTP – *Hypertext Transfer Protocol* – протокол передачи данных в сети.

MVC – *Model View Controller* – архитектурный шаблон построения приложений, при котором логика разделена на модель, представление и контроллер.

API – *Application Programming Interface* – программный интерфейс приложения.

XML – *Extensible Markup Language* – язык разметки пользовательского интерфейса.

BLOB – *Binary Large Object –* специальный тип данных, предназначенный, в первую очередь, для хранения изображений, а также компилированного программного кода.

# ВВЕДЕНИЕ

Стремительное развитие компьютерной техники и роста объема данных привело к тому, что все большее распространение стали получать информационные системы. Автоматизированные информационные системы представляют собой совокупность различных средств, предназначенных для сбора, подготовки, хранения, обработки, поиска и предоставления информации. Основной целью автоматизации отраслей является повышение качества исполнения процесса, улучшение качества информационной продукции и услуг, а также повышение оперативности обслуживания пользователей.

Создается все больше автоматизированных приложений для продажи различных видов продукции. Использование автоматизированных информационных систем для реализации приложений, направленных на продажу определенных видов продукции, позволяет усовершенствовать этот процесс, сделать его автономным, быстрым, а также с минимальным количеством ошибок, которые могли бы возникнуть из-за человеческого фактора.

Автоматизированная система учета продаж мебельной продукции и расчета прибыли с одной стороны дает возможность самостоятельно осуществлять заказы, а также следить за их обработкой, с другой стороны вести учет продаж мебельной продукции, контроль заказов, а также получать сведения о прибыли за различный промежуток времени.

Целью данного курсового проекта является совершенствование расчёта прибыли от продаж путём создания автоматизированной системы анализа продаж.

Задачи, которые необходимо выполнить в ходе выполнения курсового проекта:

–исследовать механизм работы системы продажи мебельных товаров;

–ознакомиться с уже существующими программами для реализации системы продажи мебельной продукции;

–на основе аналогов проанализировать слабые стороны данного механизма;

–разработать функциональную модель основного процесса;

–провести тестирование разработанного программного продукта;

–выполнить отладку ошибок, обнаруженных в процессе.

Объектом исследования является салон мебели.

# 1 АНАЛИЗ И МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ ПРОГРАММНОГО СРЕДСТВА

## **1.1 Описание предметной области**

Предметной областью, рассматриваемой в рамках данного курсового проекта, является автоматизация расчёта прибыли от продаж на примере мебельного магазина.

Её цель состоит в улучшении качества и скорости обслуживания клиентов, упрощении ведения учёта и продажи товара, а также в возможности расчёта и получения информации о прибыли от продаж за выбранный промежуток времени.

Расчёт прибыли от продаж представляет собой процесс определения чистой денежной выгоды, которую компания получает от продажи своих товаров или услуг. Этот расчёт является ключевым инструментом для оценки финансовой эффективности бизнеса и помогает предпринимателям принимать обоснованные решения по управлению.

Для расчёта прибыли от продаж необходимо учесть следующие финансовые показатели:

* выручка – представляет собой общую сумму денег, которую компания получила от продаж своих товаров или услуг за определенный период.
* себестоимость – представляет собой сумму затрат, которые связаны с производством и продажей товаров или услуг. Включает затраты на материалы, оплату труда, аренду помещения и т.д.
* валовая прибыль – это разница между выручкой и себестоимостью. Показывает сколько компания заработала на продаже своих товаров или услуг, не учитывая операционные расходы.
* операционные расходы – представляют собой затраты, которые связаны с основной деятельностью компании, например, налоги, оплату труда, аренду помещения, рекламу и т.д.
* чистая прибыль – это разница между валовой прибылью и операционными расходами. Она показывает сколько компания заработала после вычета всех затрат на производство и продажу товаров или услуг.

Расчет прибыли от продаж является важным инструментом управления бизнесом, позволяющим компаниям принимать обоснованные решения, управлять своей деятельностью, оптимизировать затраты и повышать операционную эффективность.

Расчет прибыли от продаж в магазине – это процесс определения финансовой прибыли, которую магазин получает от продажи товаров.

Автоматизированный расчет прибыли от продаж в мебельных магазинах – это процесс, в котором используются специальные программы и инструменты для сбора, анализа и обработки финансовых данных, связанных с продажами мебели.

Основной целью автоматизированного расчета прибыли от продаж является сокращение времени и усилий, необходимых для сбора и обработки данных, а также минимизация возможных ошибок, связанных с ручным вводом данных.

Автоматизированный расчет прибыли от продаж позволяет компаниям быстро получать данные о выручке, себестоимости и других финансовых показателях, а также получить детализированную информацию о продажах каждого товара и анализировать тенденции.

Автоматизированный расчет прибыли от продаж является важным инструментом управления для компаний, позволяющим быстро получать финансовые отчеты, анализировать показатели продаж и принимать решения на основе актуальных данных.

В данной предметной области есть три основных участника: пользователи, ассистенты и администраторы.

Пользователи ищут и выбирают различные товары и оформляют заказы на основе собранной корзины. Ассистенты обрабатывают заказы и работают с продуктами. Администраторы имеют самые приоритетные права, такие как создание учетных записей ассистентов, удаление учетных записей и просмотр информации о доходах за определенный период времени.

Для учета мебельной продукции используется отдельная таблица базы данных с подробной информацией по каждому конкретному виду мебельной продукции. Получая данные из этой базы, клиенты просматривают информацию о мебельных товарах, помещают их в корзину и оформляют заказ.

Для управления предметами мебели, помещенными в корзину, и заказами каждому пользователю предоставляется доступ к специальной таблице базы данных, которая содержит информацию о предметах в корзине и персональных заказах. Находясь в корзине, пользователь может указать количество каждого предмета, который необходимо заказать.

Существующие приложения предлагают примерно одинаковую функциональность с точки зрения пользователя. Они предоставляют пользователю возможность выбирать мебельную продукцию по категориям, искать и сортировать продукцию, узнавать дополнительную информацию о конкретном товаре и добавлять товары в корзину. В корзине пользователям предлагается возможность установить количество товаров, возможность удалить товары, а также оформить заказ. Пользователи также имеют возможность отслеживать статус действующих заказов и просматривать историю предыдущих заказов. Кроме того, существующие программы предоставляют возможность создания учетной записи, упрощая формирование и отслеживание заказов.

Таким образом, можно сделать вывод, что для того, чтобы новая система могла конкурировать на рынке, она не должна уступать уже действующим системам или более специализированной. Для этого необходимы развитые системы поиска и фильтрации, простота обработки заказов, доступ к истории заказов, возможность просмотра и изменения личной информации.

## **1.2. Разработка функциональной модели предметной области**

Для создания функциональной модели был выбран стандарт IDEF0.

IDEF0—это нотация графического моделирования, используемая для создания функциональных моделей, которые представляют структуру и функции систем, а также потоки информации и материальные объекты, которые связывают эти функции.

Четыре стороны блока имеют разные роли: левая сторона – «вход», правая – «выход», верхняя – «управление», нижняя – «механизм».

Методология IDEF0 – один из популярных подходов к описанию бизнес-процессов. Его особенности:

* использование контекстной диаграммы;
* поддержка декомпозиции;
* доминирование;
* выделение 4 типов стрелок.

Контекстная диаграмма. Самая верхняя диаграмма, на которой объект моделирования представлен единственным блоком с граничными стрелками. Эта диаграмма называется A-0. Стрелки на этой диаграмме отображают связи объекта моделирования с окружающей средой. Диаграмма A-0 устанавливает область моделирования и ее границу. На рисунке 1.1 представлена контекстная диаграмма верхнего уровня, представляющая функциональную модель «Покупка товаров в мебельном магазине».

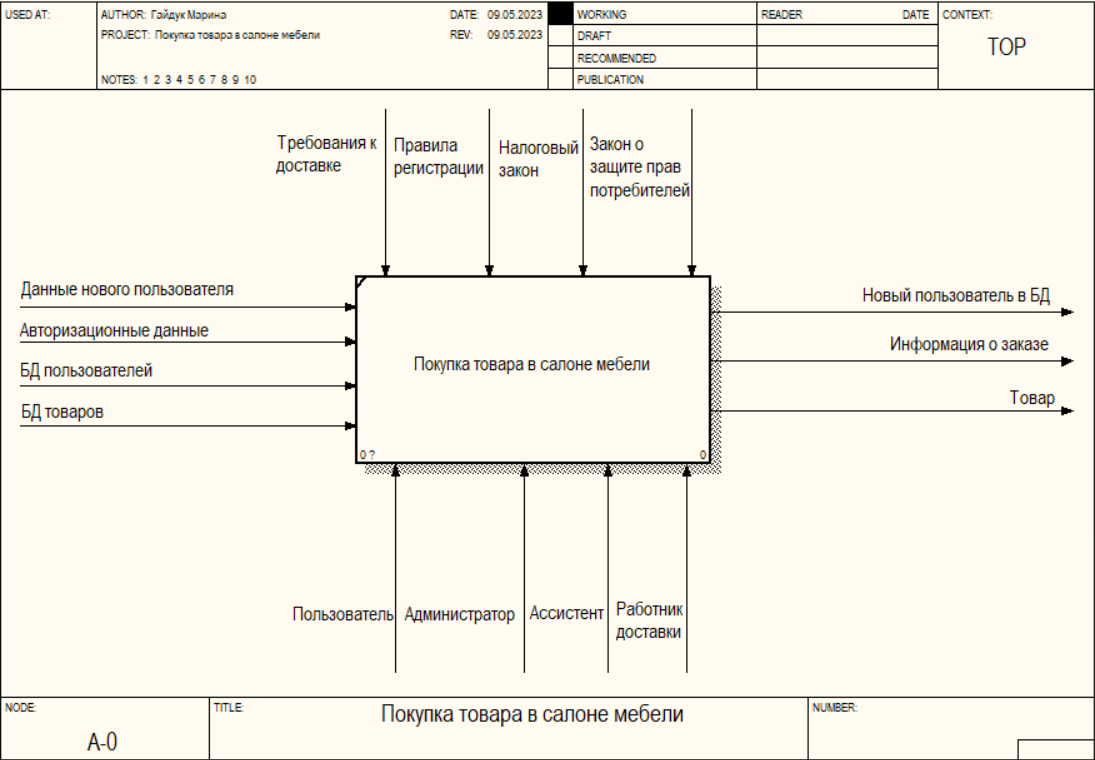


Рисунок 1.1 – Контекстная диаграмма верхнего уровня

Поддержка декомпозиции. Нотация IDEF0 поддерживает последовательную декомпозицию процесса до требуемого уровня детализации. Дочерняя диаграмма, создаваемая при декомпозиции, охватывает ту же область, что и родительский процесс, но описывает ее более подробно. Согласно методологии IDEF0 при декомпозиции стрелки родительского процесса переносятся на дочернюю диаграмму в виде граничных стрелок. В результате декомпозиции диаграммы верхнего уровня, представляющей функциональную модель «Покупка товаров в мебельном магазине», получена структура, изображённая на рисунке 1.2 диаграмма декомпозиции.

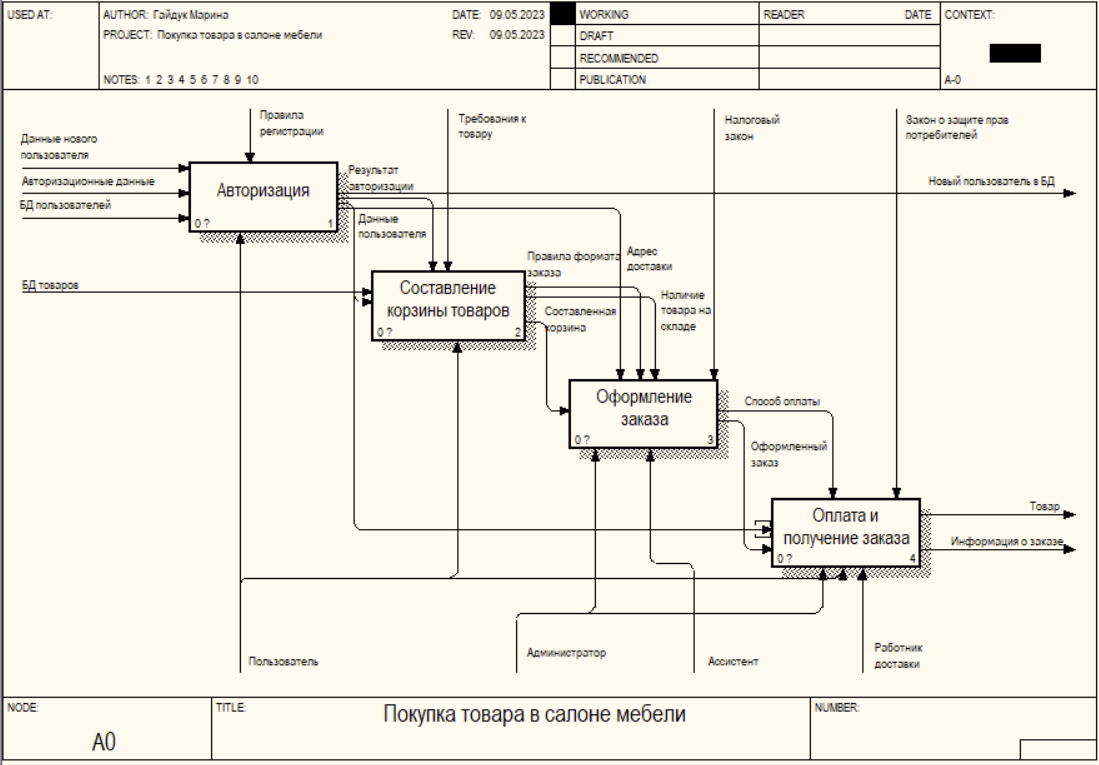
****

Рисунок 1.2 - Декомпозиция контекстной диаграммы верхнего уровня

На рисунках 1.3 - 1.6 представлены диаграммы декомпозиции для блоков «Авторизация», «Составление корзины товаров», «Оформление заказа» и «Оплата и получение заказа».

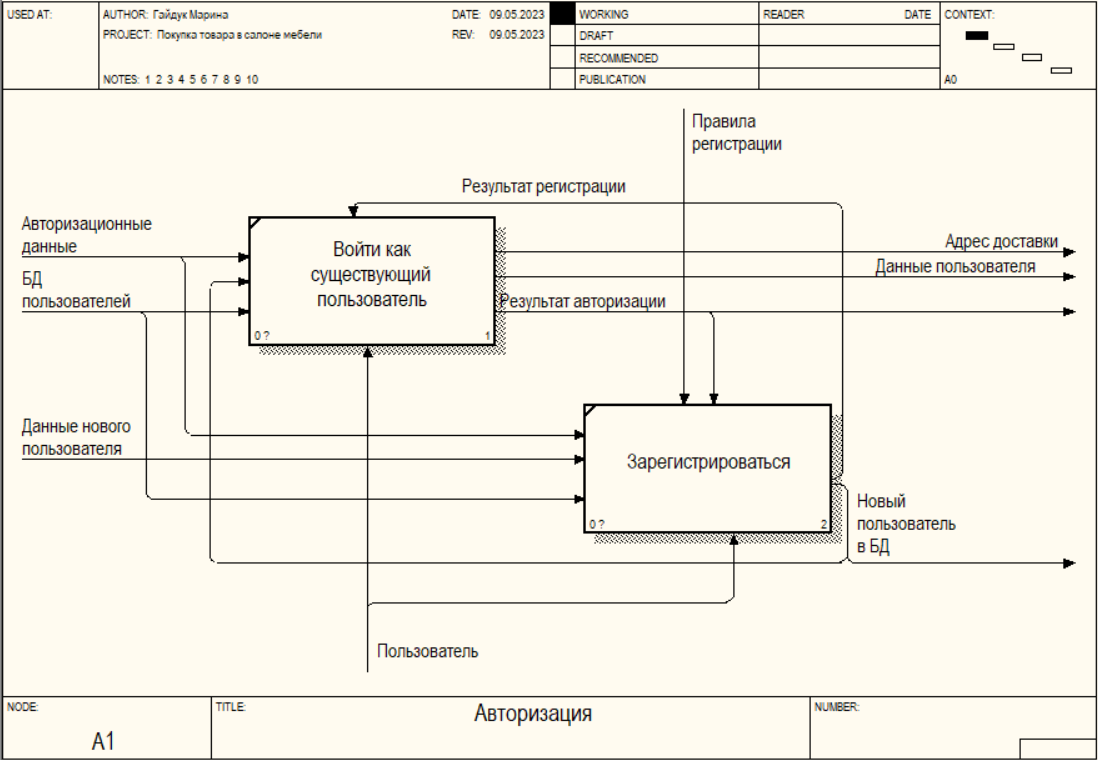


Рисунок 1.3 - Декомпозиция блока «Авторизация»

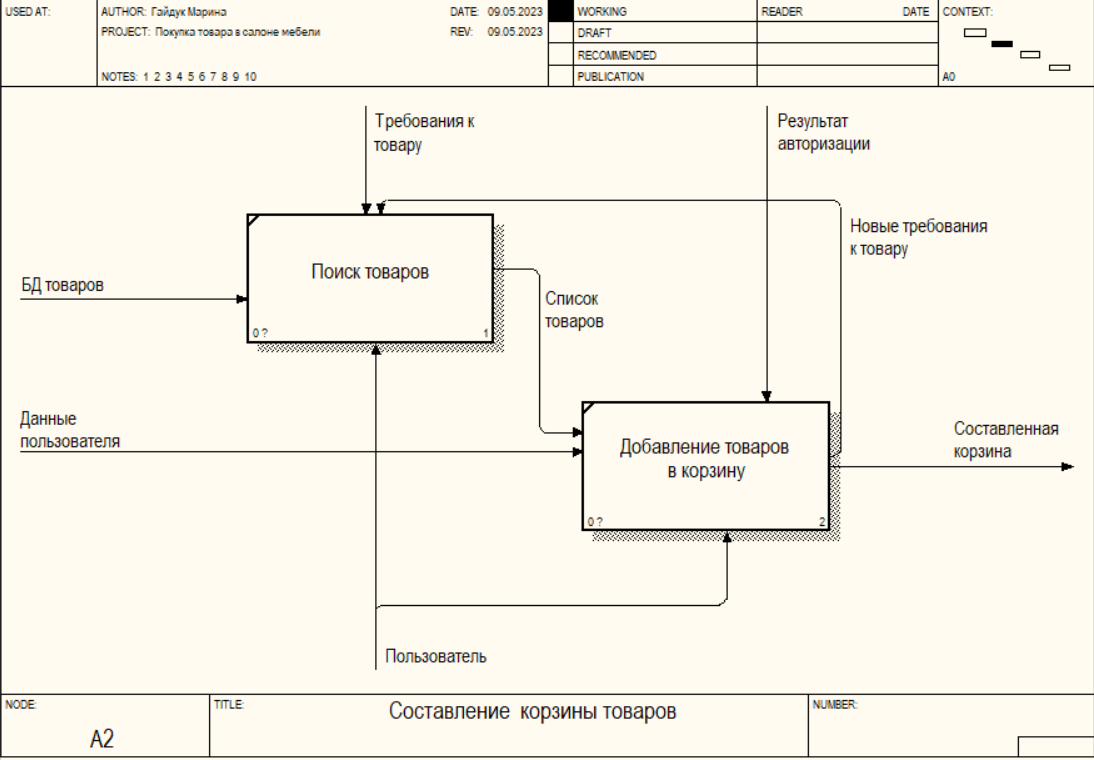


Рисунок 1.4 - Декомпозиция блока «Составление корзины товаров»

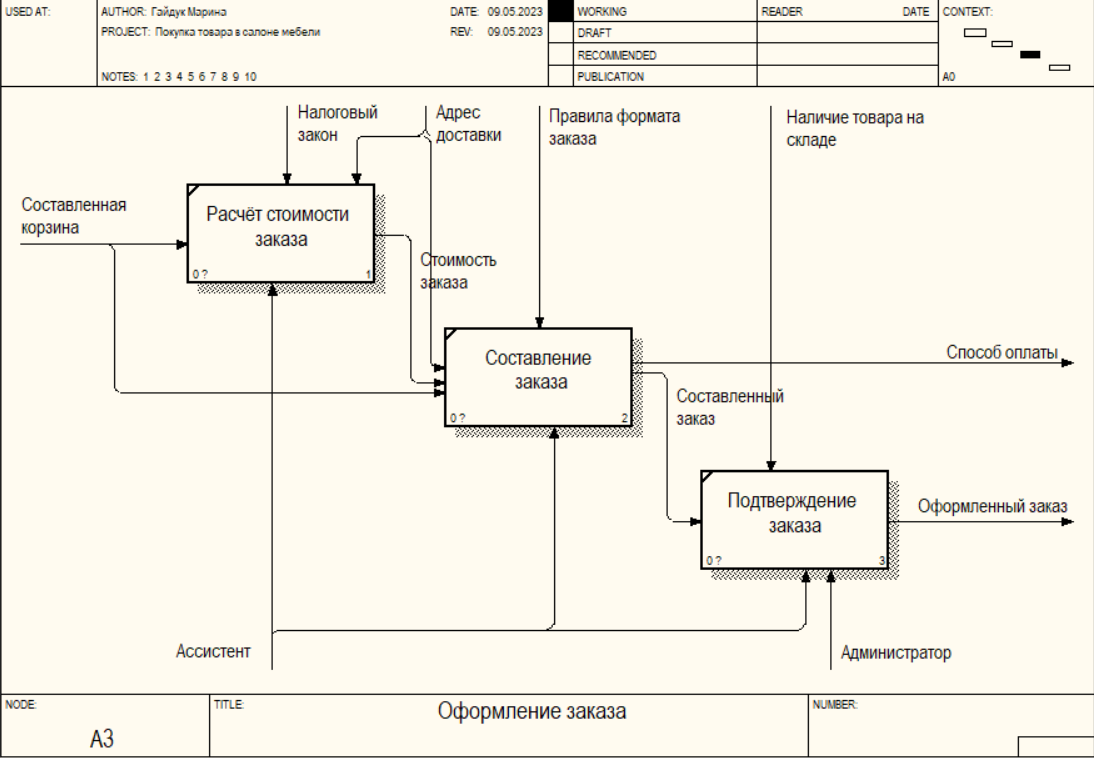


Рисунок 1.5 - Декомпозиция блока «Оформление заказа»

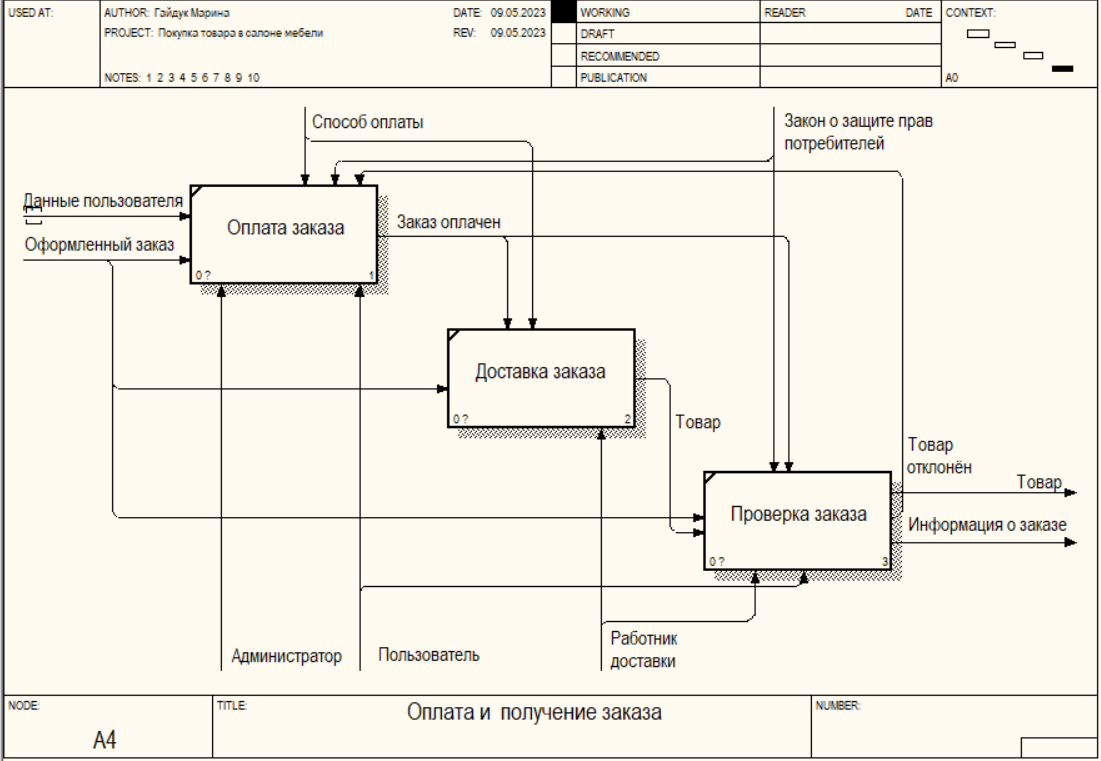


Рисунок 1.6 - Декомпозиция блока «Оплата и получение заказа»

Доминирование. Блоки модели IDEF0 на неконтекстной диаграмме должны располагаться по диагонали - от левого верхнего угла диаграммы до правого нижнего в порядке присвоенных номеров. Блоки на диаграмме, расположенные вверху слева, "доминируют" над блоками, расположенными внизу справа. "Доминирование" понимается как влияние, которое блок оказывает на другие блоки диаграммы. Расположение блоков на листе диаграммы отражает авторское понимание доминирования. Таким образом, топология диаграммы показывает, какие функции оказывают большее влияние на остальные.

Выделение 4 типов стрелок. Выделяются следующие типы стрелок: "Вход", "Выход", "Механизм", "Управление". Входы преобразуются или расходуются процессом, чтобы создать то, что появится на его выходе. Управления определяют условия, необходимые процессу, чтобы произвести правильный выход. Выходы - данные или материальные объекты, произведенные процессом. Механизмы идентифицируют средства, поддерживающие выполнение процесса [1].

Таким образом, блок IDEF0 показывает преобразование входа в выход с помощью механизмов с учетом управляющих воздействий.

## **1.3. Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований**

Согласно заданию к курсовому проекту, задача состоит в разработке приложения для расчета прибыли от продаж на примере мебельного салона. Приложение должно быть выполнено в архитектуре клиент-сервер с многопоточным сервером с организацией взаимодействия с базой данных на объектно-ориентированном языке Java. Бизнес-логика системы должна быть реализована на серверной части приложения. На сервере должна быть предусмотрена возможность параллельной обработки запросов. Доступ к данным в СУБД должен осуществляться через драйвер, предоставляемый производителем СУБД или через использование специальных технологий. Схема базы данных должна обеспечивать ссылочную целостность данных.

Сторона пользователя должна иметь возможность взаимодействовать с системой (сервером), которая в свою очередь должна принимать данные от пользователя, осуществлять обработку информации, отправлять обработанные данные к пользователю, а также заносить информацию в базу данных и осуществлять различные операции с данными согласно действиям пользователя. Необходимо предусмотреть механизм авторизации всех пользователей. Количество ролей – не менее 2-3 в зависимости от специфики предметной области (администратор, сотрудник компании, гость).

Интерфейс должен представлять собой оконное приложение с использованием стандартных библиотек пользовательского интерфейса. Его следует представлять только на русском языке.

В разрабатываемом приложении предусмотрены три действующих лица: пользователь, ассистент и админ. В зависимости от действующего лица в программе предусмотрены ограничения на выполнение некоторых функций. Каждое действующее лицо имеет удобный и интуитивно понятный интерфейс в приложении.

Пользователь имеет следующие функциональные требования:

* регистрация и авторизация;
* просмотр и изменение личной информации – выводит на экран всю личную информацию о клиенте и дает возможность изменить ее;
* просмотр товаров конкретной категории;
* поиск и сортировка товаров;
* фильтрация товаров – предоставляет возможность отсортировать продукцию по возрастанию или убыванию цены;
* получение подробной информации о товаре – выводит на экран всю информацию об имеющихся товарах;
* добавление товара в корзину – позволяет пользователю как удалять из корзины, так и добавлять в корзину различные товары;
* просмотр корзины – позволяет получить полную информацию о корзине: добавленные товары и их количество;
* изменение товаров в корзине;
* формирование заказа на основе корзины – осуществляет создание заказа, содержание которого идентично тому, которое составляет пользовательскую корзину. Пользователь может осуществить заказ только находясь в корзине.;
* просмотр и отмена активных заказов;
* просмотр завершенных заказов.

При регистрации происходит ввод личных данных, таких как:

* логин;
* пароль;
* имя;
* фамилия;
* номер телефона;
* почта;
* адрес.

После регистрации логин, пароль, а также полная информация о клиенте заносятся в базу данных с пользователями.

Авторизация происходит путем ввода личного логина и пароля. При некорректно введенной информации программа выдаст ошибку и попросит ввести логин и пароль снова.

Метод полной очистки корзины удаляет всю информацию о товарах из пользовательской корзины.

Ассистент имеет следующие функциональные требования:

* авторизация;
* просмотр информации о товарах – предоставляет информацию о выбранном товаре и позволяет корректировать этот товар;
* добавление нового товара – позволяет добавить новый товар определенной категории, и имеющий определенное описание;
* корректировка информации о товаре;
* просмотр активных заказов – позволяет получить полный список всех активных заказов от всех пользователей;
* осуществление заказа с учетом его количества – позволяет ответить на любой осуществленный пользователем заказ, дав отрицательный или положительный ответ.

Администратор имеет следующие функциональные требования:

* авторизация;
* просмотр пользователей и ассистентов – выводит полный список всех имеющихся в базе данных пользователей и ассистентов;
* регистрация ассистентов – позволяет создать аккаунт для ассистента для дальнейшего его использования;
* удаление аккаунтов пользователей и ассистентов – позволяет удалить по уникальному номеру аккаунт и все связанные с ним данные;
* получение статистики о прибыли – позволяет получить данные о прибыли за определенный промежуток времени, включает в себя построение графиков статистики прибыли за разные промежутки времени;
* формирование отчета за промежуток времени.

Диаграмма вариантов использования (англ. use-case diagram) – диаграмма, описывающая, какой функционал разрабатываемой программной системы доступен каждой группе пользователей [2].

На рисунках 1.7 - 1.9 представлены диаграммы вариантов использования администратором, ассистентом и пользователем.

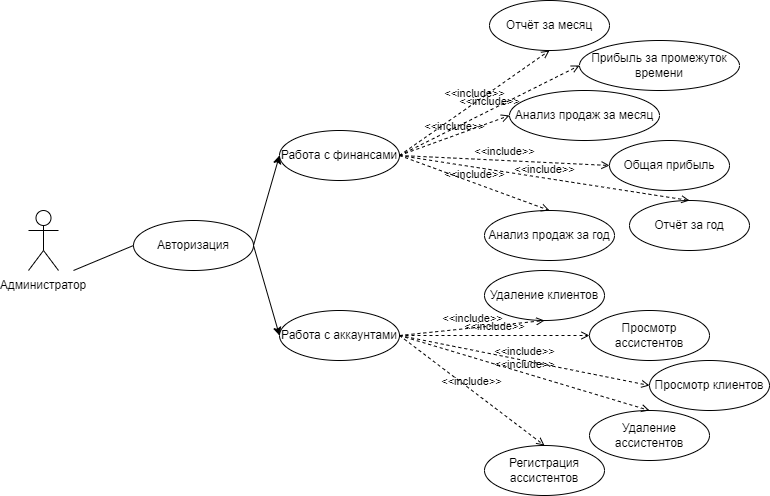


Рисунок 1.7 – Диаграмма варианта использования программы администратором

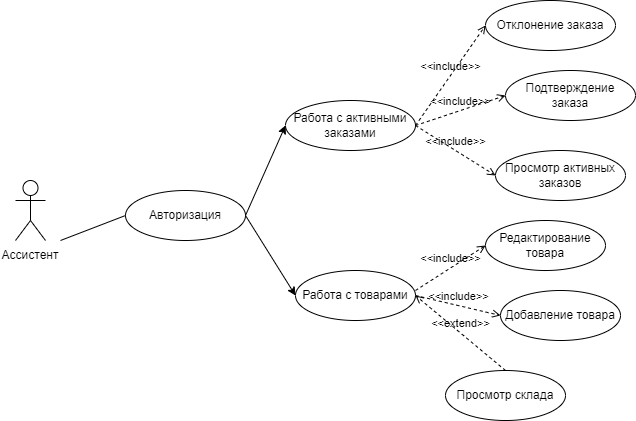


Рисунок 1.8 – Диаграмма варианта использования программы ассистентом

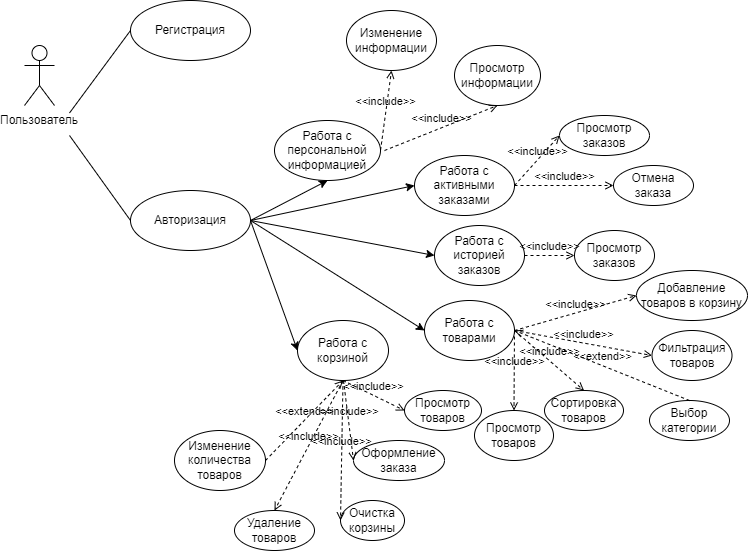


Рисунок 1.9 – Диаграмма варианта использования программы пользователем

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества актеров, взаимодействующих с системой с помощью так называемых вариантов использования. Вариант использования представляет собой последовательность действий (транзакций), выполняемых системой в ответ на событие, инициируемое некоторым внешним объектом (действующим лицом). Вариант использования описывает типичное взаимодействие между пользователем и системой.

## **1.4. Разработка информационной модели предметной области**

Информационная модель — модель объекта, представленная в виде информации, описывающей существенные для данного рассмотрения параметры и переменные величины объекта, связи между ними, входы и выходы объекта и позволяющая путём подачи на модель информации об изменениях входных величин моделировать возможные состояния объекта [3].

Для построения информационной модели предметной области была использована программа CA ERwin Data Modeler, опирающаяся на методологию IDEF1X.

Хотя терминология IDEF1X практически совпадает с терминологией IDEF1, но существует ряд фундаментальных отличий в теоретических концепциях этих методологий. Сущность в IDEF1X описывает собой совокупность или набор экземпляров похожих по свойствам, но однозначно отличающихся друг от друга по одному или нескольким признакам. Каждый экземпляр является реализацией сущности. Таким образом, сущность в IDEF1X описывает конкретный набор экземпляров реального мира, в отличие от сущности в IDEF1, которая представляет собой абстрактный набор информационных отображений реального мира [4].

Чаще всего методология IDEF1X используется для описания данных в целях последующей автоматизации их обработки с помощью систем управления базами данных.

В ходе разработки информационной модели предметной области были выделены следующие сущности:

* категория мебели;
* мебель;
* элемент корзины;
* элемент заказа;
* заказ;
* пользователь;
* ассистент;
* админ;
* обслуженный заказ;
* элемент обслуженного заказа.

На рисунке 1.10 представлена ER-диаграмма автоматизируемой предметной области.

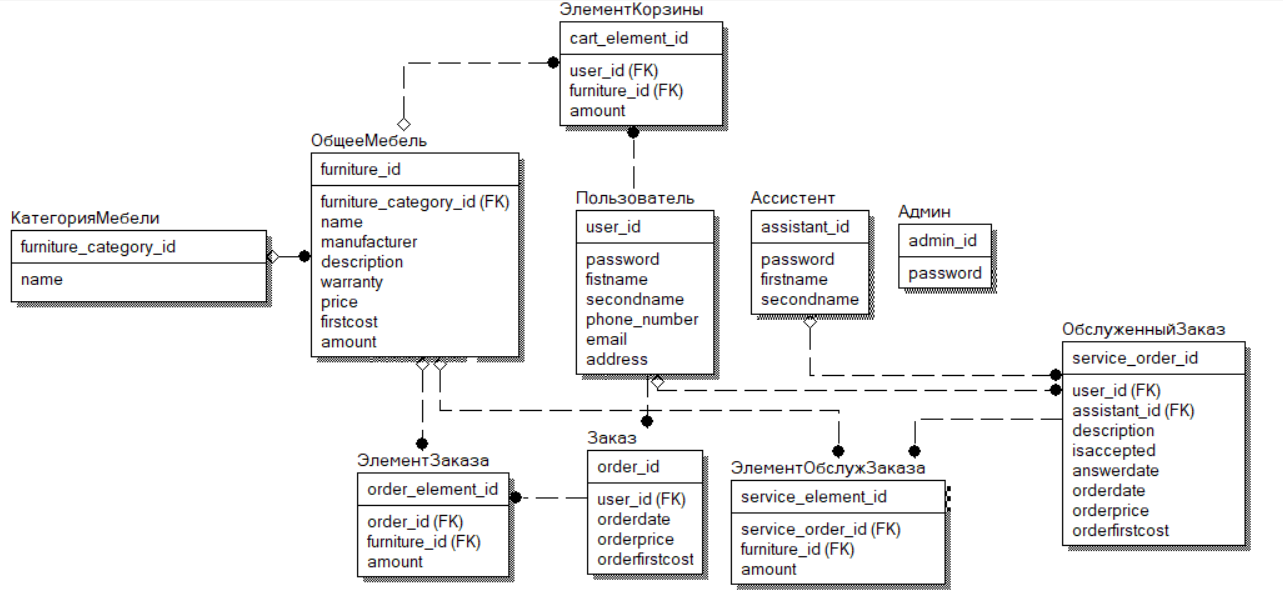


Рисунок 1.10 – ER-диаграмма автоматизируемой предметной области

Сущности, входящие в ER-диаграмму, и их атрибуты:

Сущность КатегорияМебели:

* furniture\_category\_id – содержит уникальный ID категории мебели;
* name ­– содержит название категории мебели.

Сущность ОбщееМебель:

* furniture\_id – содержит уникальный ID экземпляра мебели;
* name – содержит название мебели;
* furniture\_category\_id – содержит ID категории мебели;
* manufacturer – содержит производителя;
* description – содержит описание мебели;
* warranty – содержит гарантию;
* price – содержит цену;
* firstcost – содержит себестоимость;
* amount – содержит количество.

Сущность ЭлементКорзины:

* cart\_element\_id – содержит уникальный ID товара в корзине;
* user\_id – содержит уникальный ID пользователя;
* furniture\_id – содержит ID экземпляра мебели;
* amount – содержит количество товара в корзине.

Сущность ЭлементЗаказа:

* order\_element\_id – содержит уникальный ID элемента заказа;
* order\_id – содержит уникальный ID заказа;
* furniture\_id – содержит ID экземпляра мебели;
* amount – содержит количество товара в заказе.

Сущность Заказ:

* order\_id(содержит уникальный ID заказа);
* user\_id(содержит ID пользователя);
* orderdate(содержит дату заказа);
* orderprice(содержит стоимость заказа);
* orderfirstcost(содержит себестоимость заказа).

Сущность Пользователь:

* user\_id – содержит уникальный ID пользователя;
* password – содержит зашифрованный пароль;
* firstname – содержит имя пользователя;
* secondname – содержит фамилию пользователя;
* phone\_number – содержит номер телефона;
* email – содержит почту;
* address – содержит адрес.

Сущность Ассистент:

* assistant\_id – содержит уникальный ID ассистента;
* password – содержит зашифрованный пароль;
* firsname – содержит имя ассистента;
* secondname – содержит фамилию ассистента.

Сущность Админ:

* admin\_id – содержит уникальный ID админа;
* password – содержит зашифрованный пароль.

Сущность ОбслуженныйЗаказ:

* serviced\_order\_id – содержит уникальный ID обслуженного заказа;
* user\_id – содержит ID пользователя;
* assistant\_id – содержит ID ассистента;
* description – содержит описание заказа;
* isaccepted – содержит подтверждение заказа;
* answerdate – содержит дату ответа;
* orderdate – содержит дату заказа;
* orderprice – содержит стоимость заказа;
* orderfirstcost – содержит себестоимость заказа.

Данная ER-диаграмма приведена к условиям третьей нормальной формы – неключевой атрибут сущности функционально зависит только от первичного ключа и ни от чего другого.

## **1.5. UML-модели представления программного средства и их описание**

UML, или Unified Modeling Language, — унифицированный язык моделирования. Это графический язык, который с помощью диаграмм и схем описывает разнообразные процессы и структуры. Это не язык программирования, но чаще всего UML применяют именно в IT — с его помощью можно автоматически генерировать код. Важная особенность UML — этот язык поддерживает объектно-ориентированный подход, где все сущности представлены как объекты с определенными свойствами и методами [5].

Диаграмма последовательности. Она используется для более детального описания логики сценариев использования. Диаграммы последовательностей обычно содержат объекты, которые взаимодействуют в рамках сценария, сообщения, которыми они обмениваются, и возвращаемые результаты, связанные с сообщениями [6]. На рисунке 1.11 представлена диаграмма последовательности фильтрации товаров.

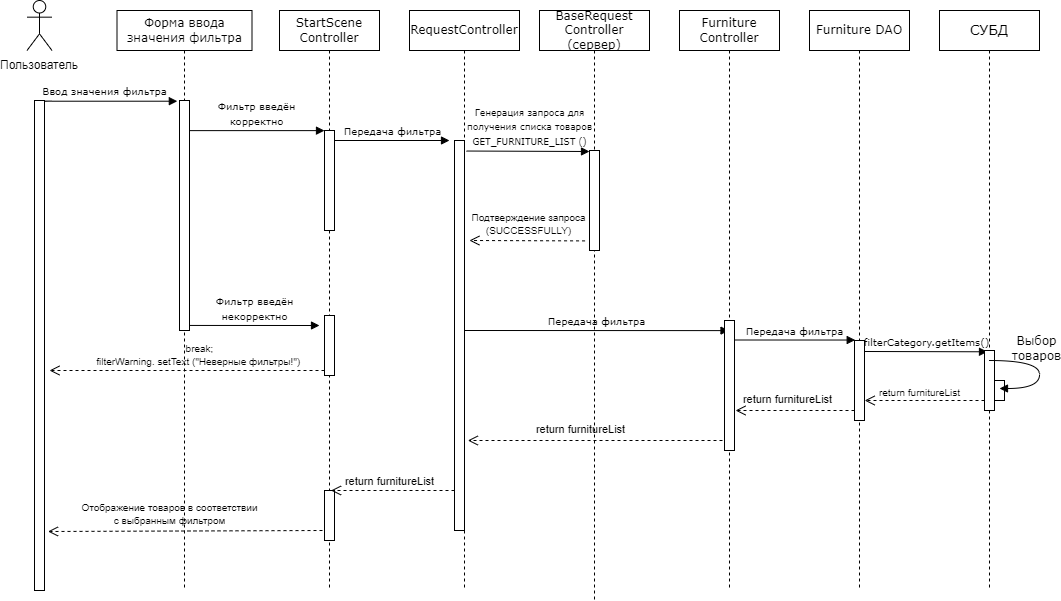


Рисунок 1.11 – Диаграмма последовательности фильтрации товаров

Данная диаграмма показывает, каким образом пользователь отфильтрует товары. Изначально он выбирает необходимые ему фильтры, если фильтры введены некорректно, то контроллер сцены выводит соответствующее предупреждение пользователю. Если же фильтры введены корректно, контроллер сцены передает информацию о фильтрах контроллеру запросов, который генерирует запрос на получение списка товаров и после получения подтверждения отсылает фильтры на контроллер сервера. Контроллер товаров сервера передает фильтры на прослойку между базой данных и приложением, которая в свою очередь на основании фильтров генерирует запрос на выборку из базы данных. После получения списка отфильтрованных товаров инициируется цепочка возвращения данных, которая заканчивается отображением списка товаров пользователю.

Диаграмма развёртывания. Диаграмма развёртывания показывает топологию системы и распределение компонентов системы по ее узлам, а также соединения - маршруты передачи информации между аппаратными узлами. Это единственная диаграмма, на которой применяются “трехмерные” обозначения: узлы системы обозначаются кубиками [7]. Диаграмма развертывания разрабатываемого программного средства представлена на рисунке 1.12.

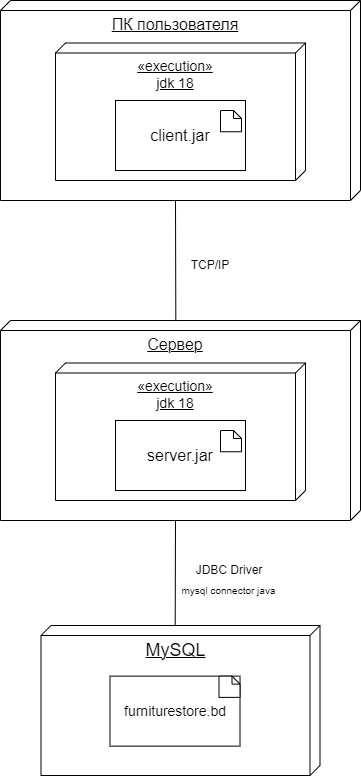


Рисунок 1.12 – Диаграмма развертывания разрабатываемого

программного средства

Для ПК пользователя и для ПК сервера, необходимо наличие среды выполнения jdk 18+. В качестве операционной системы используется Windows 10. Взаимодействие между клиентом и сервером осуществляется посредством TCP/IP соединения. Для соединения ПК пользователя и сервера используется специальный пакет Java java.net. Для соединения с базой данных используется стандарт JDBC, а в качестве драйвера mysql-connector-java. Для работы с базами данных используется СУБД MySQL.

Диаграмма классов. Диаграмма классов определяет типы классов системы и различного рода статические связи, которые существуют между ними. На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами [8].

Для описания архитектуры проекта были разработаны диаграммы классов ключевых областей. На рисунке 1.13 представлена диаграмма классов архитектуры запроса. Имеется интерфейс запроса, определяющий основные функциональные возможности, а также его реализация в виде класса запроса. Класс запроса также реализует интерфейс Serializable, поскольку передается от клиента к серверу и наоборот.

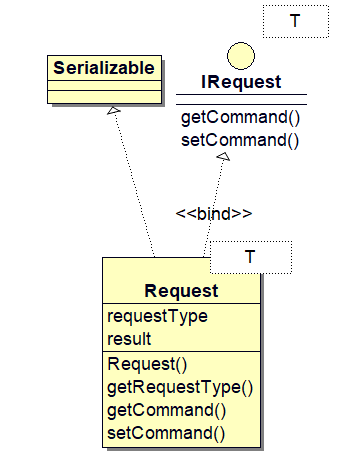


Рисунок 1.13 – Диаграмма классов архитектуры запроса

Через класс запроса осуществляется передача команд от сервера к клиенту, и наоборот. Наборы команд, при помощи которых осуществляется взаимодействие между клиентом и сервером, представлены на рисунке 1.14.

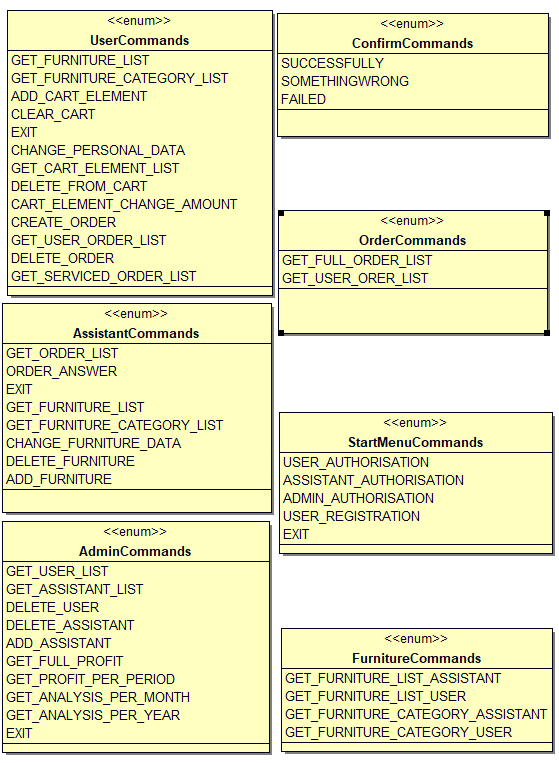


Рисунок 1.14 – Наборы команд, при помощи которых осуществляется взаимодействие между клиентом и сервером

На рисунке 1.15 представлена диаграмма классов моделей данных. Эти модели используются как объектные представления данных из реляционной базы данных. Каждая такая модель является сущностью и способна хранить информацию конкретной таблицы базы данных.

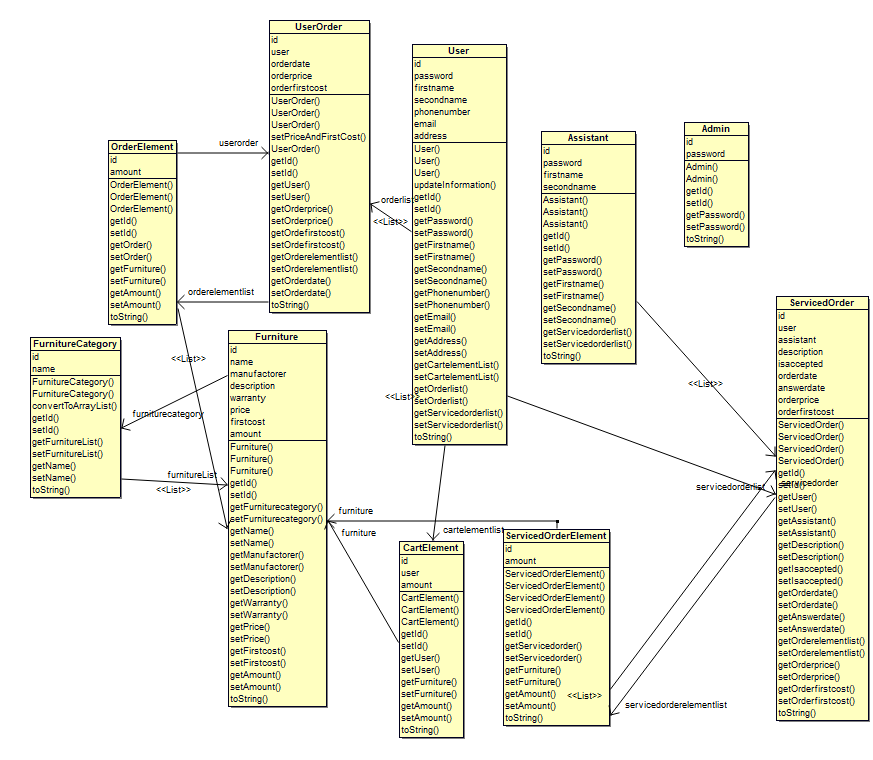


Рисунок 1.15 – Диаграмма классов моделей данных

Третьей областью является Data Access Object (DAO) – прослойка между базой данных и системой. DAO абстрагирует сущности системы и определяет их отображение на базу данных, определяет общие методы использования соединения, его получение и закрытие. Вершиной иерархии DAO является абстрактный класс или интерфейс с описанием общих методов, которые будут использоваться при взаимодействии с базой данных. Как правило, это методы поиска, удаление по ключу, обновление и т. д. На рисунке 1.16 представлена диаграмма классов DAO системы.

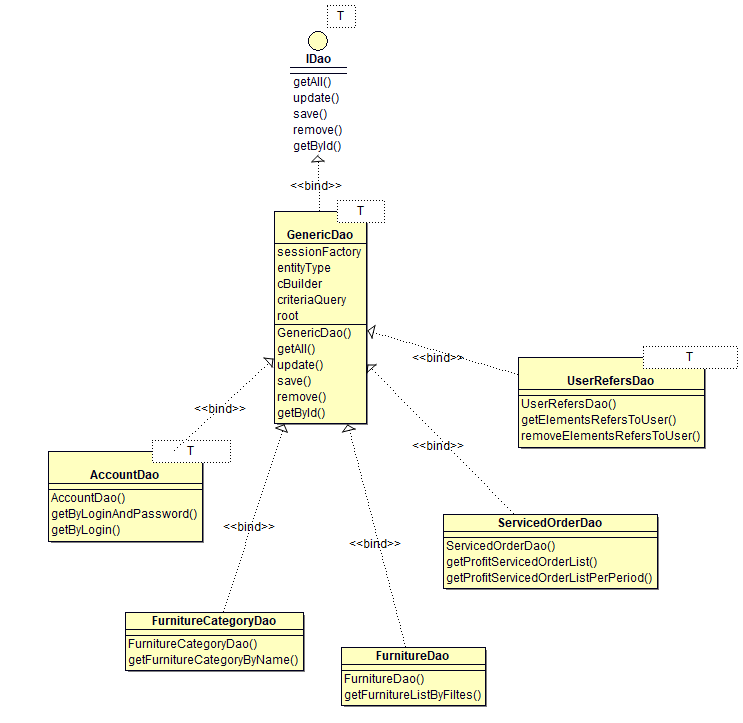


Рисунок 1.16 – Диаграмма классов DAO системы

В данной главе были представлены диаграммы классов областей, связанных с получением, хранением и пересылкой данных.

# 2. ПРОЕКТИРОВАНИЕ И КОНСТРУИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

## **2.1. Постановка задачи**

Задача заключается в реализации автоматизированной системы учета продажи товаров в мебельном салоне и расчете прибыли за определенный период.

Задачи, которые необходимо выполнить в рамках курсового проекта:

* изучение механизма работы системы продажи мебельных товаров;
* ознакомление с существующими программами для реализации системы продажи мебельной продукции;
* анализ слабых мест данного механизма на основе аналогов;
* разработка функциональных моделей основного процесса;
* тестирование разработанного программного продукта;
* отладка ошибок, обнаруженных в процессе;
* создать базу данных;
* спроектировать иерархию классов;
* реализовать клиент-серверное приложение;
* создать простой и удобный пользовательский интерфейс.

Для выполнения поставленных задач необходимо использовать различные информационные технологии при сборе, подготовке, хранении, обработке, поиске и предоставлении информации.

## **2.2. Обоснование выбора компонентов и технологий для реализации программного средства**

Разрабатываемое программное средство написано на языке программирования Java. Java представляет собой строго типизированный объектно-ориентированный язык программирования. Основные его достоинства:

* автоматическое управление памятью;
* расширенные возможности обработки исключительных ситуаций;
* возможность использования классов, имеющих цифровую подпись;
* набор средств фильтрации ввода-вывода;
* средства для создания многопоточности;
* наличие встроенных средств для создания сетевых приложений (имеется обширная программная библиотека для передачи данных по самым распространённым протоколам: FTP, HTTP, TCP/IP; работает механизм вызова удалённых методов).

Приложение реализовано в архитектуре «клиент-сервер». Задачи, которые необходимо выполнить, распределены между программой-клиентом и программой-сервером. Программы-серверы ожидают от клиентских программ запросы и предоставляют им свои ресурсы в виде данных или сервисных функций. Клиент и сервер могут располагаться на одной машине, но, поскольку программа-сервер должна выполнять запросы от множества программ-клиентов, обычно ее размещают на отдельной вычислительной машине с высокой производительностью.

В качестве стандарта связи между клиентом и сервером выбран протокол TCP/IP. Протокол TCP/IP определяет, как происходит весь процесс обмена данными: разделение на пакеты, адресация, передача, маршрутизация и получение в точке назначения. TCP/IP требует минимального централизованного управления. Этот протокол обеспечивает надежность сетей с возможностью автоматического восстановления после сбоя любого устройства. TCP/IP обладает рядом плюсов:

* работает независимо от ОС;
* поддерживает множество протоколов маршрутизации;
* использует клиент-серверную архитектуру с высокой масштабируемостью;
* не создает ненужной нагрузки на сеть или компьютер.

Для создания пользовательского интерфейса на стороне клиента использована платформа JavaFX. JavaFX — платформа на основе Java для создания приложений с насыщенным графическим интерфейсом. JavaFX предоставляет большие возможности по сравнению с рядом других подобных платформ, в частности, по сравнению со Swing. К основным возможностям относятся:

* большой набор элементов управления;
* встроенная поддержка MVC;
* возможности по работе с мультимедиа, двухмерной и трехмерной графикой;
* встроенный API для работы с многопоточностью;
* декларативный способ описания интерфейса с помощью языка разметки FXML;
* возможность стилизации интерфейса с помощью CSS.

Для создания и управления базой данных используется СУБД MySQL. MySQL — это реляционная система управления базами данных с открытым исходным кодом, которая отлично подходит для небольших и средних проектов. Преимущества [СУБД](https://web-creator.ru/articles/about_databases) MySQL:

* простота использования;
* высокая скорость работы;
* гибкость;
* надежность хранимой информации;
* поддержка нескольких типов таблиц: MyISAM, InnoDB;
* масштабируемость и производительность.

Гибкость данной СУБД обеспечивается поддержкой большого количества типов таблиц. MySQL предоставляет специальный драйвер “mysql-connector-java” для работы с JDBC, который и используется в разрабатываемом программном средстве.

## **2.3. Архитектурные решения**

Разрабатываемый программный продукт представляет собой клиент-серверное приложение.

Клиент-сервер – программное обеспечение, расположенное в пределах одной (но чаще – нескольких) машины, взаимодействующее друг другом посредством сетевых протоколов. Клиент – устройство, которое отправляет на сервер запросы. Соответственно, сервер принимает запросы, после чего обрабатывает команды и предоставляет клиенту ответ в установленном формате.

Связь между сервером и клиентом в разрабатываемом приложении осуществляется при помощи протокола TCP/IP (Transmission Control Protocol/Internet Protocol, протокол управления передачей/протокол интернета).

В качестве СУБД для разрабатываемого приложения выбрана MySQL. MySQL является одной из самых востребованных систем на сегодняшний день. Она отлично подходит для небольших и средних проектов, особенно часто ее применяют при разработке сайтов. Для работы приложения и СУБД MySQL используется стандарт JDBC и драйвер СУБД MySQL “my-sql-connector”.

## **2.4. Описание алгоритмов, реализующих ключевую бизнес-логику разрабатываемого программного средства**

Разрабатываемый продукт представляет собой клиент-серверное приложение. Запросы администратора, ассистента и пользователя обрабатываются не локально, а отправляются на сервер, который обрабатывает полученный запрос, после чего осуществляется ответ пользователю. Все алгоритмы находятся именно на серверной части проекта.

Для того, чтобы получить доступ к программе, администратору, ассистенту и пользователю необходимо ввести логин и пароль. При неправильном вводе данных будет предложен повторный ввод или возвращение в начальное меню.

После входа администратору, ассистенту или пользователю предоставляется соответствующее меню с определенным набором функций, доступное пока они не выйдут из личного аккаунта в главное меню. У администратора имеется набор функций для работы с товарами, аккаунтами, а также с финансами, у ассистента имеется набор функций для работы со складом, товарами, а также активными заказами, у пользователя имеется набор функций для работы с личными данными, товарами, корзиной, активными и заказами, а также с историей заказов. Данный алгоритм представлен графически на рисунке 2.1.

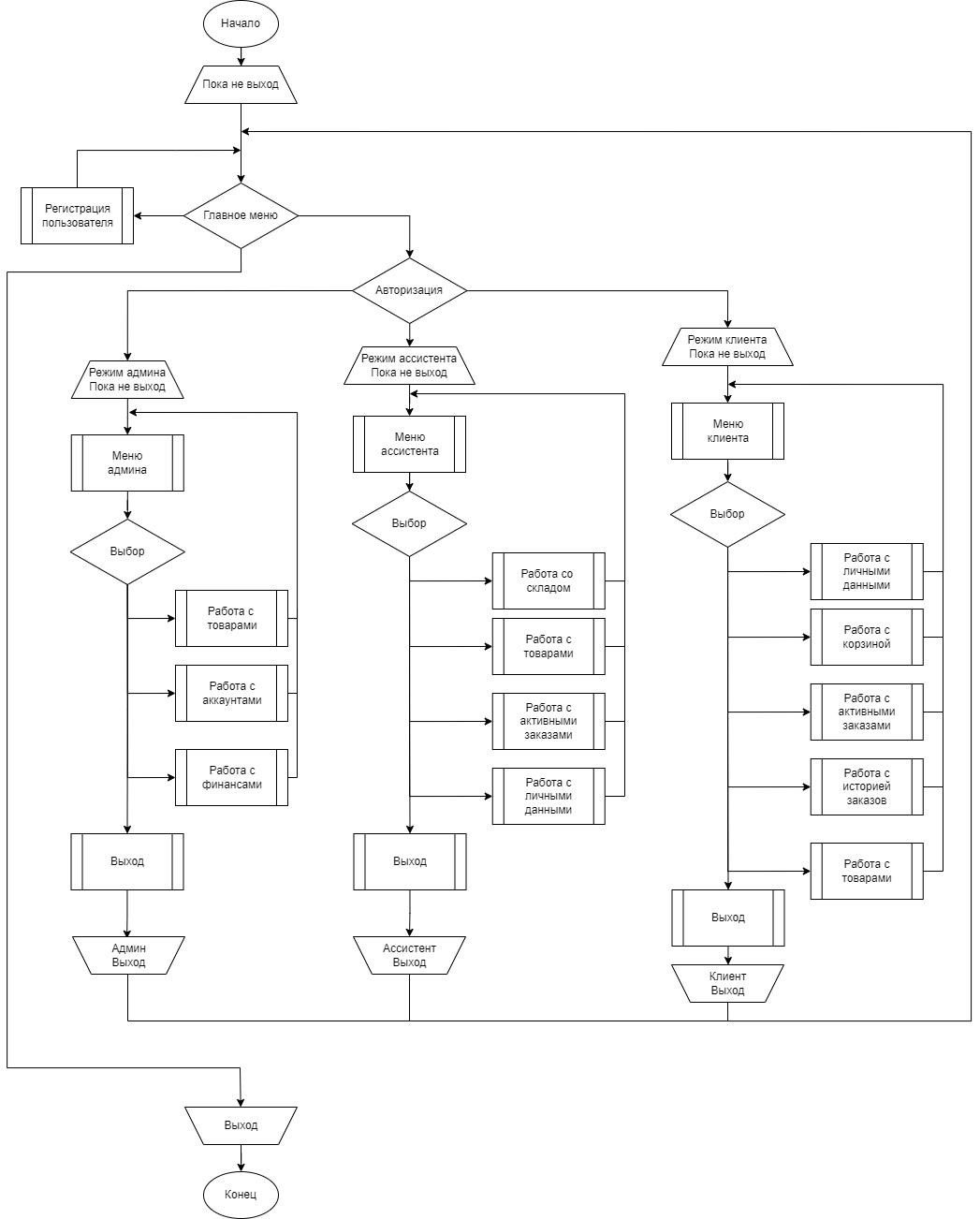


Рисунок 2.1 - Блок-схема алгоритма работы программного средства

Ключевым процессом является расчет прибыли и получение анализа продаж за различные промежутки времени. Алгоритм вычисления прибыли за все время основан на обслуженных заказ. Учитываются все заказы, подтверждённые ассистентами, поскольку они приносят прибыль. Чтобы рассчитать прибыль за все время, необходимо выбрать все обслуженные заказы с положительным ответом и сложить прибыль каждого из них. Прибыль рассчитывается по формуле:

По состоянию на 10.03.2023 налог на прибыль в Республике Беларусь составляет 20%, а подоходный налог для физ. лиц – 13%. Блок-схема алгоритма расчета общей прибыли за все время представлена на рисунке 2.2.

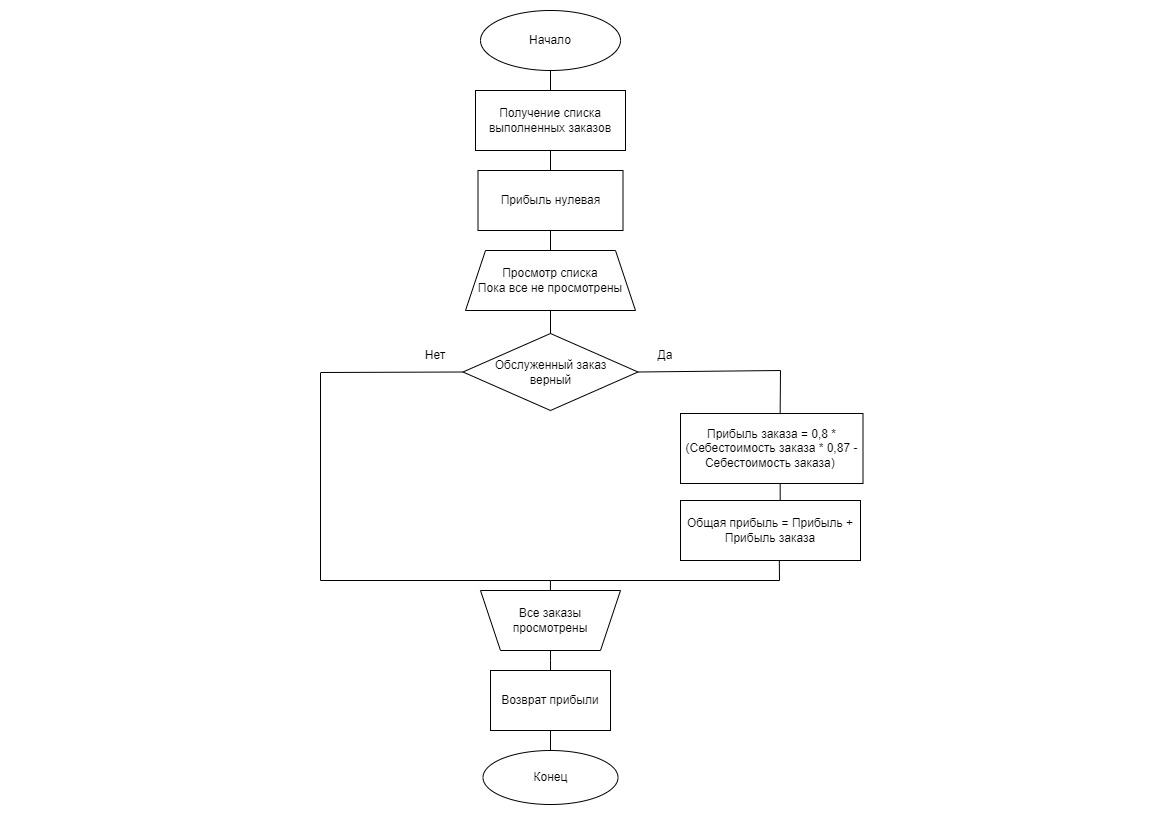


Рисунок 2.2 - Блок-схема алгоритма расчета прибыли за все время

Отличие расчёта прибыли за определённый период от расчёта прибыла за всё время заключается в том, что помимо проверки ответа ассистента, проверяется также дата ответа на заказ, если она попадает в данный промежуток времени, то такой заказ учитывается при расчете прибыли. Блок-схема алгоритма расчета прибыли за определенный промежуток времени представлена на рисунке 2.3.

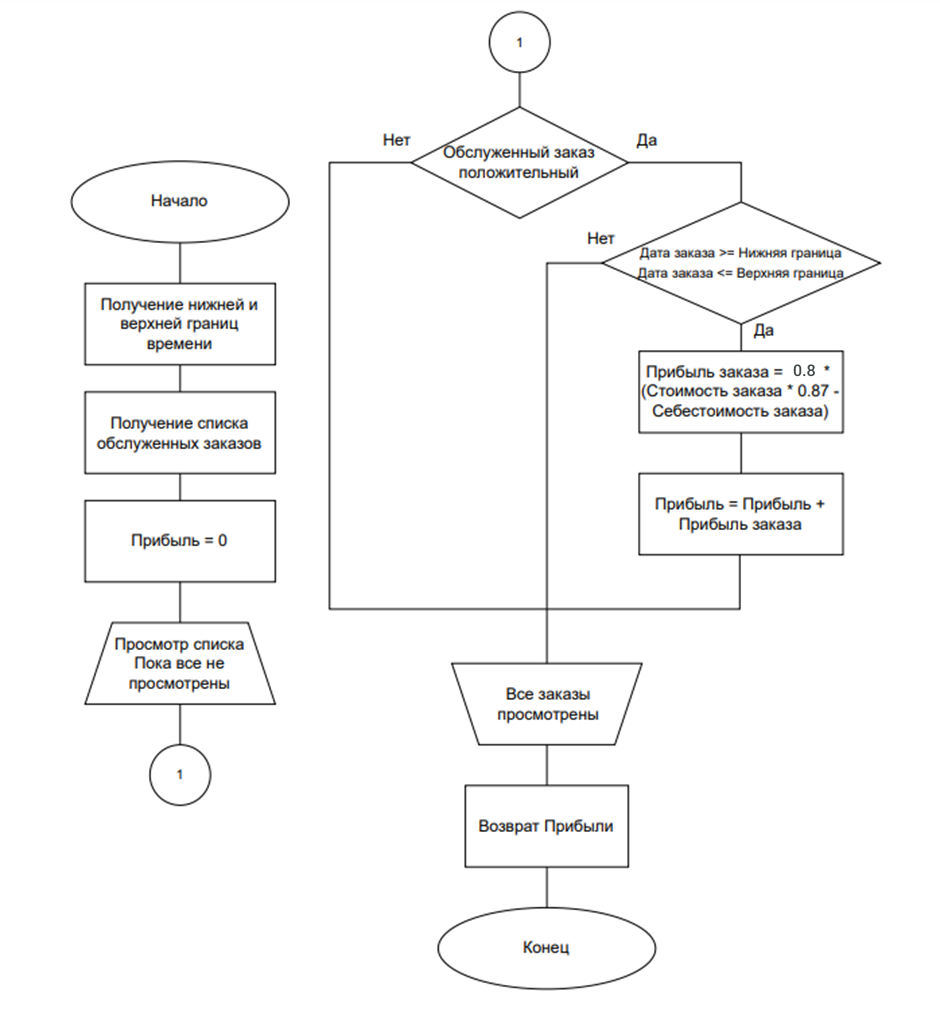


Рисунок 2.3 - Блок-схема алгоритма расчета прибыли за определенный

промежуток времени

## **2.5. Проектирование пользовательского интерфейса**

Пользовательский интерфейс – это внешний вид продукта, способ общения между пользователем и программой. Интерфейс влияет на то, насколько успешным будет продукт.

Пользовательский интерфейс должен обладать определенными качествами:

* доступность. Чтобы пользователь мог взаимодействовать с интерфейсом на интуитивном уровне и не допускать ошибок, интерфейс следует делать лёгким и понятным в использовании.
* соответствие контексту. Разные страницы могут содержать контент разного типа. Следует адаптировать каждую страницу под соответствующий ей контент, создавать элементы управления, которые упростят пользователю работы с сайтом.
* функциональность. Интерфейс должен как минимум отражать функциональность системы и давать возможность успешной работы пользователям различной квалификации.

Разрабатываемая система имеет оконный интерфейс. Данный интерфейс разработан с помощью JavaFX. JavaFX – библиотека, содержащая инструментарий пользовательского графического интерфейса для языка программирования Java. JavaFX использует современный графический процессор с помощью аппаратно-ускоренной графики.

Интерфейс разработан с помощью SceneBuilder – это автономного приложения, которое генерирует разметку FXML. Все компоненты системы взаимодействуют друг с другом и связаны.

При запуске системы появляется окно авторизации. Пользователю предлагается ввести логин и пароль для входа, если у пользователя нет аккаунта он может зарегистрироваться. При входе клиент выбирает, в качестве кого он хочет зайти. В зависимости от выбранной роли, клиент имеет различные функциональные возможности и, следовательно, различные оконные интерфейсы.

Таким образом, оконный интерфейс клиента содержит:

* область просмотра и изменения личной информации;
* форма фильтрации и поиска;
* область описания товаров;
* область просмотра активных заказов;
* область просмотра обслуженных заказов;
* область просмотра корзины.

Оконный интерфейс ассистента содержит:

* область просмотра склада;
* область изменения информации товара;
* область просмотра активных заказов;
* область ответа на заказ;
* область добавления нового товара.

Оконный интерфейс администратора содержит:

* область просмотра пользователей;
* область просмотра ассистента;
* область добавления ассистента;
* область о получения статистики о прибыли.

## **2.6. Методы и средства, используемые для обеспечения безопасности данных**

Безопасность данных — это защитные меры, применяемые для защиты данных от несанкционированного доступа и сохранения конфиденциальности, целостности и доступности данных. Лучшие практики по обеспечению безопасности данных включают такие методы защиты данных, как шифрование данных, управление ключами, редактирование данных, разделение данных на подмножества и маскирование данных, а также контроль доступа привилегированных пользователей, аудит и мониторинг [9].

В разрабатываемом программном средстве используется метод «encrypt» для создания зашифрованного представления строки пароля с целью хранения паролей в безопасном виде.

Когда пользователь создает новый аккаунт, его пароль сохраняется на сервере в зашифрованном виде. При входе в систему, введенный пользователем пароль также шифруется и сравнивается с хранимым на сервере зашифрованным паролем. Если хеши совпадают, то вход в систему разрешается.

Это позволяет обеспечить безопасность пользовательских паролей в случае, если база данных сервера попадет в руки злоумышленников.

Данный метод использует алгоритм хеширования SHA-512, который является одной из наиболее безопасных хеш-функций. Она принимает на вход байтовое представление строки пароля, которое предварительно преобразуется в кодировку UTF-8, чтобы обеспечить совместимость с различными языками и операционными системами.

Кроме того, используется ещё одна мера безопасности для защиты пароля от несанкционированного доступа – скрывание пароля в базе данных с использованием типа BLOB (Binary Large Object). Вместо того, чтобы хранить пароль в открытом текстовом формате в базе данных, он сохранияется в зашифрованном виде в объекте BLOB.

# 3. ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА

Тестирование программного средства – это метод проверки соответствия фактического программного продукта ожидаемым требованиям, который также необходим, чтобы убедиться, что продукт не содержит дефектов [10].

В разрабатываемом программном средстве ключевым аспектом тестирования стало тестирование работы с базой данных. Были написаны тесты для проверки корректного подключения к базе данных, вставки, удаления, обновления данных, а также перевод даты из одного типа в другое. Результаты тестирования представлены на рисунке 3.1.

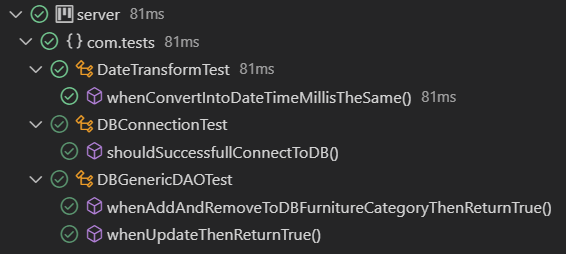


Рисунок 3.1 – Результаты тестирования

С целью проверки работоспособности разрабатываемого программного средства было проведено тестирование, отражающее возможные исключительные ситуации, которые возникают при некорректных действиях клиента.

Первым, где могут возникнуть ошибки, является пункт регистрации пользователя. Он может не ввести значения обязательных полей или использовать пользовательское имя, которое уже занято. Предупреждение о данных ошибках представлено на рисунках 3.2 и 3.3.

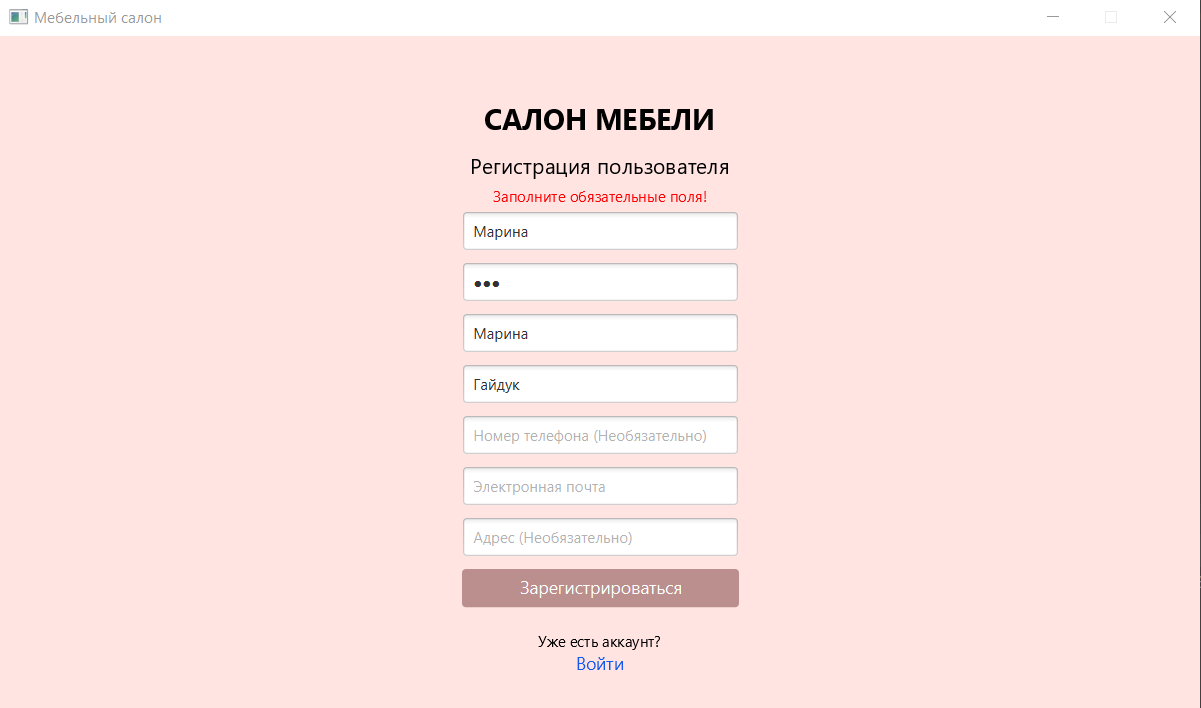


Рисунок 3.2 – Предупреждение об ошибке при регистрации

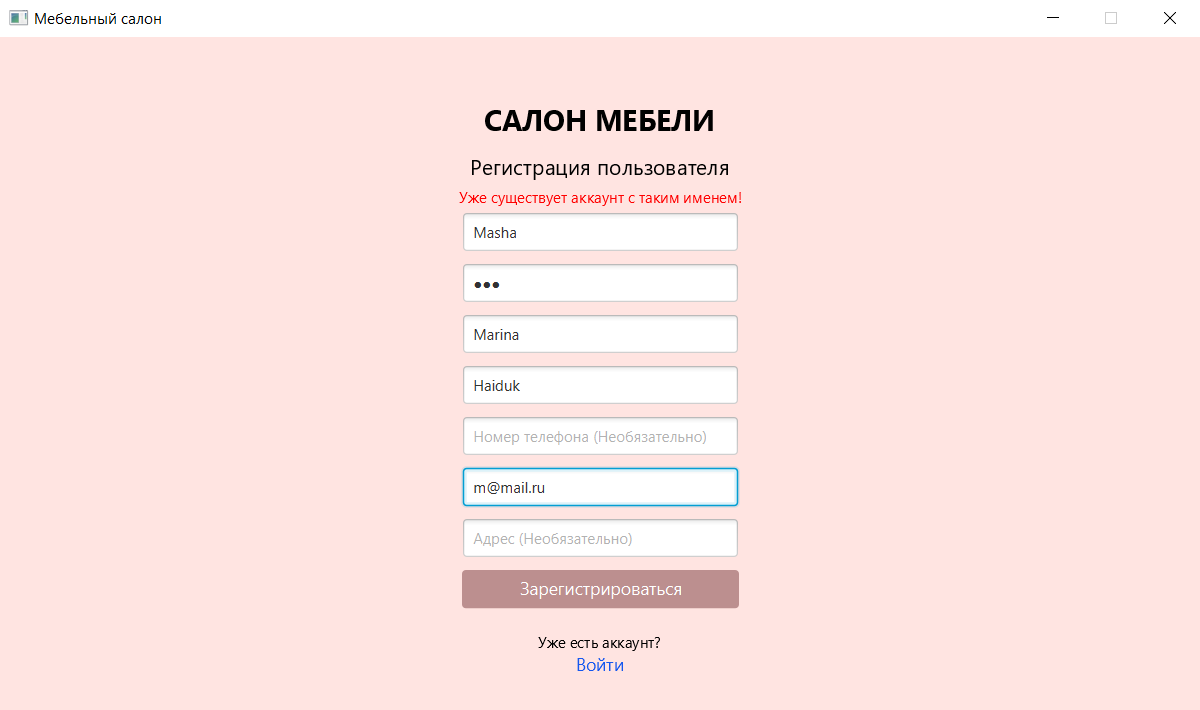


Рисунок 3.3 – Предупреждение об ошибке при регистрации

На следующем шаге при авторизации пользователь может не ввести ни одного значения или ввести значение только одного поля. В этом случае он увидит предупреждение, представленное на рисунке 3.4. При вводе имени пользователя, которого не существует, или при вводе неправильного пароля, клиент также увидит предупреждение, представленное на рисунке 3.5.

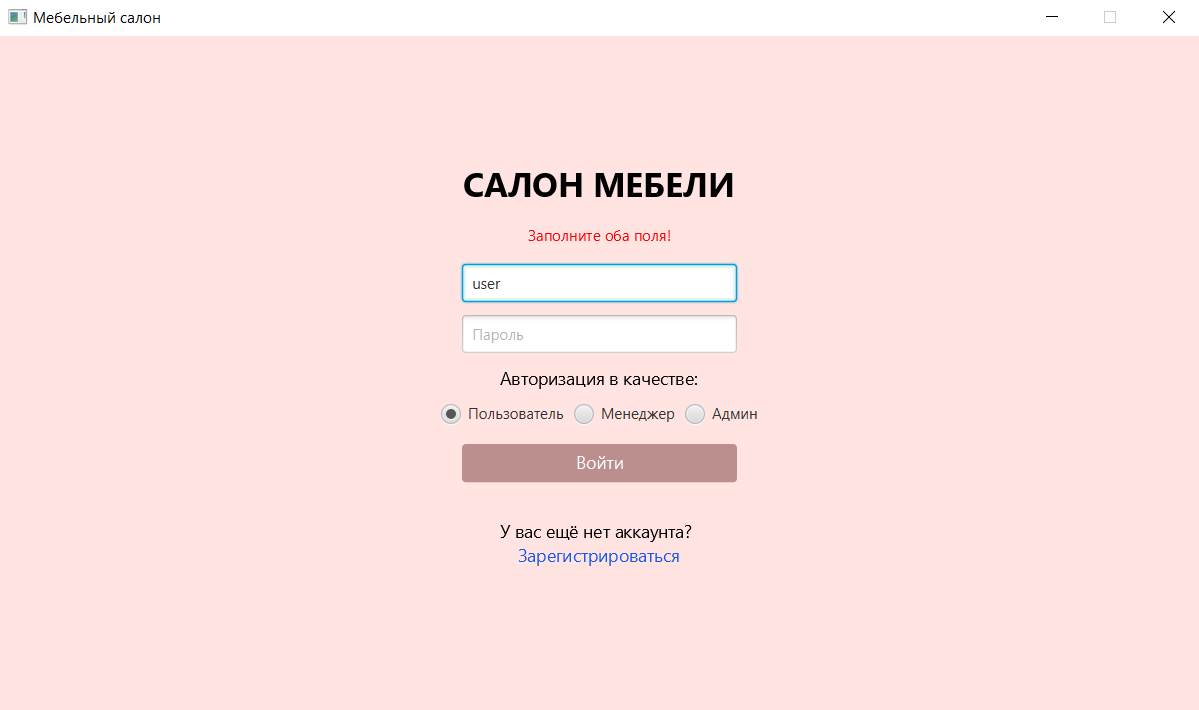


Рисунок 3.4 – Уведомление об ошибке при авторизации

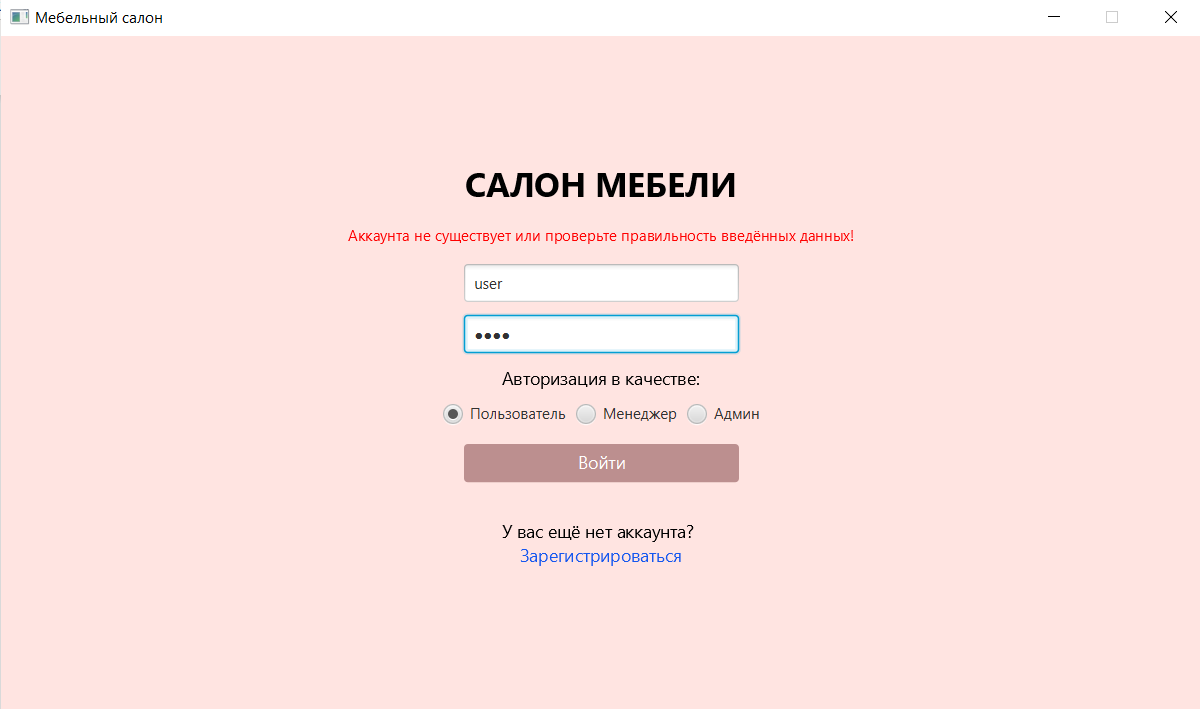


Рисунок 3.5 – Уведомление об ошибке при авторизации

При авторизации в качестве пользователя клиенту предоставлена форма фильтрации, где он может выбрать необходимые фильтры для поиска товаров. Это достаточно уязвимое место с точки зрения исключительных ситуаций. Пользователь может не заполнить нужные поля, заполнить их неправильно, ввести отрицательные значения или неверные границы. Во всех этих случаях пользователь увидит предупреждение о некорректно введенных фильтрах. Пример предупреждения о некорректно введенных фильтрах представлен на рисунках 3.6, 3.7.

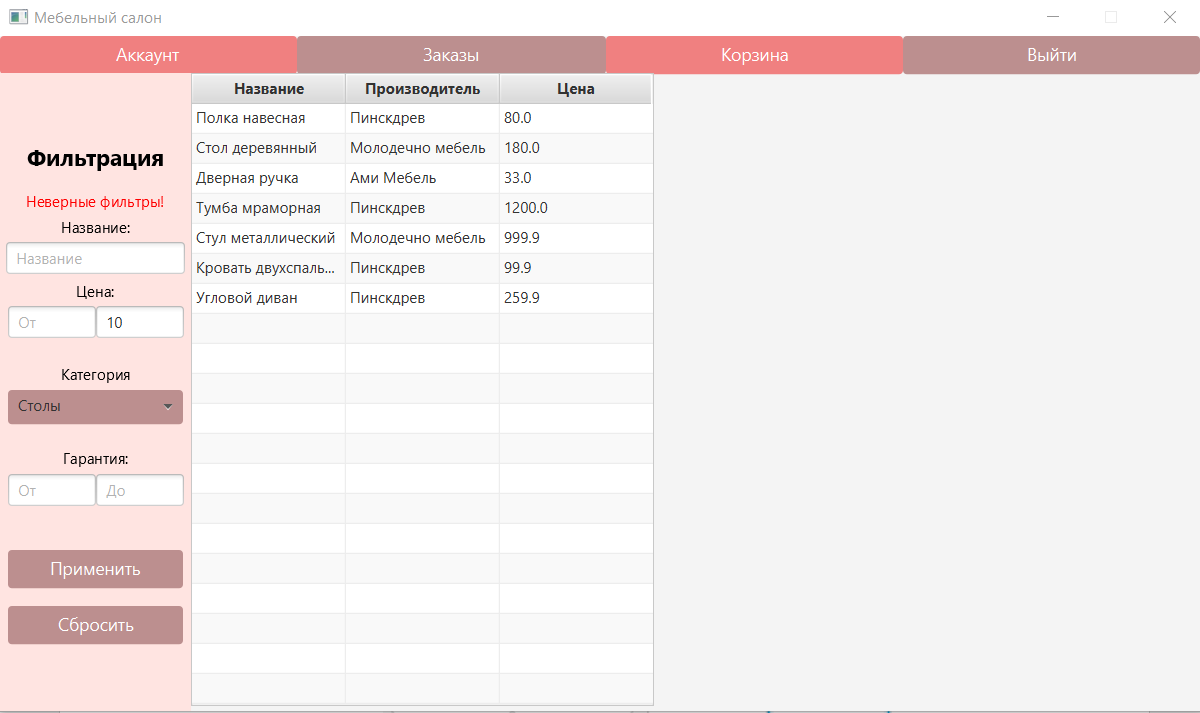


Рисунок 3.6 – Предупреждение о некорректно введённых фильтрах

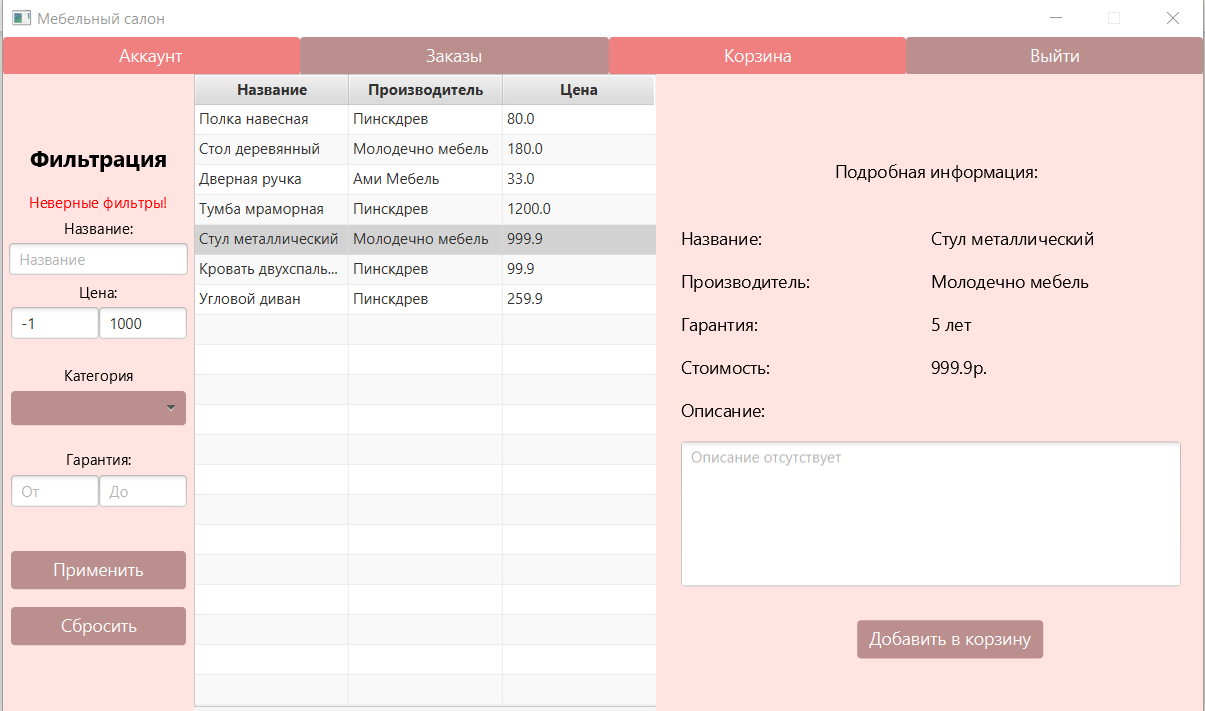


Рисунок 3.7 – Предупреждение о некорректно введённых фильтрах

Кроме того, у пользователя есть возможность изменять личную информацию. При изменении личной информации пользователь, так же, как и при регистрации, может не указать значения обязательный полей. В этом случае он получит предупреждение о неправильно введённых данных, представленное на рисунке 3.8.

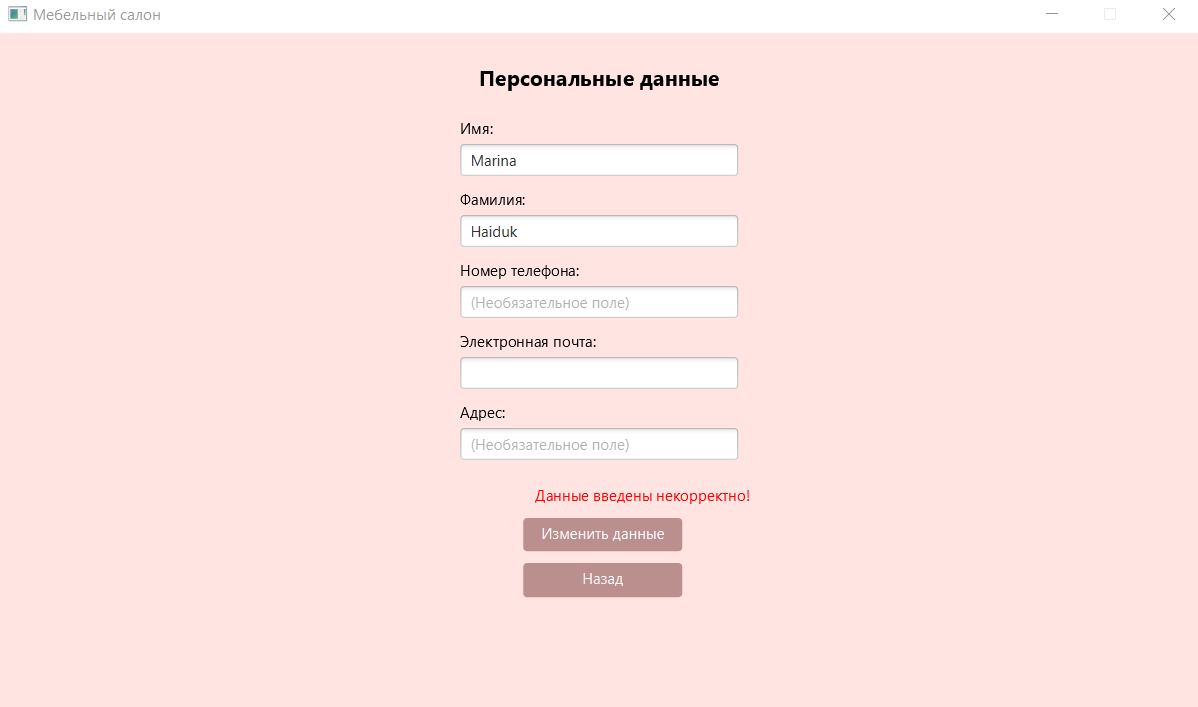


Рисунок 3.8 – Предупреждение о неправильно введённых данных

Таким образом было проведено тестирование работы базы данных, в результате которого все тесты были успешно выполнены. Также были протестированы основные моменты, где пользователь может нарушить ход выполнения программы путём создания исключительных ситуаций. В таких случая разрабатываемое программное средство выводит сообщения предупреждения, которые указывают, что пользователь ввел некоторые данные некорректно.

# 4. РУКОВОДСТВО ПО РАЗВЁРТЫВАНИЮ И ИПОЛЬЗОВАНИЮ ПРОГРАММНОГО СРЕДСТВА

## **4.1 Руководство по установке (развёртыванию) программного средства**

Поскольку разработанное программное средство является клиент-серверным приложением, необходимо выполнить установку как серверной части, так и клиентской.

Для установки серверной части необходимо иметь версию jdk 18 или выше. Также необходимо скачать JDBC.

Кроме того, необходимо установить какую-либо СУБД и запустить скрипт базы данных из приложения В. Рекомендуется использовать MySQL, поскольку он предоставляет специальный драйвер “mysql-connector-java” для работы с JDBC, который используется в разрабатываемом программном средстве.

Для настройки доступа приложения к СУБД необходимо внести изменения в файл hibernate.cfg.xml.

Сервер работает на порту 3333. Для изменения порта необходимо внести изменения в класс «Server».

При установке клиентской части в настройках конфигурации следует указать путь к javafx-sdk-18.0.1.

## **4.2 Руководство пользователя**

Основная цель руководства пользователя заключается в обеспечении пользователя необходимой информацией для самостоятельной работы с программой или автоматизированной системой.

После успешного запуска приложения у пользователя открывается стартовое окно, представленное на рисунке 4.1.

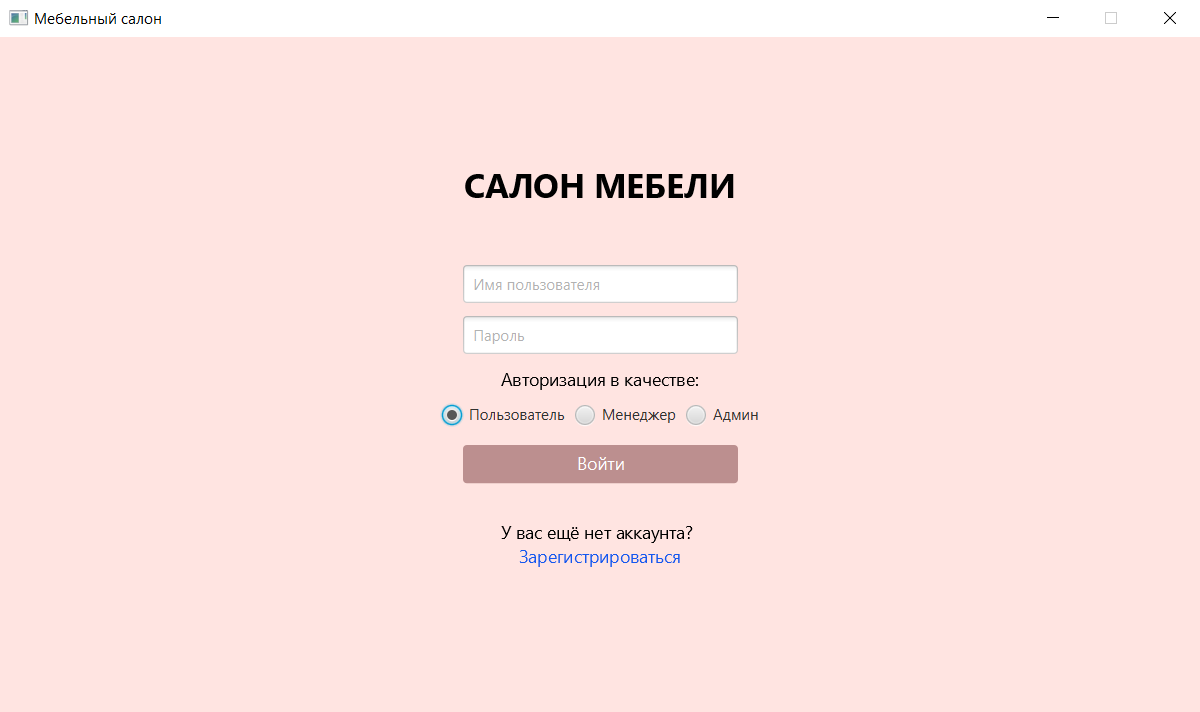


Рисунок 4.1 – Стартовое окно

После этого пользователю доступно 4 вариант дальнейший действий:

* войти в качестве пользователя;
* войти в качестве ассистента;
* войти в качестве администратора;
* зарегистрировать новый аккаунт пользователя.

Форма регистрации нового аккаунта пользователя представлена на рисунке 4.2.

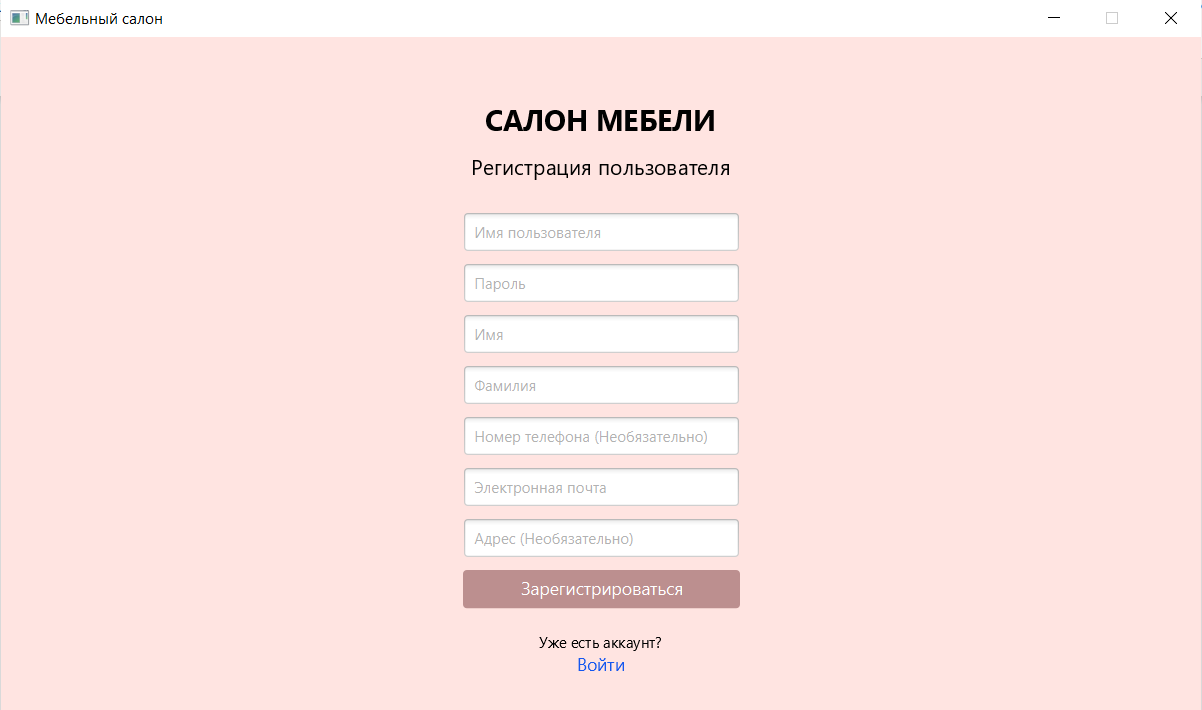


Рисунок 4.2 – Форма регистрации аккаунта пользователя

После ввода всех обязательных полей и создания имени пользователя, которого ещё нет в системе, происходит успешная регистрация пользователя и снова открывается стартовое окно для входа. На рисунке 4.3 представлена авторизация в качестве пользователя.

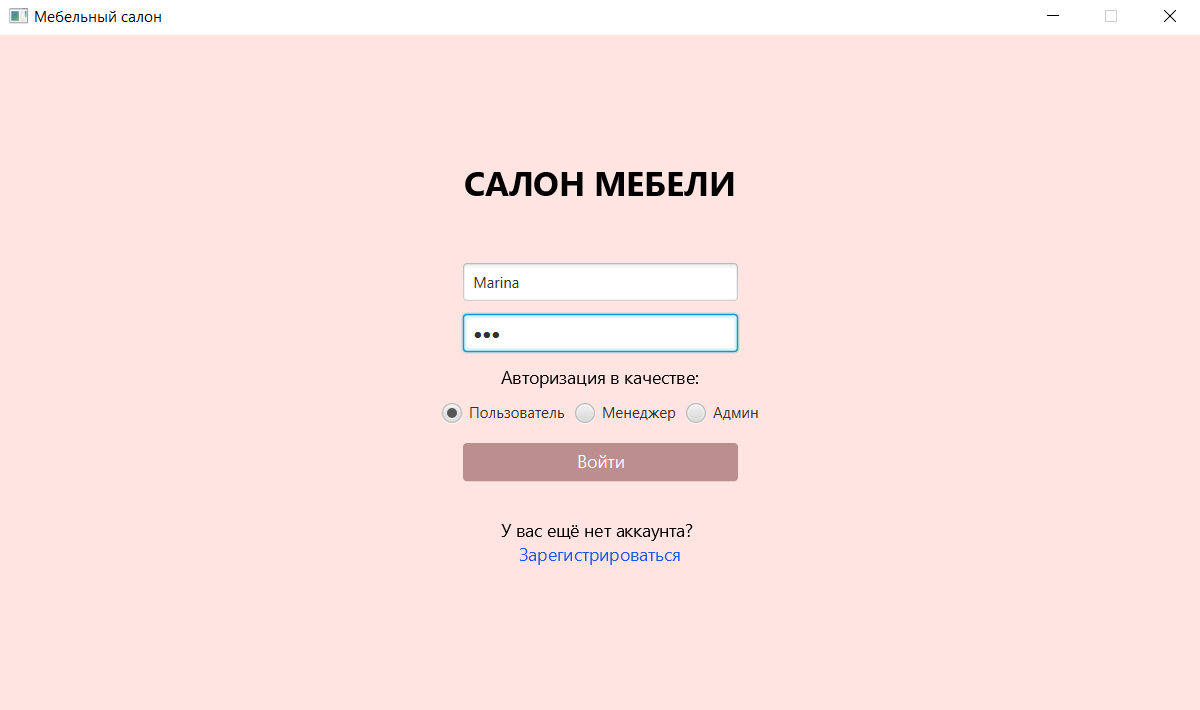


Рисунок 4.3 – Авторизация в качестве пользователя

После всех выполненных действий пользователь видит окно пользователя, в котором отображено список товаров магазина, форма фильтров и поиска, подробная информация выбранного товара, вкладки личной информации, заказов, корзины, а также кнопка «Выйти». На рисунке 4.4 представлено окно пользователя.

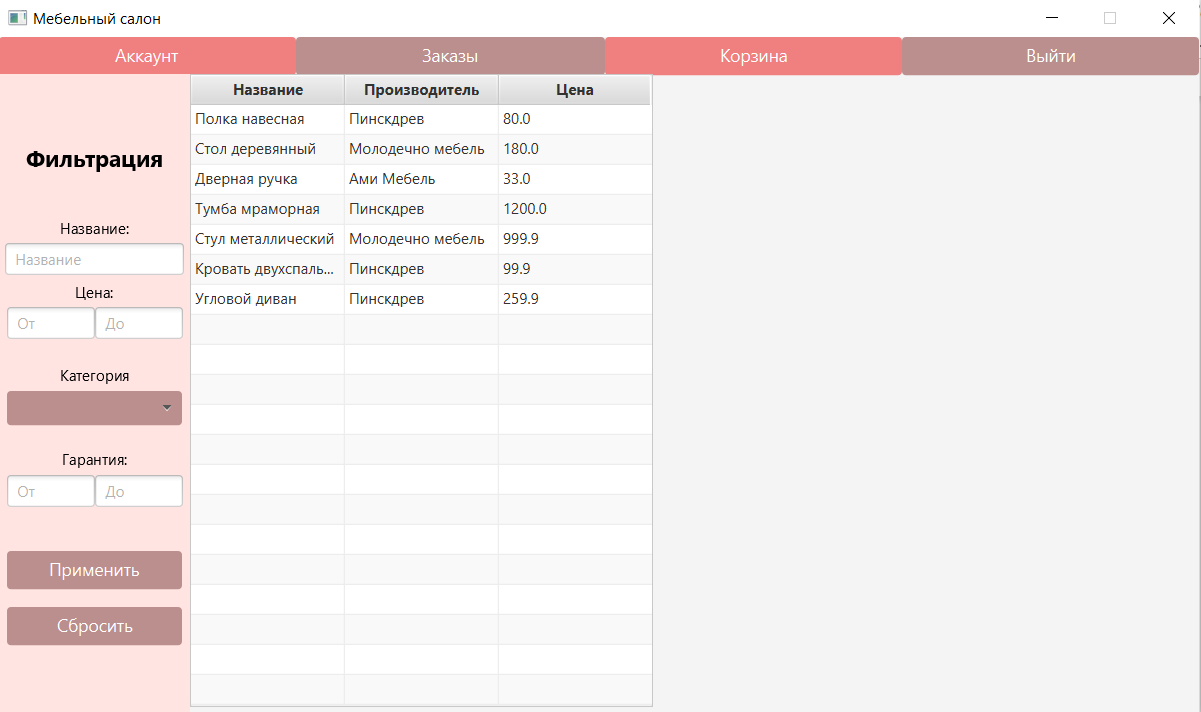


Рисунок 4.4 – Окно пользователя

В форме фильтрации и поиска пользователь имеет возможность настроить такие фильтры, как «Цена», «Гарантия», «Категория», а также можно произвести поиск по названию товара. После выбора необходимых фильтров и нажатия кнопки «Применить», показывается список товаров, удовлетворяющих заданным фильтрам. На рисунке 4.5 представлен результат поиска по выбранным фильтрам.

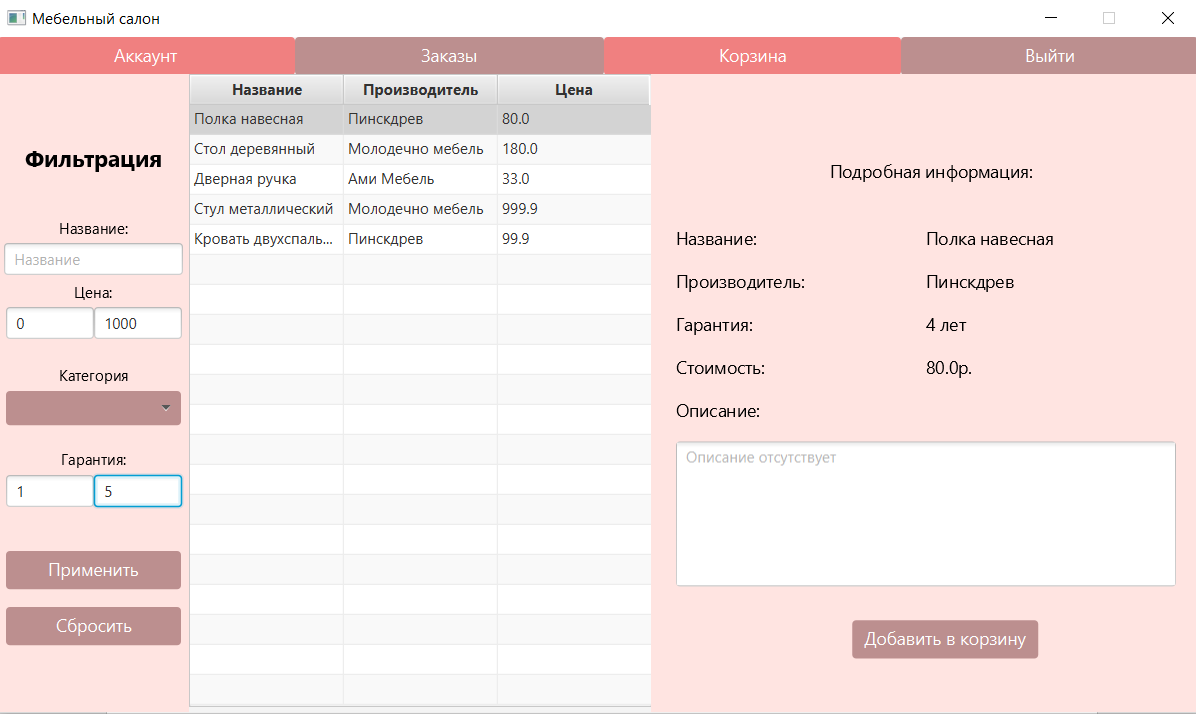


Рисунок 4.5 – Результат поиска по выбранным фильтрам

Под подробным описанием товара находится кнопка «Добавить в корзину», при нажатии на которую товар добавляется в пользовательскую корзину. Если товар уже в корзине, пользователь получает предупреждение об этом. На рисунке 4.6 представлено добавление товара в корзину.

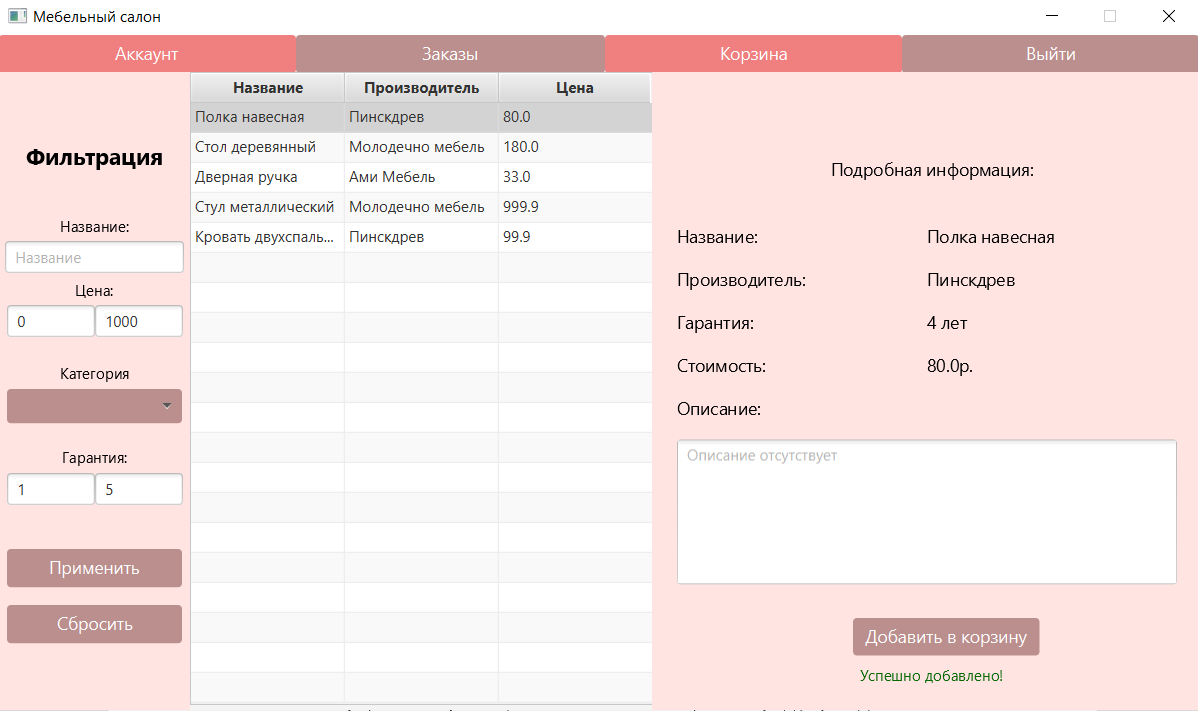


Рисунок 4.6 – Добавление товара в корзину

При нажатии на вкладку «Аккаунт» пользователю предоставляется возможность просмотреть и изменить личную информацию. Это показано на рисунке 4.7.

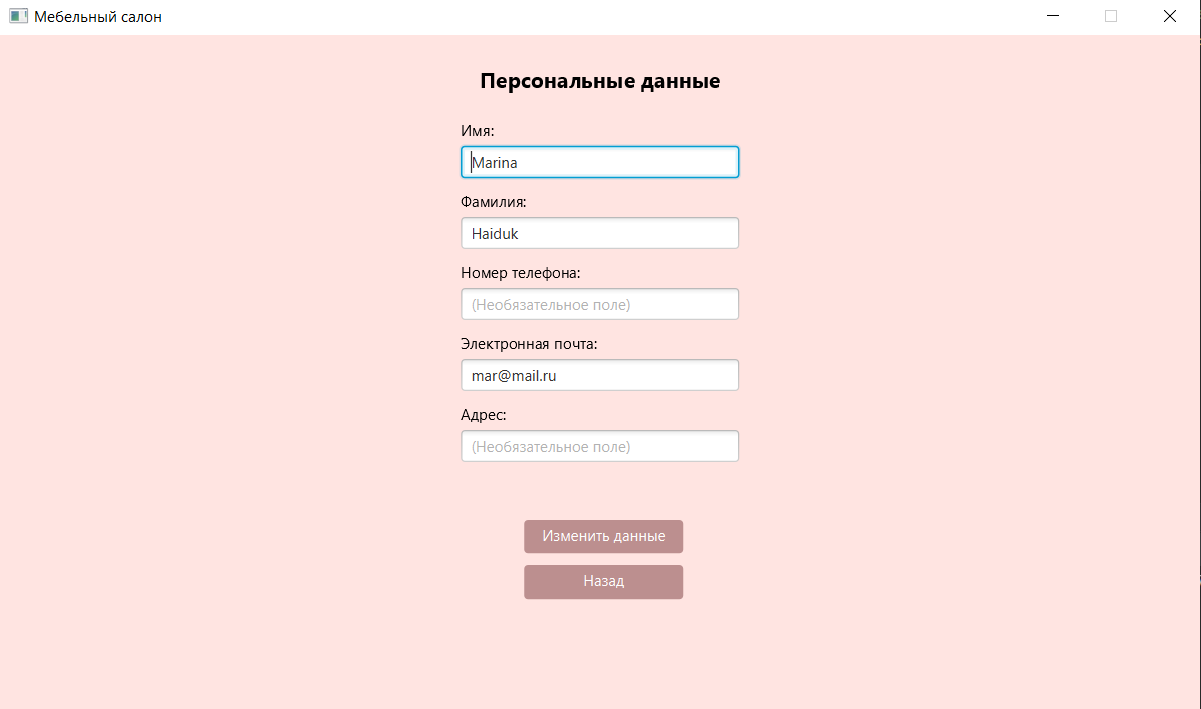


Рисунок 4.7 – Персональные данные пользователя

Как упоминалось выше, пользователь имеет возможность изменения личной информации. При верно введенных данных и нажатии кнопки «Изменить данные», пользователь получает сообщение о том, что данные изменены. Пример изменения данных представлен на рисунке 4.8.

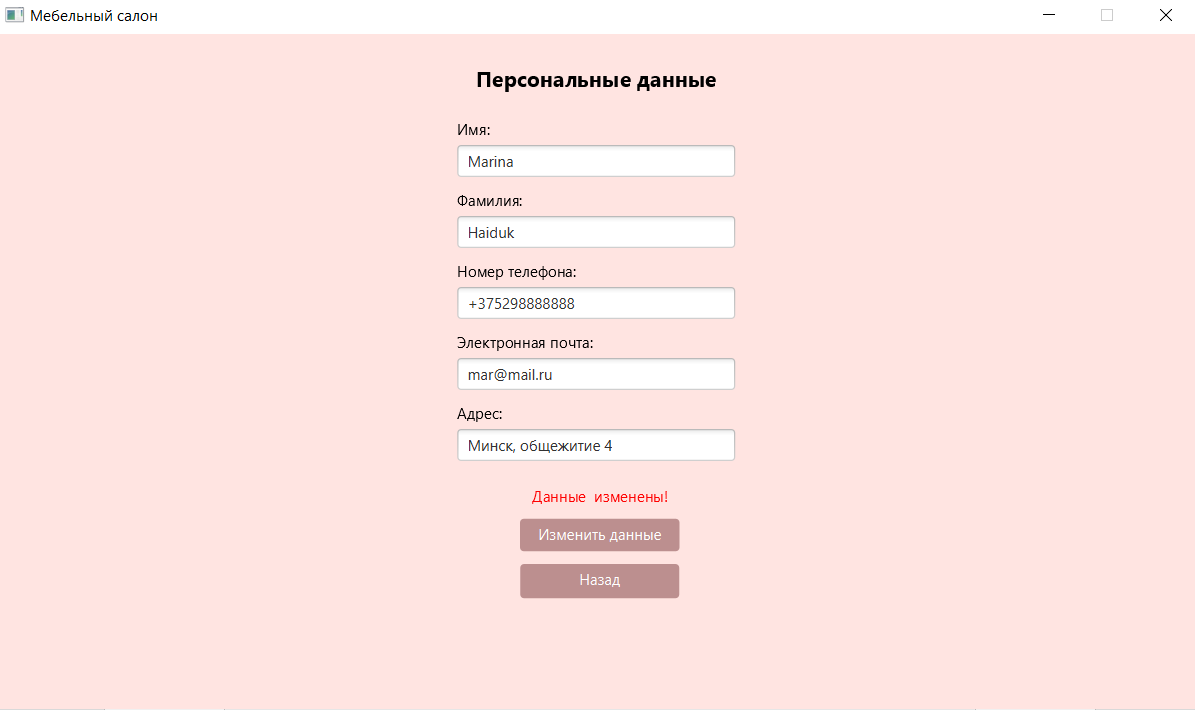


Рисунок 4.8 – Изменение персональных данных пользователя

При переходе на вкладку «Корзина» у пользователя открывается страница персональной корзины. Здесь он может увидеть добавленные товары, их количество, а также описание. Пользователь имеет возможность удалить товар из корзины, изменить его количество, оформить заказ, а также очистить корзину. На рисунке 4.9 представлена пользовательская корзина, а также изменение количества товара.

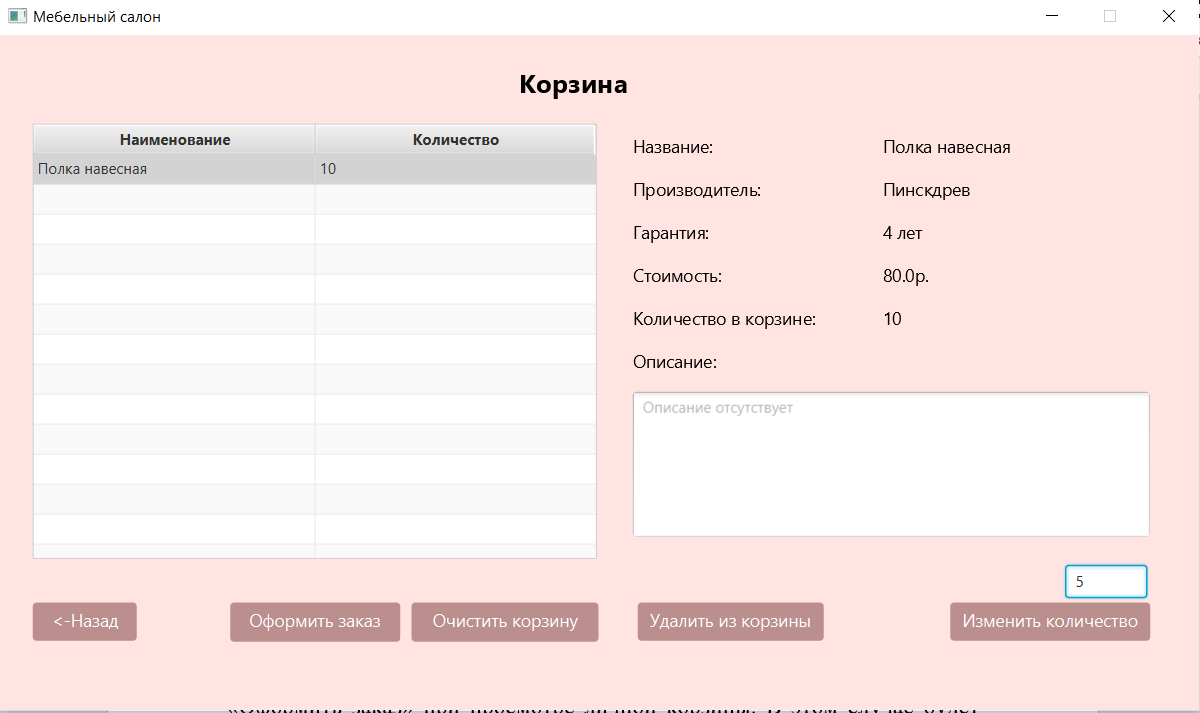


Рисунок 4.9 – Пользовательская корзина и изменение количества товара

При нажатии на кнопку «Оформить заказ» при просмотре личной корзины будет оформлен заказ с таким же составом товаров и количеством как в корзине. Чтобы получить всю информацию о заказе пользователю необходимо войти во вкладку «Заказы». На рисунке 4.10 представлено оформление заказа.

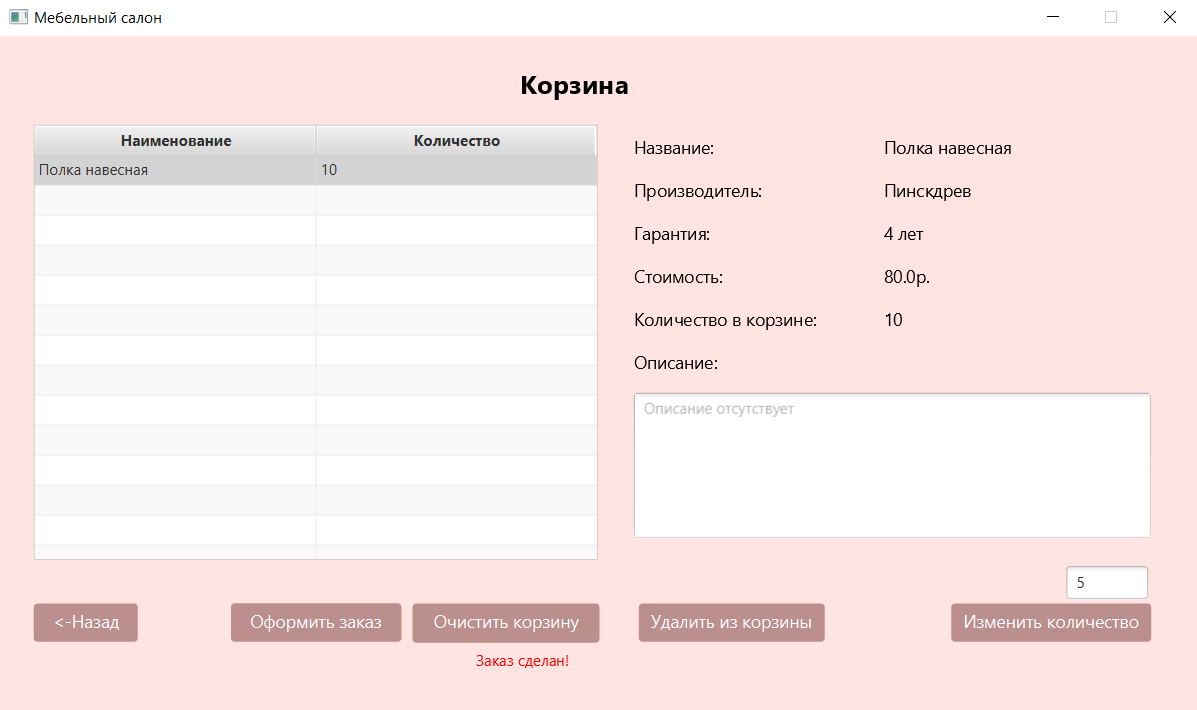


Рисунок 4.10 – Оформление заказа

Во вкладке «Заказы» пользователь видит информацию об активных заказах, он может просмотреть информацию о заказе, а также отменить его. На рисунке 4.11 представлено окно заказов пользователя.

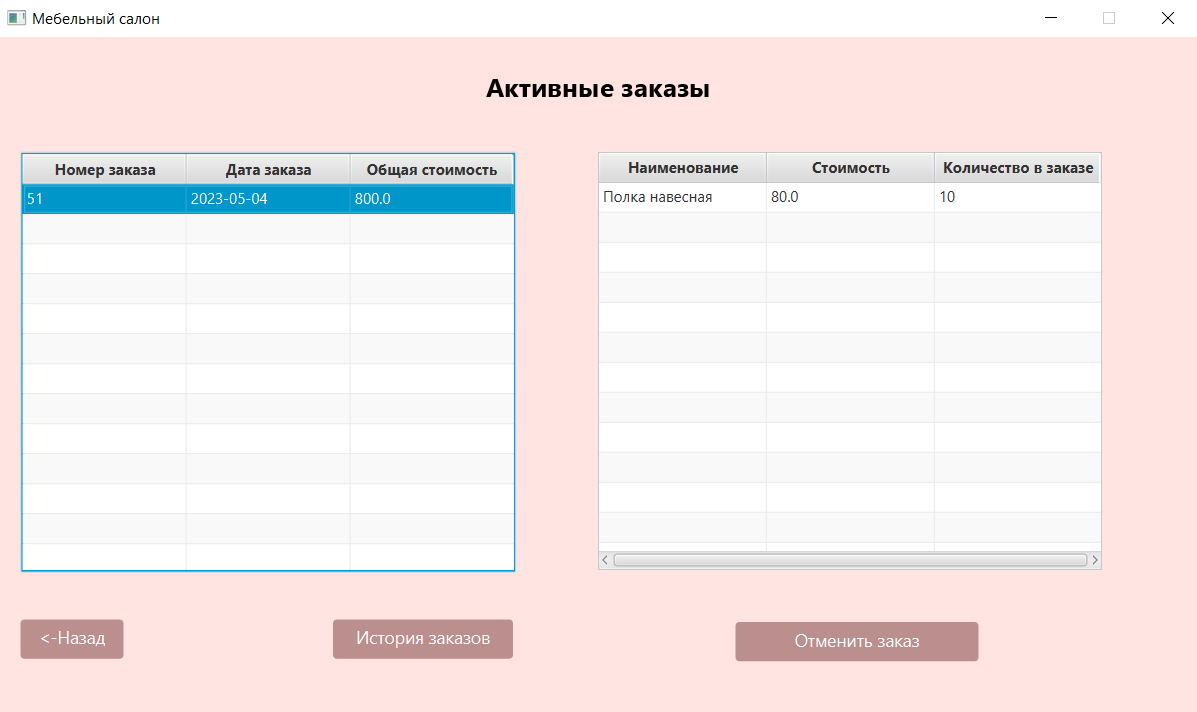


Рисунок 4.11 – Окно заказов

При нажатии на кнопку «История заказов» в этом же окне, пользователь переходит в окно просмотра истории заказов. Здесь отображено состояние заказа, имя и фамилию менеджера, который ответил на заказ, а также само сообщение ответа. На рисунке 4.12 представлено окно истории заказов.

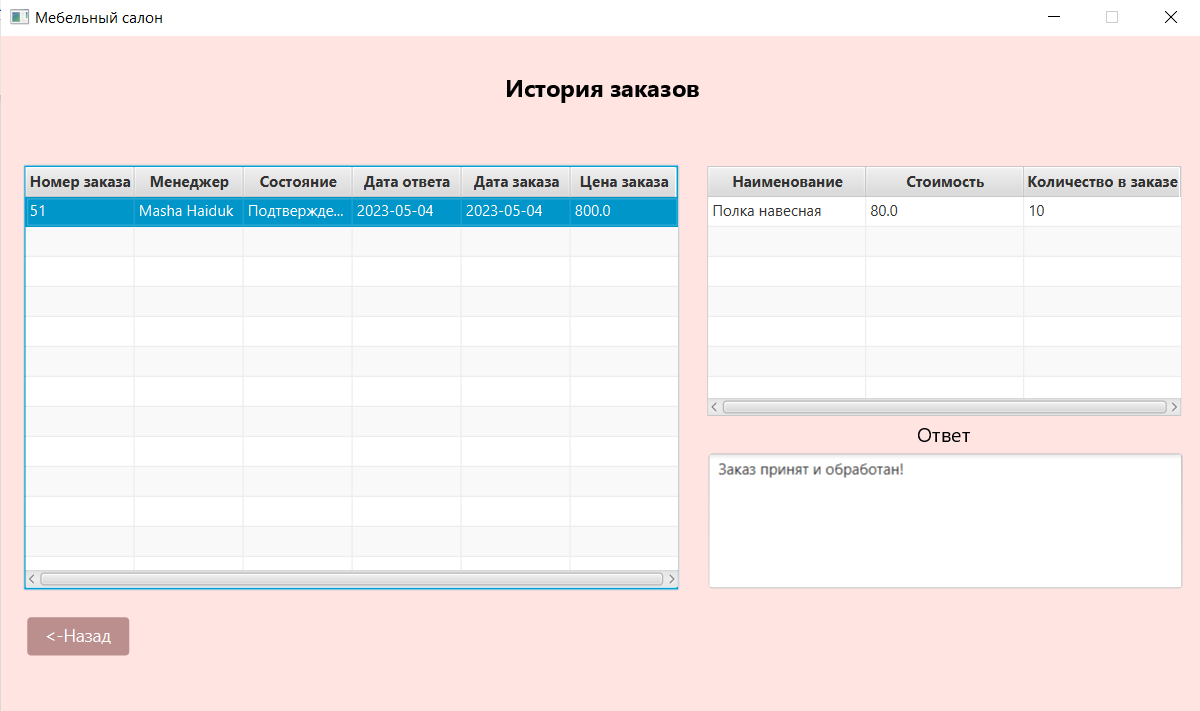


Рисунок 4.12 – Окно истории заказов

При нажатии на вкладку «Выйти» пользователю открывается стартовое окно приложения.

Для авторизации в качестве администратора необходимо ввести имя администратора и пароль, выбрать соответствующую роль и нажать на кнопку «Войти».

После этого открывается начальное окно администратора, где находятся такие элементы, как список всех пользователей, список всех ассистентов, форма регистрации нового ассистента, кнопки обновления списков пользователей и ассистентов, а также кнопка «Финансы», при нажатии на которую администратор переходит в окно работы с финансами. Стартовое окно администратора представлено на рисунке 4.13.

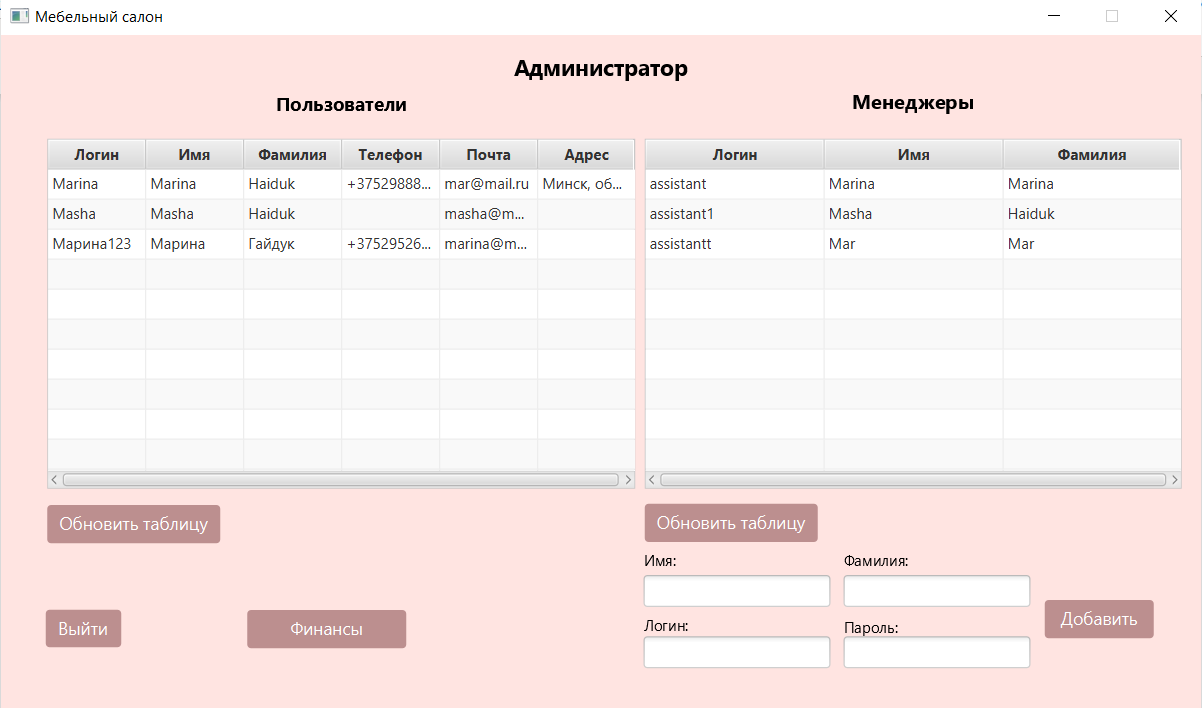


Рисунок 4.13 – Стартовое окно администратора

При выборе аккаунта из списков пользователей или менеджеров у администратора появляются кнопки: «Удалить пользователя» и «Удалить менеджера» соответственно. При нажатии на них происходит удаление выбранного аккаунта. При заполнении формы добавления нового менеджера и нажатии на кнопку «Добавить», администратор регистрирует нового менеджера и добавляет его в список менеджеров. Пример регистрации нового менеджера представлен на рисунке 4.14.

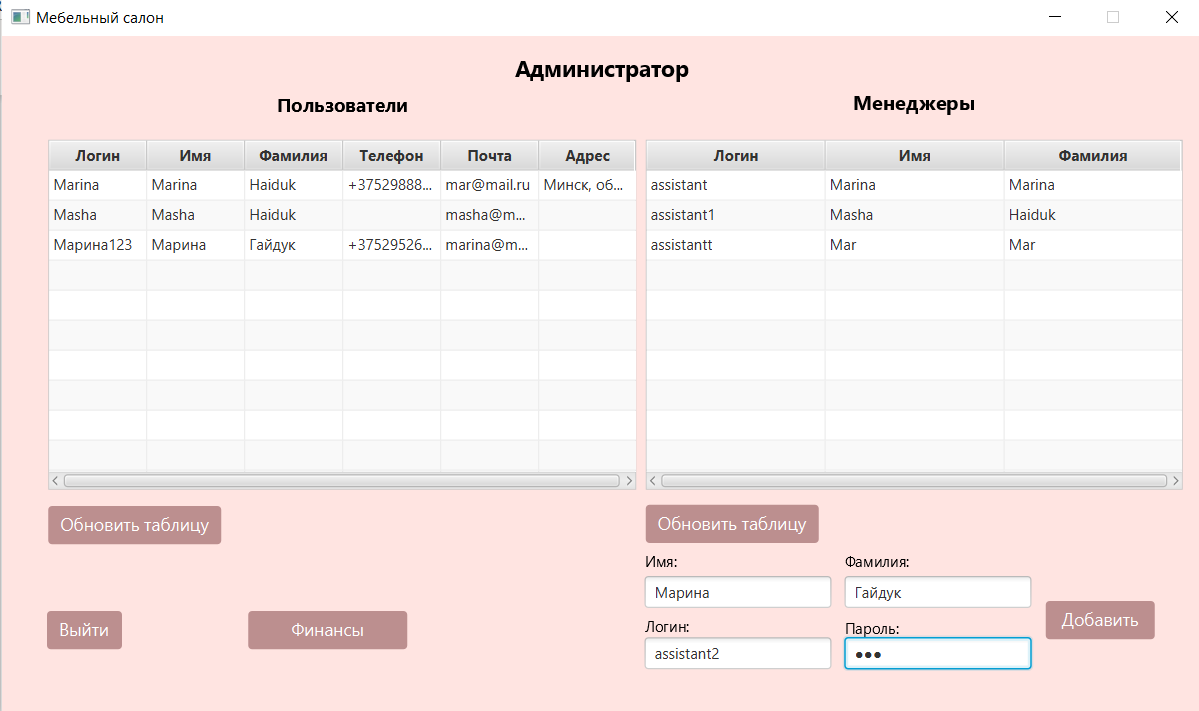


Рисунок 4.14 – Регистрация нового менеджера

При нажатии на кнопку «Финансы» открывается окно финансов. В этом окне администратор видит информацию о полной прибыли, а также множество форм, взаимодействуя с которыми можно получить информацию о прибыли за определенный промежуток времени, а также статистику за месяц или год. Окно финансов представлено на рисунке 4.15.

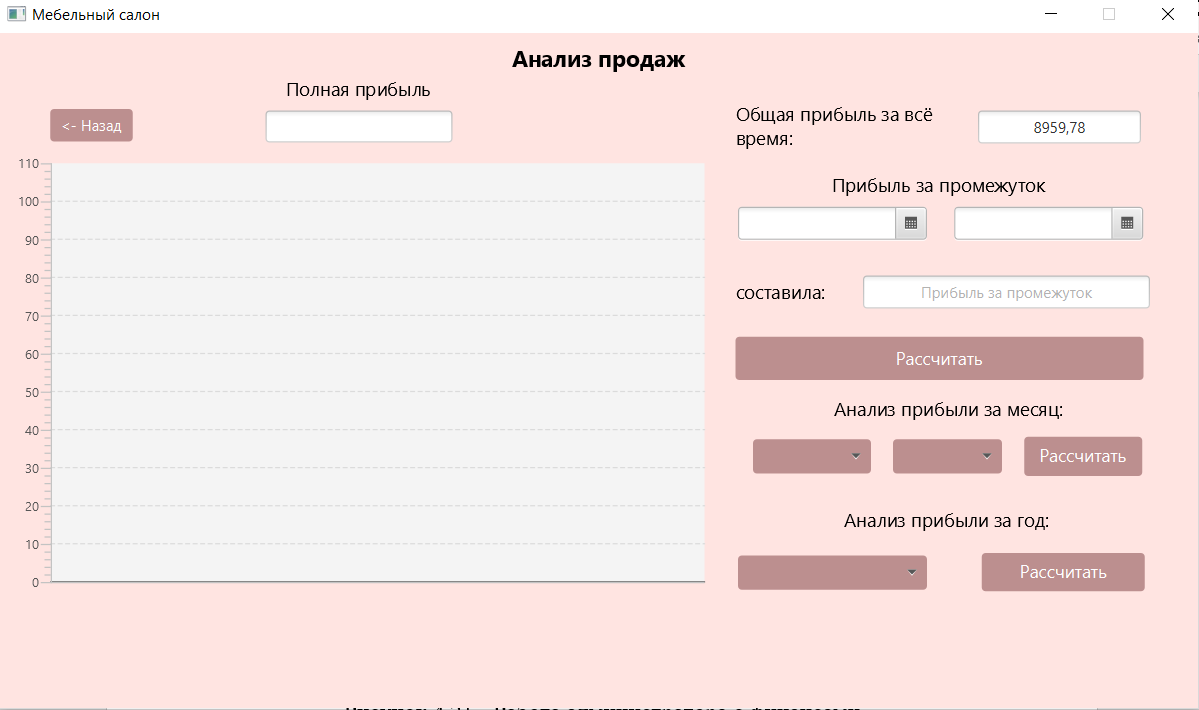


Рисунок 4.15 – Окно финансов

На рисунке 4.16 представлен пример получения прибыли за определенный промежуток времени.

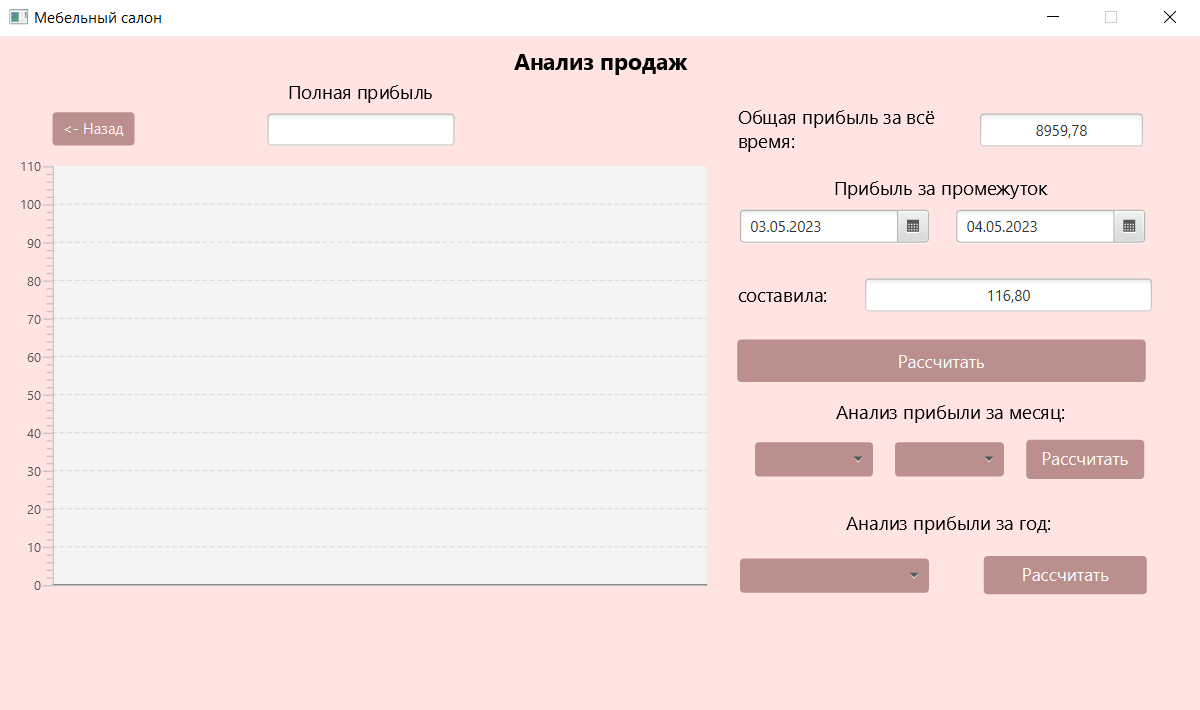


Рисунок 4.16 – Получение прибыли за промежуток времени

На рисунке 4.17 представлен пример получения анализа продаж за определённый месяц.

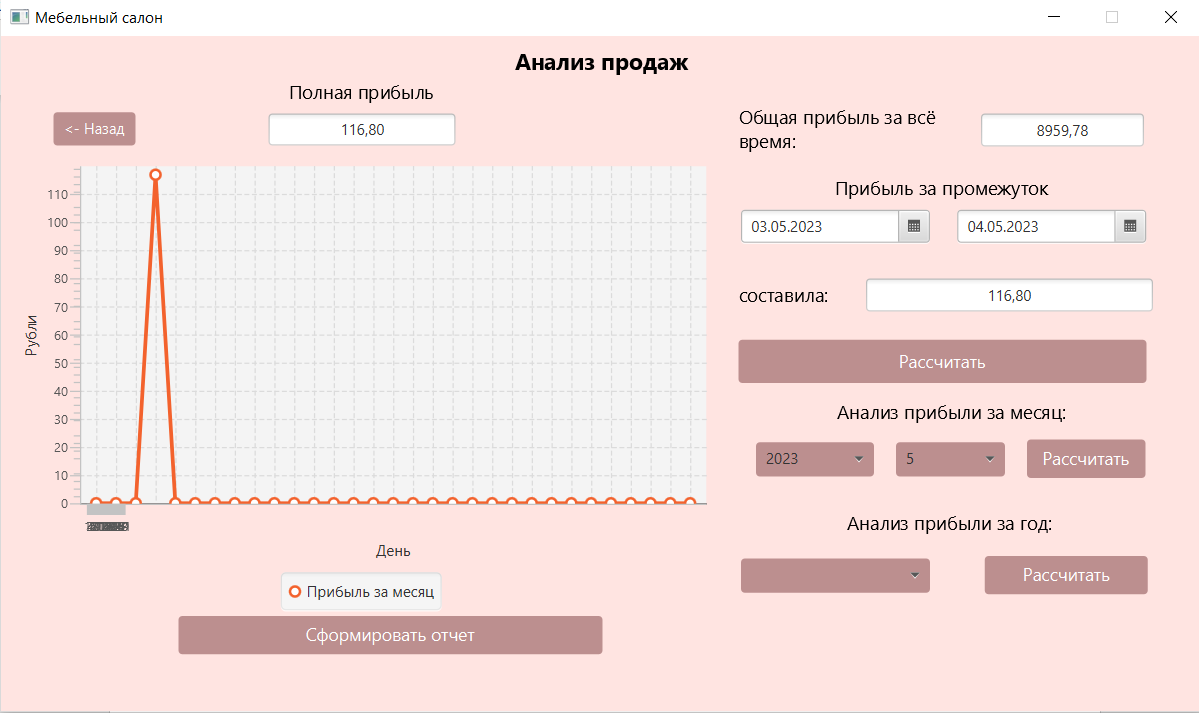


Рисунок 4.17 – Пример получения анализа продаж за определённый месяц

Пример получения анализа прибыли за определенный год представлен на рисунке 4.18.

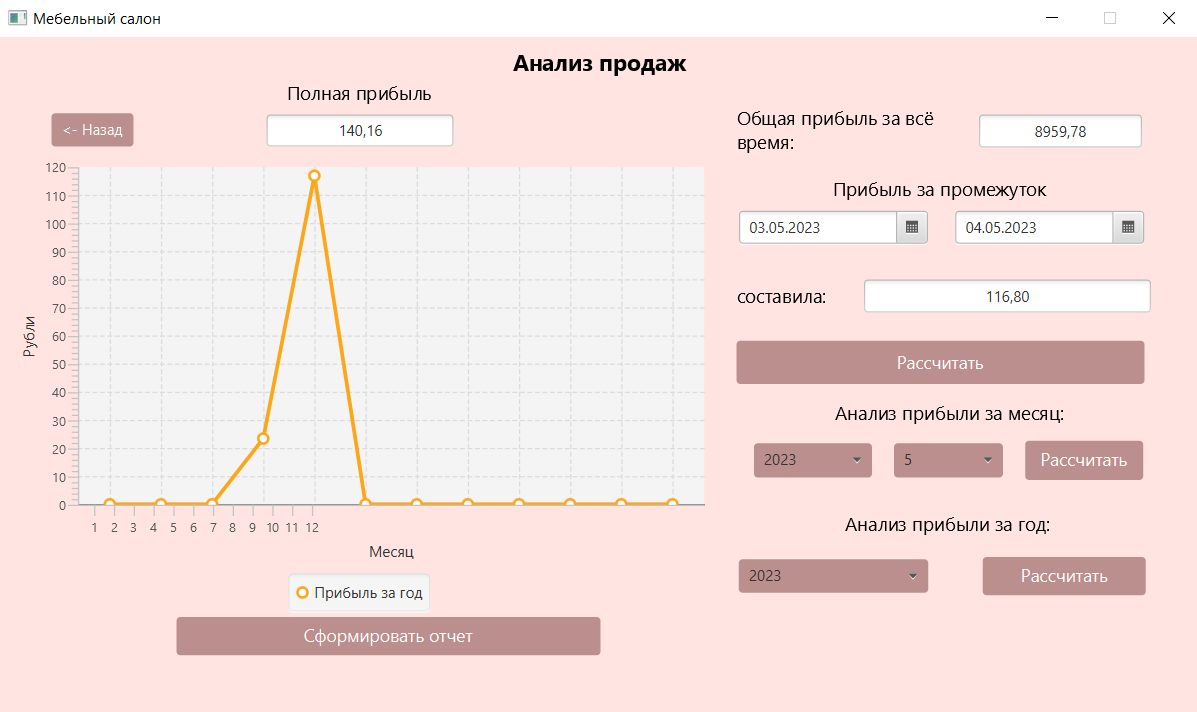


Рисунок 4.18 – Получение анализа прибыли за определённый год

При получении анализа продаж за год или месяц администратор имеет возможность сформировать отчет. Для этого ему необходимо нажать на кнопку «Сформировать отчет». Отчет сохраняется в виде файла на устройстве администратора и содержит полную информацию о прибыли за год или месяц. Пример сформированного отчета за определенный год приведен на рисунке 4.19.

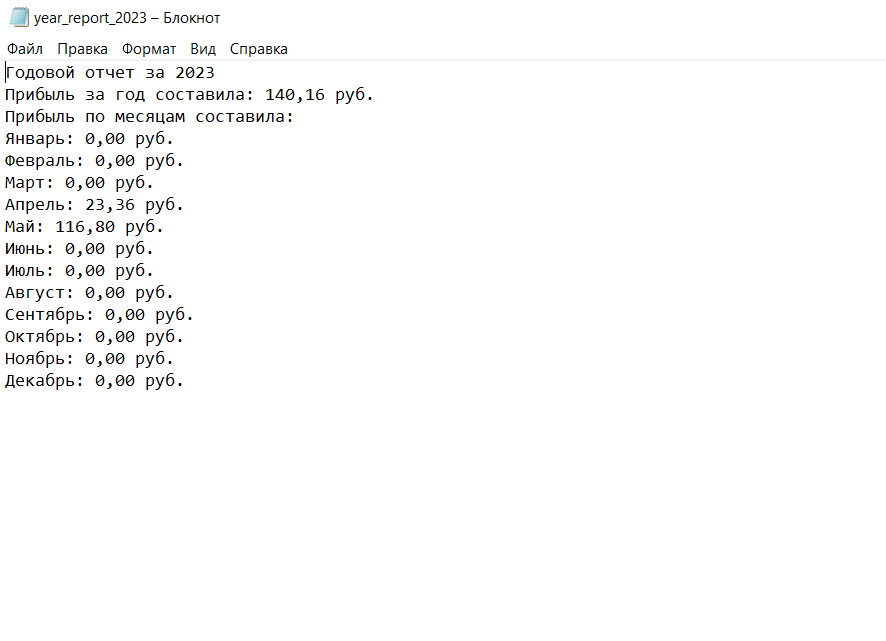


Рисунок 4.19 – Сформированный отчёт за 2023 год

Для входа в качестве ассистента необходимо ввести имя ассистента и пароль, выбрать соответствующую роль и нажать на кнопку «Войти».

После этого откроется стартовое окно ассистента, где находится перечень активных заказов от всех пользователей приложения. Также там находится информация о том, в качестве какого именно ассистента был выполнен вход, имеются кнопки выхода, просмотра склада товаров, а также кнопка обновления перечня заказов. Стартовое окно ассистента представлено на рисунке 4.20.

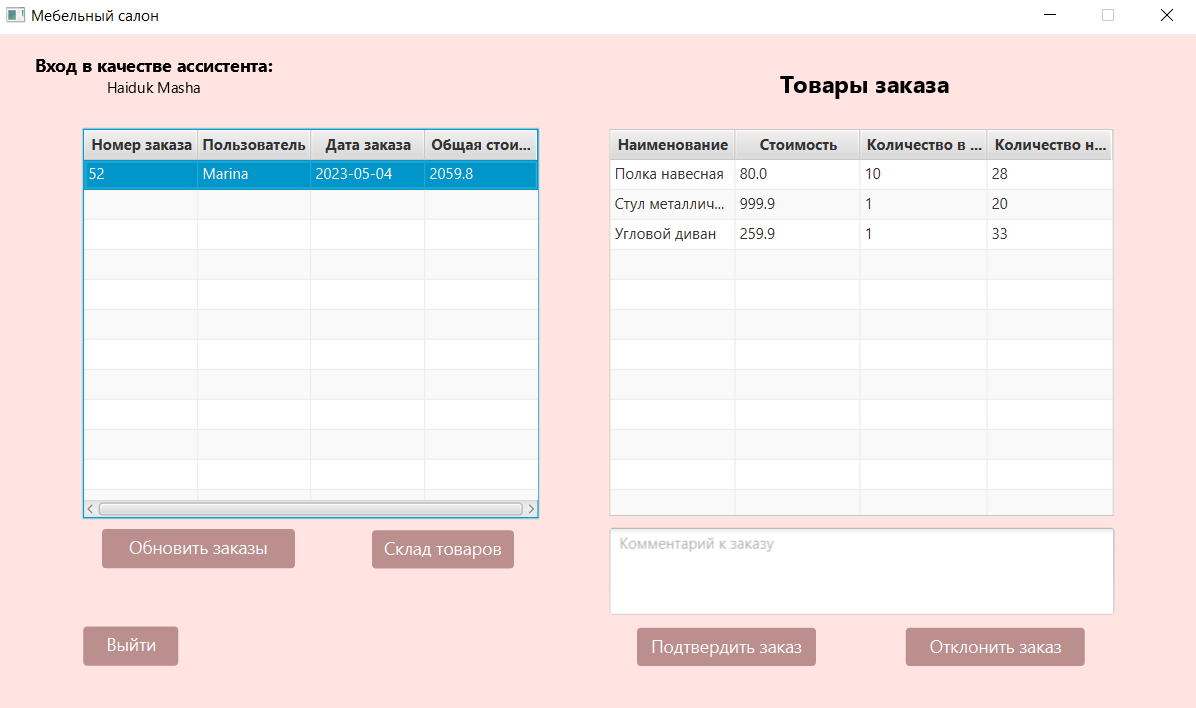


Рисунок 4.20 – Стартовое окно ассистента

При выборе заказа из списка ассистент получает подробную информацию о данном заказе, а также имеет возможность отклонить его, либо подтвердить, если на складе находится достаточное количество товара. Также ассистент при желании может оставить комментарий к ответу.

При нажатии на кнопку «Склад товаров» ассистент переходит в окно склада товаров. Здесь находится список всех товаров мебельного салона. Ассистент видит подробную информацию о каждом из товаров, а также их количество на складе. При выборе товара из списка ассистент может изменить любую информацию о товаре, или удалить его. На рисунке 4.21 представлено окно работы со складом.

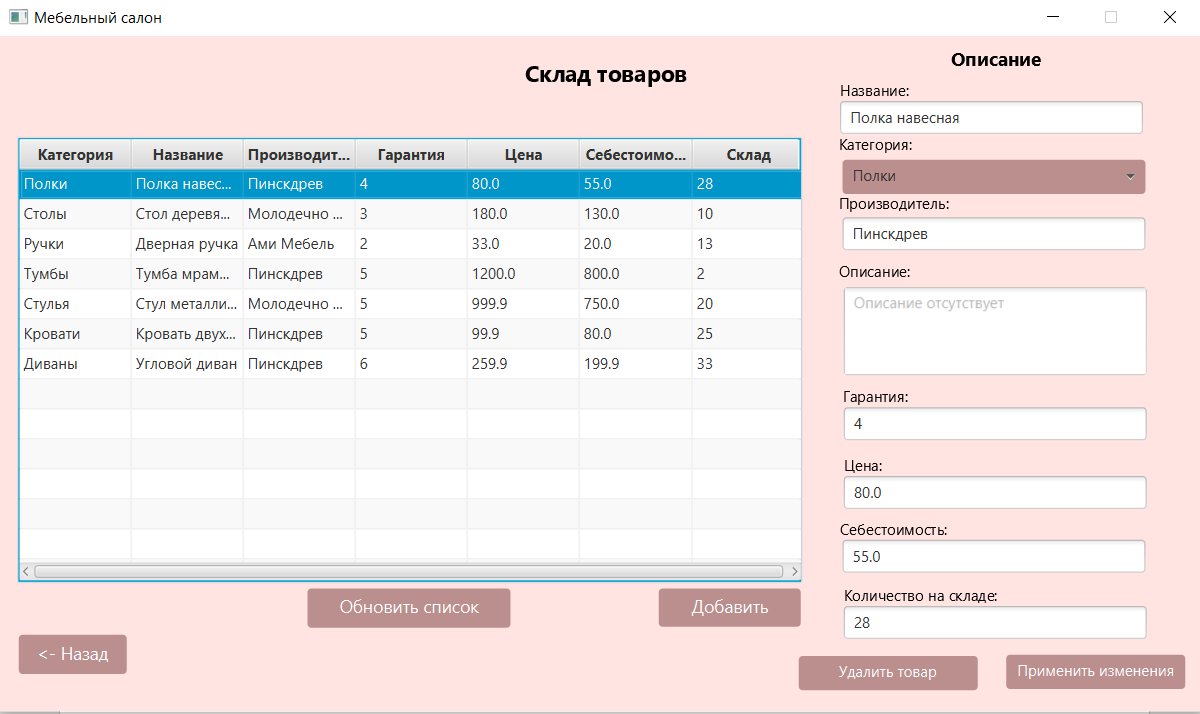


Рисунок 4.21 – Окно работы со складом

Также в этом окне при нажатии на кнопку «Добавить» ассистент имеет возможность добавить новый товар путём ввода информации о товаре в специальную форму, представленную на рисунке 4.22. Для того, чтобы добавить товар необходимо после заполнения информации о товаре нажать на кнопку «Добавить товар».

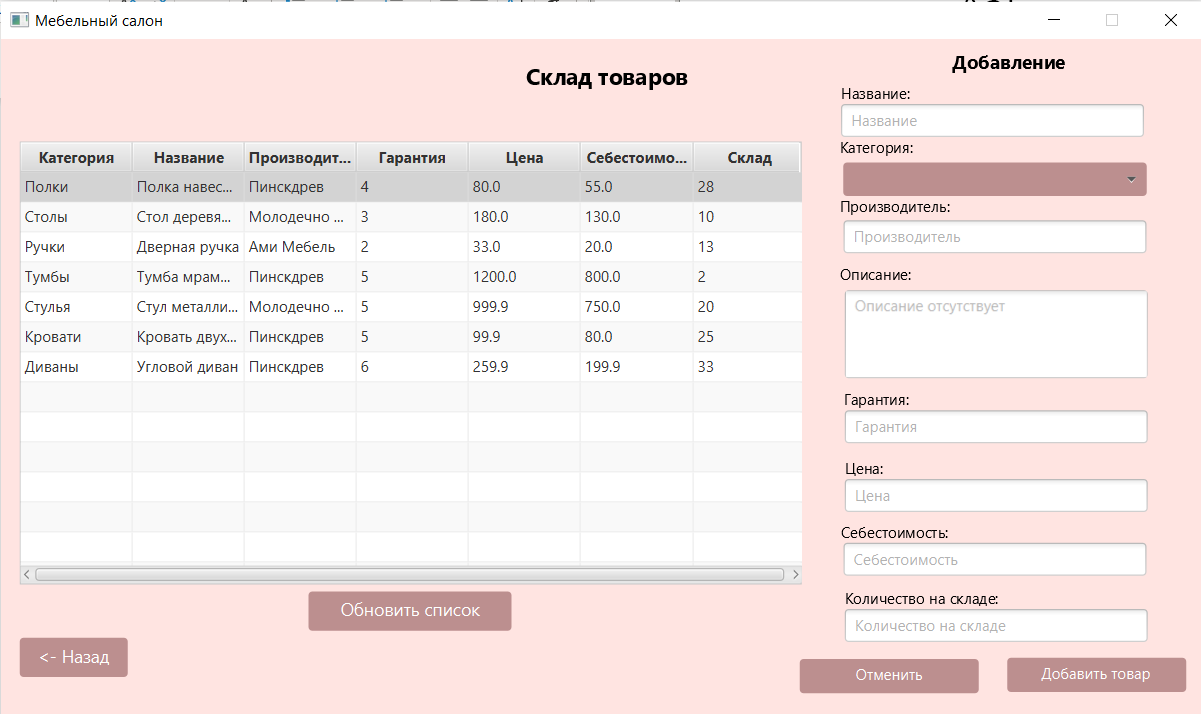


Рисунок 4.23 – Форма добавления нового товара

Таким образом, было представлено развёрнутое руководство пользователя, которое демонстрирует все варианты использования приложения в качестве пользователя, ассистента, а также администратора.

# ЗАКЛЮЧЕНИЕ

В ходе курсового проекта было разработано программное средство, представляющее собой автоматизированную систему учета продаж мебельной продукции и расчета прибыли. Программа имеет интуитивно понятный интерфейс, проста и удобна в использовании, защищена от ввода некорректных значений.

В ходе разработки программного средства использовались основные средства языка программирования Java, позволяющие наиболее эффективно реализовать данный проект на базе архитектуры клиент-сервер. Сервер обладает многопоточной архитектурой, следовательно программное средство имеет возможность одновременного подключения и обслуживания нескольких клиентов. Для хранения всей информации, относящейся к данному разработанному программному средству, используется база данных под управлением СУБД MySQL. Приложение протестировано при помощи фреймворка JUnit.

Для достижения поставленных целей в курсовом проекте была разработана объектная модель и ее программная реализация. При разработке объектной модели были определены ее составные части – классы и установлены связи между ними. Для более детального описания предметной области были разработаны UML-модели, которые позволили спроектировать основные программные схемы. Были построены блок-схемы алгоритма работы всего приложения и основных ее функций. Была разработана и описана диаграмма вариантов использования приложения, показывающая доступный администратору, ассистенту и пользователя функционал.

В ходе выполнения курсового проекта была описана инструкция развертывания приложения, а также руководство пользователя, которое поможет понять все особенности работы приложения.

Итогом курсового проекта является разработанное графическое оконное приложение, которое может помочь улучшить качество работы мебельных салонов и обслуживания клиентов.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Нотация IDEF0 [Электронный ресурс]. – Режим доступа: https://www.businessstudio.ru/wiki/docs/v4/doku.php/ru/csdesign/bpmodeling/idef0.

[2] Диаграмма вариантов использования UML [Электронный ресурс]. –

Режим доступа: <https://habr.com/ru/post/566218/>.

[3] Информационная модель [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C>.

[4] Основы методологии IDEF1X [Электронный ресурс]. – Режим доступа: <http://www.interface.ru/home.asp?artId=136>.

[5] UML [Электронный ресурс]. – Режим доступа: https://blog.skillfactory.ru/glossary/uml/.

[6] Диаграмма последовательности [Электронный ресурс]. – Режим доступа: https://flexberry.github.io/ru/fd\_sequence-diagram.html.

[7] Диаграмма развёртывания [Электронный ресурс]. – Режим доступа: https://flexberry.github.io/ru/fd\_deployment-diagram.html.

[8] Диаграмма классов [Электронный ресурс]. – Режим доступа: <https://flexberry.github.io/ru/gpg_class-diagram.html>.

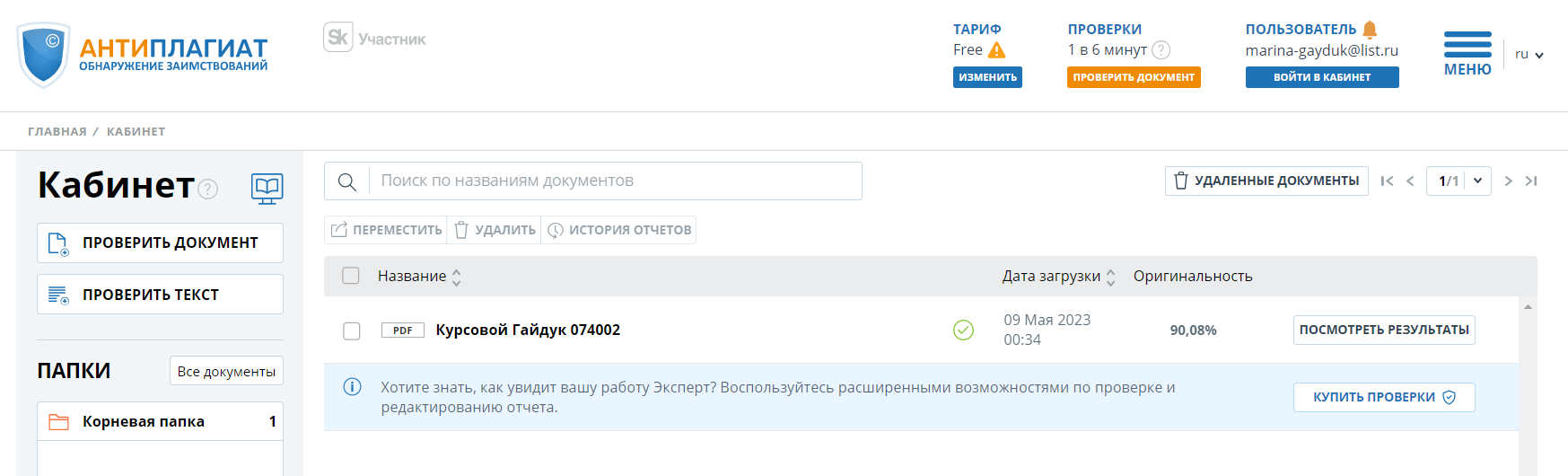
[9] Безопасность данных [Электронный ресурс]. – Режим доступа: <https://www.oracle.com/cis/security/database-security/what-is-data-security/>.

[10] Тестирование программного обеспечения [Электронный ресурс]. – Режим доступа: https://logrocon.ru/news/testing\_is.

# ПРИЛОЖЕНИЕ А

**(обязательное)**

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



# ПРИЛОЖЕНИЕ Б

**(обязательное)**

**Листинг кода алгоритмов, реализующих основную бизнес-логику**

package service;

import java.sql.Date;

import java.time.LocalDate;

import java.util.ArrayList;

import java.util.Calendar;

import java.util.List;

import db.dao.ServicedOrderDao;

import db.entity.ServicedOrder;

public class ProfitService {

private float calculateProfitPerServicedList(List<ServicedOrder> servicedOrderList)

{float totalProfit = 0.0f;

float income = 0.0f;

for (ServicedOrder servicedOrder : servicedOrderList) {

// налог на доход - 13% 03.23

income += servicedOrder.getOrderprice() \* 0.87f - servicedOrder.getOrderfirstcost();}

//налог на прибыль - 20% на 03.23

totalProfit = income \* 0.8f;

return totalProfit;}

public Date calculateDate(LocalDate localDate)

{ Calendar cal = Calendar.getInstance();

cal.set(Calendar.YEAR, localDate.getYear());

cal.set(Calendar.MONTH, localDate.getMonthValue() - 1);

cal.set(Calendar.DATE, localDate.getDayOfMonth());

cal.set(Calendar.HOUR, 0);

cal.set(Calendar.MINUTE, 0);

cal.set(Calendar.SECOND, 0);

return new Date(cal.getTimeInMillis());

public float getTotalProfit()

{var servicedOrderDao = new ServicedOrderDao(ServicedOrder.class);

float profit = calculateProfitPerServicedList(servicedOrderDao.getProfitServicedOrderList());

return profit;}

public float getProfitPerPeriod(LocalDate lowLocalDate, LocalDate highLocalDate)

{var servicedOrderDao = new ServicedOrderDao(ServicedOrder.class);

**ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ Б**

float profit = calculateProfitPerServicedList( servicedOrderDao.getProfitServicedOrderListPerPeriod(calculateDate(lowLocalDate), calculateDate(highLocalDate)));

return profit;}

public List<Float> getProfitPerMonth(LocalDate monthDate)

{var servicedOrderDao = new ServicedOrderDao(ServicedOrder.class);

var servicedOrderList = servicedOrderDao.getProfitServicedOrderList();

int year = monthDate.getYear();

int month = monthDate.getMonthValue();

var servicedOrderListByMonth = new ArrayList<ServicedOrder>();

for (ServicedOrder servicedOrder : servicedOrderList) {

if(servicedOrder.getAnswerdate().toLocalDate().getYear() == year &&

servicedOrder.getAnswerdate().toLocalDate().getMonthValue() == month)

{servicedOrderListByMonth.add(servicedOrder);}}

servicedOrderList.clear();

int maxMonthDay = 0;

if(monthDate.getMonthValue() != 2)

maxMonthDay = monthDate.getMonth().maxLength();

else if(monthDate.isLeapYear())

maxMonthDay = 29;

else

maxMonthDay = 28;

List<Float> monthDayProfitList = new ArrayList<Float>();

float totalProfitPerMonth = 0.0f;

for (int i = 0; i < maxMonthDay; i++)

{float profitPerDay = 0.0f;

for (ServicedOrder servicedOrder : servicedOrderListByMonth) {

if(servicedOrder.getAnswerdate().toLocalDate().getDayOfMonth() == i + 1)

profitPerDay += 0.8f \* (servicedOrder.getOrderprice() \* 0.87f - servicedOrder.getOrderfirstcost());}

monthDayProfitList.add(profitPerDay);

totalProfitPerMonth += profitPerDay;}

monthDayProfitList.add(totalProfitPerMonth);

return monthDayProfitList;}

public List<Float> getProfitPerYear(LocalDate yearDate)

{var servicedOrderDao = new ServicedOrderDao(ServicedOrder.class);

var servicedOrderList = servicedOrderDao.getProfitServicedOrderList();

int year = yearDate.getYear();

**ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ Б**

var servicedOrderListByYear = new ArrayList<ServicedOrder>();

for (ServicedOrder servicedOrder : servicedOrderList) {

if (servicedOrder.getAnswerdate().toLocalDate().getYear() == year)

servicedOrderListByYear.add(servicedOrder);}

servicedOrderList.clear();

var monthProfitList = new ArrayList<Float>();

var totalProfitPerYear = 0.0f;

for(int i = 0; i < 12; i++)

{float profitPerMonth = 0.0f;

for (ServicedOrder servicedOrder : servicedOrderListByYear) {

if (servicedOrder.getAnswerdate().toLocalDate().getMonthValue() == i + 1)

profitPerMonth += 0.8f \* (servicedOrder.getOrderprice() \* 0.87f - servicedOrder.getOrderfirstcost());}

monthProfitList.add(profitPerMonth);

totalProfitPerYear += profitPerMonth;}

monthProfitList.add(totalProfitPerYear);

return monthProfitList;}}

package controllers;

import java.time.LocalDate;

import request.commands.ConfirmCommands;

import request.controller.BaseRequestController;

import service.ProfitService;

public class FinanceController extends BaseRequestController{

private ProfitService profitService;

public FinanceController(BaseRequestController contr) {

super(contr);

profitService = new ProfitService();}

public void sendFullProfit()

{sendResponse(ConfirmCommands.class, ConfirmCommands.SUCCESSFULLY);

try {float fullProfit = profitService.getTotalProfit();

outputObjectStream.writeObject(fullProfit);

} catch (Exception e) {

e.getStackTrace();}}

public void sendProfitPerPeriod()

{sendResponse(ConfirmCommands.class, ConfirmCommands.SUCCESSFULLY);

try {

var lowLocalDate = (LocalDate)inputObjectStream.readObject();

**ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ Б**

var highLocalDate = (LocalDate)inputObjectStream.readObject();

var profitPerPeriod = profitService.getProfitPerPeriod(lowLocalDate, highLocalDate);

outputObjectStream.writeObject(profitPerPeriod);

} catch (Exception e) {

e.getStackTrace();}}

public void sendAnalysisPerMonth()

{sendResponse(ConfirmCommands.class, ConfirmCommands.SUCCESSFULLY);

try {

var monthLocalDate = (LocalDate)inputObjectStream.readObject();

var monthDayProfitList = profitService.getProfitPerMonth(monthLocalDate);

outputObjectStream.writeObject(monthDayProfitList);

} catch (Exception e) {

e.getStackTrace();}}

public void sendAnalysisPerYear()

{sendResponse(ConfirmCommands.class, ConfirmCommands.SUCCESSFULLY);

try {

var yearLocalDate = (LocalDate)inputObjectStream.readObject();

var monthProfitList = profitService.getProfitPerYear(yearLocalDate);

outputObjectStream.writeObject(monthProfitList);

} catch (Exception e) {

e.getStackTrace();}}}

package gui.controller;

import java.io.IOException;

import java.time.LocalDate;

import java.util.ArrayList;

import java.util.List;

import controllers.FinanceController;

import gui.GUI;

import javafx.fxml.FXML;

import javafx.scene.chart.CategoryAxis;

import javafx.scene.chart.LineChart;

import javafx.scene.chart.NumberAxis;

import javafx.scene.chart.XYChart.Data;

import javafx.scene.chart.XYChart.Series;

import javafx.scene.control.ChoiceBox;

import javafx.scene.control.DatePicker;

import javafx.scene.control.TextField;

**ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ Б**

public class AdminFinanceSceneController {

@FXML private LineChart<String, Float> profitLineChart;

@FXML private CategoryAxis profitXAxis;

@FXML private NumberAxis profitYAxis;

@FXML private TextField fullProfitField, profitPerPerioField;

@FXML private DatePicker lowDatePicker, highDatePicker;

@FXML private ChoiceBox<Integer> analysisPerMonthYearChoice, analysisPerMonthMonthChoice,

analysisPerYearYearChoice;

@FXML private TextField fullProfitByChart;

@FXML

public void initialize()

{RefreshFullProfit();

FillChoiceBoxes();}

private void RefreshFullProfit()

{float fullProfit = FinanceController.getFullProfit();

fullProfitField.setText(String.format("%.2f", fullProfit));}

private void FillChoiceBoxes()

{var yearList = new ArrayList<Integer>();

for(int i = 1990; i < 2024; i++)

yearList.add(i);

var monthList = new ArrayList<Integer>();

for(int i = 1; i < 13; i++)

monthList.add(i);

analysisPerMonthYearChoice.getItems().setAll(yearList);

analysisPerYearYearChoice.getItems().setAll(yearList);

analysisPerMonthMonthChoice.getItems().setAll(monthList);}

@FXML

private void getProfitPerPeriod()

{var lowDate = lowDatePicker.getValue();

var highDate = highDatePicker.getValue();

if(lowDate == null || highDate == null)

return;

var profitPerPeriod = FinanceController.getProfitPerPeriod(lowDate, highDate);

profitPerPerioField.setText(String.format("%.2f", profitPerPeriod));}

@FXML

private void analysisPerMonth()

{if(analysisPerMonthYearChoice.getValue() == null ||

**ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ Б**

analysisPerMonthMonthChoice.getValue() == null)

return;

var monthLocalDate = LocalDate.of(analysisPerMonthYearChoice.getValue(),

analysisPerMonthMonthChoice.getValue(),

1);

var monthDayProfitList = FinanceController.getAnalysisPerMonth(monthLocalDate);

updateChart(monthDayProfitList, "День", "Прибыль за месяц");}

@FXML

private void analysisPerYear()

{if(analysisPerYearYearChoice.getValue() == null)

return;

var yearLocalDate = LocalDate.of(analysisPerYearYearChoice.getValue(), 1, 1);

var monthProfitList = FinanceController.getAnalysisPerYear(yearLocalDate);

updateChart(monthProfitList, "Месяц", "Прибыль за год");}

private void updateChart(List<Float> periodList, String xLabel, String seriesName)

{Series<String, Float> periodSeries = new Series<String, Float>();

for(int i = 0; i < periodList.size() - 1; i++)

{periodSeries.getData().add(new Data<String, Float>(String.valueOf(i + 1), periodList.get(i)));}

profitXAxis.setLabel(xLabel);

profitYAxis.setLabel("Рубли");

periodSeries.setName(seriesName);

profitLineChart.getData().clear();

profitLineChart.getData().addAll(periodSeries);

fullProfitByChart.setText(String.format("%.2f", periodList.get(periodList.size() - 1)));}

@FXML

private void exitToAdminStartScene() throws IOException

{

GUI.setRoot("adminstartscene");

}

}

# ПРИЛОЖЕНИЕ В

**(обязательное)**

**Листинг скрипта генерации базы данных**

CREATE DATABASE IF NOT EXISTS `furniturestore` /\*!40100 DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci \*/ /\*!80016 DEFAULT ENCRYPTION='N' \*/;

USE `furniturestore`;

-- MySQL dump 10.13 Distrib 8.0.31, for Win64 (x86\_64)

--

-- Host: localhost Database: furniturestore

-- ------------------------------------------------------

-- Server version 8.0.31

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!50503 SET NAMES utf8 \*/;

/\*!40103 SET @OLD\_TIME\_ZONE=@@TIME\_ZONE \*/;

/\*!40103 SET TIME\_ZONE='+00:00' \*/;

/\*!40014 SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0 \*/;

/\*!40014 SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0 \*/;

/\*!40101 SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='NO\_AUTO\_VALUE\_ON\_ZERO' \*/;

/\*!40111 SET @OLD\_SQL\_NOTES=@@SQL\_NOTES, SQL\_NOTES=0 \*/;

--

-- Table structure for table `admin`

--

DROP TABLE IF EXISTS `admin`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `admin` (

`id` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NOT NULL,

`password` blob NOT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `id\_UNIQUE` (`id`)

**ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ В**

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `admin`

--

LOCK TABLES `admin` WRITE;

/\*!40000 ALTER TABLE `admin` DISABLE KEYS \*/;

INSERT INTO `admin` VALUES ('admin',\_binary 'З­DЛ­v\*] ¤Rщ\иTэБ\а\зҐ\*8\_#у\к±\Ш  
“\ФrcMъ\З\УNј5\Сj·ыЉђ\И­—Q\Ц\ЗSЌЖќ\ШЮђw\м');

/\*!40000 ALTER TABLE `admin` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `assistant`

--

DROP TABLE IF EXISTS `assistant`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `assistant` (

`id` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NOT NULL,

`password` blob NOT NULL,

`firstname` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NOT NULL,

`secondname` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NOT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `id\_UNIQUE` (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `assistant`

--

LOCK TABLES `assistant` WRITE;

/\*!40000 ALTER TABLE `assistant` DISABLE KEYS \*/;

**ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ В**

INSERT INTO `assistant` VALUES ('assistant',\_binary '\rҐ\Уэ-Y\Е/~ЌЯЏїO\в™0­;H®\"oїуTј\Нш»ЇЃe\ЭDAє\а\дd№3ГќCЂ[ш8\зЛ‡:','Р›РёР·Р°','Р‘РѕР±РѕРґР¶РѕРЅРѕРІ');

/\*!40000 ALTER TABLE `assistant` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `cartelement`

--

DROP TABLE IF EXISTS `cartelement`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `cartelement` (

`id` int NOT NULL AUTO\_INCREMENT,

`user` varchar(50) NOT NULL,

`furniture` int NOT NULL,

`amount` int NOT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `id\_UNIQUE` (`id`),

UNIQUE KEY `uniquecartelement` (`user`,`furniture`),

KEY `furniturecartelement` (`furniture`),

CONSTRAINT `furniturecartelement` FOREIGN KEY (`furniture`) REFERENCES `furniture` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `usercartelement` FOREIGN KEY (`user`) REFERENCES `user` (`id`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB AUTO\_INCREMENT=85 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `cartelement`

--

LOCK TABLES `cartelement` WRITE;

/\*!40000 ALTER TABLE `cartelement` DISABLE KEYS \*/;

INSERT INTO `cartelement` VALUES (83,'haiduk',13,10),(84,'user',13,2);

/\*!40000 ALTER TABLE `cartelement` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `furniture`

--

**ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ В**

DROP TABLE IF EXISTS `furniture`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `furniture` (

`id` int NOT NULL AUTO\_INCREMENT,

`furniturecategory` int NOT NULL,

`name` varchar(45) NOT NULL,

`manufactorer` varchar(45) NOT NULL,

`description` varchar(255) DEFAULT NULL,

`warranty` int DEFAULT NULL,

`price` float NOT NULL,

`firstcost` float NOT NULL,

`amount` int NOT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `id\_UNIQUE` (`id`),

KEY `furniture\_category\_idx` (`furniturecategory`),

CONSTRAINT `furniture\_category` FOREIGN KEY (`furniturecategory`) REFERENCES `furniturecategory` (`id`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB AUTO\_INCREMENT=17 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `furniture`

--

LOCK TABLES `furniture` WRITE;

/\*!40000 ALTER TABLE `furniture` DISABLE KEYS \*/;

INSERT INTO `furniture` VALUES (10,1,'Полка навесная','Пинскдрев','',4,80,55,28),(11,18,'Стол деревянный','Молодечно мебель','Стол сделан из осины. Гибкий, качественный, надёжный.',3,180,130,10),(12,32,'Дверная ручка','Ами Мебель','',2,33,20,13),(13,33,'Тумба мраморная','Пинскдрев','Тумба доступна в рассрочку.',5,1200,800,2),(14,34,'Стул металлический','Молодечно мебель','',5,999.9,750,20),(15,37,'Кровать двухспальная','Пинскдрев','',5,99.9,80,25),(16,36,'Угловой диван','Пинскдрев','Пружинный блок типа Bonnell; пружина «змейка».',6,259.9,199.9,33);/\*!40000 ALTER TABLE `furniture` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `furniturecategory`

--

**ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ В**

DROP TABLE IF EXISTS `furniturecategory`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `furniturecategory` (

`id` int NOT NULL AUTO\_INCREMENT,

`name` varchar(100) NOT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `id\_UNIQUE` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=38 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `furniturecategory`

--

LOCK TABLES `furniturecategory` WRITE;

/\*!40000 ALTER TABLE `furniturecategory` DISABLE KEYS \*/;

INSERT INTO `furniturecategory` VALUES (1,'Полки'),(17,'Кушетки'),(18,'Столы'),(32,'Ручки'),(33,'Тумбы'),(34,'Стулья'),(35,'Зеркала'),(36,'Диваны'),(37,'Кровати');/\*!40000 ALTER TABLE `furniturecategory` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `orderelement`

--

DROP TABLE IF EXISTS `orderelement`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `orderelement` (

`id` int NOT NULL AUTO\_INCREMENT,

`userorder` int NOT NULL,

`furniture` int NOT NULL,

`amount` int NOT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `id\_UNIQUE` (`id`),

KEY `furnitureorderelement\_idx` (`furniture`),

KEY `orderinorderelemenet\_idx` (`userorder`),

CONSTRAINT `furnitureorderelement` FOREIGN KEY (`furniture`) REFERENCES `furniture` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,

**ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ В**

**ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ В**

CONSTRAINT `orderinorderelemenet` FOREIGN KEY (`userorder`) REFERENCES `userorder` (`id`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB AUTO\_INCREMENT=67 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `orderelement`

--

LOCK TABLES `orderelement` WRITE;

/\*!40000 ALTER TABLE `orderelement` DISABLE KEYS \*/;

INSERT INTO `orderelement` VALUES (66,48,13,2);

/\*!40000 ALTER TABLE `orderelement` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `servicedorder`

--

DROP TABLE IF EXISTS `servicedorder`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `servicedorder` (

`id` int NOT NULL,

`user` varchar(50) DEFAULT NULL,

`assistant` varchar(50) DEFAULT NULL,

`description` varchar(255) DEFAULT NULL,

`isaccepted` tinyint NOT NULL,

`answerdate` date NOT NULL,

`orderdate` date NOT NULL,

`orderprice` float NOT NULL,

`orderfirstcost` float NOT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `id\_UNIQUE` (`id`),

KEY `userserviceorder\_idx` (`user`),

KEY `assistantserviceorder\_idx` (`assistant`),

CONSTRAINT `assistantserviceorder` FOREIGN KEY (`assistant`) REFERENCES `assistant` (`id`) ON DELETE SET NULL ON UPDATE CASCADE,

CONSTRAINT `userserviceorder` FOREIGN KEY (`user`) REFERENCES `user` (`id`) ON DELETE SET NULL ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

**ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ В**

--

-- Dumping data for table `servicedorder`

--

LOCK TABLES `servicedorder` WRITE;

/\*!40000 ALTER TABLE `servicedorder` DISABLE KEYS \*/;

INSERT INTO `servicedorder` VALUES (26,'user2','assistantt','РћС‚РєР°Р·Р°РЅРѕ!',0,'2023-03-06','2023-03-06',1730,1400),(34,'user','assistant','',1, '2023-03-08', '2023-03-06',1150,1000),(35,'user2','assistant','',1, '2023-03-31', '2023-03-06',2173,1820),(37,'user','assistant','РЅРµРЅРµРЅРµРЅРµРЅРµРЅРµРЅ',0, '2023-03-08', '2023-03-07',11150,9000),(38,'user','assistant','РЎР»РёС€РєРѕРј РјРЅРѕРіРѕ Р·Р°РєР°Р·Р°РЅРЅС‹С… РІРµС‰Рµ!',0,''2023-03-08',''2023-03-08',4059,3339.6),(39,'user','assistant','',1,'2023-03-09',''2023-03-09',8000,1012),(40,'user2','assistant','РћС‚Р»РёС‡РЅ!',1,''2023-03-09', '2023-03-09',2173,1881.2),(43,'user2','assistant','!!!',1, '2023-03-09', '2023-03-09',1230,1012),(44,'user','assistant','',1, '2023-03-09', '2023-03-09',1230,1012),(45,'user','assistantt',NULL,0, '2023-03-12', '2023-03-12',100,80),(46,'haiduk','assistant','РЈСЃРїРµС€РЅРѕ РїРѕРґС‚РІРµСЂР¶РґРµРЅРѕ! РЎРїР°СЃРёР±Рѕ Р·Р° РїРѕРєСѓРїРєСѓ!',1, '2023-03-14', '2023-03-14',12000,8000),(47,'haiduk','assistant','РЈСЃРїРµС€РЅРѕВ РїРѕРґС‚РІРµСЂР¶РґРµРЅРѕ!В РЎРїР°СЃРёР±РѕВ Р·Р°В РїРѕРєСѓРїРєСѓ!',1, '2023-03-14', '2023-03-14',12000,8000);

/\*!40000 ALTER TABLE `servicedorder` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `servicedorderelement`

--

DROP TABLE IF EXISTS `servicedorderelement`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `servicedorderelement` (

`id` int NOT NULL,

`servicedorder` int NOT NULL,

`furniture` int DEFAULT NULL,

`amount` int NOT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `id\_UNIQUE` (`id`),

KEY `furnitureserviceelement\_idx` (`furniture`),

KEY `serviceinelement\_idx` (`servicedorder`),

CONSTRAINT `furnitureserviceelement` FOREIGN KEY (`furniture`) REFERENCES `furniture` (`id`) ON DELETE SET NULL ON UPDATE CASCADE,

**ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ В**

CONSTRAINT `serviceinelement` FOREIGN KEY (`servicedorder`) REFERENCES `servicedorder` (`id`) ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `servicedorderelement`

--

LOCK TABLES `servicedorderelement` WRITE;

/\*!40000 ALTER TABLE `servicedorderelement` DISABLE KEYS \*/;

INSERT INTO `servicedorderelement` VALUES (31,26,NULL,10),(32,26,NULL,15),(42,34,NULL,50),(43,35,NULL,51),(44,35,NULL,10),(47,37,NULL,50),(48,37,NULL,100),(49,38,NULL,33),(50,38,NULL,33),(51,39,NULL,10),(52,39,NULL,10),(53,40,NULL,51),(54,40,NULL,10),(59,43,NULL,10),(60,43,NULL,10),(61,44,NULL,10),(62,44,NULL,10),(63,45,NULL,1),(64,46,13,10),(65,47,13,10);

/\*!40000 ALTER TABLE `servicedorderelement` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `user`

--

DROP TABLE IF EXISTS `user`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `user` (

`id` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NOT NULL,

`password` blob NOT NULL,

`firstname` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NOT NULL,

`secondname` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NOT NULL,

`phonenumber` varchar(20) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci DEFAULT NULL,

`email` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NOT NULL,

`address` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci DEFAULT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `id\_UNIQUE` (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;

**ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ В**

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `user`

--

LOCK TABLES `user` WRITE;

/\*!40000 ALTER TABLE `user` DISABLE KEYS \*/;

INSERT INTO `user` VALUES ('Marina',\_binary '<™ Ї\м%5MU®!Y  
Іn8\Х?!sё\У\Ь>\оL~z±Бл‹…>;\зєa;1»\\њ6!M\ЙсJBэz/Ы„…k\К\\D\В','РђР»РµРєСЃРµР№','Р‘СѓСЂРёРє','+375291234567','marina@mail.ru','РњРёРЅСЃРє, РћР±С‰РµР¶РёС‚РёРµ 1'),('Masha',\_binary '6\'ђљ)\ГЃ q\м\'ч\ЙК—ra‚®Тљ}\Э.T53\"Пі\n»ћ:mт¬, ю#Cc\ЦxVMЌ0Y0W\_`\в\У\РHMy','Р•РіРѕСЂ','Р СѓРґРµРЅРєРѕРІ','','4121244','РњРёРЅСЃРє РЎРёС‚Рё'),('user',\_binary '±Ca@LЏэTњ\ЫD<?\н\ву\е4\Ч?xчs\н—Ф¤6©эќ°^\иі%А­6C‹Cю\ИQOББ\нІAАћ,\в','РђР»РµРєСЃРµР№','user2','5305014','mar@mail.ru','РњРёРЅСЃРє'),;

/\*!40000 ALTER TABLE `user` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `userorder`

--

DROP TABLE IF EXISTS `userorder`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `userorder` (

`id` int NOT NULL AUTO\_INCREMENT,

`user` varchar(50) NOT NULL,

`orderdate` date NOT NULL,

`orderprice` float NOT NULL,

`orderfirstcost` float NOT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `id\_UNIQUE` (`id`),

KEY `user\_idx` (`user`),

CONSTRAINT `userinorder` FOREIGN KEY (`user`) REFERENCES `user` (`id`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB AUTO\_INCREMENT=49 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

**ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ В**

--

-- Dumping data for table `userorder`

--

LOCK TABLES `userorder` WRITE;

/\*!40000 ALTER TABLE `userorder` DISABLE KEYS \*/;

INSERT INTO `userorder` VALUES (48,'user', '2023-03-14',2400,1600);

/\*!40000 ALTER TABLE `userorder` ENABLE KEYS \*/;

UNLOCK TABLES;

/\*!40103 SET TIME\_ZONE=@OLD\_TIME\_ZONE \*/;

/\*!40101 SET SQL\_MODE=@OLD\_SQL\_MODE \*/;

/\*!40014 SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS \*/;

/\*!40014 SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS \*/;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

/\*!40111 SET SQL\_NOTES=@OLD\_SQL\_NOTES \*/;

-- Dump completed on 2023-04-15 18:21:32