Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет инженерно-экономический

Кафедра экономической информатики

Дисциплина «Технологии разработки программного обеспечения»

|  |  |
| --- | --- |
|  | «К ЗАЩИТЕ ДОПУСТИТЬ» |
|  | Руководитель курсового проекта  Заведующий кафедры ЭИ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д. А. Сторожев |
|  | \_\_\_.\_\_\_\_.2022 |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

на тему:

**«Разработка программы учета товаров на складе»**

БГУИР КР 1-40 05 01-12 001 ПЗ

|  |  |
| --- | --- |
|  | Выполнил студент группы 124405  Быков Алексей Дмитриевич  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовая работа представлена на проверку \_\_\_.\_\_\_\_.2022  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2022

**СОДЕРЖАНИЕ**

[Введение 4](#_Toc102253711)

[1 Анализ и моделирование предметной области программного средства 5](#_Toc102253712)

[1.1 Описание предметной области 5](#_Toc102253713)

[1.2 Разработка функциональной модели предметной области в нотации IDEF0 6](#_Toc102253714)

[1.3 Разработка BPMN-модели основного процесса предметной области 8](#_Toc102253715)

[1.4 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований 8](#_Toc102253716)

[1.5 UML-модели представления программного средства и их описание 11](#_Toc102253721)

[2 Проектирование и конструирование программного средства](#_Toc102253722) 16

[2.1 Постановка задачи 16](#_Toc102253723)

[2.2 Разработка модульной структуры программы 16](#_Toc102253724)

[2.3 Выбор способа организации данных 17](#_Toc102253725)

[2.4 Разработка перечня пользовательских функций программы 17](#_Toc102253726)

[2.5 Разработка схем алгоритмов работы программы 21](#_Toc102253727)

[3 Тестирование и проверка работоспособности программного средства…….24](#_Toc102253728)

[4 Инструкция по развертыванию приложения и сквозной тестовый пример](#_Toc102253729)..34

[4.1 Авторизация 34](#_Toc102253730)

[4.2 Модуль пользователя 36](#_Toc102253731)

[4.3 Модуль администратора 37](#_Toc102253732)

[Заключение. 40](#_Toc102253733)

[Список использованных источников. 41](#_Toc102253734)

[Приложения. 42](#_Toc102253735)

[Ведомость 47](#_Toc102253736)

**Перечень условных обозначений, символов и терминов**

|  |  |
| --- | --- |
| БД (база данных) *[*1*]* | – упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе. |
| *SQL (structured query language) [*1*]* | ­– язык программирования, используемый в большинстве реляционных баз данных для запросов, обработки и определения данных, а также контроля доступа. |
| *UML (Unified Modeling Language) [*2*]* | – система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования. Его можно использовать для визуализации, спецификации, конструирования и документирования программных систем. |
| *BPML (Business Process Model and Notation) [*3*]* | – метод составления блок-схем, отображающий этапы выполнения бизнес-процесса от начала до конца. |
| *IDEF0 [*4*]* | – методология функционального моделирования и графическая нотация, предназначенная для формализации и описания бизнес-процессов. |
| *Windows Forms [*5*]* | – платформа пользовательского интерфейса для создания классических приложений *Windows*. |

# **ВВЕДЕНИЕ**

Существует большое количество предприятий, оперирующих огромными объемами товаров. Нередко встает вопрос об учете товаров. Бумажная бухгалтерия хоть и все ещё используется, но многие предприятия переходят на электронные программы складского учета. Так как бумажная бухгалтерия имеет множество минусов. Например, сложность поиска определенного товара, так как это делается вручную; необходимость физического хранения базы товаров, что так же приводит к ряду проблем; возможный человеческий фактор, когда некоторые товары изначально неправильно были отсортированы в алфавитном порядке, что делает поиск нужных товаров ещё более затруднительным.

В настоящее время большинство предприятий нуждается в современной и качественной программе учета товара на складе, для того, чтобы вести складской учет. Такое программное обеспечение требуется не только для крупных складских и логистических предприятий, но также и для маленьких фирм или индивидуальных предпринимателей, которые предлагают свои услуги. Если не следить за остатками товаров и не контролировать их размещение, это означает, что ваш бизнес будет уходить в минус, при этом крупные суммы вам придётся списываться с собственных счетов из-за ошибок сотрудников.

Целью данной курсовой работы будет анализ предметной области складского учета, ее особенностей и поиск необходимого минимума функциональных требований для улучшения эффективности деятельности складских и бухгалтерских отделов предприятий в данной сфере для улучшения доступности данных для анализа и минимизации потерь от списания товаров.

Для достижения поставленной цели, удобства использования и облегчения процесса складского учета необходимо выполнить следующие задачи:

* проанализировать предметную область,
* разработать список необходимых функциональных требований,
* организовать связь приложения с базами данных,
* разработать алгоритм работы приложения,
* разработать приложение,
* описать функциональные возможности приложения и принцип его работы;

В качестве объекта исследования данной курсовой работы следует выделить товары на складах компаний, особенности хранения и учета информации, а также особенности безопасного хранения информации о пользователях и товарах.

# **Анализ и моделирование предметной области программного средства**

**1.1 Описание предметной области**

Проблема складского учета всегда остро стояла в предприятиях, пытающихся оптимизировать затраты и минимизировать потери. Ведение бумажного учета товара ведет к ряду проблем, которые не всегда имеют решение, а если и имеют, то затраты на их решение намного больше, чем теоретическая прибыль, которую получит от этого предприятие.

Среди проблем бумажного складского учета можно выделить следующее *[*6*]*:

* удаленность офиса от склада — трудно контролировать деятельность сотрудников и производимые ими операции;
* случайные или специальные ошибки работников при заполнении документов вручную, приписки, хищения и т.д;
* отсутствие прозрачности при проведении операций закупки/продажи;
* несогласованность действий персонала друг с другом из-за отсутствия общей информационной базы;
* большие временные затраты на заполнение и сдачу документации в бухгалтерию;
* отсутствие представления об остатках товара на складе;
* возможные проблемы при резервировании товара для отдельных клиентов;
* трудности при отслеживании движения товара по цепочке производитель-склад-покупатель;
* необходимость частых внеплановых проверок состояния склада;
* большие затраты времени и человеческих ресурсов на проведение инвентаризаций.

Автоматизация склада обеспечивает эффективное и рациональное управление складскими процессами, что, в свою очередь, является необходимым условием стабильного развития бизнеса.

Программа складского учета позволяет избежать большинства ошибок и проблем, связанных с человеческим фактором (перегруза, недогруза, ошибок в учете, воровства и многих других).

Благодаря автоматизации работы склада возрастает скорость обработки заказов, в несколько раз ускоряется проведение складских операций, увеличивается пропускная способность склада, оптимизируется использование его площадей, на порядок снижаются потери товара. За счет сокращения объема рутинной работы повышается эффективность работы складского персонала. Так же, введение программы складского учета позволяет уменьшить количество необходимого персонала на обслуживание складских помещений, что в свою очередь повышает прибыль предприятия.

* 1. **Разработка функциональной модели предметной области в нотации *IDEF0***

Для увеличения эффективности анализа предметной области обратимся к схемам, которые в наглядном виде показывают происходящие на предприятиях процессы. На рисунке 1.1 представлен нулевой уровень разложения процесса складского учета.

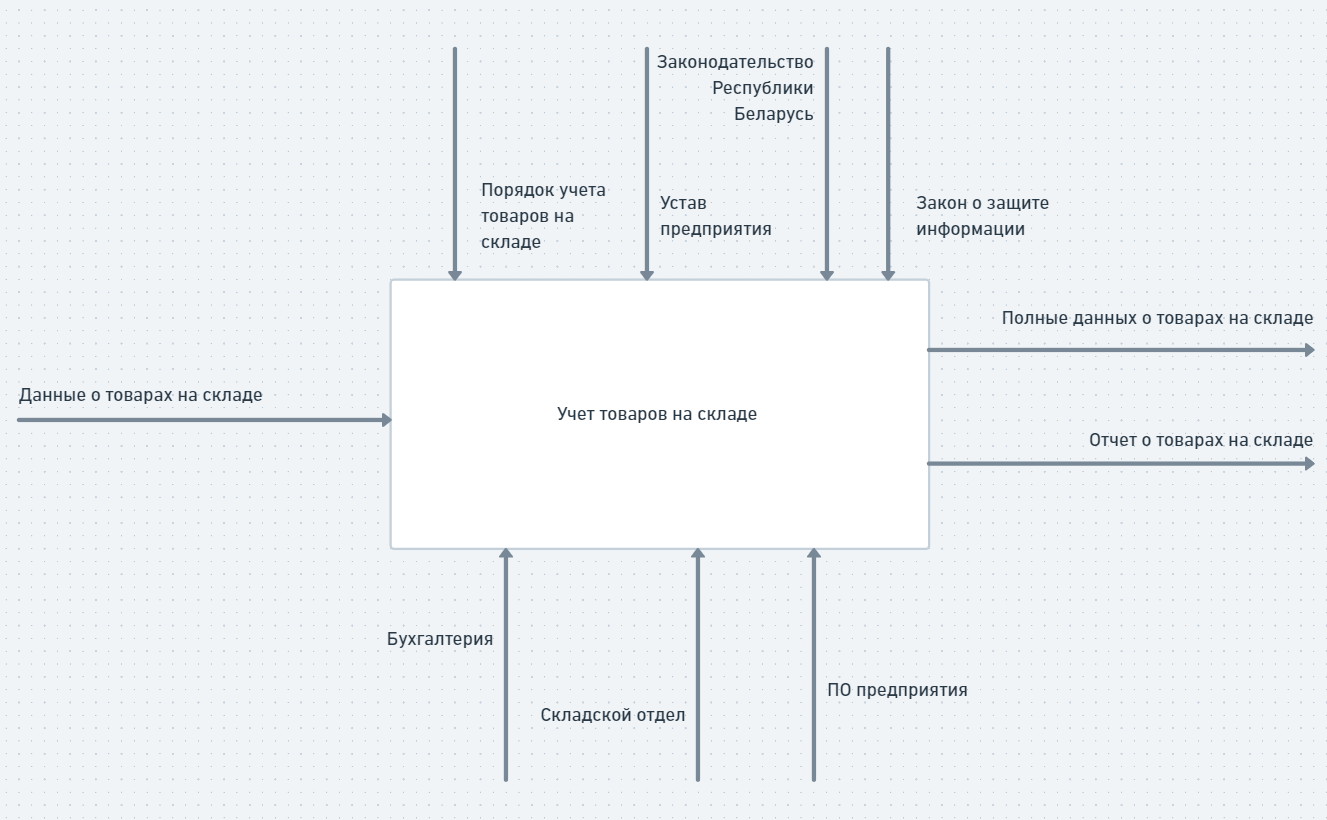


Рисунок 1.1 – Контекстная диаграмма складского учета

На рисунке 1.2 показано разложение процесса складского учета на подпроцессы, характеризующие один из вариантов учета товаров на складе предприятия. На вход подаются данные о товарах, которые в итоге заносятся в базу данных для дальнейшей оптимизации взаимодействия с данными о товарах на складе.

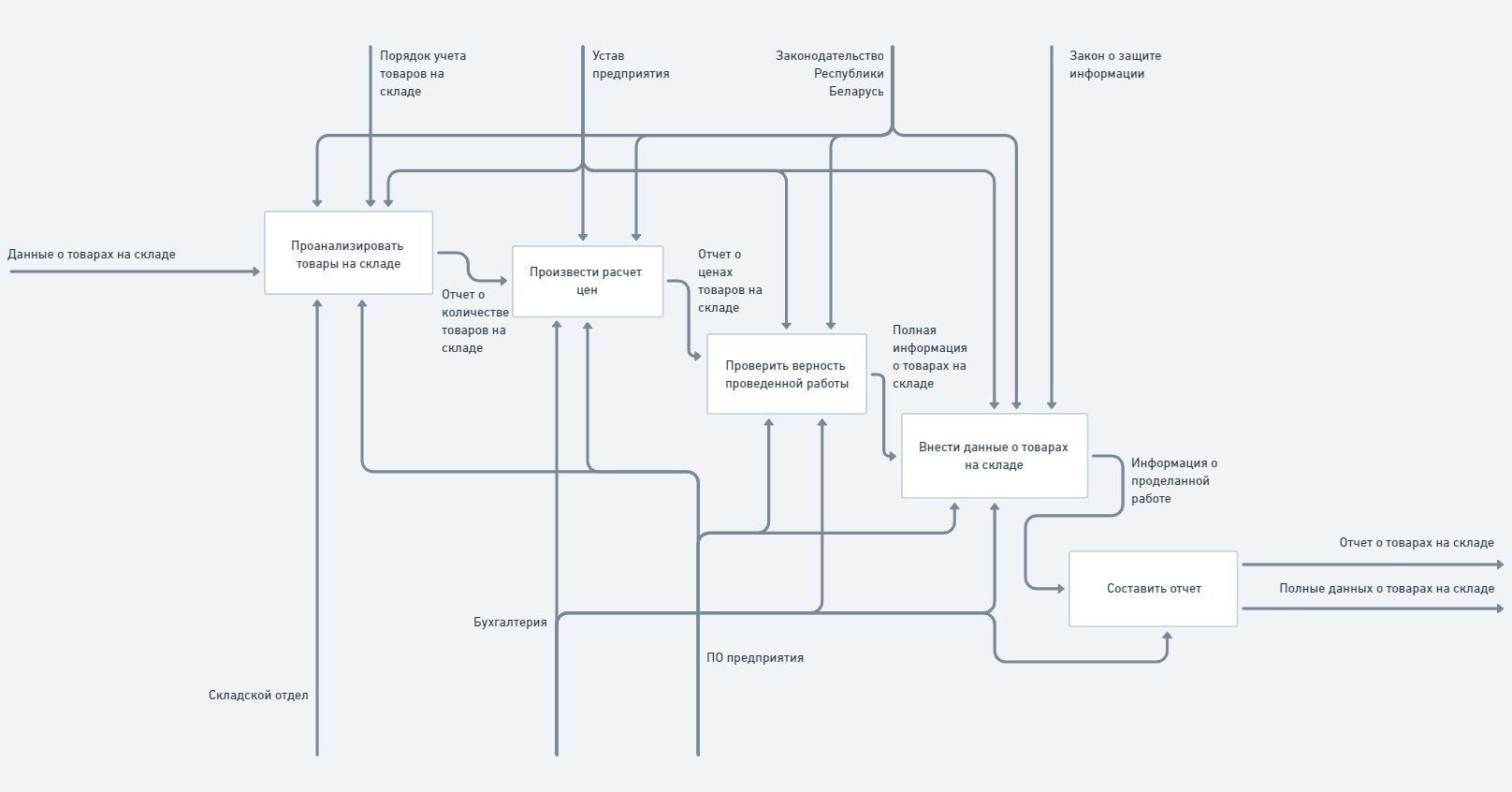


Рисунок 1.2 – Декомпозиция первичной контекстной диаграммы

На рисунке 1.3 можно увидеть декомпозицию второго уровня, показывающую в деталях процесс анализа товаров на складе.

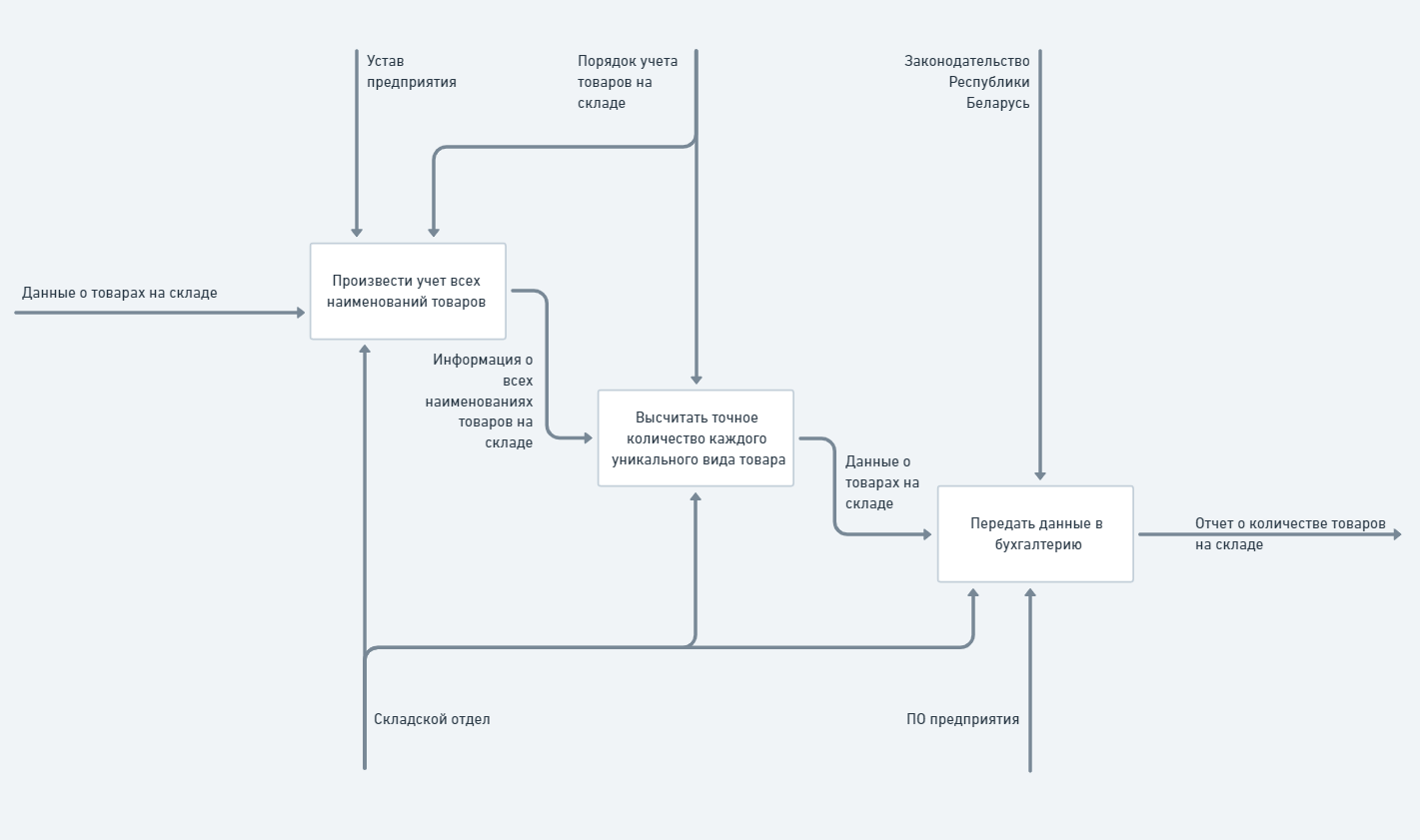


Рисунок 1.3 – Декомпозиция блока “Проанализировать товары на складе”

Оставшиеся блоки диаграммы не нуждаются в дополнительной декомпозиции, так как представляют из себя достаточно простые задачи. Наиболее важным процессом в приведенной диаграмме является процесс внесения данных о товарах на складе компании в базу данных, где информация будет храниться и изменяться в последующем времени. Так как именно правильное размещение данных позволит в дальнейшем оптимально использовать ПО складского учета.

## **Разработка *BPMN*-модели основного процесса предметной области**

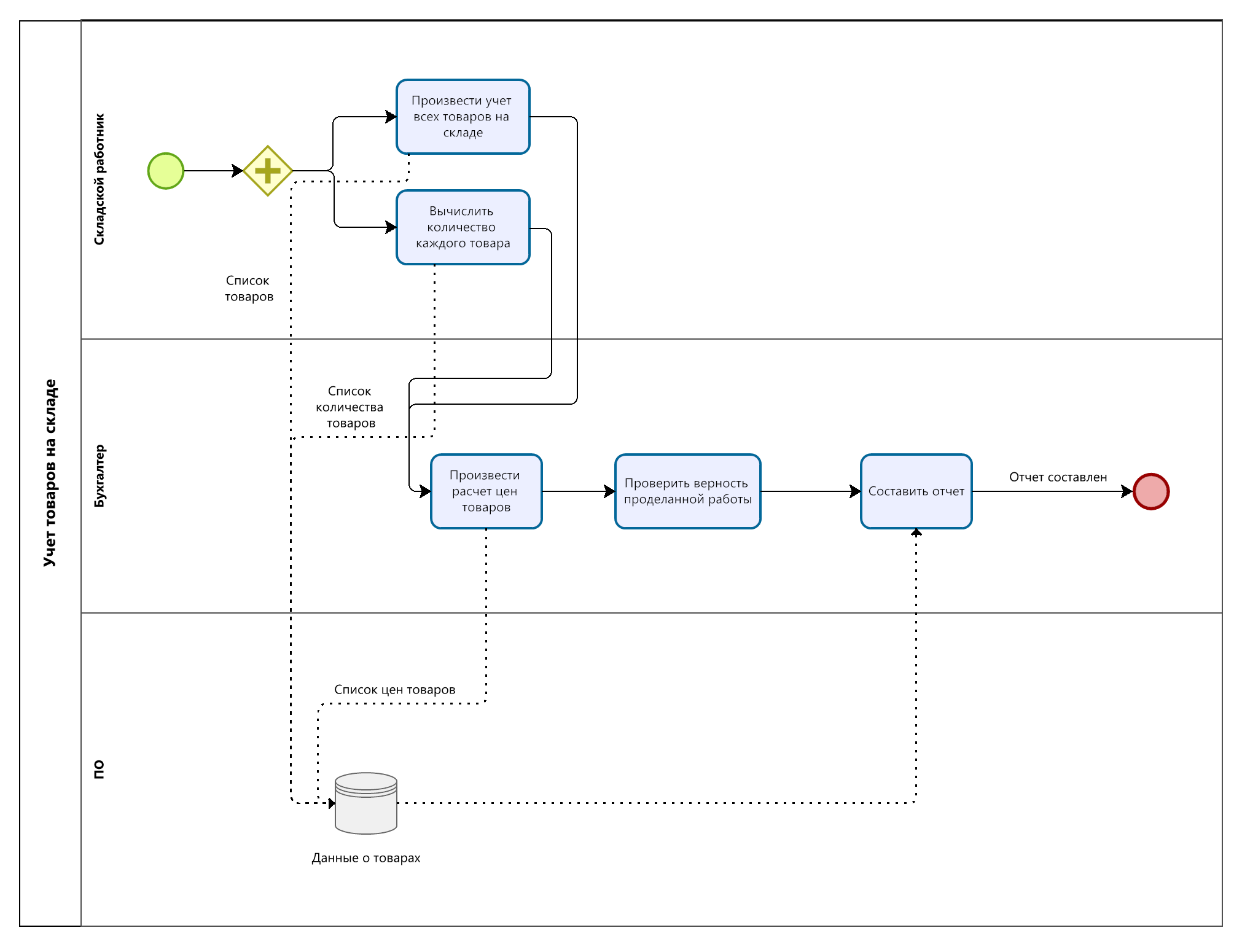


Рисунок 1.4 – *BPMN*-модель расчёта зарплаты сотрудника предприятия

* 1. **Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований**

### **1.4.1 Текст варианта задания**

Разработка программы учета товаров на складе.

Программа предоставляет сведения о товарах, имеющихся на складе: наименование товара, количество единиц товара, цена единицы товара, дата поступления товара на склад, ФИО, зарегистрировавшего товар.

Индивидуальное задание: вывести в алфавитном порядке список товаров, хранящихся более *x* месяцев, стоимость которых превышает *y* рублей (*x, y* вводятся с клавиатуры).

Общее для всех вариантов задание: реализовать авторизацию для входа в систему, функционал администратора и функционал пользователя.

**1.4.2 Исходные данные**

1. Язык программирования *С#.*
2. Среда разработки *Microsoft Visual Studio*.
3. Интерфейс программирования приложений – *Windows Forms*.
4. Вид приложения – оконное графическое.
5. Парадигма программирования – объектно-ориентированная.
6. Способ организации данных – классы.
7. Способ хранения данных – локальные БД с использованием *SQL*.
8. Система управления БД – *MySQL*.
9. Каждая логически завершенная задача программы должна быть реализована в виде функции.
10. Код должен соответствовать правилам, определенным в документе *«С# Code Convention».*

### **1.4.3 Функциональные требования**

Для разграничения доступа к разным частям программы, сделует разделить пользователей на две роли: администратор и пользователь. В свою очередь, роль администратора должна включать в себя все функции роли пользователя, а так же иметь дополнительный функционал, позволяющий управлять содержанием баз данных.

Приложение должно соответствовать следующим критериям для обеспечения удобства пользования:

* логичность интерфейса;
* отсутствие визуальной перегруженность;
* интуитивность понимания программы;
* обеспечение быстрого доступа ко всем функциям.

Структура учетной записи должна состоять из:

* *id;*
* *login;*
* *password;*
* *admin*.

Где поле *«admin»* представляет собой булевую переменную, позволяющий отделить обычных пользователей от администраторов.

Информация о товаре должна соответствовать следующему шаблону:

* *id;*
* *product;*
* *quantity;*
* *price;*
* *date of registration;*
* *registrants name.*

В функционал роли пользователя должны входить следующие функции:

* просмотр всей таблицы с информацией о товарах на складе,
* поиск информации по любому полю таблицы товаров;
* получение текстового отчета с содержимым таблицы;
* подсчет заработной платы по отделам;
* нахождение товаров, хранящихся больше *X* месяцев, стоимость которых превышает *Y* рублей;
* возможность выйти из аккаунта.

Роль администратора должна включать в себя весь функционал роли пользователя, а также приведенные ниже функции:

* добавление нового поля в БД товаров с информацией о товаре;
* добавление нового поля в БД пользователей с информацией о пользователе;
* изменение полей информации о пользователе;
* изменение полей информации о товаре;
* удаление информации о выбранном пользователе из БД;
* удаление информации о выбранном товаре из БД;
* переход между таблицами со списком пользователей и товаров на складе.

В целях безопасности и ускорения доступа к информации, хранение данных осуществляется в локальных базах данных.

### **1.4.4 Требования к программной реализации**

1. Все переменные и константы должны иметь осмысленные имена в рамках тематики варианта курсовой работы. Переменным рекомендуется присваивать имена, состоящие из букв нижнего регистра; для формирования составного имени используется нижнее подчеркивание (например, *number\_of\_students*) или «верблюжья нотация» (например, *flagExit*). Константам рекомендуется присваивать имена, состоящие из букв верхнего регистра (например, *SIZE\_ARR\_OF ACCOUNTS, FILE\_OF\_ACCOUNTS*).
2. Имена функций должны быть осмысленными, строиться по принципу глагол + существительное (например, *addAccount, findStudentBySurname*). Если функция выполняет проверку и возвращает результат типа *bool*, то ее название должно начинаться с глагола *is* (например, *isNumberNumeric, isLoginUnique*).
3. Не допускается использование оператора прерывания *goto*.
4. Код не должен содержать неименованных числовых констант.
5. («магических» чисел), неименованных строковых констант (например, имен файлов и др.). Подобного рода информацию следует представлять, как глобальные константы. По правилам качественного стиля программирования тексты всех информационных сообщений, выводимых пользователю в ответ на его действия, также оформляются как константы.
6. Код необходимо комментировать (как минимум в части объявления структур, массивов/векторов, прототипов функций, нетривиальной логики).
7. Код не должен дублироваться.
8. Одна функция решает только одну задачу.
9. Следует избегать глубокой вложенности условных и циклических конструкций: вложенность блоков должна быть не более трех.
10. Следует избегать длинных функций: текст функции должен умещаться на один экран.
11. Следует выносить код логически независимых модулей в отдельные файлы.
    1. ***UML*-модели представления программного средства и их описание**

Диаграмма вариантов использования – диаграмма, описывающая, какой функционал разрабатываемой программной системы доступен каждой группе пользователей. В разработанном приложении существует две группы с возможностями, представленными на рисунке 1.5.

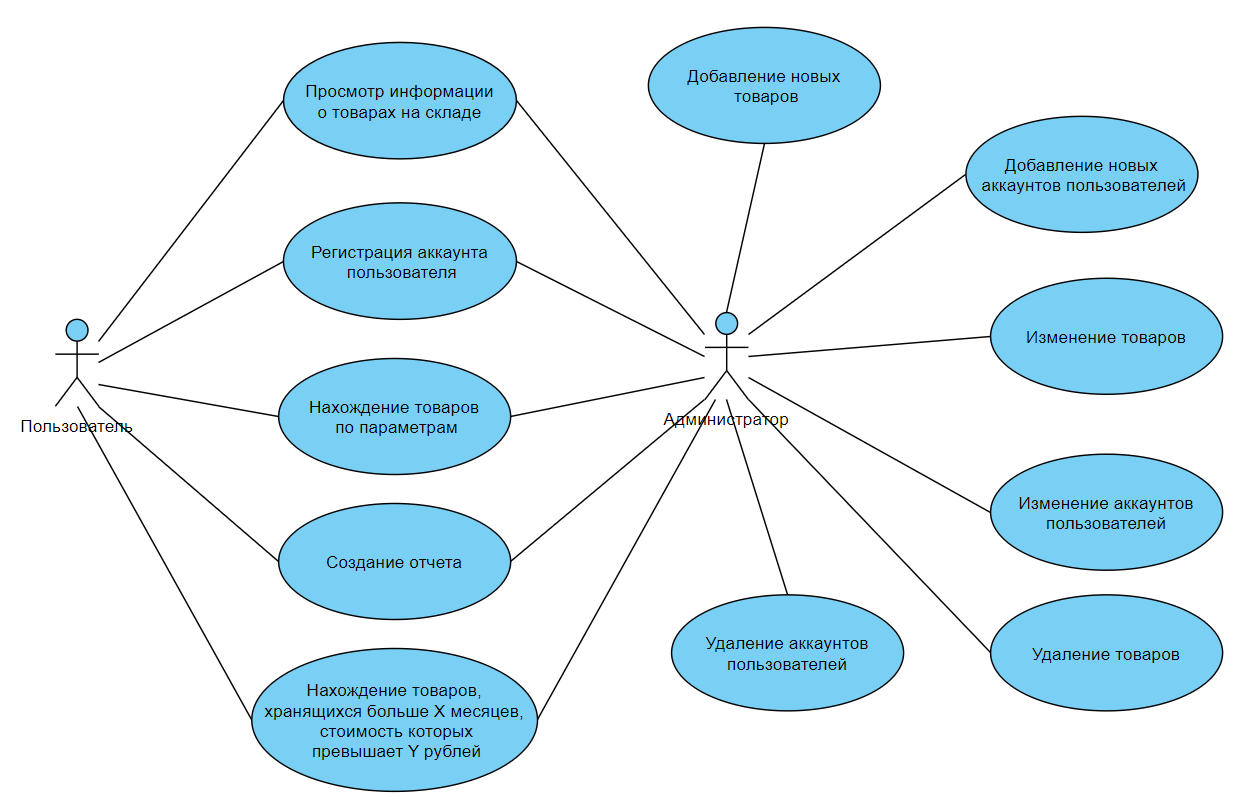
****

Рисунок 1.5 – Диаграмма вариантов использования

Диаграмма компонентов позволяет визуализировать взаимодействие отдельных модулей программы, содержащих связанные компоненты (рисунке 1.6).

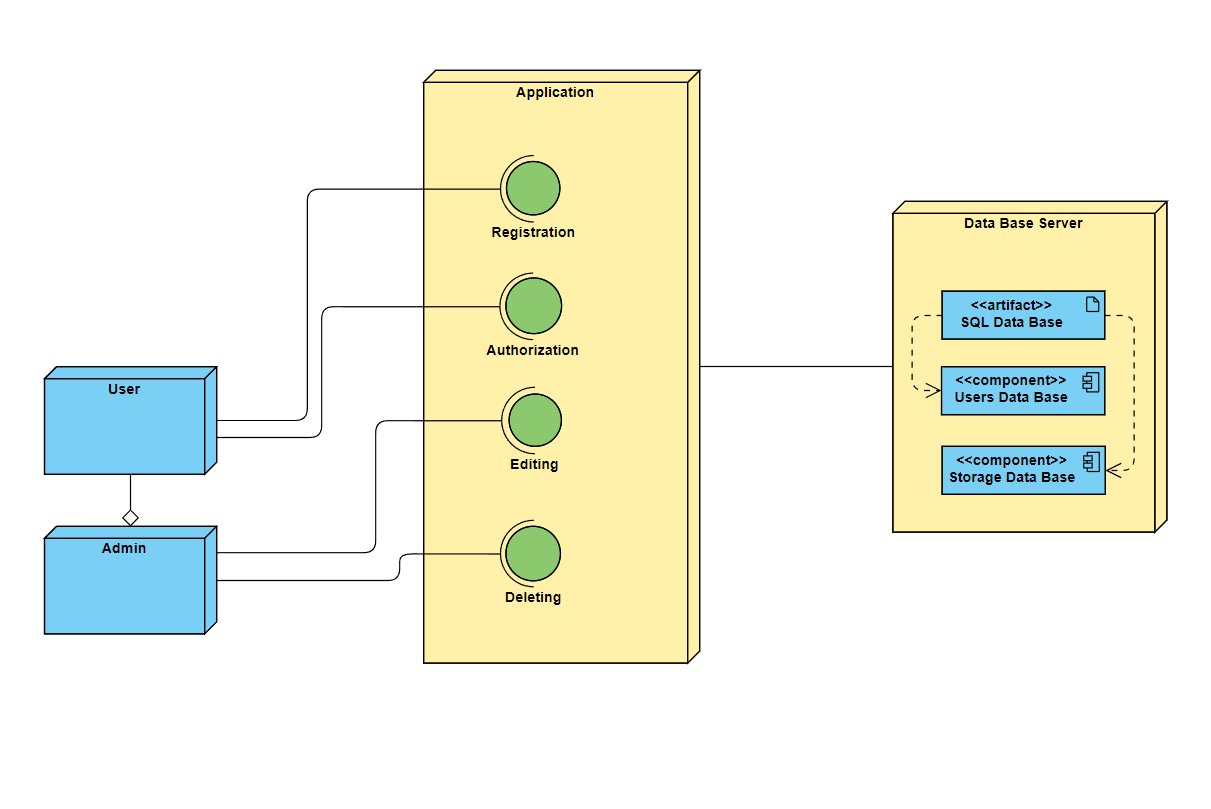


Рисунок 1.6 – Диаграмма компонентов

Диаграмма состояний позволяет описать различные состояния динамической системы. Соответствующая диаграмма представлена на рисунке 1.7.

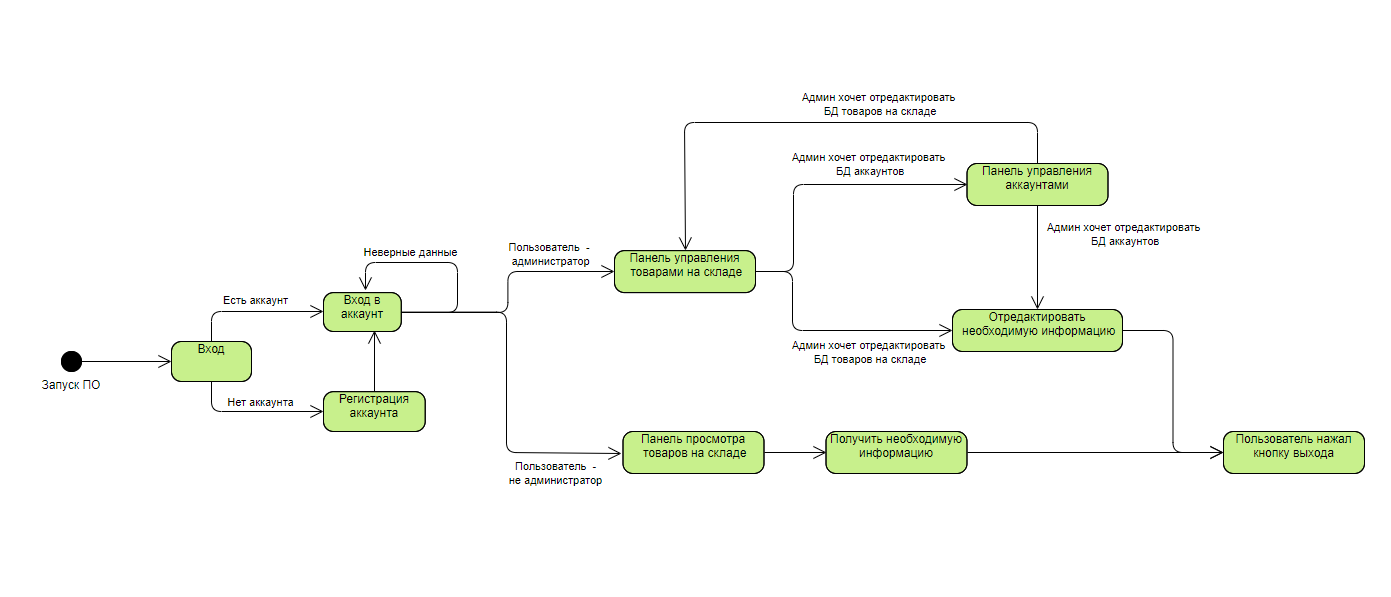
****

Рисунок 1.7 – Диаграмма состояний

Диаграммы развертывания используются для визуализации аппаратных процессоров/узлов/устройств системы, каналов связи между ними и размещения программных файлов на этом аппаратном обеспечении. Диаграмма развертывания представлена на рисунке 1.8.

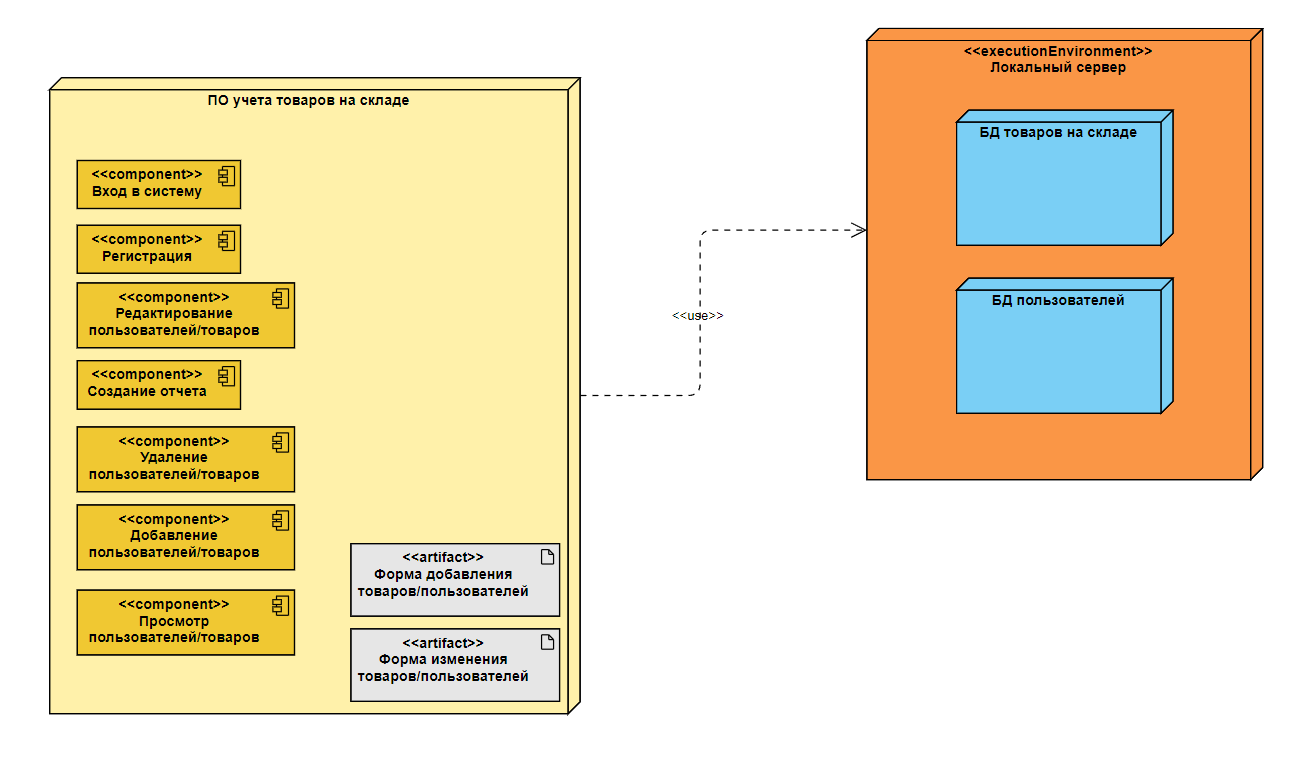
****

Рисунок 1.8 – Диаграмма развертывания

Диаграмма последовательностей относится к диаграммам взаимодействия *UML*, описывающим поведенческие аспекты системы. Другими словами, диаграмма последовательностей отображает временные особенности передачи и приема сообщений объектами. На рисунках 1.9, 1.10 и 1.11 диаграммы последовательностей дают разные сценарии использования программы со стороны использования ее разными пользователями, с правами администратора и без.

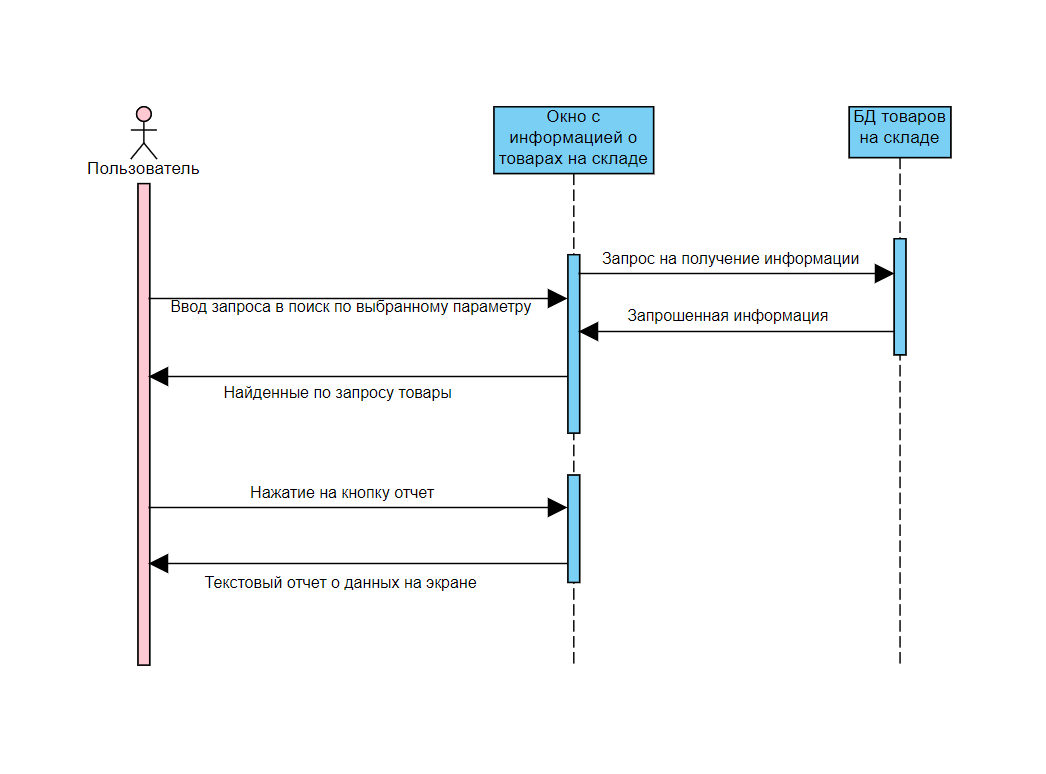


Рисунок 1.9 – Регистрация пользователя в системе

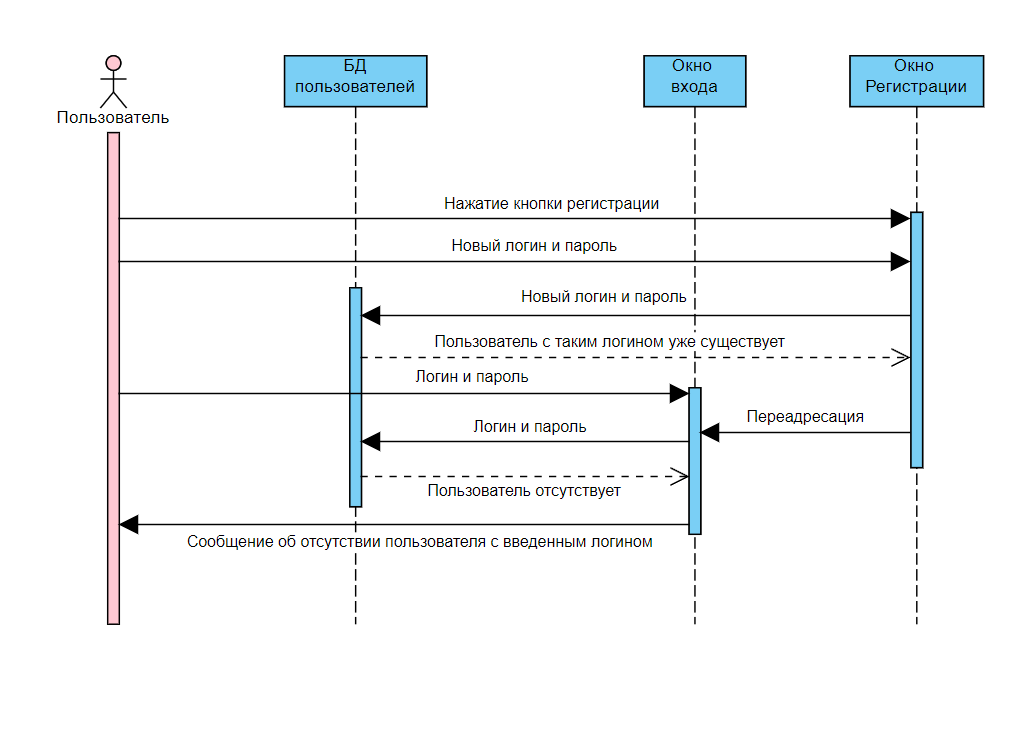
****

Рисунок 1.10 – Взаимодействие пользователя с системой

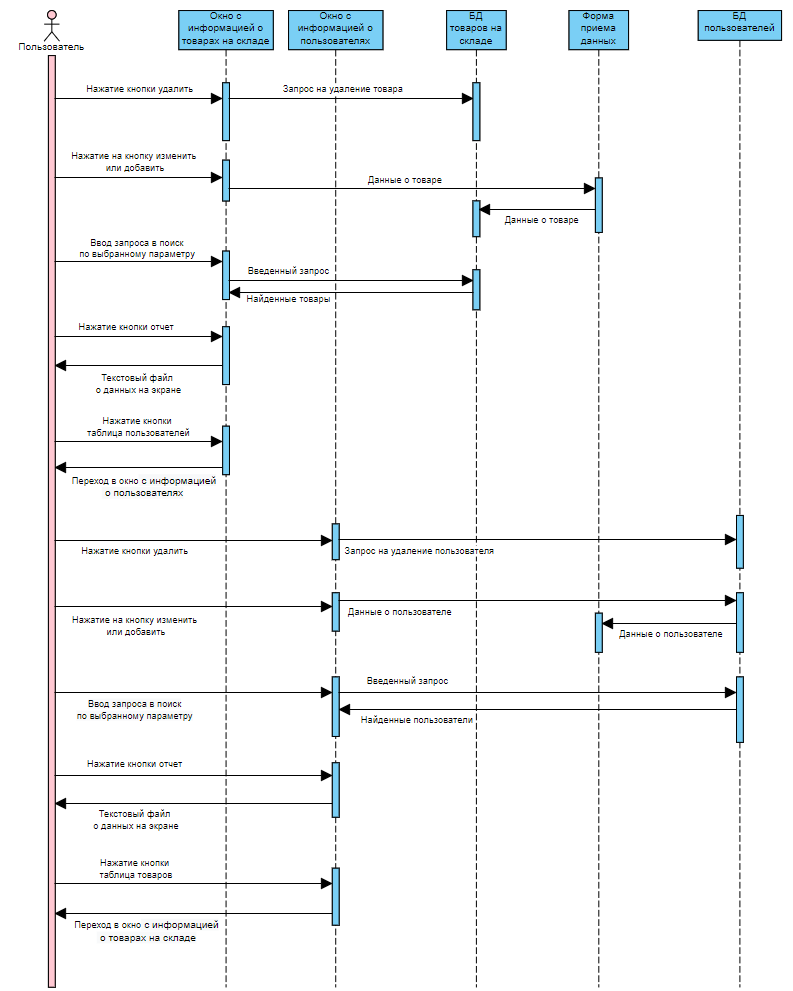


Рисунок 1.11 – Взаимодействие администратора с системой

1. **ПРОЕКТИРОВАНИЕ И КОНСТРУИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА** 
   1. **Постановка задачи**

Настоящая цель данной работы – разработка программного обеспечения складского учета, удовлетворяющего всем функциональным требованиям.

* 1. **Разработка модульной структуры программы**

Поскольку был выбрал объектно-ориентированный подход, большая часть компонентов программы представлена классами. Схема модульной организации программы представлена на рисунке 2.1.

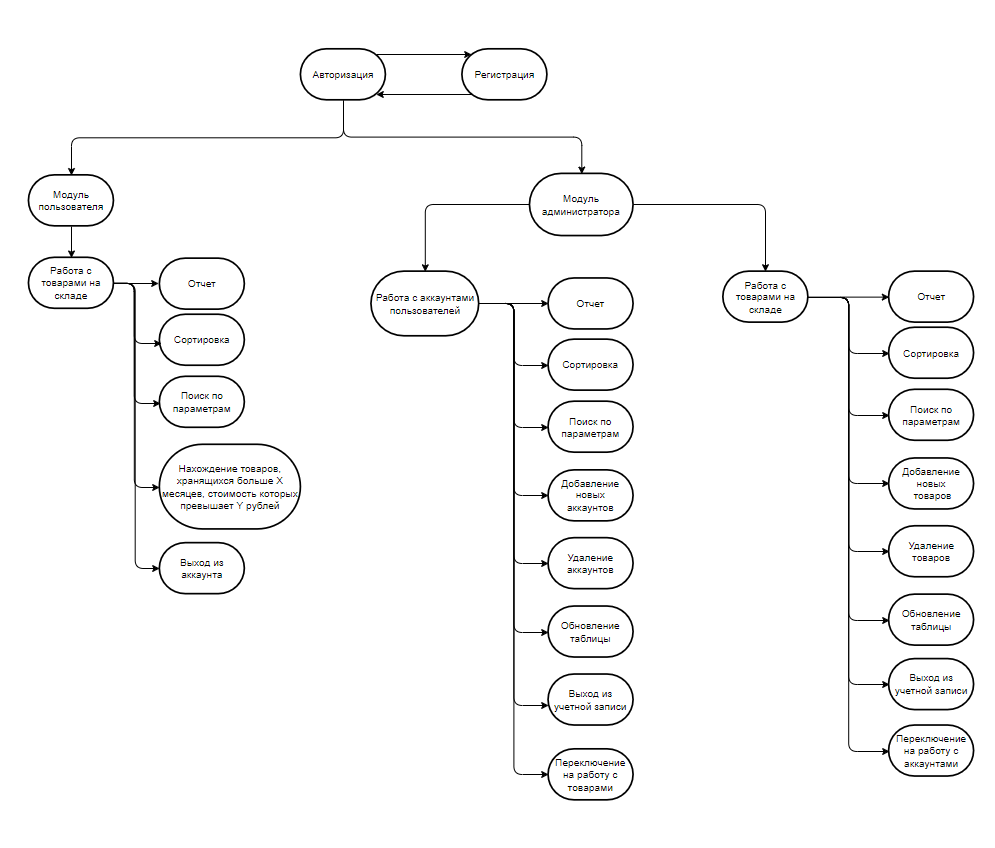


Рисунок 2.1 – Модульная схема структуры программы

Каждый класс в свою очередь подразделяется на функции, выполняющие отведенные им единичные задачи. Это можно наблюдать на примере модулей «работа с товарами на складе» и «работа с аккаунтами пользователей». В данном случае оба этих модуля представляют собой классы, визуализирующие окно работы с информацией при помощи инструмента *Windows Forms*. Наполнением этих классов являются функции-методы, позволяющие реагировать на взаимодействия пользователя с программой.

* 1. **Выбор способа организации данных**

Учитывая развитие технологий в текущей рыночной ситуации, предприятия имеют запрос на удобное систематизированное хранилище информации, которое в свою очередь обеспечивает безопасность и сохранность данных, а также имеет возможность в постоянном масштабировании.

Базы данных отлично подходят для выполнения выше поставленных задач. В том числе, использование систем управления БД, таких как *MySQL* (выбранная для выполнения проекта), обеспечивает высокую производительность запросов к базам данных, что позволяет оптимизировать затраты на закупку оборудования для осуществления работы ПО.

Так же, текущее развитие технологий позволяет связываться с подобными хранилищами информации в реальном времени, а также хранить огромные объемы информации в понятном и организованном виде. Именно поэтому для систематизации хранения данных был выбран формат баз данных с использованием системы управления *MySQL*, которые в данном случае подключаются к локальному серверу. Но программа сделана таким образом, что изменение локального сервера на удаленный не доставит никаких проблем. Достаточно будет поменять несколько строк параметров сервера.

* 1. **Разработка перечня пользовательских функций программы**

Для удобства восприятия информации и систематизации планирования, список пользовательских функций будет разбит на группы, представляемых собой класс. Каждый класс обладает набором методов, позволяющие выполнять необходимые в этом классе процессы. Всю программу можно разделить модули, в разработанной программе каждый модуль представлен отдельным файлом, что улучшает читаемость написанного ПО, а также ускоряет дальнейшую модернизацию программы, если таковая потребуется.

Всю программу можно разделить на следующие модули:

* модуль входа в систему (форма *LoginForm*);
* модуль регистрации (форма *RegistrationForm*);
* модуль пользователя (форма *MainUser*);
* модуль администратора с данными о товарах на складе (форма *AdminMain*);
* модуль администратора с данными о аккаунтах пользователей (форма *AdminUsers*);
* модуль для редактирования информации о товарах на складе (форма *UpdateForm*);
* модуль для редактирования информации о аккаунтах пользователей (форма *UpdateUserForm*).
* модуль хранящий в себе пути подключения к базам данных (*DataBase*)

Для обращения к базам данных существует модуль *DataBase*, который хранит в себе пути подключения, что упрощает смену сервера до изменения параметров в одном лишь этом модуле. Список методов модуля:

* public void OpenConnection()
* public void CloseConnection()
* public MySqlConnection GetConnection()

Первая форма, которая встречает пользователя при запуске ПО – форма авторизации (*LoginForm*). Представленные в ней методы приведены ниже:

* public LoginForm()
* private void LoginButton\_Click(object sender, EventArgs e)
* private void LoginBox\_Enter(object sender, EventArgs e)
* private void LoginBox\_Leave(object sender, EventArgs e)
* private void PassBox\_Enter(object sender, EventArgs e)
* private void PassBox\_Leave(object sender, EventArgs e)
* private void toRegistration\_Click(object sender, EventArgs e)
* private void LoginForm\_FormClosing(object sender, FormClosingEventArgs e)

Если пользователь не имеет учетной записи, он может перейти на форму регистрации (*RegistrationForm*). Список методов которой состоит из:

* public RegistrationForm()
* private void RegistrationButton\_Click(object sender, EventArgs e)
* public bool IsUserExist()
* private void LoginBox\_Enter(object sender, EventArgs e)
* private void LoginBox\_Leave(object sender, EventArgs e)
* private void PassBox\_Enter(object sender, EventArgs e)
* private void PassBox\_Leave(object sender, EventArgs e)
* private void PassBox2\_Enter(object sender, EventArgs e)
* private void PassBox2\_Leave(object sender, EventArgs e)
* private void toRegistration\_Click(object sender, EventArgs e)
* private void RegistrationForm\_FormClosed(object sender, FormClosedEventArgs e)

Далее рассмотрим модуль пользователя (*MainUser*). Методы, которыми представлена форма пользователя, приведены ниже:

* public void AllTable()
* public MainUser()
* private bool IsVoid()
* private void SearchBox\_Enter(object sender, EventArgs e)
* private void SearchBox\_Leave(object sender, EventArgs e)
* private void XBox\_Enter(object sender, EventArgs e)
* private void XBox\_Leave(object sender, EventArgs e)
* private void YBox\_Enter(object sender, EventArgs e)
* private void YBox\_Leave(object sender, EventArgs e)
* private void TaskButton\_Click(object sender, EventArgs e)
* private void SearchButton\_Click(object sender, EventArgs e)
* private void LoginButton\_Click(object sender, EventArgs e)
* private void radioButtonProduct\_CheckedChanged(object sender, EventArgs e)
* private void radioButtonPrice\_CheckedChanged(object sender, EventArgs e)
* private void radioButtonQuantity\_CheckedChanged(object sender, EventArgs e)
* private void radioButtonName\_CheckedChanged(object sender, EventArgs e)
* private void MainUser\_FormClosing(object sender, FormClosingEventArgs e)
* private void MinPriceBox\_Enter(object sender, EventArgs e)
* private void MaxPriceBox\_Leave(object sender, EventArgs e)
* private void MaxPriceBox\_Enter(object sender, EventArgs e)
* private void MinPriceBox\_Leave(object sender, EventArgs e)
* private void ReportButton\_Click(object sender, EventArgs e)

Модуль администратора с данными о товарах на складе (*AdminMain*) состоит из:

* private void FillTable(DataTable table)
* public void AllTable()
* public AdminMain()
* private bool IsVoid()
* private void LoginButton\_Click(object sender, EventArgs e)
* private void radioButtonProduct\_CheckedChanged(object sender, EventArgs e)
* private void radioButtonPrice\_CheckedChanged(object sender, EventArgs e)
* private void radioButtonQuantity\_CheckedChanged(object sender, EventArgs e)
* private void radioButtonName\_CheckedChanged(object sender, EventArgs e)
* private void SearchButton\_Click(object sender, EventArgs e)
* private void SearchBox\_Enter(object sender, EventArgs e)
* private void SearchBox\_Leave(object sender, EventArgs e)
* private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)
* private void button1\_Click(object sender, EventArgs e) //new row button
* private void button2\_Click(object sender, EventArgs e) //update table button
* private void AnotherTableButton\_Click(object sender, EventArgs e)
* private void AdminMain\_FormClosing(object sender, FormClosingEventArgs e)
* private void MinPriceBox\_Enter(object sender, EventArgs e)
* private void MaxPriceBox\_Leave(object sender, EventArgs e)
* private void MaxPriceBox\_Enter(object sender, EventArgs e)
* private void MinPriceBox\_Leave(object sender, EventArgs e)
* private void ReportButton\_Click(object sender, EventArgs e)
* private void TaskButton\_Click\_1(object sender, EventArgs e)
* private void XBox\_Enter(object sender, EventArgs e)
* private void XBox\_Leave(object sender, EventArgs e)
* private void YBox\_Enter(object sender, EventArgs e)
* private void YBox\_Leave(object sender, EventArgs e)

В ходе работы программы, администратору может понадобиться изменить или добавить новый элемент в таблицу. В таком случае вызывается дополнительное всплывающее окно, в которое можно вписать информацию о товаре. В программе это окно представлено модулем для редактирования информации о товарах на складе (*UpdateForm*). Данный модуль состоит из:

* public UpdateForm(string id,bool isAdd)
* private void UpdateButton\_Click(object sender, EventArgs e)
* private void toAdminMain\_Click(object sender, EventArgs e)

Модуль администратора с данными о аккаунтах пользователей (*AdminUsers*) представлен в программе следующими методами:

* public AdminUsers()
* private void FillTable(DataTable table)
* public void AllTable()
* private bool IsVoid()
* private void LoginButton\_Click(object sender, EventArgs e)
* private void SearchButton\_Click(object sender, EventArgs e)
* private void SearchBox\_Enter(object sender, EventArgs e)
* private void SearchBox\_Leave(object sender, EventArgs e)
* private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)
* private void AnotherTableButton\_Click(object sender, EventArgs e)
* private void UpdateButton\_Click(object sender, EventArgs e)
* private void AddButton\_Click(object sender, EventArgs e)
* private void radioButtonAdmin\_CheckedChanged(object sender, EventArgs e)
* private void radioButtonLogin\_CheckedChanged(object sender, EventArgs e)
* private void radioButtonPassword\_CheckedChanged(object sender, EventArgs e)
* private void AdminUsers\_FormClosing(object sender, FormClosingEventArgs e)
* private void ReportButton\_Click(object sender, EventArgs e)

Так же, как и с добавлением или изменением товаров, администратору иногда необходимо изменить или добавить пользователя. В таком случае открывается всплывающее окно для редактирования информации о аккаунтах пользователей (*UpdateUserForm*). Наполнением формы являются:

* public UpdateUserForm(string id, bool isAdd, string login)
* public bool IsUserExist()
* private void UpdateButton\_Click(object sender, EventArgs e)
* private void toAdminMain\_Click(object sender, EventArgs e)
  1. **Разработка схем алгоритмов работы программы**

Главной точкой входа для приложения выступает файл *Program.cs*. его можно считать равноценным функции main в консольных приложениях. С его выполнения начинается работа разработанного программного обеспечения по складскому учету. Алгоритм его работы представляет собой изображенное на рисунке 2.2.

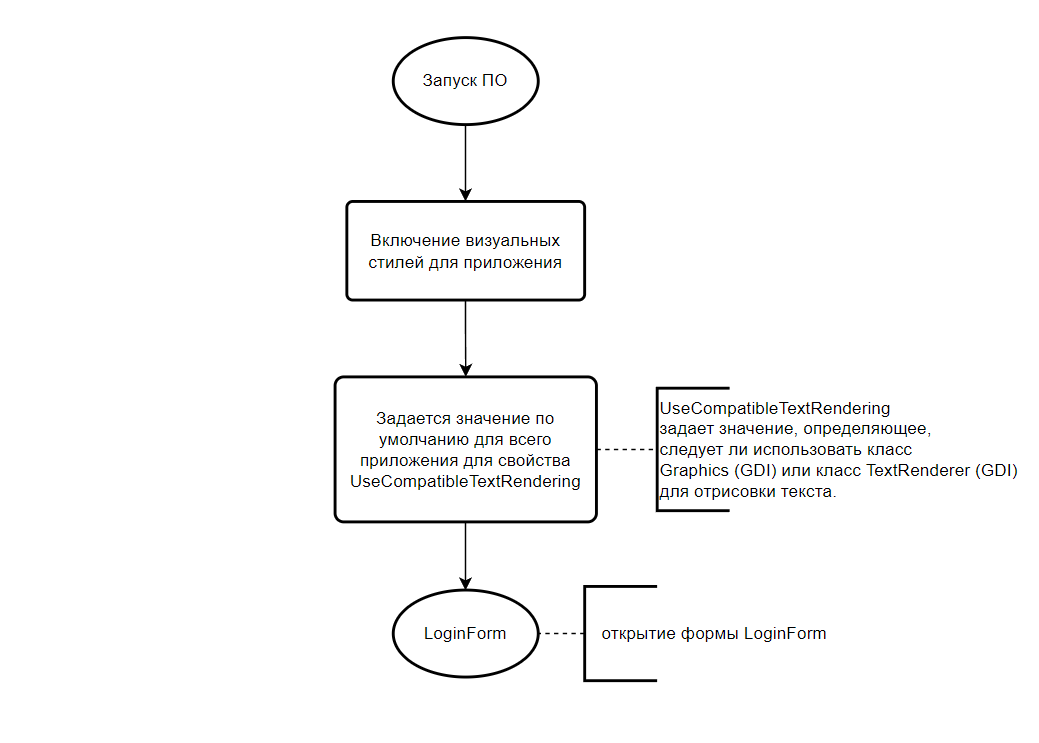


Рисунок 2.2 – Алгоритм работы файла *Program.cs*

Итогом работы файла Program.cs является запуск формы *LoginForm*. Графическое представление работы которого представлено на рисунке 2.3.

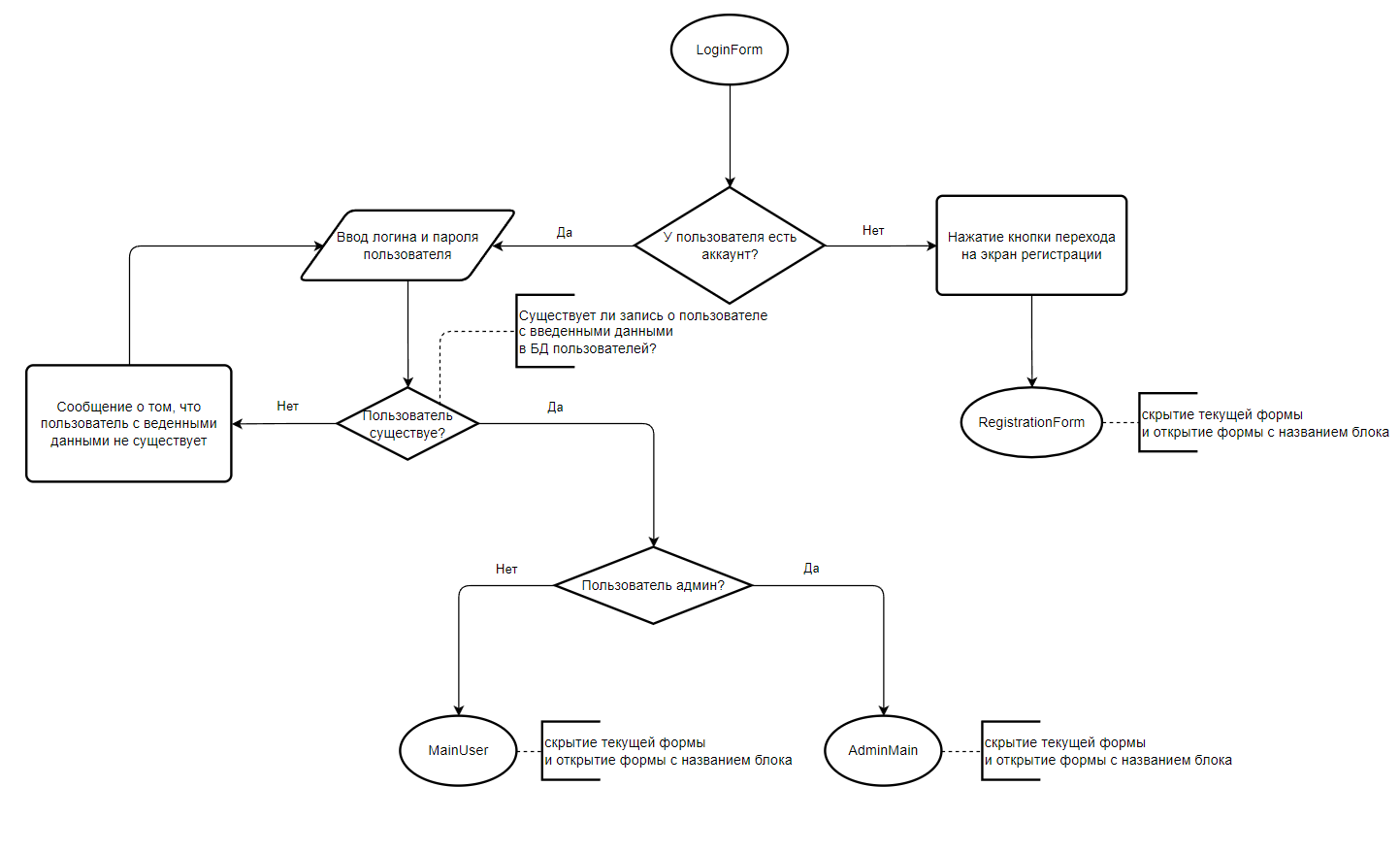


Рисунок 2.3 – Алгоритм работы формы LoginForm

На рисунке 2.4 представлена графическая схема выполнения индивидуального задания, представленного в тексте задания моего варианта.

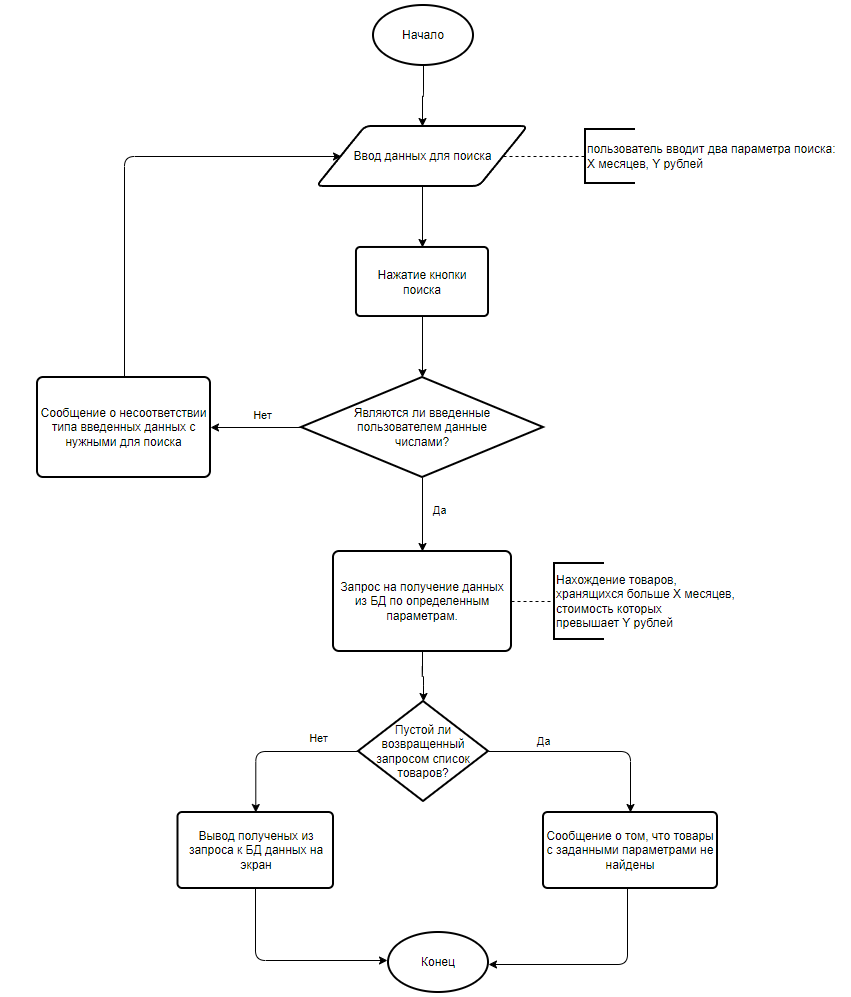


Рисунок 2.4 – Алгоритм работы функции выполнения индивидуального задания

Иногда администратору может понадобиться возможность редактирования данных о записи определенного пользователя. Для такого случая в программе предусмотрена кнопка «Изменить», при нажатии на которую срабатывает функция *UpdateButton\_Click*. Ход работы которой представлен на рисунке 2.5.

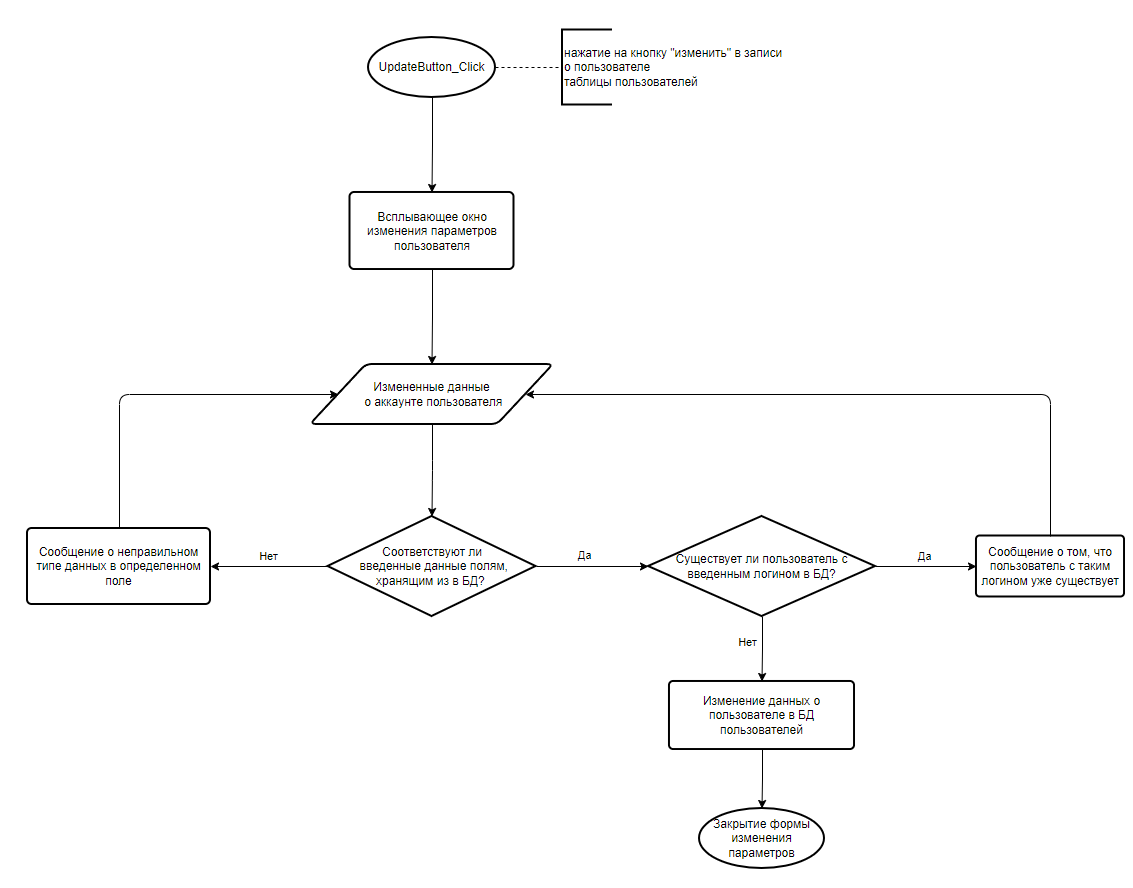


Рисунок 2.5 – Алгоритм работы функции изменения данных о пользователе

1. **ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА**

Первое исключение, с которым может столкнуться пользователь – попытка входа в аккаунт, запись о котором не существует в БД пользователей. В таком случае будет выведено следующее сообщение, представленное на рисунке 3.1.

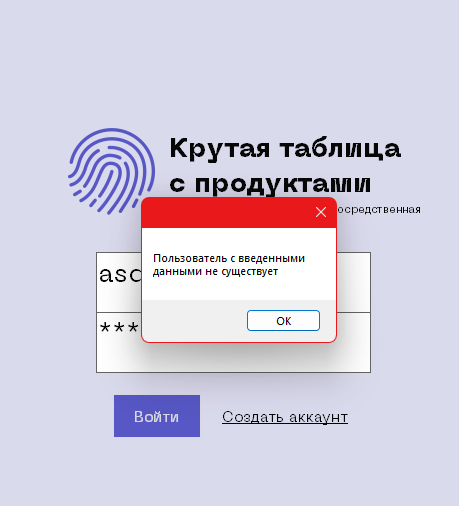


Рисунок 3.1 – Попытка входа в несуществующий аккаунт

Перейдём к экрану регистрации, тут программа может выдать сообщения на рисунке 3.2, просящие ввести данные. Они вызываются если соответствующие поля являются пустыми.

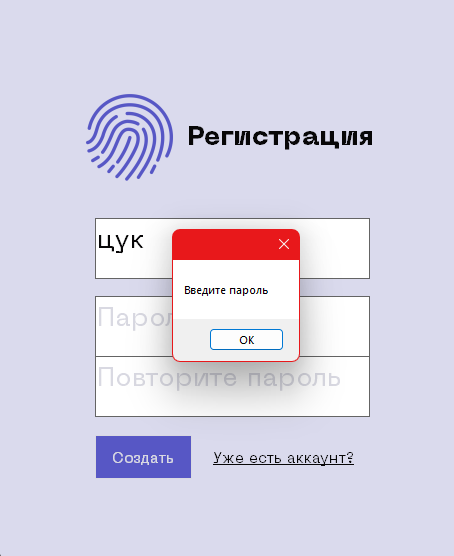
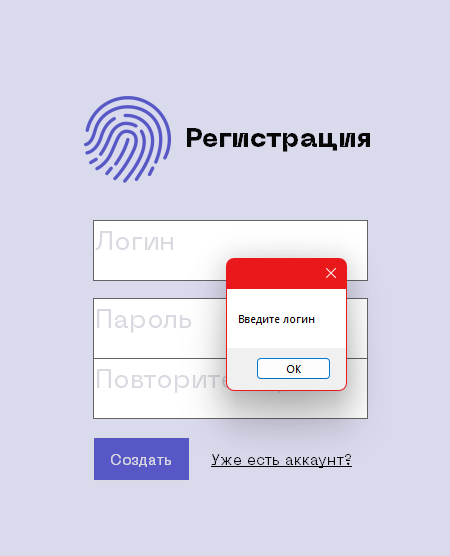


Рисунок 3.2 – Сообщение о пустых полях регистрации

Следующие два исключения, которые может получить пользователь – попытка создания аккаунта с уже существующим в БД пользователей логином (рисунок 3.3) и несовпадение паролей, введенных пользователем в два поля для подтверждения пароля (рисунок 3.4).

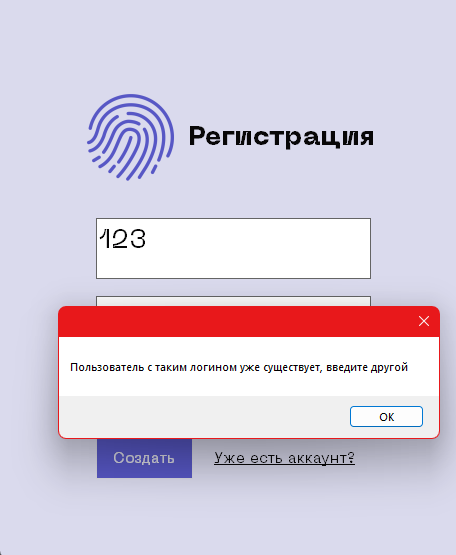


Рисунок 3.3 – Попытка создания аккаунта с уже существующим логином

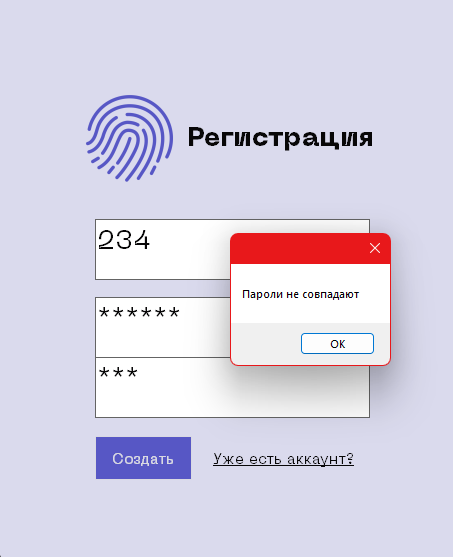


Рисунок 3.4 – Попытка создания аккаунта с несовпадением двух паролей, введенных пользователем

Перейдем к рассмотрению исключений, которые могут произойти по ходу выполнения модуля пользователя.

В модуле пользователя предусмотрело сообщение, которые выводится при не нахождении соответствующей введенным параметрам поиска информации. Оно представлено на рисунке 3.5.

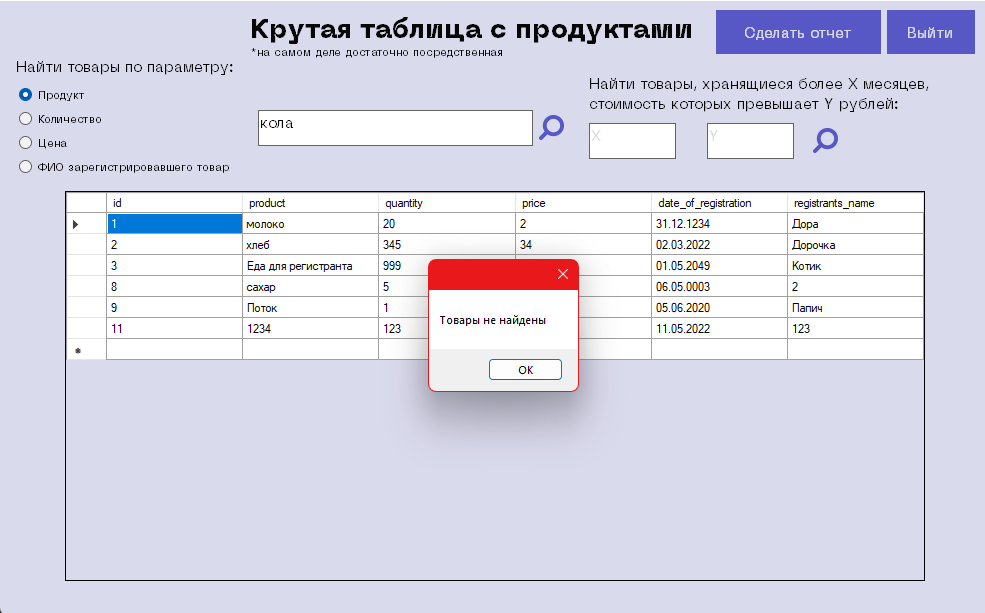


Рисунок 3.5 – Сообщение «Товары не найдены»

Следующее сообщение, с которым может столкнуться пользователь – несоответствие введенной информации с типом переменной, по которой будет происходить поиск. Выведенное сообщение об ошибке представлено на рисунке 3.6.

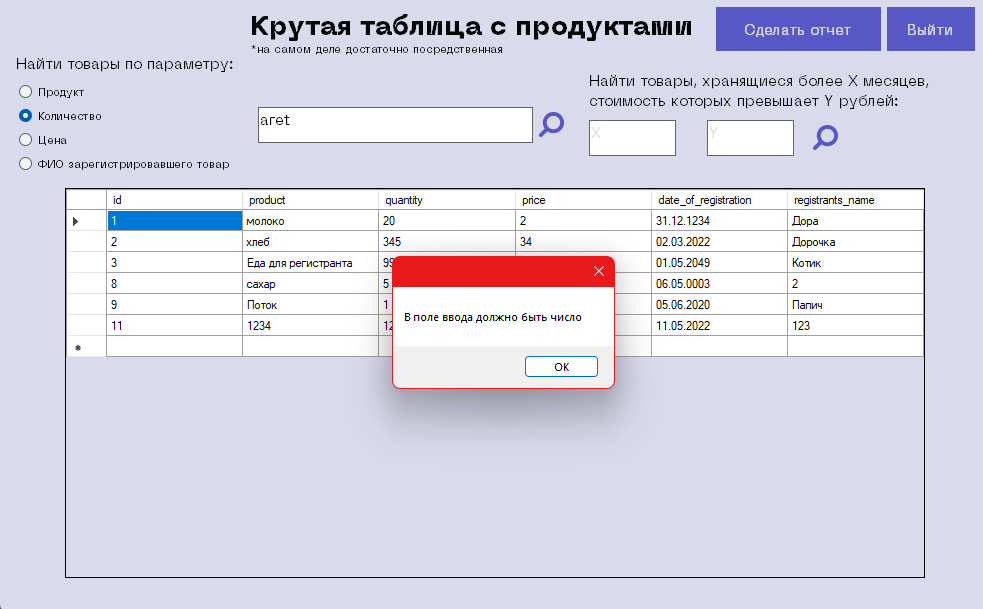


Рисунок 3.6 – Сообщение «Товары не найдены»

В модуле администратора существую все те же сообщения об ошибках поиска, что и при работе модуля пользователя. Однако, модуль администратора так же имеет собственные дополнительные исключения.

При попытке удаления записи о товаре, на экран выводится следующее окно подтверждения действия (рисунок 3.7).

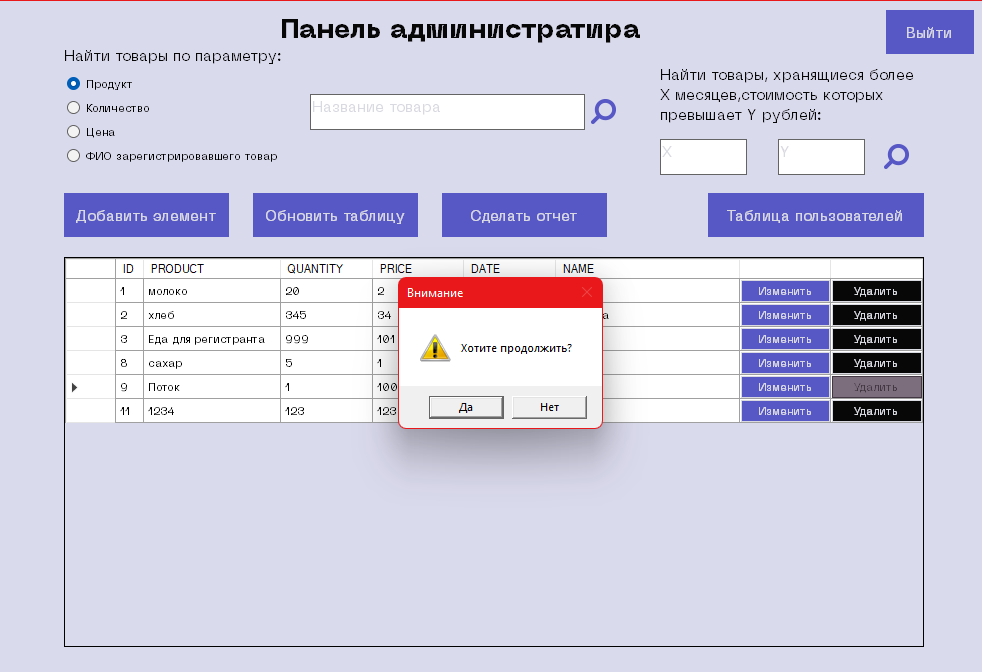


Рисунок 3.7 – Окно подтверждения удаления товара

Так же, при работе с модулем администратора может потребоваться изменить или добавить новую запись, в таком случае вызывается модуль изменения и добавления товаров.

В нем предусмотрены исключения о несоответствии формата данных. Выводящиеся сообщения представлены на рисунках 3.8 и 3.9.

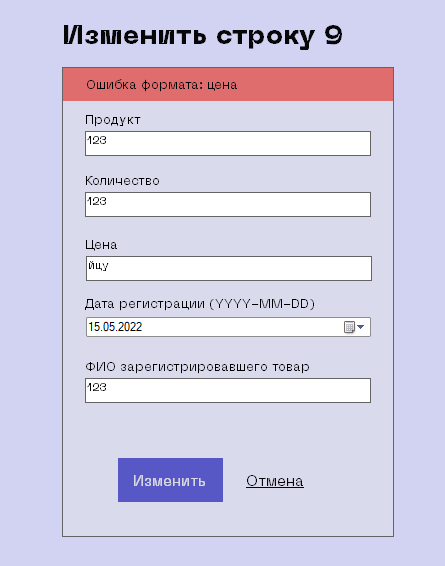
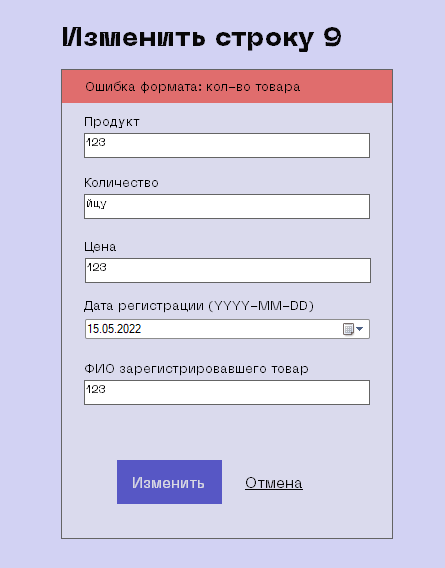


Рисунок 3.8 – Несоответствие формата при изменении записи

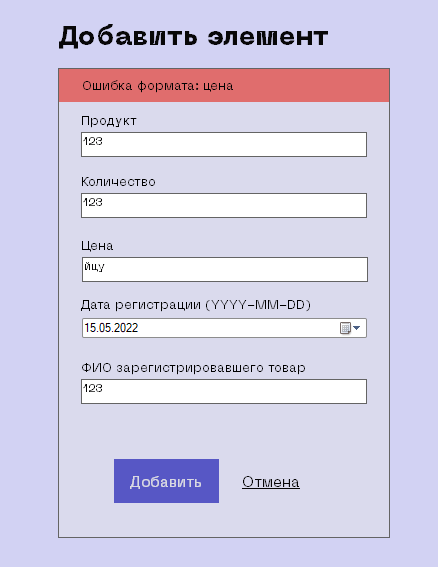
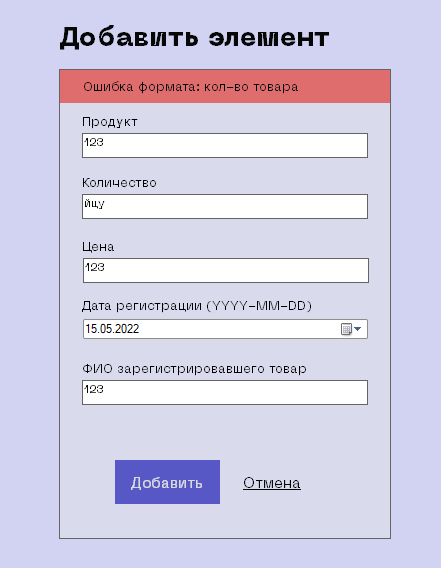


Рисунок 3.9 – Несоответствие формата при добавлении записи

Так же предусмотрено окно подтверждения действия по добавлению или изменению записи. Ниже, на рисунке 3.10, представлено выполнение данного окна.

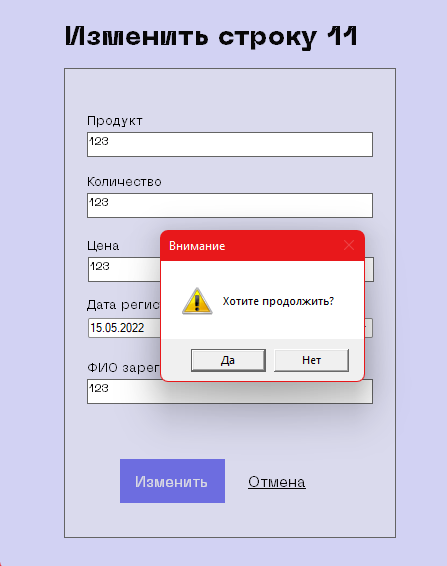
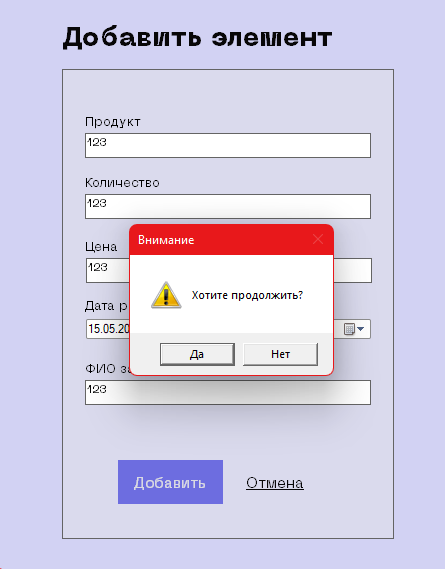


Рисунок 3.10 – Окно подтверждения действия по добавлению или изменению записи

Перейдем к обработке возможных исключений при работе администратора с БД пользователей.

Так же, как и при работе с таблицей товаров, существует сообщении о не нахождении подходящих под запрос записей (рисунок 3.11).

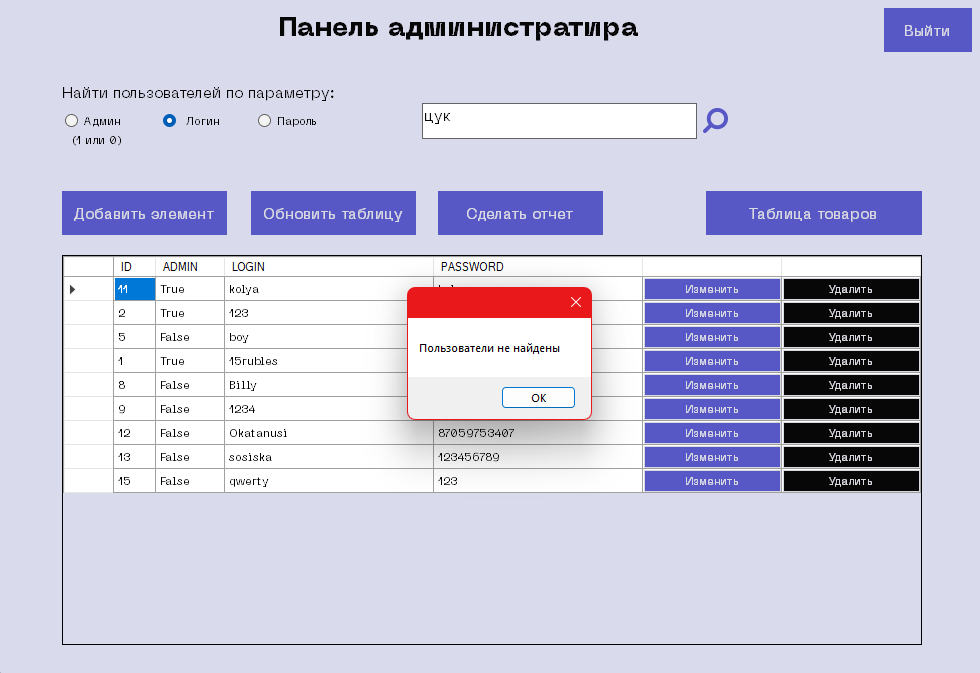


Рисунок 3.11 – Сообщении о не нахождении записей подходящих под запрос

Во избежание потери доступа к возможностям администратора, существует первично созданный аккаунт администратора, который нельзя ни удалить, ни изменить. При попытке сделать обратное, будет выведено следующее сообщение (рисунок 3.12).

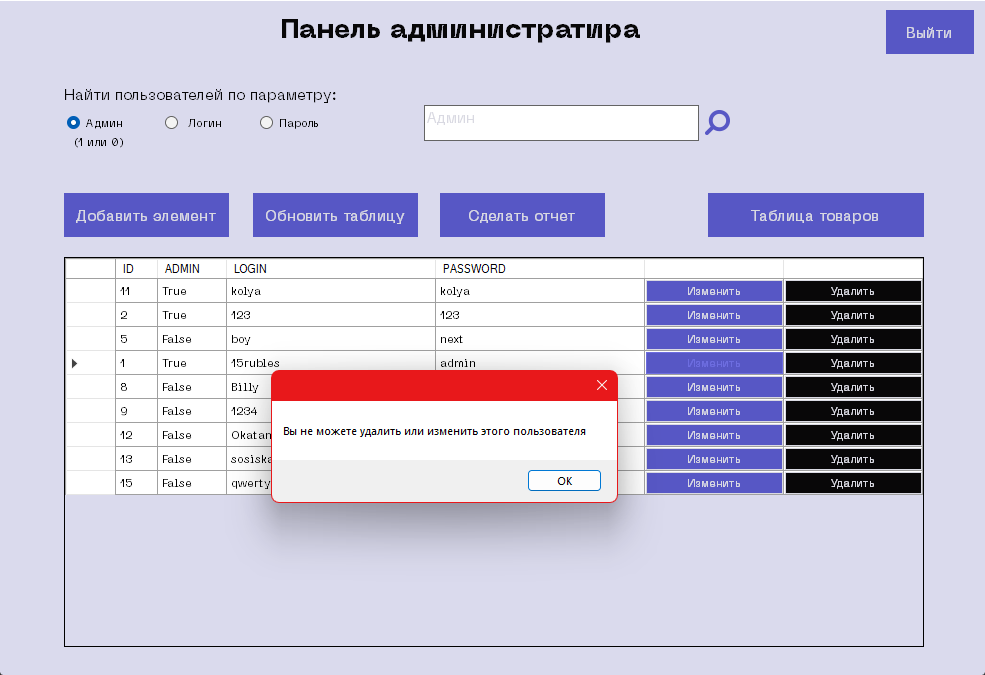


Рисунок 3.12 – Сообщении о попытке удалить или изменить первоначального администратора

При попытке удаления записи пользователя, не являющегося первоначальным администратором, буде выведено следующее окно подтверждения (рисунок 3.13).

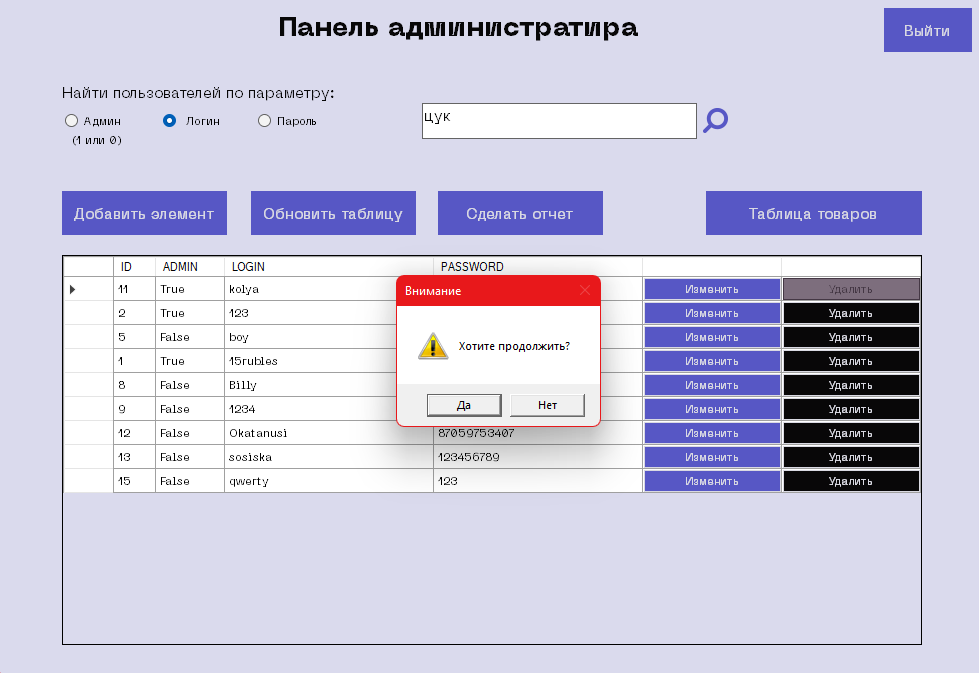


Рисунок 3.13 – Окно подтверждения удаления пользователя

При добавлении или изменения записи пользователя, вызывается соответствующее окно. Внутри него предусмотрено два исключения: поле пароля или логина не должны быть пустыми (рисунок 3.14), а также введенный логин не должен встречаться в уже существующей БД пользователей (рисунок 3.15).

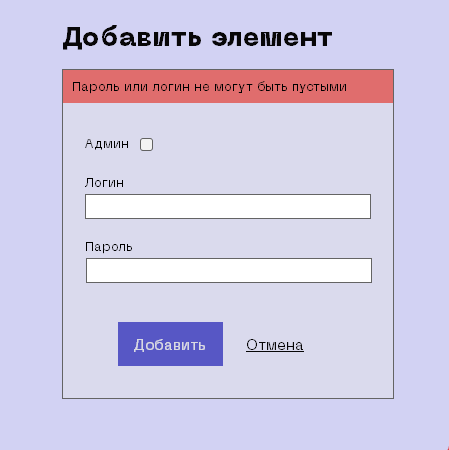
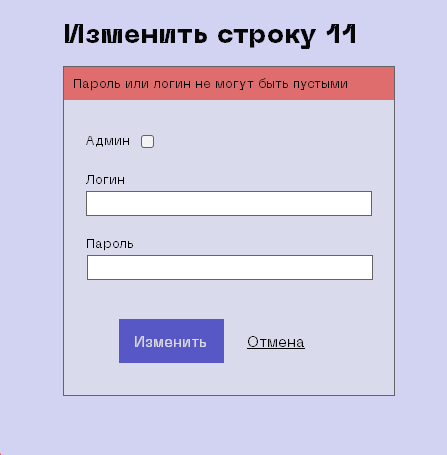


Рисунок 3.14 – Сообщение о пустых полях пароля или логина

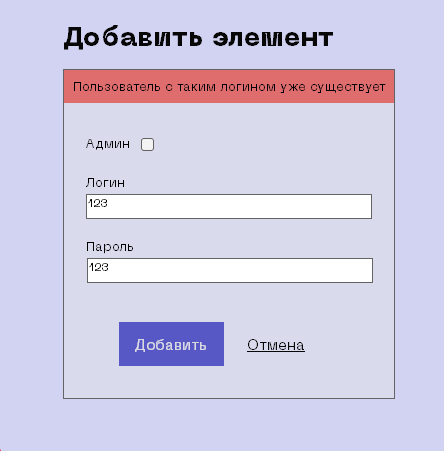
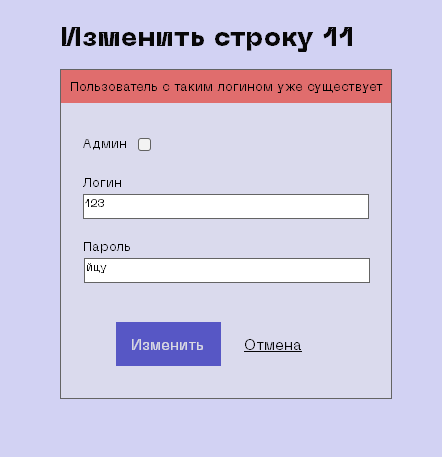


Рисунок 3.15 – Сообщение о существовании пользователя с введенным логином

Так же, как и в случае с добавление или изменением записей о товарах, при добавлении или изменении записей пользователей вызывается окно подтверждения действия (рисунок 3.16).

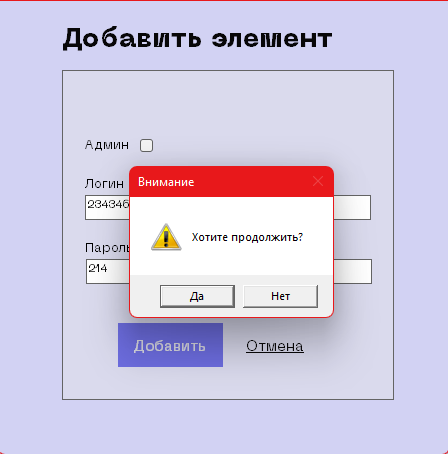
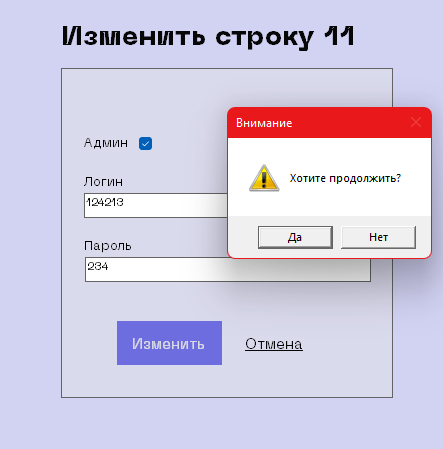


Рисунок 3.16 – Окно подтверждения действия

1. **ИНСТРУКЦИЯ ПО РАЗВЕРТЫВАНИЮ ПРИЛОЖЕНИЯ И СКВОЗНОЙ ТЕСТОВЫЙ ПРИМЕР**

**Общие сведения**. Разработанное программное обеспечение предназначено для складского учета товаров.

В программном обеспечении реализованы следующие функции:

* ввод, хранение и изменение исходных данный о товарах, для дальнейшего просмотра и использования при расчетах;
* поиск и сортировка БД товаров по определенным пользователем параметрам;
* формирование отчетных форм введенных в БД данных, с возможностью сортировки и предварительной выборки данных по параметрам.

**Установка и настройка системы**. Для функционирования приложения требуется установка и настройка серверной части. Она осуществляется стандартным образом. Тестовая база данных *Shop.sql*, поставляемая с приложением, находится в корневой папке проекта.

Запуск приложения происходит с помощью открытия файла *Shop.exe.*

* 1. **Авторизация**

При запуске приложения первое что встречает пользователя – окно входа. Оно предоставляет возможность как входа в систему, так и перехода к форме регистрации (рисунок 4.1).

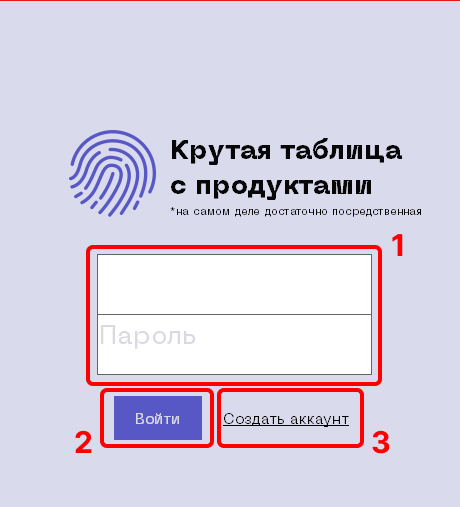


Рисунок 4.1 – Окно входа

На окне представлены следующие элементы:

1. Поля ввода логина и пароля.
2. Кнопка, при нажатии на которую происходит попытка входа, в случае правильного ввода информации, открывается окно, соответствующее статусу пользователя (администратор или нет).
3. Кнопка, ведущая к окну регистрации.

Окно регистрации (рисунок 4.2) представляет собой следующие элементы:

1. Поля ввода логина, пароля и подтверждения пароля.
2. Кнопка создания аккаунта.
3. Кнопка, возвращающая к экрану входа.

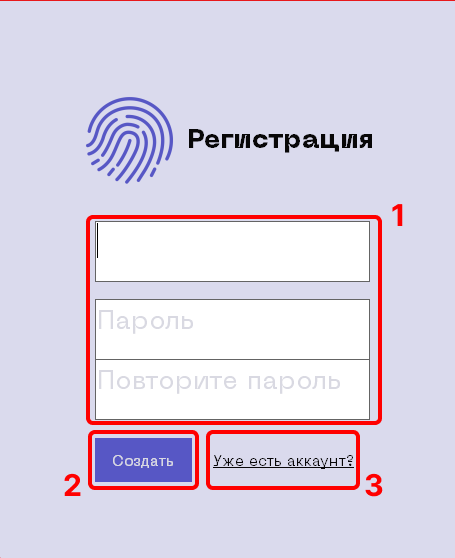


Рисунок 4.2 – Окно регистрации

* 1. **Модуль пользователя**

После авторизации, если пользователь не является администратором, он попадает к окну просмотра информации о товарах на складе (рисунок 3), со следующими элементами:

1. Кнопки выбора параметра для поиска товаров.
2. Поле ввода информации для поискового запроса, а также кнопка поиска (иконка увеличительной лупы).
3. Копка получения отчета, при нажатии на которую открывается текстовый файл с отображающейся на данный момент в таблице информацией.
4. Кнопка выхода из аккаунта.
5. Поля ввода информации для выполнения индивидуального задания и кнопка поиска.
6. Ряд кнопок. При нажатии на какое-либо слово, происходит сортировка по данной колонке таблицы.

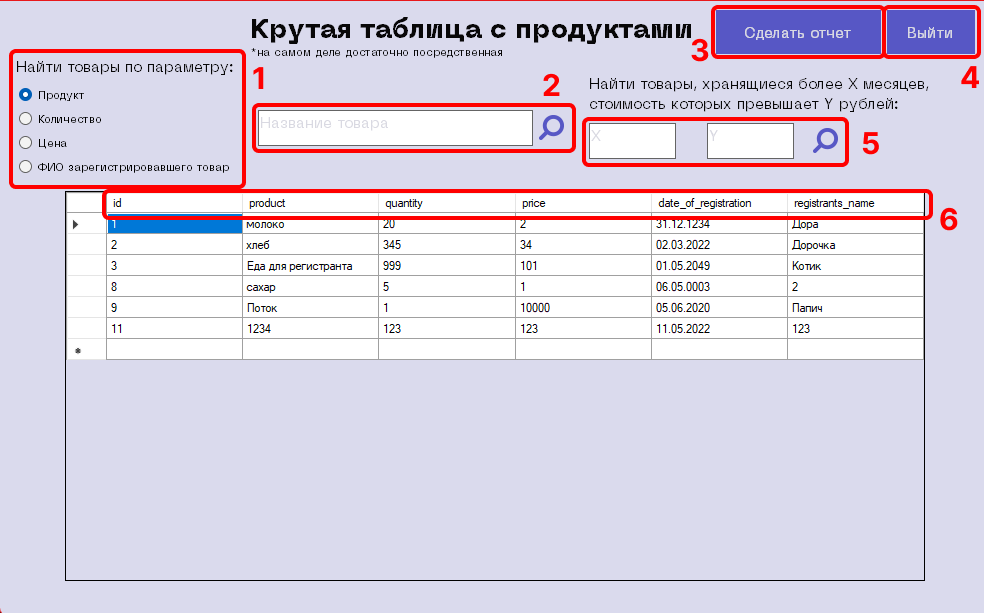


Рисунок 4.3 – Окно пользователя

* 1. **Модуль администратора**

Если пользователь является администратором, то вместо вышеописанного окна, открывается окно администратора с данными о товарах на складе. Репрезентации данного окна представлена на рисунке 4.4.

Так же, данное окно можно описать следующими элементами:

1. Кнопки выбора параметра для поиска товаров.
2. Поле ввода информации для поискового запроса, а также кнопка поиска (иконка увеличительной лупы).
3. Кнопка выхода из аккаунта.
4. Поля ввода информации для выполнения индивидуального задания и кнопка поиска.
5. Кнопка добавления новой записи в БД товаров.
6. Кнопка обновления таблицы, которая заполняет ее данными, находящимися в данный момент в БД товаров.
7. Копка получения отчета, при нажатии на которую открывается текстовый файл с отображающейся на данный момент в таблице информацией.
8. Кнопка, открывающая окно администратора с таблицей аккаунтов пользователей.
9. Ряд кнопок. При нажатии на какое-либо слово, происходит сортировка по данной колонке таблицы.
10. Кнопки изменения и удаления записи о товаре в данной строке таблицы.

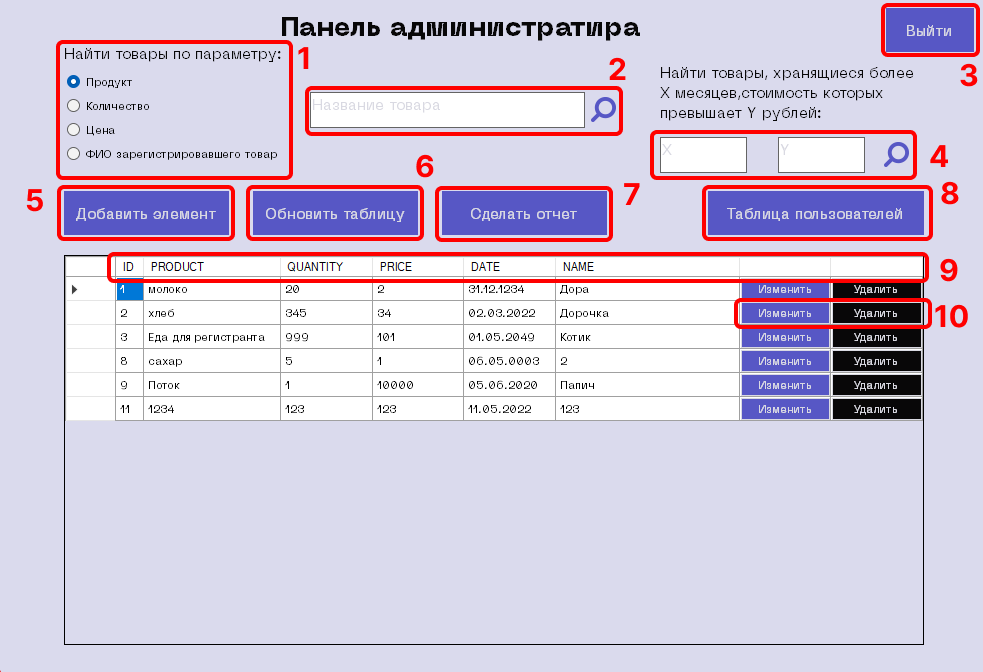


Рисунок 4.4 – Окно администратора с данными о товарах

Окно администратора с аккаунтами пользователей представлено на рисунке 4.5. Оно состоит из следующих элементов:

1. Кнопки выбора параметра для поиска товаров.
2. Поле ввода информации для поискового запроса, а также кнопка поиска (иконка увеличительной лупы).
3. Кнопка выхода из аккаунта.
4. Поля ввода информации для выполнения индивидуального задания и кнопка поиска.
5. Кнопка добавления новой записи в БД товаров.
6. Кнопка обновления таблицы, которая заполняет ее данными, находящимися в данный момент в БД товаров.
7. Копка получения отчета, при нажатии на которую открывается текстовый файл с отображающейся на данный момент в таблице информацией.
8. Кнопка, открывающая окно администратора с таблицей аккаунтов пользователей.
9. Ряд кнопок. При нажатии на какое-либо слово, происходит сортировка по данной колонке таблицы.
10. Кнопки изменения и удаления записи о товаре в данной строке таблицы.

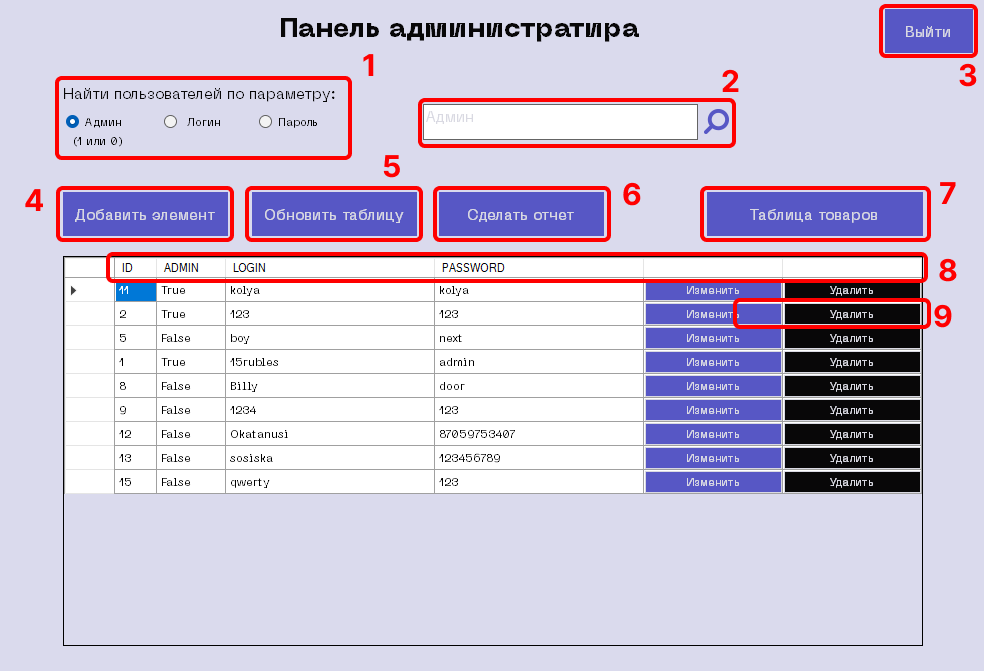


Рисунок 4.5 – Окно администратора с данными об аккаунтах пользователей

# **ЗАКЛЮЧЕНИЕ**

В курсовой работе представлен анализ предметной области складского учета, ее основных особенностей. Также найден необходимый минимум функциональных требований к программному обеспечению для улучшения эффективности деятельности складских и бухгалтерских отделов компаний в данной сфере.

По итогу проделанной работы, предоставлено ПО, позволяющее обеспечить весь необходимый минимальный функционал для оптимизации учета товаров на складах предприятий.

Разработанное программное средство обеспечивает выполнение следующих функций:

* ввод и хранение исходных данных для формирования промежуточных и результатных данных: таблица товаров на складе, таблица пользователей ПО;
* формирование и хранение данных промежуточных и итоговых расчетов: данные о товарах и аккаунтах пользователей по критериям отбора, сортировка товаров и аккаунтов по критериям;
* формирование отчетных форм с любыми видами запросов к ПО: сортировка, поиск;
* модуль администратора, позволяющий добавлять и изменять данные в имеющихся БД (БД пользователей, БД товаров).

Система имеет приятный и доступный интерфейс, полный набор функций, необходимых для складского учета.

Для достижения поставленной цели, успешно решены следующие задачи:

* проанализирована предметная область;
* разработан список необходимых функциональных требований;
* организована связь приложения с базами данных;
* разработан алгоритм работы приложения;
* разработано приложение;
* описаны функциональные возможности приложения и принцип его работы.

**Список использованных источников**

1. Статья «Что такое база данных?» [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://www.oracle.com/cis/database/what-is-database/>.
2. Статья «Зачем нам UML? Или как сохранить себе нервы и время» [Электронный ресурс]. – Электронные данные. – Режим доступа: https://habr.com/ru/post/458680/.
3. Статья «Что такое нотация моделирования бизнес-процессов» [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.lucidchart.com/pages/ru/bpmn.
4. Статья «IDEF0. Знакомство с нотацией и пример использования» [Электронный ресурс]. – Электронные данные. – Режим доступа: https://trinion.org/blog/idef0-znakomstvo-s-notaciey-i-primer-ispolzovaniya.
5. Статья «Руководство по классическим приложениям (Windows Forms .NET)» [Электронный ресурс]. – Электронные данные. – Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/desktop/winforms/overview/?view=netdesktop-6.0.
6. Статья «10 основных сложностей в работе неавтоматизированного склада» [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.dirmagazina.ru/article/534-10-osnovnyh-slojnostey-v-rabote-neavtomatizirovannogo-sklada.

**Приложение А**

**(обязательное)**

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

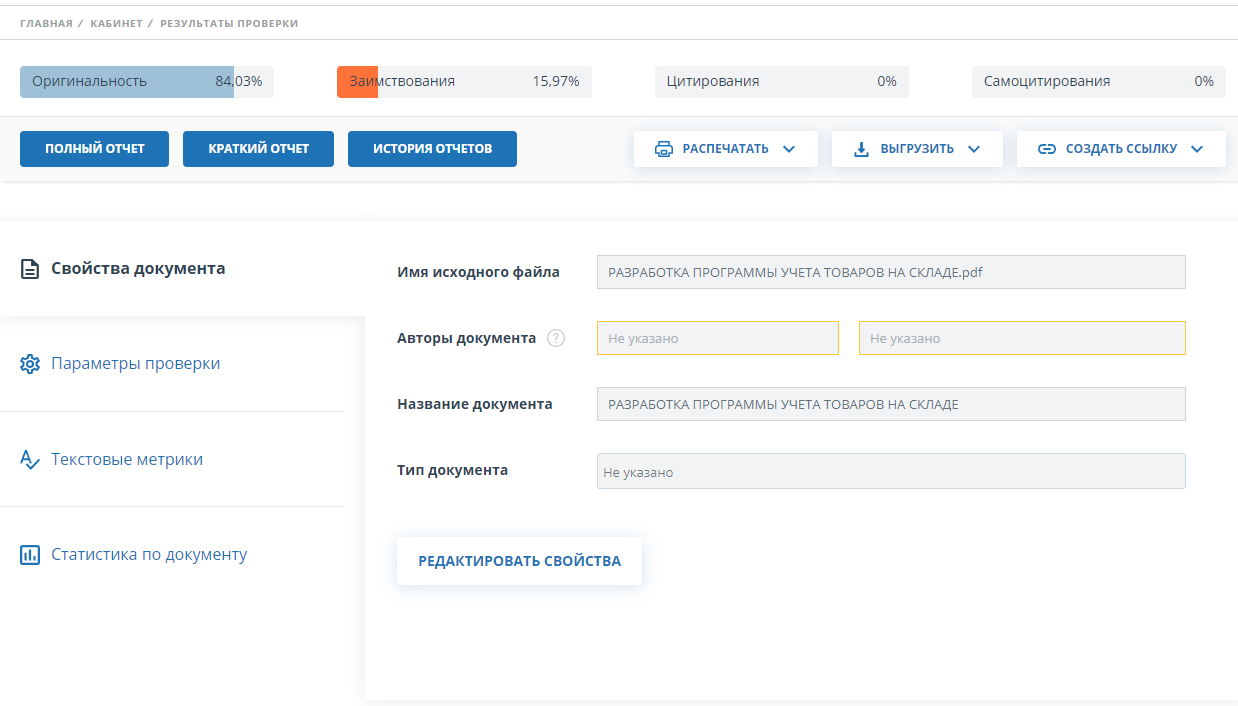
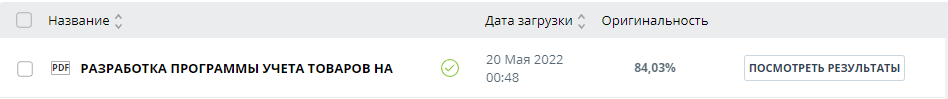


Рисунок А.1 – Отчет о проверке на заимствования в системе «Антиплагиат»

**Приложение Б**

**(обязательное)**

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

//try to registrate

private void RegistrationButton\_Click(object sender, EventArgs e)

{

if (LoginBox.Text == "Логин")

{

MessageBox.Show("Введите логин");

return;

}

if (PassBox.Text == "Пароль")

{

MessageBox.Show("Введите пароль");

return;

}

if (PassBox2.Text == "Повторите пароль")

{

MessageBox.Show("Введите пароль повторно");

return;

}

if (PassBox.Text != PassBox2.Text)

{

MessageBox.Show("Пароли не совпадают");

return;

}

DataBase db = new DataBase();

MySqlCommand command = new MySqlCommand("INSERT INTO `users` (`login`, `password`) VALUES (@uL, @uP);", db.GetConnection());

command.Parameters.Add("@uL", MySqlDbType.VarChar).Value = LoginBox.Text;

command.Parameters.Add("@uP", MySqlDbType.VarChar).Value = PassBox.Text;

if (IsUserExist()) return;

db.OpenConnection();

if (command.ExecuteNonQuery() == 1)

{

this.Hide();

LoginForm form = new LoginForm();

form.Show();

}

else MessageBox.Show("Ошибка при регистрации");

db.CloseConnection();

}

public bool IsUserExist()

{

DataBase db = new DataBase();

DataTable table = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand command = new MySqlCommand("SELECT \* FROM `users` WHERE `login` = @uL", db.GetConnection());

command.Parameters.Add("@uL", MySqlDbType.VarChar).Value = LoginBox.Text;

adapter.SelectCommand = command;

adapter.Fill(table);

if (table.Rows.Count > 0)

{

MessageBox.Show("Пользователь с таким логином уже существует, введите другой");

return true;

}

else return false;

}

//update information about product

private void UpdateButton\_Click(object sender, EventArgs e)

{

int conwert, month;

bool isXInt = Int32.TryParse(QuantityBox.Text, out conwert);

if (!isXInt)

{

Errorlabel.Text = FOR\_ALL + QUANTITY;

ErrorPanel.Visible = true;

return;

}

isXInt = Int32.TryParse(PriceBox.Text, out conwert);

if (!isXInt)

{

Errorlabel.Text = FOR\_ALL + PRICE;

ErrorPanel.Visible = true;

return;

}

DialogResult result = MessageBox.Show("Хотите продолжить?", "Внимание", MessageBoxButtons.YesNo, MessageBoxIcon.Warning, MessageBoxDefaultButton.Button1, MessageBoxOptions.DefaultDesktopOnly);

if (result == DialogResult.Yes)

{

string date = dateTimePicker1.Value.Year + "-" + dateTimePicker1.Value.Month + "-" + dateTimePicker1.Value.Day;

DataBase db = new DataBase();

DataTable table = new DataTable();

MySqlCommand command = new MySqlCommand();

if (isAddPush) command.CommandText = "INSERT INTO `storage` (`id`, `product`, `quantity`, `price`, `date\_of\_registration`, `registrants\_name`) VALUES (NULL, @prod, @quant, @price, @date, @name);";

else command.CommandText = "UPDATE `storage` SET `product` = @prod, `quantity` = @quant, `price` = @price, `date\_of\_registration` = @date, `registrants\_name` = @name WHERE `storage`.`id` = @id";

command.Connection = db.GetConnection();

command.Parameters.Add("@prod", MySqlDbType.VarChar).Value = ProductBox.Text;

command.Parameters.Add("@quant", MySqlDbType.VarChar).Value = QuantityBox.Text;

command.Parameters.Add("@price", MySqlDbType.VarChar).Value = PriceBox.Text;

command.Parameters.Add("@date", MySqlDbType.VarChar).Value = date;

command.Parameters.Add("@name", MySqlDbType.VarChar).Value = NameBox.Text;

if (!isAddPush) command.Parameters.Add("@id", MySqlDbType.VarChar).Value = idPush;

command.ExecuteNonQuery();

this.Hide();

}

}

//personal task

private void TaskButton\_Click\_1(object sender, EventArgs e)

{

if (XBox.Text != X && YBox.Text != Y)

{

int x, y;

bool isXInt = Int32.TryParse(XBox.Text, out x);

bool isYInt = Int32.TryParse(YBox.Text, out y);

if (isXInt && isYInt)

{

DataBase db = new DataBase();

DataTable table = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand command = new MySqlCommand("SELECT \* FROM `storage` WHERE `price` > @y AND TIMESTAMPDIFF(MONTH,`date\_of\_registration`,CURDATE()) > @x ", db.GetConnection());

command.Parameters.Add("@x", MySqlDbType.VarChar).Value = x;

command.Parameters.Add("@y", MySqlDbType.VarChar).Value = y;

adapter.SelectCommand = command;

adapter.Fill(table);

if (table.Rows.Count > 0)

{

FillTable(table);

dataGridView1.Sort(dataGridView1.Columns["Prod"], ListSortDirection.Ascending);

}

else MessageBox.Show(NOT\_FOUNT);

}

else

{

XBox.Text = X;

XBox.ForeColor = ColorTranslator.FromHtml("#D9D9E2");

YBox.Text = Y;

YBox.ForeColor = ColorTranslator.FromHtml("#D9D9E2");

MessageBox.Show("X и Y должны быть числами");

}

}

}

//try to login

private void LoginButton\_Click(object sender, EventArgs e)

{

String userLogin = LoginBox.Text;

String userPass = PassBox.Text;

DataBase db = new DataBase();

DataTable table = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand command = new MySqlCommand("SELECT \* FROM `users` WHERE `login` = @uL AND `password` = @uP",db.GetConnection());

command.Parameters.Add("@uL", MySqlDbType.VarChar).Value = userLogin;

command.Parameters.Add("@uP", MySqlDbType.VarChar).Value = userPass;

adapter.SelectCommand = command;

adapter.Fill(table);

if (table.Rows.Count > 0)

{

if (table.Rows[0][1].ToString() == "True")

{

AdminMain form = new AdminMain();

this.Hide();

form.Show();

}

else

{

this.Hide();

MainUser form = new MainUser();

form.Show();

}

}

else MessageBox.Show(USER\_DOES\_NOT\_EXIST);

}

//make a report

private void ReportButton\_Click(object sender, EventArgs e)

{

string path = "report.txt";

File.WriteAllText(path, String.Empty);

using (StreamWriter writer = new StreamWriter(path, true))

{

for (int i = 0; i < dataGridView1.Columns.Count - 2; i++)

writer.Write(dataGridView1.Columns[i].HeaderText + " ");

writer.WriteLine();

for (int i = 0; i < dataGridView1.Rows.Count; i++)

{

for (int j = 0; j < dataGridView1.Columns.Count - 2; j++)

{

writer.Write(dataGridView1.Rows[i].Cells[j].Value + " ");

}

writer.WriteLine();

}

}

var proc = new System.Diagnostics.Process();

proc.StartInfo.FileName = "report.txt";

proc.StartInfo.UseShellExecute = true;

proc.Start();

}

**Ведомость документов курсовой работы**