

Тема работы: Разработка информационной системы учёта товаров на складах на языке Java

**Разработчик, Geekbrains**

**Аскерова Оксана Васильевна**

**2024 г**

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#bookmark4)

1. Анализ требований на разработку 7
   1. Описание бизнес-требований 7
   2. Разработка модели данных 10
      1. Выделение существительных и глаголов 10
      2. [Разработка концептуальной карты 11](#bookmark12)
      3. [Разработка ER-моделей 12](#bookmark14)
   3. Моделирование бизнес-процессов 13
      1. Описание предметной области 13
      2. [Табличное представление результатов анализа процесса 14](#bookmark19)
      3. [BPMN2 модель процессов 15](#bookmark21)
   4. [Разработка пользовательских требований 17](#bookmark23)
      1. [Диаграмма вариантов использования 17](#bookmark25)
      2. [Спецификация вариантов использования 18](#bookmark27)
   5. [Разработка функциональных требований 24](#bookmark29)
   6. [Разработка нефункциональных требований 25](#bookmark31)
   7. [Выводы по главе 26](#bookmark33)
2. [Проектирование информационной системы учета товаров на складах 27](#bookmark35)
   1. [Архитектура веб-приложения 27](#bookmark37)
   2. [Функциональная модель системы 29](#bookmark39)
   3. [Разработка алгоритмов системы 33](#bookmark41)
      1. [Алгоритм функции приемки 33](#bookmark43)
      2. [. Алгоритм функции списания 34](#bookmark45)
      3. [Алгоритм функции перемещения 35](#bookmark47)
   4. [Структура базы данных веб-приложения 36](#bookmark49)
   5. [Выводы по главе 41](#bookmark51)
3. Реализация информационной системы 42
   1. Реализация веб-приложения 42
      1. Описание методов сервера 43
   2. [Формирование документов 51](#bookmark72)
      1. [Заполнение шаблона документов 52](#bookmark74)
      2. [Сохранение документов 53](#bookmark76)
   3. [Описание интерфейса веб-приложения 54](#bookmark78)
      1. [Страница «Главная» 54](#bookmark80)
      2. Страница «Регистрация» 54
      3. [Страница «Вход в систему» 55](#bookmark82)
      4. [Страница «Товары» 56](#bookmark84)
      5. [Страница «Склады» 58](#bookmark86)
      6. [Страница «Сводка» 59](#bookmark88)
      7. [Страница «Приемка» 60](#bookmark90)
      8. [Страница «Перемещение» 61](#bookmark92)
      9. [Страница «Списание» 62](#bookmark94)

[3.3 Выводы по главе 62](#bookmark96)

[ЗАКЛЮЧЕНИЕ 63](#bookmark98)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 65](#bookmark100)

ПРИЛОЖЕНИЕ А 67

ПРИЛОЖЕНИЕ Б 72

ВВЕДЕНИЕ

ООО «Авто-Про» - компания, занимающаяся продажей автотоваров. У компании работает несколько магазинов на территории Татарстана и Самарской области. Благодаря выгодным ценам и высокому качеству товары компании пользуются спросом, количество клиентов растет. Руководители собираются расширять сеть магазинов, а именно открывать точки в других регионах России.

Однако компания все чаще сталкивается с проблемой ведения учета товаров на складах. На данный момент складской учет ведется вручную. Если раньше при малых оборотах товара сотрудники легко справлялись с учетом, то теперь эта задача становится трудновыполнимой. В первую очередь страдает скорость работы. Время на поиски необходимых документов, проведение подсчетов и сверок при большом количестве товаров существенно увеличивается. Также становится тяжело контролировать количество остатков и запасов на складах, возникают ошибки, и возрастает риск воровства и потери товаров. Актуальность работы обусловлена необходимостью автоматизировать складской учет и ведение документооборота для повышения скорости и качества работы персонала склада.

Объектом исследования являются компания ООО «Авто-Про», которая занимается продажей товаров для автомобилей.

Предметом исследования является процесс учета товаров на складах данной компании.

Целью данной работы является улучшение точности учета, сокращение времени на выполнение операций, снижение рисков ошибок и повышение общей эффективности работы за счет автоматизации процессов.

Для достижения поставленной цели требуется решить следующие задачи:

* провести анализ деятельности организации, в частности процесса складского учета;
* разработать модели процесса;
* разработать пользовательские требования;
* разработать функциональные и нефункциональные требования;
* рассмотреть архитектуру веб-приложения;
* разработать функциональную модель системы;
* разработать алгоритмы системы;
* рассмотреть структуру базы данных приложения;
* реализовать систему.

Структура работы обусловлена предметом, целью и задачами исследования. Работа состоит из введения, трех глав и заключения. Введение раскрывает актуальность, объект, предмет, цель, задачи работы «Разработка информационной системы учета товаров на складах на языке Java». В первой главе произведен анализ требований на разработку. Вторая глава посвящена проектированию информационной системы, а третья реализации системы. В заключении подводятся итоги исследования, формируются окончательные выводы по рассматриваемой теме.

1 Анализ требований на разработку

1.1 Описание бизнес-требований

Организация ООО «Авто-Про» специализируется на продаже автотоваров. В настоящее время у компании есть четыре розничных магазина, расположенных в различных городах Республики Татарстан и Самарской области. Однако они сталкиваются с проблемами учета товаров на своих складах. Все операции, связанные с приемкой товаров, фиксацией поступлений и списаний, перемещениями товаров в настоящее время выполняются вручную. Это занимает много времени и может приводить к ошибкам. В связи с этим компания приняла решение автоматизировать данный процесс, чтобы упростить его выполнение и устранить возможные ошибки.

Возможности бизнеса: эффективный контроль за количеством товаров на складах и уровнем запасов; снижение риска потерь и кражи товаров; надежное хранение и доступность информации; упрощение процесса ведения документооборота.

Бизнес-цели: достижение снижения потерь и кражи товаров на 60% к концу II квартала 2023 года; успешная тестовая эксплуатация системы во II квартале 2023 года. Критерии успеха: сокращение времени, затрачиваемого на выполнение операций по учету товаров; реализация основного функционала проекта к концу II квартала 2023 года.

Видение решения. Решение представляет собой современное веб­приложение, разработанное на языке Java с использованием фреймворка Spring Boot. Оно будет предоставлять удобный интерфейс для персонала склада, позволяющий легко принимать товары от поставщиков, генерировать необходимые документы и выполнять различные операции с товарами. Благодаря автоматизации процесса учета товаров на складе ожидается 7

значительное повышение эффективности работы складского персонала, сокращение времени на учет товаров и повышение точности данных.

Бизнес-риски: риск превышения бюджета проекта, что может привести к нехватке финансирования и ограничениям в реализации планов; риск задержки сроков сдачи проекта из-за возникновения проблем в процессе разработки, например, технических проблем или изменений в требованиях; нежелание сотрудников компании переходить на новую систему и

возможности саботажа со стороны персонала, что может привести к

сопротивлению и снижению эффективности внедрения.

Предположения и зависимости: доступ к изменению данных будет возможен только для авторизованных пользователей; система будет способна работать эффективно независимо от количества товаров.

Основные функции решения: авторизация; добавление, редактирование и удаление складов и товаров; поиск товаров по названию; вывод сводки по товарам на складах; приемка товаров; списание товаров со склада; перемещение товаров между складами; формирование складских документов.

MVP: авторизация; добавление, редактирование и удаление складов и товаров; вывод сводки по товарам на складах. Прочие версии продукта:

Версия 2: формирование этикеток и ценников, экспорт данных в 1С Бухгалтерию, мобильное приложение. Стейкхолдеры проекта приведены в таблице 1.1:

Таблица 1.1 - Профили заинтересованных лиц.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Стейкхолдеры** | **Ценность** | **Вероятное отношение к**  **продукту** | **Ограничения** |
| 1 | Руководитель | Повышение эффективности работы компании | Заинтересованность в минимальных затратах | Стоимость разработки и обслуживания системы |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Продолжение таблицы 1.1 | | | | |
| 2 | Персонал склада | Упрощение ведения учета товаров на складах | Заинтересованность в автоматизации рутинных задач и уменьшении бумажных работ | Сложность переобучения |
| 3 | Продавцы | Предоставление точной информации клиентам | Заинтересованность в быстром получении актуальной информации о наличии товаров | Сложность обучения |
| 4 | Бухгалтеры | Обеспечение  точности данных в бухгалтерской отчетности | Заинтересованность в возможности получения точной информации для финансового отдела | Бюджет |

Приоритеты проекта: все функции MVP должны быть выполнены; критическая функциональность должна работать без ошибок; версия MVP должны быть запущена в тестовую эксплуатацию в III квартале 2023 года; расходы на разработку 500 000 рублей; предполагаемый персонал: проджект- менеджер, бэк-енд разработчик на Java, фронт-энд разработчик, специалист по интеграции.

- руководитель:

1. просмотр сводки по товарам и складам;
2. мониторинг операций приемки, списания и перемещения; товаров для контроля эффективности работы склада;
3. изучение информации о документах.

- персонал склада:

1. добавление товаров на склад;
2. выполнение операции приемки товаров;
3. списание товаров со склада;
4. перемещения товаров между складами;
5. формирование складских документов.

- продавцы:

1. поиск товаров по названию;
2. просмотр сводки по товарам для оценки доступности и количества товаров.

- бухгалтеры:

1. мониторинг финансовых операций;
2. просмотр сводки по товарам и складам;
3. экспорт данных в бухгалтерскую систему.
   1. Разработка модели данных
      1. Выделение существительных и глаголов

Выделение существительных и глаголов. Проанализировав бизнес- требования был сформирован список из существительных-объектов, глаголов и существительных-ролей, который представлен в таблице 1.2.

Таблица 1.2 - Существительные, глаголы и роли.

|  |  |  |
| --- | --- | --- |
| **Существительные** | **Глаголы** | **Роли** |
| Товар | Создать | Руководитель |
| Склад | Хранить | Персонал склада |
| Запасы | Искать | Продавец |
| Приемка | Выполнить | Бухгалтер |
| Списание | Получить |  |
| Перемещение | Управлять |  |
| Документ |  |  |

* + 1. Разработка концептуальной карты

В результате проведения анализа были выявлены три основные сущности в предметной области: «Товар», «Склад» и «Запасы». «Товары» хранятся на складе, а «Склад» содержит информацию о «Запасах».

«Персонал склада» осуществляет «Приемку» товаров, а также при необходимости «Списание» товаров со склада или «Перемещение» товаров между складами. После каждой операции формируются соответствующие «Документы», содержащие информацию о проведенных операциях.

«Бухгалтер» осуществляет анализ и учет всех складских «Документов» для бухгалтерской отчетности и поддержания финансовой документации в актуальном состоянии. «Продавцы» используют информацию о «Запасах», чтобы уведомить клиентов о наличии или отсутствии товаров на складе. «Руководитель», основываясь на данных о «Запасах», принимает решения о закупке товаров.

На рисунке 1.1 представлена концептуальная карта предметной области.

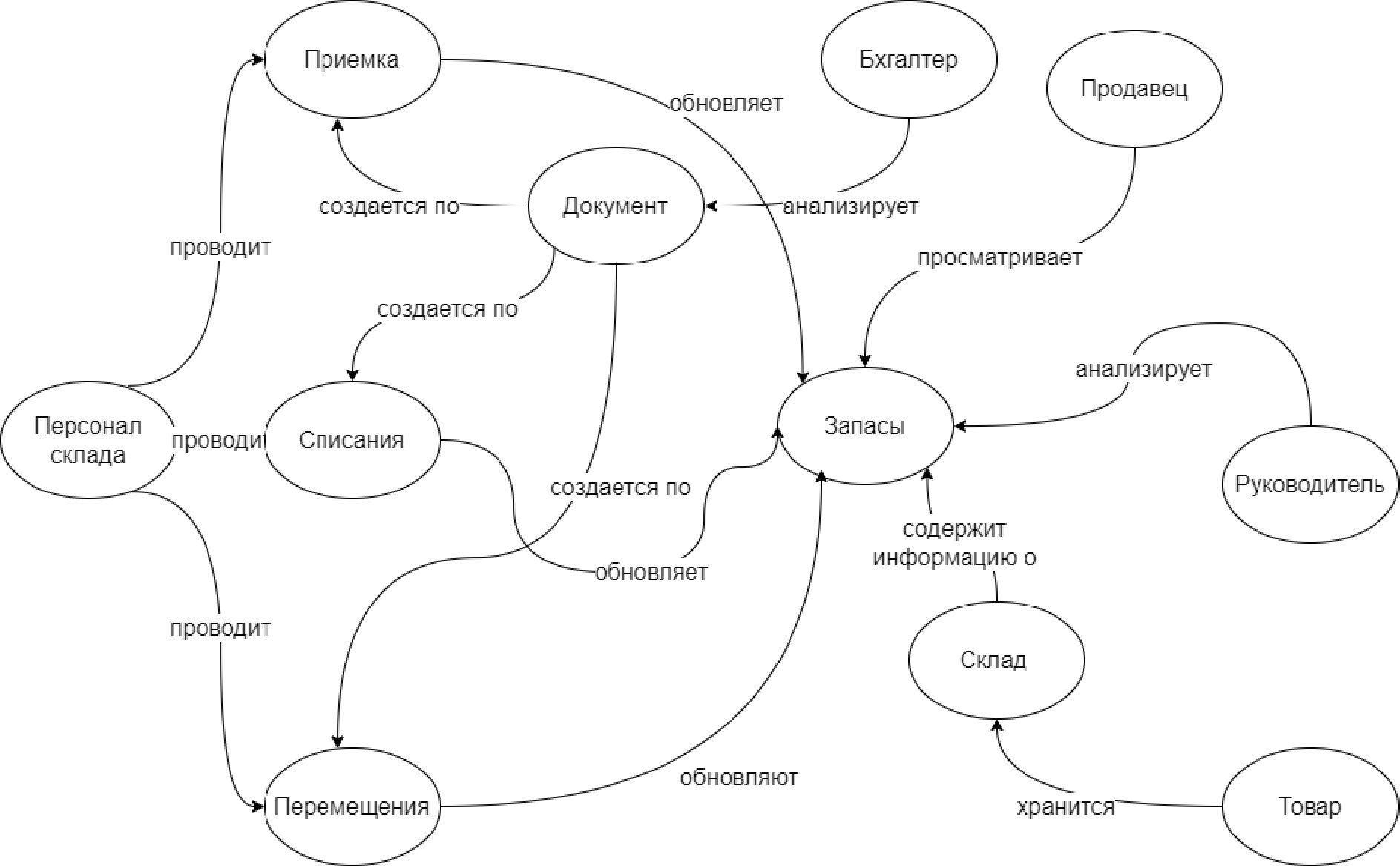


Рисунок 1.1 - Концептуальная карта предметной области

* + 1. Разработка ER-моделей

ER-модель основных сущностей процесса учета товаров на складах без атрибутов представлена на рисунке 1.2 [11]:

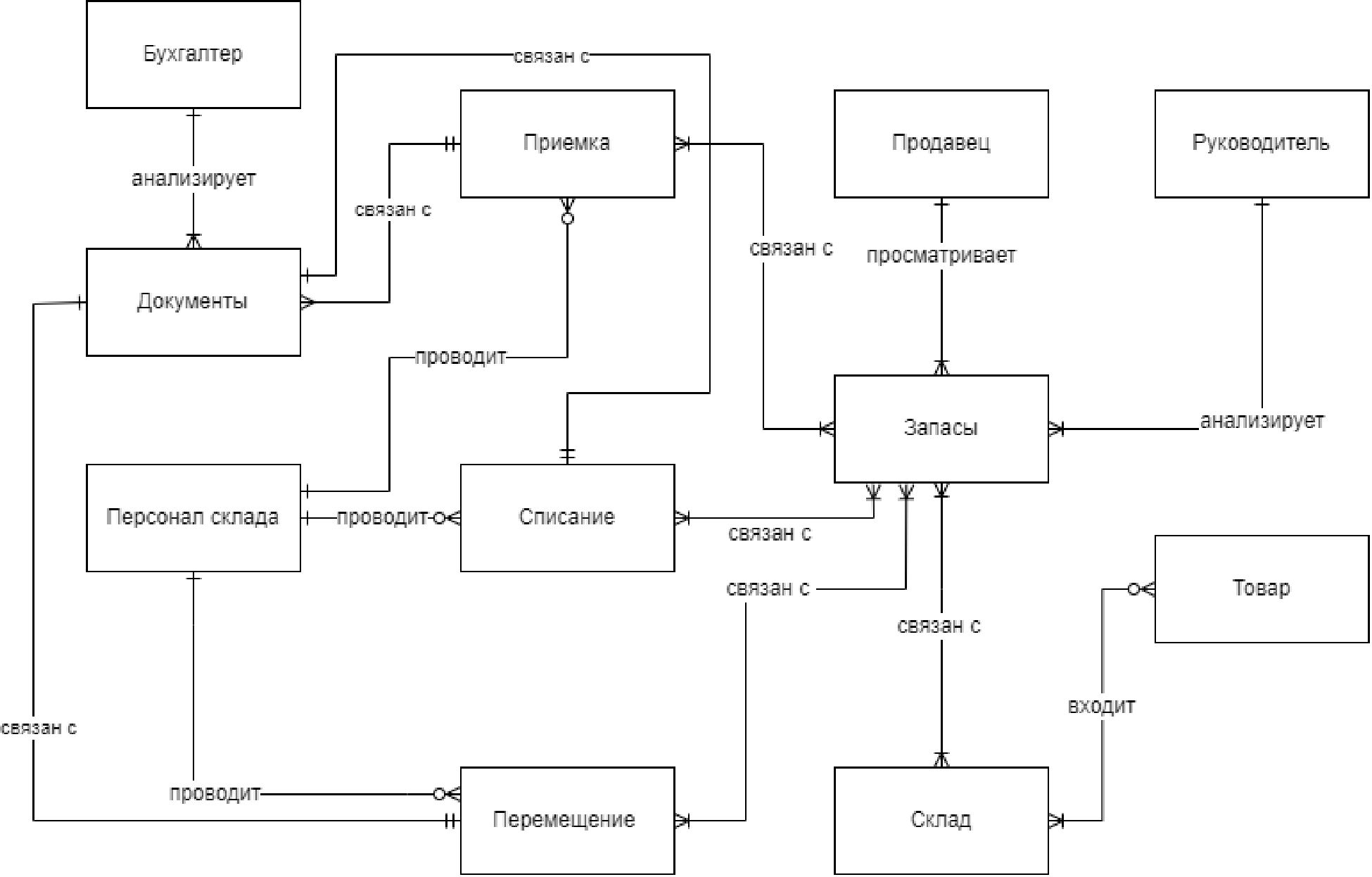


Рисунок 1.2 - ER-модель процесса учёта товаров без атрибутов

Товар может входить в один или несколько складов.

Склад может содержать ноль или множество товаров. Склад и Запасы связаны между собой множественным отношением.

Персонал склада проводит множество приемок, списаний и перемещений. Данные операции выполняются от лица одного пользователя и могут быть связаны с несколькими запасами.

Один документ относится только к одной приемке, списанию и перемещению. При этом перемещение и списание имеют по одному документы, а приемка имеет несколько документов.

Бухгалтер анализирует несколько документов. Руководитель и Продавец анализирует несколько запасов.

На рисунке 1.3 продемонстрирована ER-модель с атрибутами

сущностей [11].

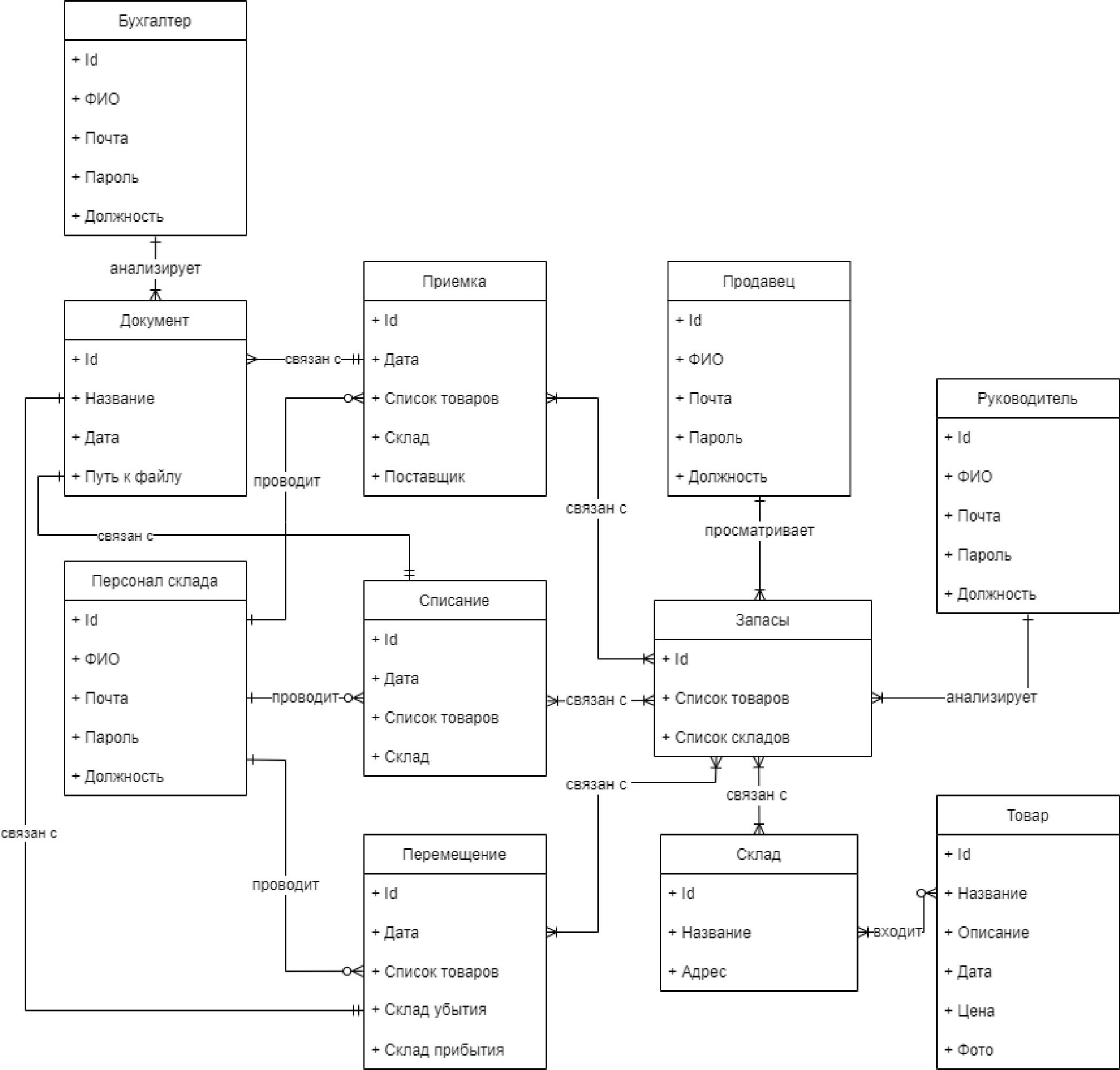


Рисунок 1.3 - ER-модель процесса учёта товаров с атрибутами сущностей

* 1. Моделирование бизнес-процессов
     1. Описание предметной области

ООО «Авто-Про» занимается продажей широкого ассортимента товаров для автомобилей. С постоянным расширением ассортимента возникают определенные проблемы с учетом товаров на складах. Прежде 13

всего, увеличение числа позиций товаров приводит к увеличению номенклатурных единиц, что затрудняет работу сотрудников, поиск нужных товаров занимает длительное время. Во-вторых, большой объем документации создает сложности при подсчетах и сверках. Возникают проблемы с точностью и надежностью учета товаров, что может привести к потерям, кражам и просрочкам.

В целом, проблемы с учетом товаров на складах существенно затрудняют работу компании, негативно влияют на ее эффективность, точность учета и безопасность товаров.

* + 1. Табличное представление результатов анализа процесса

В таблице 1.3 представлен анализ процесса учета товаров.

Таблица 1.3 - Анализ процесса учета товаров.

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Роль** |
| 1 | Товар поступил на продажу | Результат | Продавец |
| 2 | Списание товара | Действие | Персонал склада |
| 3 | Перемещение товара | Действие | Персонал склада |
| 4 | Формирование документов | Действие | Персонал склада |
| 5 | Занесения информации о товаре | Действие | Персонал склада |
| 6 | Написание претензии поставщику | Действие | Персонал склада |
| 7 | Товар был принят | Решение | Персонал склада |
| 8 | Товар был отклонен | Решение | Персонал склада |
| 9 | Приемка товаров | Действие | Персонал склада |
| 10 | Товар не доставлен | Событие | Персонал склада |
| 11 | Товар доставлен | Событие | Персонал склада |
| 12 | Заключение договора с поставщиком | Действие | Руководитель |

|  |  |  |  |
| --- | --- | --- | --- |
| 13 | Возникла необходимость закупить товар | Событие | Руководитель |

* + 1. BPMN2 модель процессов

Сформируем диаграммы BPMN для бизнес-процессов учета товаров на

складах [1, 13]. Диаграмма BPMN продемонстрирована на рисунке 1.4:

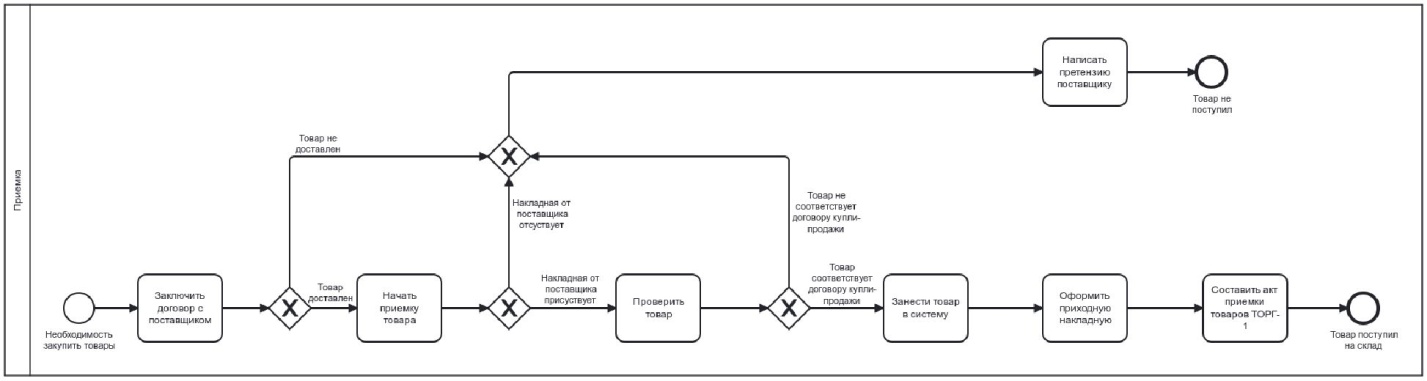


Рисунок 1.4 - BPMN диаграмма процесса «Приемка»

При необходимости закупить товар заключается договор с поставщиком. После этого ожидается доставка товара. Когда товар поступает, кладовщик начинает процесс приемки товаров. В первую очередь проводится проверка товара на брак, соответствие количества, массы и комплектности. Если товар соответствует условиям договора купли-продажи, кладовщик вносит информацию о товаре в систему, оформляет приходную накладную и акт приемки ТОРГ-1 при необходимости. Товар начинает числиться на складе. В случае если поставщик не доставил товар или не предоставил накладную, либо товар не соответствует требованиям качества, составляется претензия на поставщика [10].

Описанный процесс приемки товара позволяет обеспечить контроль над поставками, проверку качества и количества товаров, а также обновление информации в системе учета.

Следующим рассмотрим процесс Перемещения товаров между складами. BPMN-диаграмма процесса представлена на рисунке 1.5:

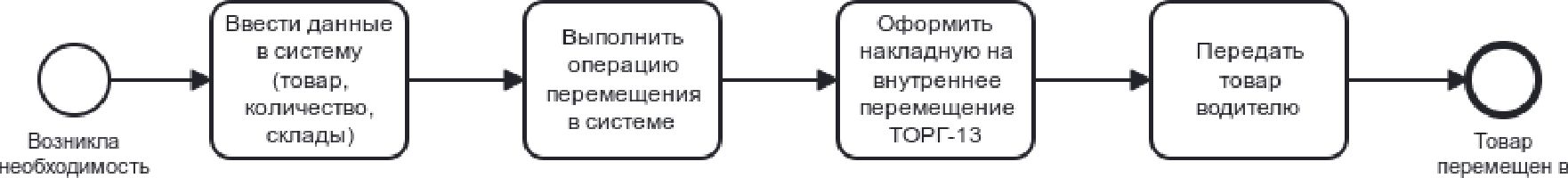
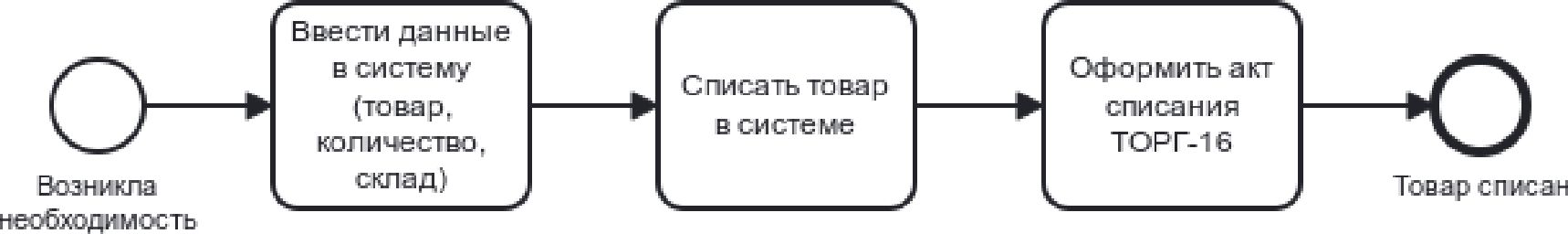


Рисунок 1.5 - BPMN диаграмма процесса «Перемещения товаров»

После того, как товар поступает на склад, он может быть перемещен в другой склад внутри организации. Кладовщик осуществляет данное перемещение, указывая необходимые детали, такие как товары, склад отправления и склад назначения в системе. После перемещения товара, оформляется накладная на внутреннее перемещение ТОРГ-13 и товар передается водителю, который осуществляет его доставку на указанный склад [10].

Далее рассмотрим процесс списания товаров, BPMN-диаграмма

которого продемонстрирована на рисунке 1.6:



списать товар

Рисунок 1.6 - BPMN диаграмма процесса «Списания»

Для выполнения операции списания товара необходимо ввести информацию о товаре, его количестве и выбранном складе. После выполнения операции списания товара, оформляется акт списания товаров под названием ТОРГ-16. Система автоматически обновляет остатки по указанному товару, списывая соответствующее количество товара с выбранного склада [10].

Бланки документов представлены в приложении А [12].

* 1. Разработка пользовательских требований

Пользовательские требования описывают задачи, которые пользователи должны выполнять с помощью системы, или взаимодействие между пользователями и системой с целью достижения полезного результата для заинтересованных лиц. Они определяют необходимую функциональность, чтобы обеспечить выполнение сценариев использования системы [2].

* + 1. Диаграмма вариантов использования

Диаграмма вариантов использования является графическим инструментом моделирования, который помогает описать функциональность системы и взаимодействие между актерами и системой. Она предоставляет обзор того, как различные пользователи будут взаимодействовать с системой и какие функциональные возможности доступны каждому актеру [14].

К основным элементам диаграммы вариантов использования относятся: 1. Актеры, представляющие различные типы пользователей или внешние системы.

1. Варианты использования, предоставляющие функциональные возможности системы.
2. Отношения между актерами и вариантами использования.

Ниже на рисунке 1.7 представлена диаграмма вариантов использования для веб-приложения учета товаров на складах.

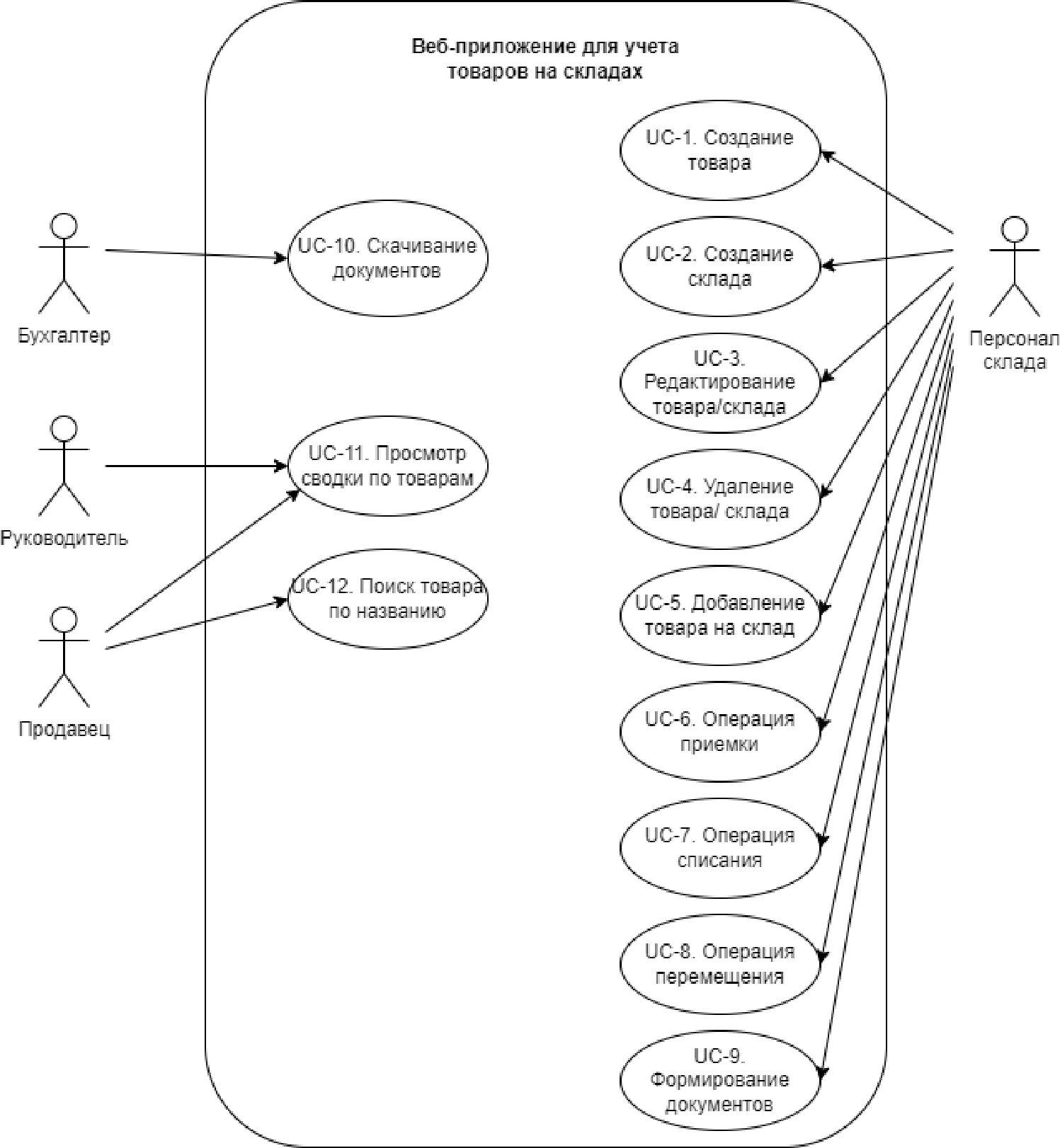


Рисунок 1.7 - Диаграмма вариантов использования

* + 1. Спецификация вариантов использования

В данной главе описаны спецификации вариантов использования. Каждая спецификация описывает конкретный сценарий или функциональность системы, опираясь на требования и потребности пользователей.

В таблице 1.4 представлена спецификация варианта использования для функции создания нового товара в системе.

Таблица 1.4 - Спецификация варианта использования «Создание товара».

|  |  |
| --- | --- |
| **UC-1. Создание товара** | |
| **Определение/ Ценность** | Операция предназначена для добавления товара в систему. |
| **Пользователь** | Персонал склада. |
| **Предусловие** | Пользователь авторизован и инициировал выполнение команды. |
| **Основной сценарий** | 1. Пользователь заполняет данными форму для создания товара и сохраняет данные. 2. Система добавляет товар в базу данных и отображает его в списке товаров. |
| **Альтернативные сценарии** | 2а. Обязательные поля не заполнены.  2а.1 Система отображает сообщение об ошибке.  2а.2 Пользователь заполняет необходимые поля.  2а.3 Работа возобновляется с шага 2. |
| **Результат** | Товар добавлен в систему. |

В таблице 1.5 представлена спецификация варианта использования для функции создания нового склада в системе.

Таблица 1.5 - Спецификация варианта использования «Создание склада».

|  |  |
| --- | --- |
| **UC-2. Создание склада** | |
| **Определение/**  **Ценность** | Операция предназначена для добавления склада в систему. |
| **Пользователь** | Персонал склада. |
| **Предусловие** | Пользователь авторизован и инициировал выполнение команды. |
| **Основной сценарий** | 1. Пользователь заполняет данными форму для создания склада и сохраняет данные. 2. Система добавляет склад в базу данных и отображает его в списке складов. |
| **Альтернативные сценарии** | 2а. Обязательные поля не заполнены.  2а.1 Система отображает сообщение об ошибке.  2а.2 Пользователь заполняет необходимые поля.  2а.3 Работа возобновляется с шага 2. |
| **Результат** | Склад добавлен в систему. |

В таблице 1.6 представлена спецификация варианта использования для функции редактирования информации о товаре/складе в системе.

Таблица 1.6 - Спецификация варианта использования «Редактирование

товара/склада».

|  |  |
| --- | --- |
| **UC-3. Редактирование товара/склада** | |
| **Определение/**  **Ценность** | Операция предназначена для редактирования товара/склада. |

|  |  |
| --- | --- |
| **Пользователь** | Персонал склада. |
| **Предусловие** | Пользователь авторизован и инициировал выполнение команды. |
| **Основной сценарий** | 1. Пользователь вносит изменения в форме редактирования товара/склада и сохраняет данные. 2. Система обновляет данные о товаре/складе в базе данных и отображает товар/склад в списке. |
| **Альтернативные сценарии** | 2а. Обязательные поля не заполнены.  2а.1 Система отображает сообщение об ошибке.  2а.2 Пользователь заполняет необходимые поля.  2а.3 Работа возобновляется с шага 3. |
| **Результат** | Товар/склад обновлен. |

В таблице 1.7 представлена спецификация варианта использования для функции удаления товара/склада из системы.

Таблица 1.7 - Спецификация варианта использования «Удаление

товара/склада».

|  |  |
| --- | --- |
| **UC-4. Удаление товара/склада** | |
| **Определение/**  **Ценность** | Операция предназначена для удаления товара/склада. |
| **Пользователь** | Персонал склада. |
| **Предусловие** | Пользователь авторизован. |
| **Основной сценарий** | 1. Пользователь нажимает на кнопку удаления   товара/склада.   1. Система удаляет товар/склад из базы данных. |
| **Альтернативные сценарии** | - |
| **Результат** | Товар/склад удален. |

В таблице 1.8 представлена спецификация варианта использования для функции добавления товара на склад.

Таблица 1.8 - Спецификация варианта использования «Добавление товара на склад».

|  |  |
| --- | --- |
| **UC-5. Добавление товара на склад** | |
| **Определение/**  **Ценность** | Операция предназначена для добавления товара на склад. |
| **Пользователь** | Персонал склада. |
| **Предусловие** | Пользователь авторизован и инициировал выполнение команды. |
| **Основной сценарий** | 1. Пользователь заполняет данными форму добавления товара на склад и сохраняет данные. 2. Система проверяет наличие товара на складе. Если товар |

|  |  |
| --- | --- |
|  | уже есть на складе, то обновляет его количество в базе данных. Если товара нет на складе, то добавляет его на склад. |

Продолжение таблицы 1.8

|  |  |
| --- | --- |
| **Альтернативные сценарии** | 2а. Данные не корректны.  2а.1 Система выводит сообщение об ошибке.  2а.2 Пользователь вводит корректные данные.  2а.3 Работа возобновляется с шага 2. |
| **Результат** | Товар добавлен на склад. |

В таблице 1.9 представлена спецификация варианта использования для функции приемки товаров. Данный процесс включает проверку товара, его количества и качество, и последующее занесение данных в систему.

Таблица 1.9 - Спецификация варианта использования «Приемка».

|  |  |
| --- | --- |
| **UC-6. Приемка** | |
| **Определение/**  **Ценность** | Операция предназначена для выполнения приемки товаров. |
| **Пользователь** | Персонал склада. |
| **Предусловие** | Пользователь авторизован и инициировал выполнение команды. |
| **Основной сценарий** | 1. Пользователь заполняет данными форму для создания приемки. 2. Система обновляет количество товаров на складах и заносит информацию о приемке в базу данных. |
| **Альтернативные сценарии** | 2а. Данные не корректны.  2а.1 Система выводит сообщение об ошибке.  2а.2 Пользователь вводит корректные данные.  2а.3 Работа возобновляется с шага 2. |
| **Результат** | Приемка выполнена. |

В таблице 1.10 представлена спецификация варианта использования для функции списания товаров из-за брака, повреждения или иным причинам.

Таблица 1.10 - Спецификация варианта использования «Списание».

|  |  |
| --- | --- |
| **UC-7. Списание** | |
| **Определение/**  **Ценность** | Операция предназначена для списания товаров. |
| **Пользователь** | Персонал склада. |
| **Предусловие** | Пользователь авторизован и инициировал выполнение команды. |
| **Основной сценарий** | 1. Пользователь заполняет форму для создания списания и сохраняет данные. 2. Система обновляет количество товаров на складах и |

заносит информацию о списании в базу данных.

|  | заносит информацию о списании в базу данных. |
| --- | --- |

Продолжение таблицы 1.10

|  |  |
| --- | --- |
| **Альтернативные сценарии** | 2а. Данные не корректны.  2а.1 Система выводит сообщение об ошибке.  2а.2 Пользователь вводит корректные данные.  2а.3 Работа возобновляется с шага 2. |
| **Результат** | Списание товаров выполнено. |

В таблице 1.11 представлена спецификация варианта использования для функции перемещения товаров между складами.

Таблица 1.11 - Спецификация варианта использования «Перемещение».

|  |  |
| --- | --- |
| **UC-8. Перемещение** | |
| **Определение/ Ценность** | Операция предназначена для перемещения товаров между складами. |
| **Пользователь** | Персонал склада. |
| **Предусловие** | Пользователь авторизован и инициировал выполнение команды. |
| **Основной сценарий** | 1. Пользователь заполняет форму для создания перемещения и сохраняет данные. 2. Система обновляет количество товаров на складах и заносит информацию о перемещении в базу данных. |
| **Альтернативные сценарии** | 2а. Данные не корректны.  2а.1 Система выводит сообщение об ошибке.  2а.2 Пользователь вводит корректные данные.  2а.3 Работа возобновляется с шага 2. |
| **Результат** | Перемещение товаров выполнено. |

В таблице 1.12 представлена спецификация варианта использования для функции формирования складских документов, необходимых для операций приемки, списания и перемещения товаров.

Таблица 1.12 - Спецификация варианта использования «Формирование документов».

|  |  |
| --- | --- |
| **UC-9. Формирование документов** | |
| **Определение/**  **Ценность** | Операция предназначена для формирования документов. |
| **Пользователь** | Персонал склада. |
| **Предусловие** | Пользователь авторизован. |
| **Основной сценарий** | 1. Пользователь нажимает на кнопку формирования документов на странице с операцией. 2. Система формирует документ на основе данных об операции и сохраняет ее в папке документов. |
| **Альтернативные сценарии** | - |

|  |  |
| --- | --- |
| **Результат** | Документы сформированы. |

В таблице 1.13 представлена спецификация варианта использования для скачивания сформированных документов.

Таблица 1.13 - Спецификация варианта использования «Скачивание

документов».

|  |  |
| --- | --- |
| **UC-10. Скачивание документов** | |
| **Определение/ Ценность** | Операция предназначена для скачивания документов. |
| **Пользователь** | Бухгалтер. |
| **Предусловие** | Пользователь авторизован. |
| **Основной сценарий** | 1. Пользователь открывает список документов, выбирает нужный документ и нажимает на кнопку скачивания документа. 2. Система скачивает файл по его пути. |
| **Альтернативные сценарии** | 2а. Файл не найден.  2а.1 Система выводит сообщение об ошибке. |
| **Результат** | Документ скачан. |

В таблице 1.14 представлена спецификация варианта использования для просмотра сводки, в которой отображены товары и их количество на складах.

Таблица 1.14 - Спецификация варианта использования «Просмотр сводки по товарам».

|  |  |
| --- | --- |
| **UC-11. Просмотр сводки по товарам** | |
| **Определение/ Ценность** | Операция предназначена для просмотра запасов товаров на складах. |
| **Пользователь** | Руководитель, Продавец. |
| **Предусловие** | Пользователь авторизован и инициировал выполнение команды. |
| **Основной сценарий** | 1. Система выдает пользователю сводку по товарам. 2. Пользователь нажимает на товар. 3. Система открывает подробную информацию о товаре. |
| **Альтернативные сценарии** | - |
| **Результат** | Сводка просмотрена. |

В таблице 1.15 представлена спецификация варианта использования для поиска товаров по их названию.

Таблица 1.15 - Спецификация варианта использования «Поиск товаров».

|  |  |
| --- | --- |
| **UC-12. Поиск товаров** | |
| **Определение/ Ценность** | Операция предназначена для поиска товаров по названию. |
| **Пользователь** | Руководитель, Продавец. |
| **Предусловие** | Пользователь авторизован и инициировал выполнение команды. |
| **Основной сценарий** | 1. Пользователь вводит название товара. 2. Система выводит товары с совпадающим названием. |
| **Альтернативные сценарии** | - |
| **Результат** | Поиск товара выполнен. |

* 1. Разработка функциональных требований

После проведения анализа пользовательских требований были выделены и разработаны функциональные требования, которые описывают основные функции и возможности, необходимые для удовлетворения потребностей пользователей. Эти функциональные требования были представлены в таблице 1.16.

Таблица 1.16 - Функциональные требования.

|  |  |
| --- | --- |
| **Идентификатор** | **Функции/Требования** |
| **F1** | Работа с товарами и складами |
| **FR-1.1** | Система должны предоставлять возможность создавать товары и склады |
| **FR-1.2** | Система должны предоставлять возможность  редактировать товары и склады |
| **FR-1.3** | Система должны предоставлять возможность удалять товары и склады |
| **FR-1.4** | Система должны предоставлять возможность  добавления товара на склад |
| **FR-1.5** | Система должны предоставлять возможность  просматривать карточки товаров и складов |
| **FR-1.6** | Система должны выводить сводку по товарам |
| **FR-1.7** | Система должны предоставлять возможность поиска товара по названию |
| **F2** | Операции |
| **FR-2.1** | Система должны предоставлять возможность проводить приемку товаров |
| **FR-2.2** | Система должны предоставлять возможность проводить списание товаров |

Продолжение таблицы 1.16

|  |  |
| --- | --- |
| **FR-2.3** | Система должны предоставлять возможность проводить перемещение товаров |
| **F3** | Документы |
| **FR-3.1** | Система должна предоставлять возможность  формировать документы для операций |
| **FR-3.2** | Система должна предоставлять возможность  пользователю скачивать документы |

* 1. Разработка нефункциональных требований

В данном разделе описаны нефункциональные требования к системе, а именно: пользовательские интерфейсы, требования к удобству пользования, производительности и безопасности [3].

В таблице 1.17 отображены требования к пользовательскому интерфейсу разрабатываемой системы.

Таблица 1.17 - Пользовательские интерфейсы.

|  |  |
| --- | --- |
| **UI-1** | Сводка по товарам должна отображаться в виде таблицы |
| **UI-2** | Незаполненные обязательные поля на форме ввода и  редактирования данных должны быть выделены красным цветом |
| **UI-3** | Поле ввода текста в формах должно поддерживать прокрутку в случае ввода большого текста |

Требования к удобству пользования представлены в таблице 1.18:

Таблица 1.18 - Требования к удобству пользования.

|  |  |
| --- | --- |
| **USE-1** | Система должна ограничивать пользователя при выборе  некорректных значений для выполнения операций. |
| **USE-2** | Интерфейс должен быть логически продуман. Например, поиск товара должен предшествовать действиям над ним, таким как редактирование или удаление. В такой же логической цепи должны выводиться действия ко всем элементам системы |

Требования к производительности описаны в таблице 1.19:

Таблица 1.19 - Требования к производительности.

|  |  |
| --- | --- |
| **PER-1** | Веб-страницы должны загружаться не более 5 секунд |

|  |  |
| --- | --- |
| **PER-2** | Операции с товарами выполняются системой не более чем за 10 секунд |
| **PER-3** | Операция по формированию документов не более 15 секунд |

В таблице 1.20 описаны требования к безопасности.

Таблица 1.20 - Требования к безопасности.

|  |  |
| --- | --- |
| **SEC-1** | Для работы с системой нужно быть авторизованным |
| **SEC-2** | Каждый пользователь в зависимости от своей роли имеет различные права доступа |

* 1. Выводы по главе

В данной главе был проведен анализ требований на разработку информационной системы для учета товаров на складах, а именно:

1. Провести анализ процесса складского учета компании.
2. Разработаны модели процессов.
3. Разработаны пользовательские требования.
4. Разработаны функциональные требования.
5. Разработаны нефункциональные требования.

В начале были разработаны бизнес-требования, включающие исходные данные, возможности, цели, риски, видение решения, критерии успеха, предположения и зависимости. Также были определены основные функции решения и описаны стейкхолдеры. С использованием анализа бизнес- требований были созданы концептуальная карта и ER-модели. Были разработаны BPMN-модели для процесса учета товаров и рассмотрены варианты использования с соответствующими спецификациями. Для системы были выделены функциональные и нефункциональные требования, а также были установлены бизнес-правила, которые должны соблюдаться. Данные аспекты являются основой для дальнейшей разработки и реализации системы, обеспечивая понимание ее функциональности и требований.

2 Проектирование информационной системы учета товаров на складах

Основной целью разработки информационной системы является автоматизация и оптимизация процессов учета товаров на складе. Система должна обеспечить точный и надежный учет товаров, отслеживать их движение и формировать необходимые документы [4].

Для ввода веб-сервиса в эксплуатации необходимо соблюдение следующих условий:

* требуемое приспособление системы к изменениям;
* сохранение работоспособности системы в различных вероятных условиях;
* требуемое время отклика системы на запрос;
* требуемую надежность;
* требуемую защиту информации от несанкционированного доступа;
* требуемую безопасность.

Для разработки информационной системы требуется определить:

* архитектуру веб-приложения;
* функциональную модель системы;
* алгоритмы основных функции системы;
* структуру базы данных приложения.
  1. Архитектура веб-приложения

Под архитектурой информационной системы понимается структура, организация и взаимосвязь ее компонентов, которые определяют общую концепцию и подход к проектированию и разработке системы [5].

Для разработки веб-приложения будет использоваться фреймворк Spring Boot на языке Java [6]. Этот фреймворк основан на Spring Framework, но предоставляет более простой и удобный подход к разработке, конфигурации и развертыванию приложений [7].

Архитектура Spring Boot не строго определена, но, как правило, включает следующие слои:

1. Слой представления: на этом слое располагаются контроллеры, которые принимают HTTP-запросы и взаимодействуют с пользовательским интерфейсом или клиентской частью приложения. Они обрабатывают запросы, вызывают соответствующие сервисы и возвращают ответы клиенту.
2. Бизнес-слой: на этом слое находятся сервисы, которые содержат бизнес-логику приложения. Они обрабатывают бизнес-запросы, выполняют операции над данными, преобразовывают их и взаимодействуют с другими компонентами приложения, такими как слой доступа к данным.
3. Слой хранения данных: этот слой отвечает за управление и доступ к данным приложения. Он включает в себя логику, отвечающую за взаимодействие с базой данных или другими источниками данных, выполнение операций CRUD и обработку запросов на доступ к данным. В Spring Boot обычно используются технологии доступа к данным, такие как JPA или Spring Data.
4. База данных: на этом уровне находится фактическая база данных или другой источник данных, с которыми приложение

взаимодействует. Здесь хранятся данные приложения и выполняются операции над ними, обеспечивая постоянное хранение и доступность данных.

Слои взаимодействуют друг с другом, обеспечивая четкое разделение обязанностей и модульность в архитектуре Spring Boot. Архитектура фреймворка Spring Boot продемонстрирована на рисунке 2.1:

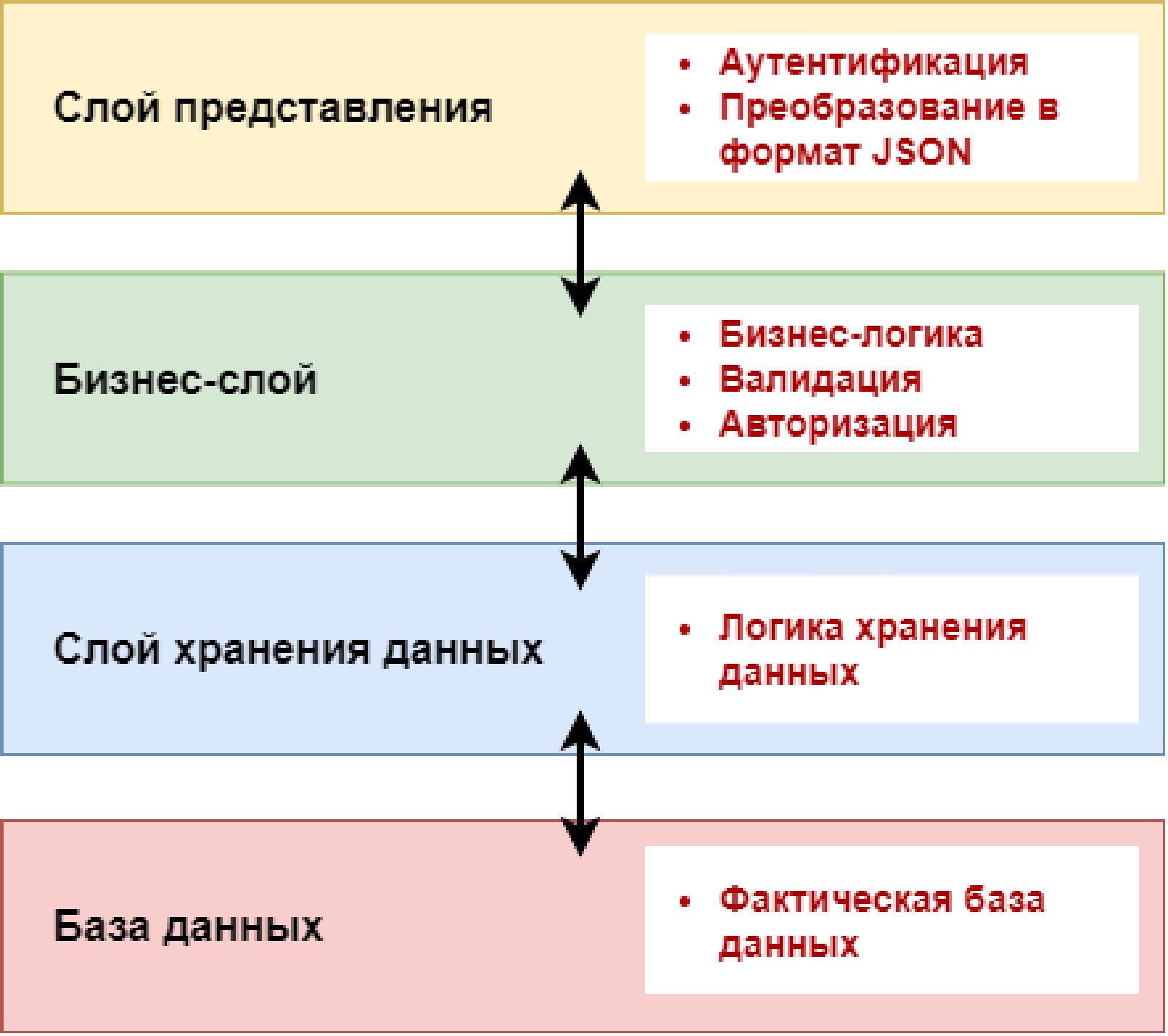


Рисунок 2.1 - Архитектура Spring Boot

* 1. Функциональная модель системы

Функциональная модель системы используется для полного и подробного описания функциональности, которую должна предоставлять система. Она помогает понять, как система будет выполнять свои задачи и как пользователи будут взаимодействовать с системой для достижения своих целей. Функциональная модель может быть представлена в виде диаграммы, графа, блок-схемы и т.д.

Учитывая требования, которые были предъявлены к информационной системе, был сформирован список функций, которые должна выполнятьсистема. На основе них были выявлены входные и выходные данные. Далее на рисунке 2.2 отображена функциональная модель веб-приложения.

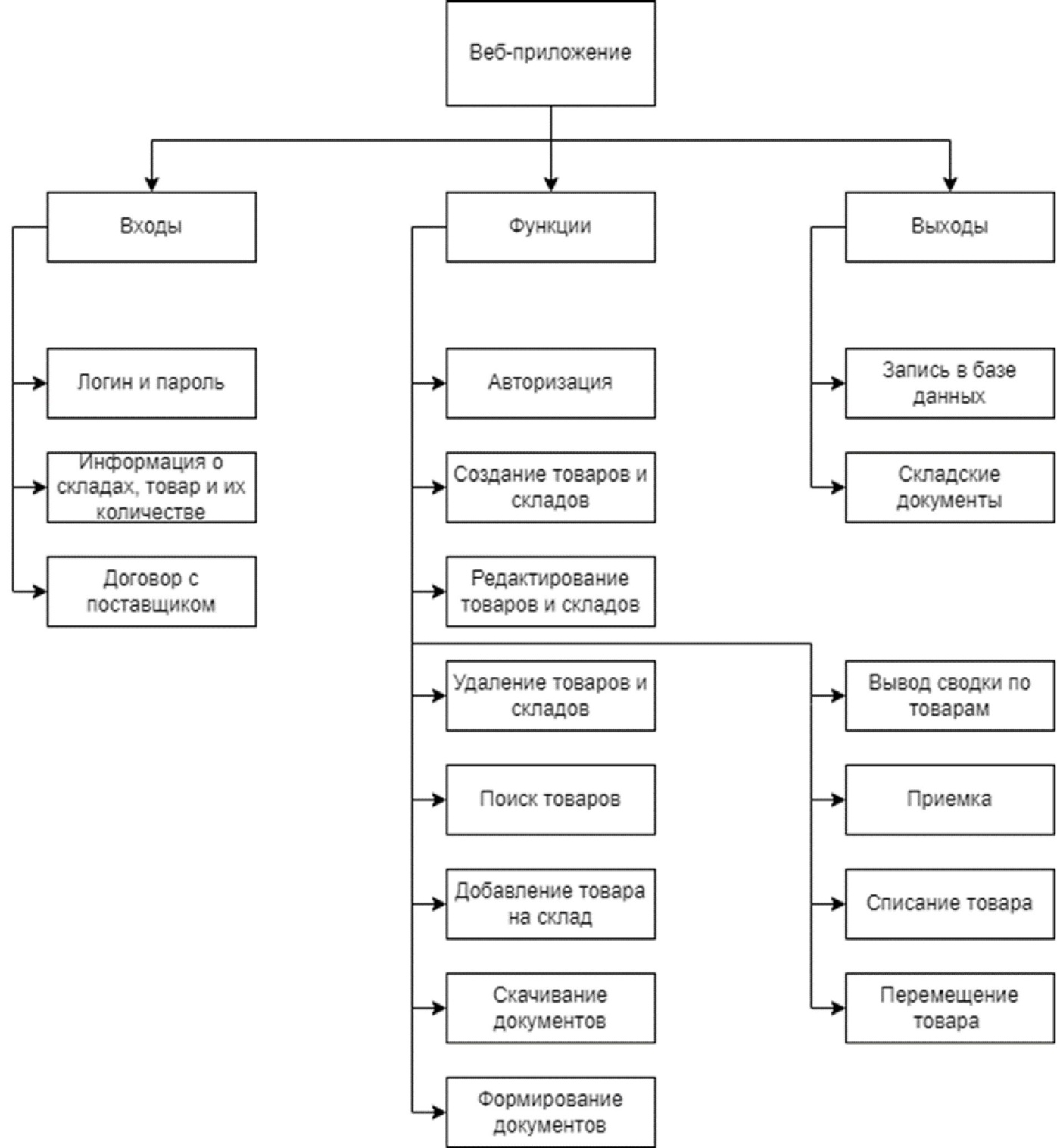


Рисунок 2.2 - Функциональная модель веб-приложения

С помощью этапа анализа были выявлены основные функции:

* авторизация в системе;
* создание товаров и складов;
* редактирование товаров и складов;
* удаление товаров и складов;
* добавление товара на склад;
* поиск товара по названию;
* формирование документов;
* скачивание документов;
* приемка;
* списание товара;
* перемещение товара;
* вывод сводки по товарам.

Для выполнения функций требуются входные данные:

* логин и пароль для авторизации;
* информация о складах, товарах и их количестве;
* договор с поставщиком.

Результатом выполнения функций являются:

* запись в базе данных в системе;
* сформированные складские документы.

Для визуализации процессов и связи между функциями были разработаны IDEF0 диаграммы [15]. На рисунке 2.3 представлена IDEF0 диаграмма процесса учета товаров на складах:

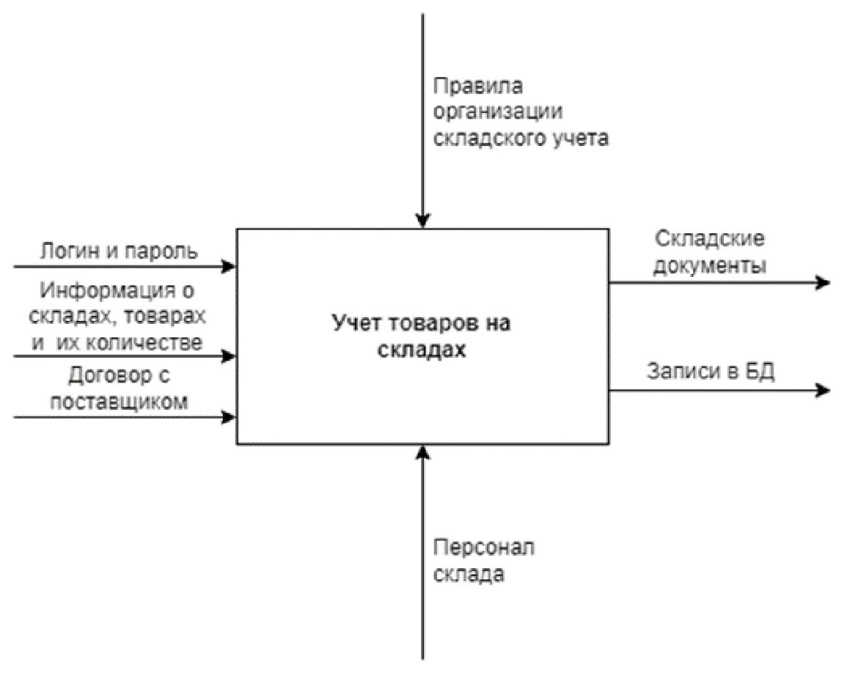


Рисунок 2.3 - Контекстная диаграмма IDEF0 процесса учета товаров на

складах

Далее на рисунках 2.4 и 2.5 продемонстрированы IDEF-диаграммы первого уровня декомпозиции и декомпозиции второго уровня для функции

приемки:

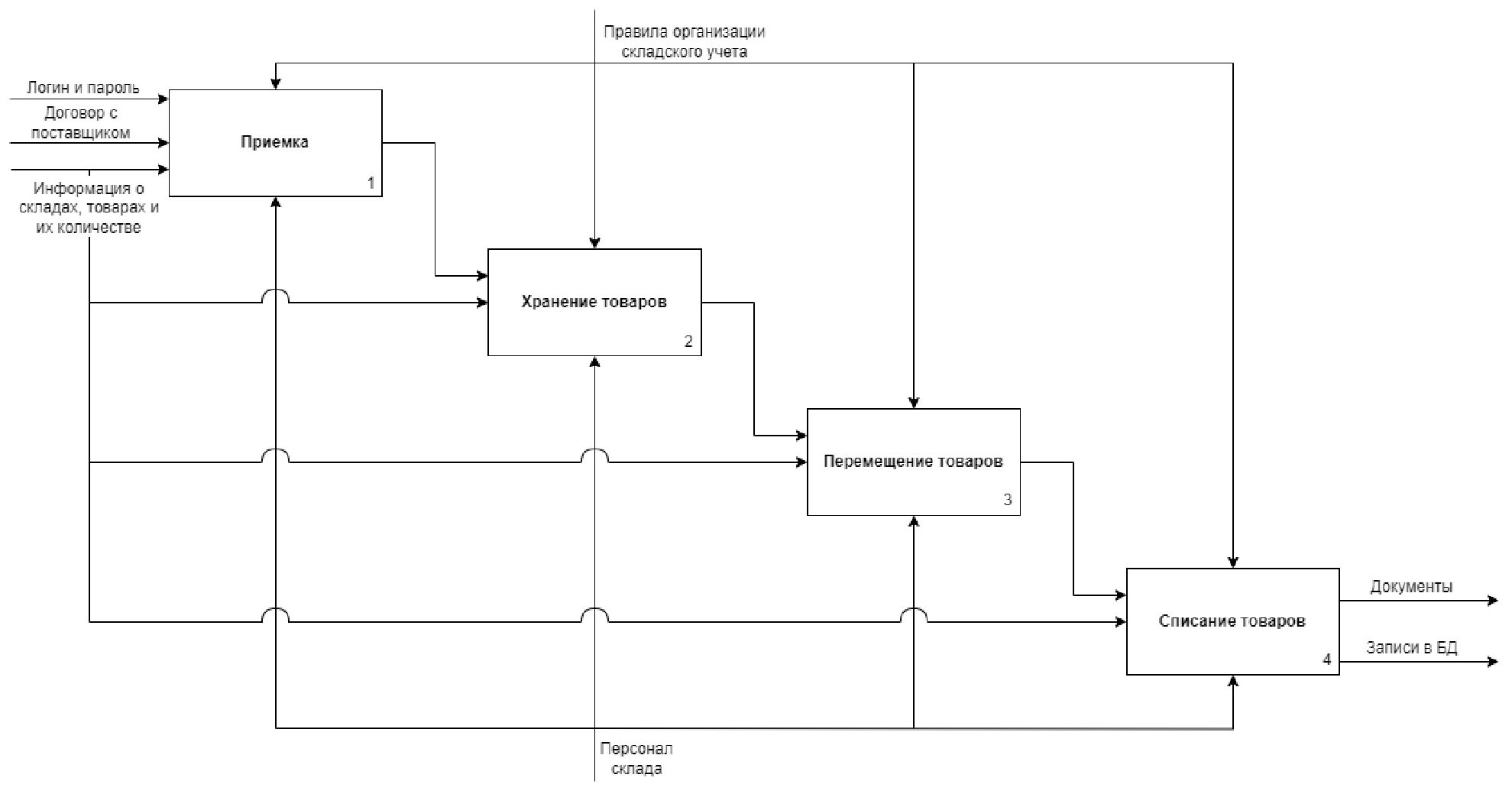


Рисунок 2.4 - Диаграмма первого уровня декомпозиции

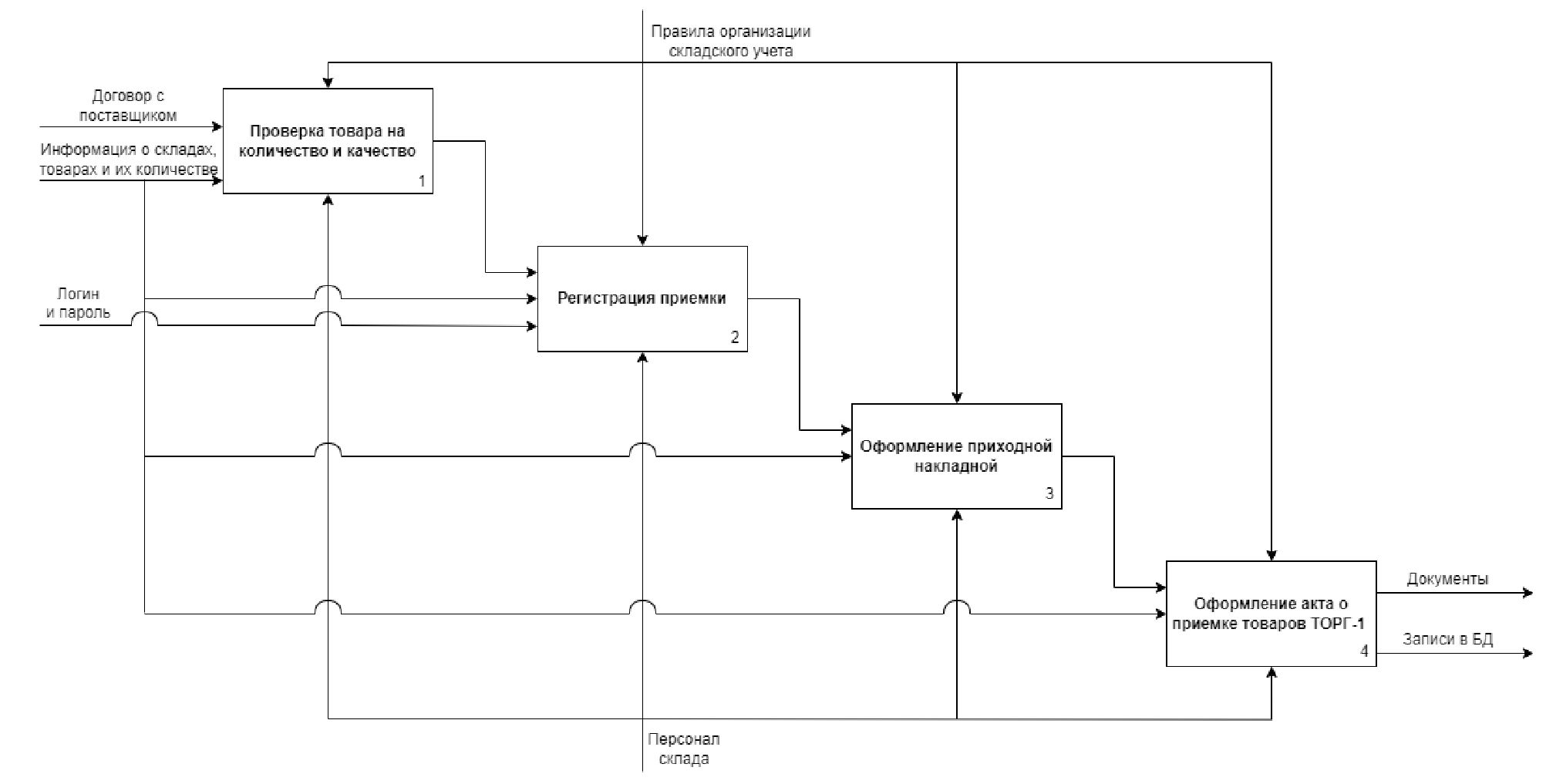


Рисунок 2.5 - Диаграмма декомпозиции для приемки

* 1. Разработка алгоритмов системы

В данной главе представлены алгоритмы основных функций, которые выполняет система. Для удобства визуализации последовательности действий были созданы UML-диаграммы деятельности. Эти диаграммы представляют собой графическое представление шагов и потоков выполнения, которые происходят при выполнении определенных функций системы [8].

Для обеспечения безопасности и контроля доступа, к функциям системы предоставлен доступ только авторизованным пользователям с ролью администратора. Это означает, что только пользователи, имеющие соответствующие права и авторизацию, могут использовать функции, связанные с административными задачами и привилегиями системы. Это важно для защиты системы от несанкционированного доступа и обеспечения конфиденциальности и целостности данных.

* + 1. Алгоритм функции приемки

Основной функцией разрабатываемой информационной системы является функция приемки товаров.

Пользователь (персонал склада) инициирует запрос для начала приемки. Система открывает страницу приемки, где от пользователя требуется ввести данные о складе, товарах из накладной от поставщика и их количество. Если все заполнено корректно, система сохраняет данные, обновляет базу данных и выводит список приемок. Далее можно создать складские документы: приходную накладную и акт приемки товаров ТОРГ-1.

Ниже на рисунке 2.6 представлена UML диаграмма деятельности функции «Приемка»:

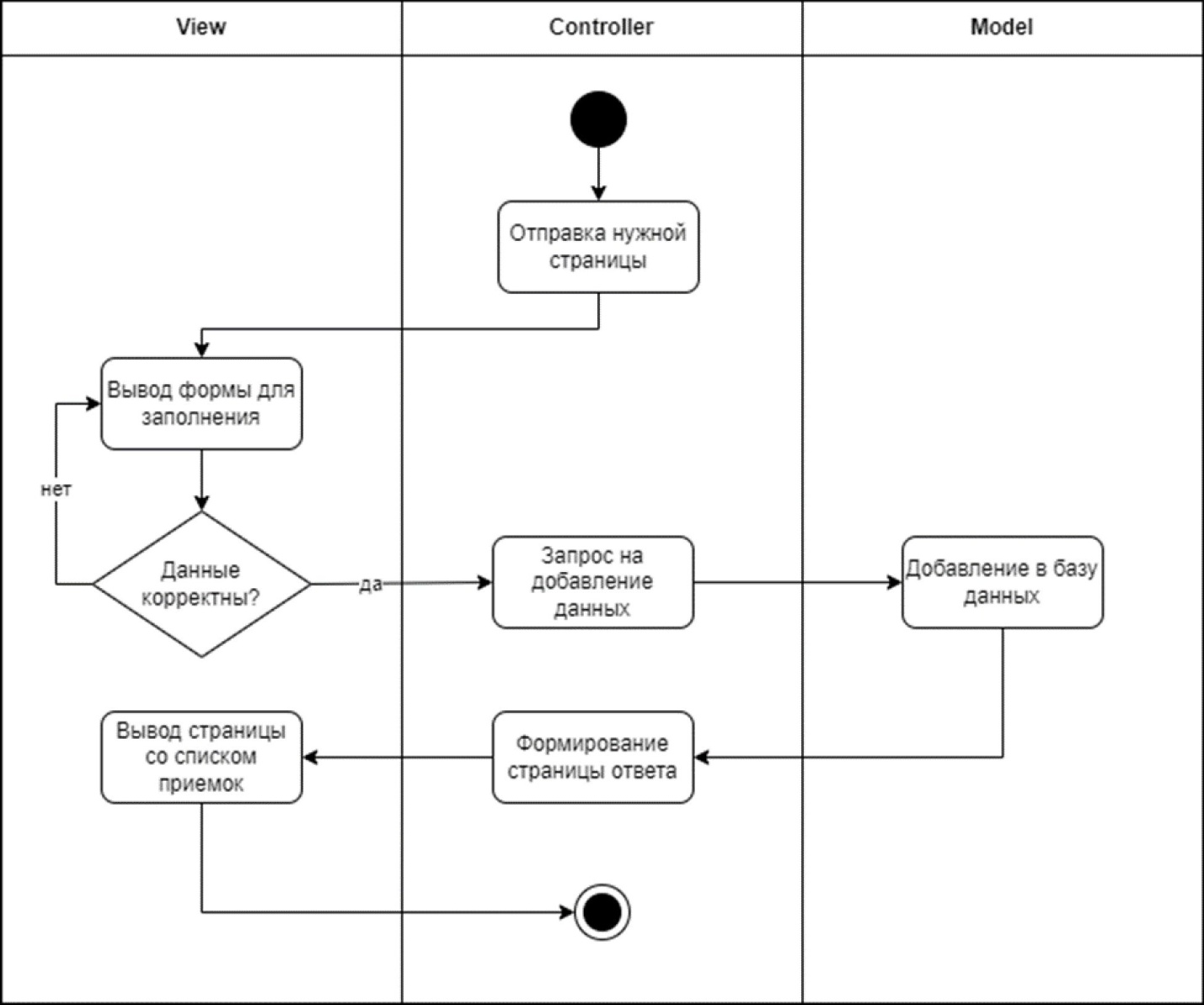


Рисунок 2.6 - UML диаграмма деятельности функции «Приемка»

* + 1. . Алгоритм функции списания

Данная функция предназначена для списания товаров. После того как пользователь отправляет запрос для списания, система выводит форму, в которой необходимо указать склад, товары и их количество, подлежащее списанию. Если все данные корректны, система сохранит данные и обновит базу данных. После выполнения данной операции становится доступно формирование акта списания ТОРГ-16.

Ниже на рисунке 2.7 представлена UML диаграмма деятельности функции «Списания»:

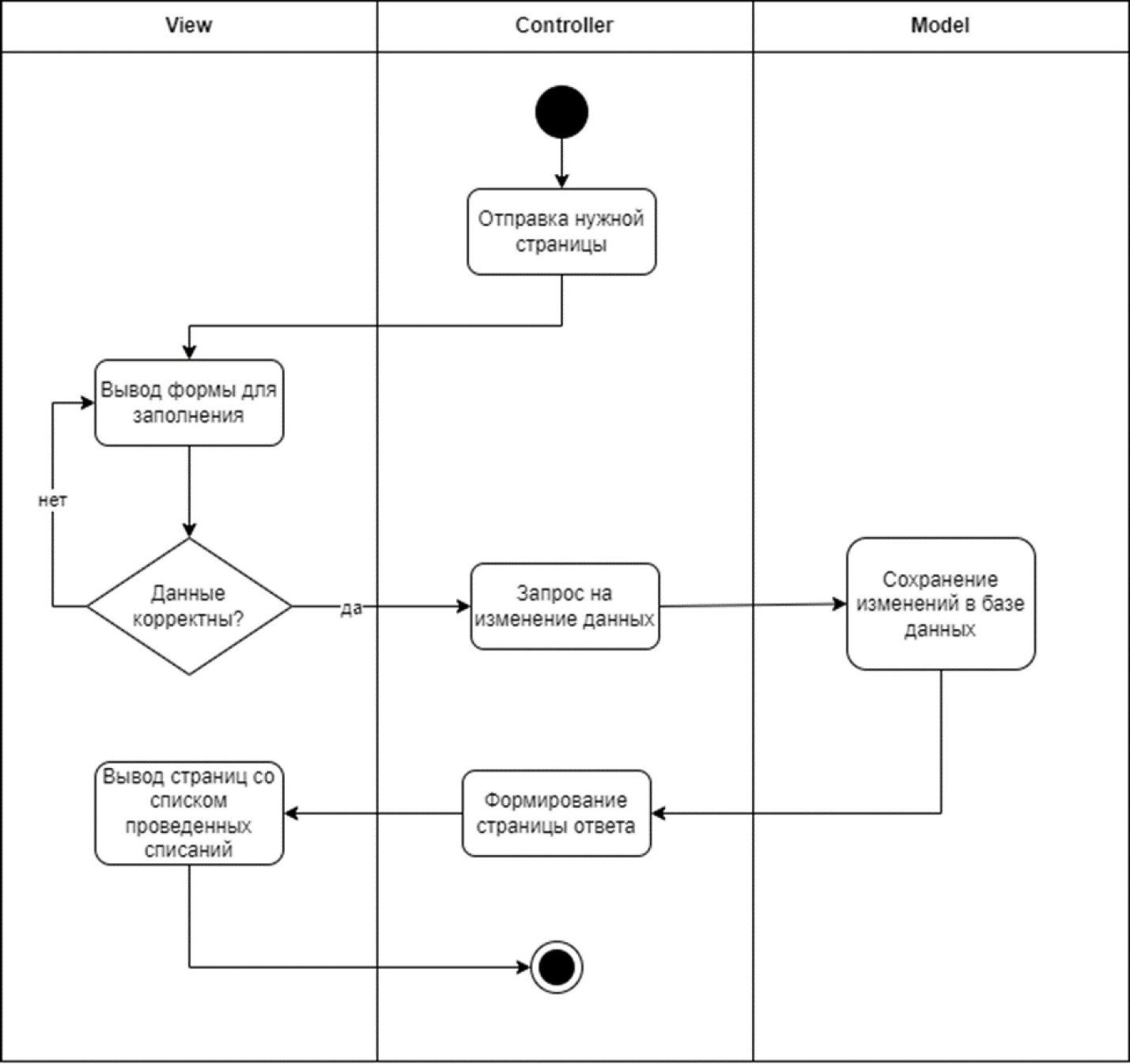


Рисунок 2.7 - UML диаграмма деятельности функции «Списания»

* + 1. Алгоритм функции перемещения

После поступления товара на склад, он может быть перемещён в другой склад организации. На запрос о перемещении, система открывает страницу с формой для заполнения. Пользователь должен ввести данные о складе отправления и назначения, а также список товаров и их количество. Если данные будут корректны, система сохранит данные и обновит базу данных. Далее при необходимости можно сформировать накладную на внутреннее перемещение ТОРГ-13.

На рисунке 2.8 представлена UML диаграмма деятельности функции

«Перемещения»:

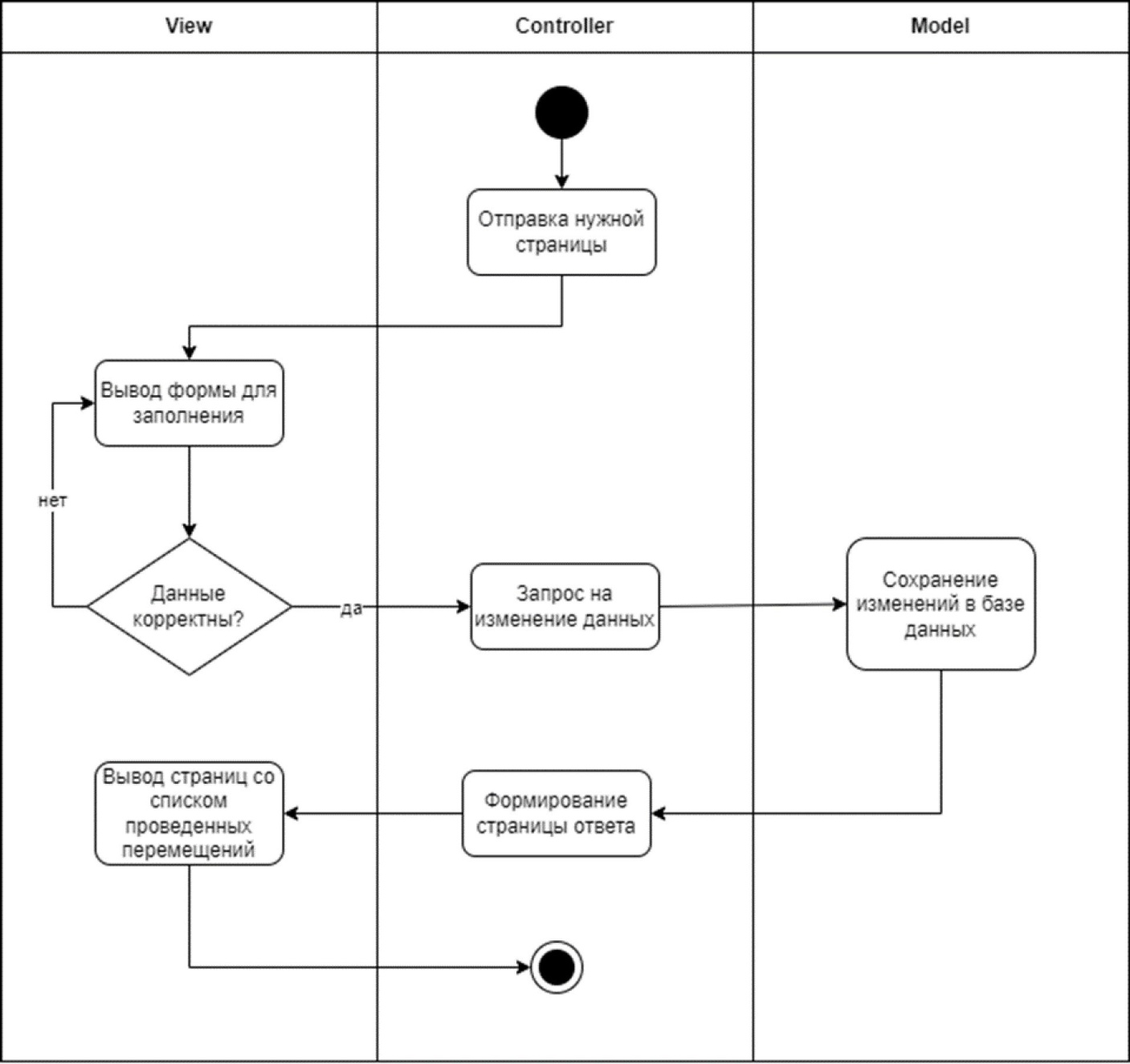


Рисунок 2.8 - UML диаграмма деятельности функции «Перемещения»

* 1. Структура базы данных веб-приложения

Для хранения данных в разрабатываемой системе будет использоваться СУБД PostgreSQL. База данных будет содержать следующие таблицы:

1. User - хранит данные о зарегистрированных пользователях системы.
2. Item - содержит информацию о товарах.
3. Image - используется для хранения изображений, связанных с товарами.
4. Stock - содержит информацию о складах, на которых хранятся товары.
5. StockItem - устанавливает связь между конкретным товаром и его наличием на определенном складе.
6. Receipt - хранит информацию о приемках.
7. ReceiptItem - содержит информацию о товарах и их количестве, которые входят в приемку.
8. Transfer - хранит информацию о перемещениях товаров между складами.
9. TransferItem - содержит информацию о товарах и их количестве, которые входят в перемещение.
10. WriteOff - хранит информацию о списаниях товаров.
11. WriteOffItem - содержит информацию о товарах и их количестве, которые входят в списание.
12. Document - содержит информацию о документах, которые были сформированы.

Каждая из таблиц базы данных содержит определённые типы объектов, которые описаны в таблицах 2.1-2.12:

Таблица 2.1 - таблица с пользователями User базы данных.

|  |  |  |
| --- | --- | --- |
| **№** | **Имя поля** | **Тип поля** |
| 1 | id | Целочисленный |
| 2 | firstname | Текстовый |
| 3 | lastname | Текстовый |
| 4 | email | Текстовый |
| 5 | role | Текстовый |
| 6 | hash password | Текстовый |

Таблица 2.2 - таблица с товарами Item базы данных.

|  |  |  |
| --- | --- | --- |
| **№** | **Имя поля** | **Тип поля** |
| 1 | id | Целочисленный |
| 2 | created at | Дата |
| 3 | name | Текстовый |

|  |  |  |
| --- | --- | --- |
| 4 | description | Текстовый |
| 5 | price | Целочисленный |
| 6 | preview image id | Целочисленный |

Таблица 2.3 - таблица с изображениями Image базы данных.

|  |  |  |
| --- | --- | --- |
| **№** | **Имя поля** | **Тип поля** |
| 1 | Id | Целочисленный |
| 2 | content type | Текстовый |
| 3 | name | Текстовый |
| 4 | original file name | Текстовый |
| 5 | size | Целочисленный |
| 6 | item id | Целочисленный |
| 7 | bytes | Байтовый |
| 8 | is preview image | Булевый |

Таблица 2.4 - таблица со складами Stock базы данных.

|  |  |  |
| --- | --- | --- |
| **№** | **Имя поля** | **Тип поля** |
| 1 | id | Целочисленный |
| 2 | name | Текстовый |
| 3 | address | Текстовый |

Таблица 2.5 - таблица с информацией о товарах на складах StockItem базы

данных.

|  |  |  |
| --- | --- | --- |
| **№** | **Имя поля** | **Тип поля** |
| 1 | id | Целочисленный |
| 2 | item id | Целочисленный |
| 3 | stock id | Целочисленный |
| 4 | quantity | Целочисленный |

Таблица 2.6 - таблица с приемками Receipt базы данных.

|  |  |  |
| --- | --- | --- |
| **№** | **Имя поля** | **Тип поля** |
| 1 | id | Целочисленный |
| 2 | created at | Дата |
| 3 | quantity | Целочисленный |
| 4 | supplier | Текстовый |
| 5 | stock id | Целочисленный |
| 6 | user id | Целочисленный |

Таблица 2.7 - таблица с товарами приемок ReceiptItem базы данных.

|  |  |  |
| --- | --- | --- |
| **№** | **Имя поля** | **Тип поля** |

|  |  |  |
| --- | --- | --- |
| 1 | id | Целочисленный |
| 2 | item id | Целочисленный |
| 3 | quantity | Целочисленный |
| 4 | receipt id | Целочисленный |

Таблица 2.8 - таблица с перемещениями Transfer базы данных.

|  |  |  |
| --- | --- | --- |
| **№** | **Имя поля** | **Тип поля** |
| 1 | id | Целочисленный |
| 2 | created at | Дата |
| 3 | quantity | Целочисленный |
| 4 | from stock id | Целочисленный |
| 5 | to stock id | Целочисленный |
| 6 | user id | Целочисленный |

Таблица 2.9 - таблица с товарами перемещений Transferitem базы данных.

|  |  |  |
| --- | --- | --- |
| **№** | **Имя поля** | **Тип поля** |
| 1 | id | Целочисленный |
| 2 | item id | Целочисленный |
| 3 | quantity | Целочисленный |
| 4 | transfer id | Целочисленный |

Таблица 2.10 - таблица со списаниями WriteOff базы данных.

|  |  |  |
| --- | --- | --- |
| **№** | **Имя поля** | **Тип поля** |
| 1 | id | Целочисленный |
| 2 | created at | Дата |
| 3 | quantity | Целочисленный |
| 4 | stock id | Целочисленный |
| 5 | user id | Целочисленный |

Таблица 2.11 - таблица с товарами списаний WriteOffItem базы данных.

|  |  |  |
| --- | --- | --- |
| **№** | **Имя поля** | **Тип поля** |
| 1 | id | Целочисленный |
| 2 | item id | Целочисленный |
| 3 | quantity | Целочисленный |
| 4 | write off id | Целочисленный |

Таблица 2.12 - таблица с информацией о документах Document базы данных.

|  |  |  |
| --- | --- | --- |
| **№** | **Имя поля** | **Тип поля** |
| 1 | id | Целочисленный |
| 2 | created at | Дата |
| 3 | name | Целочисленный |

|  |  |  |
| --- | --- | --- |
| 4 | Path | Целочисленный |
| 5 | operation id | Целочисленный |

На рисунке 2.9 продемонстрирована взаимосвязь таблиц с помощью полей:

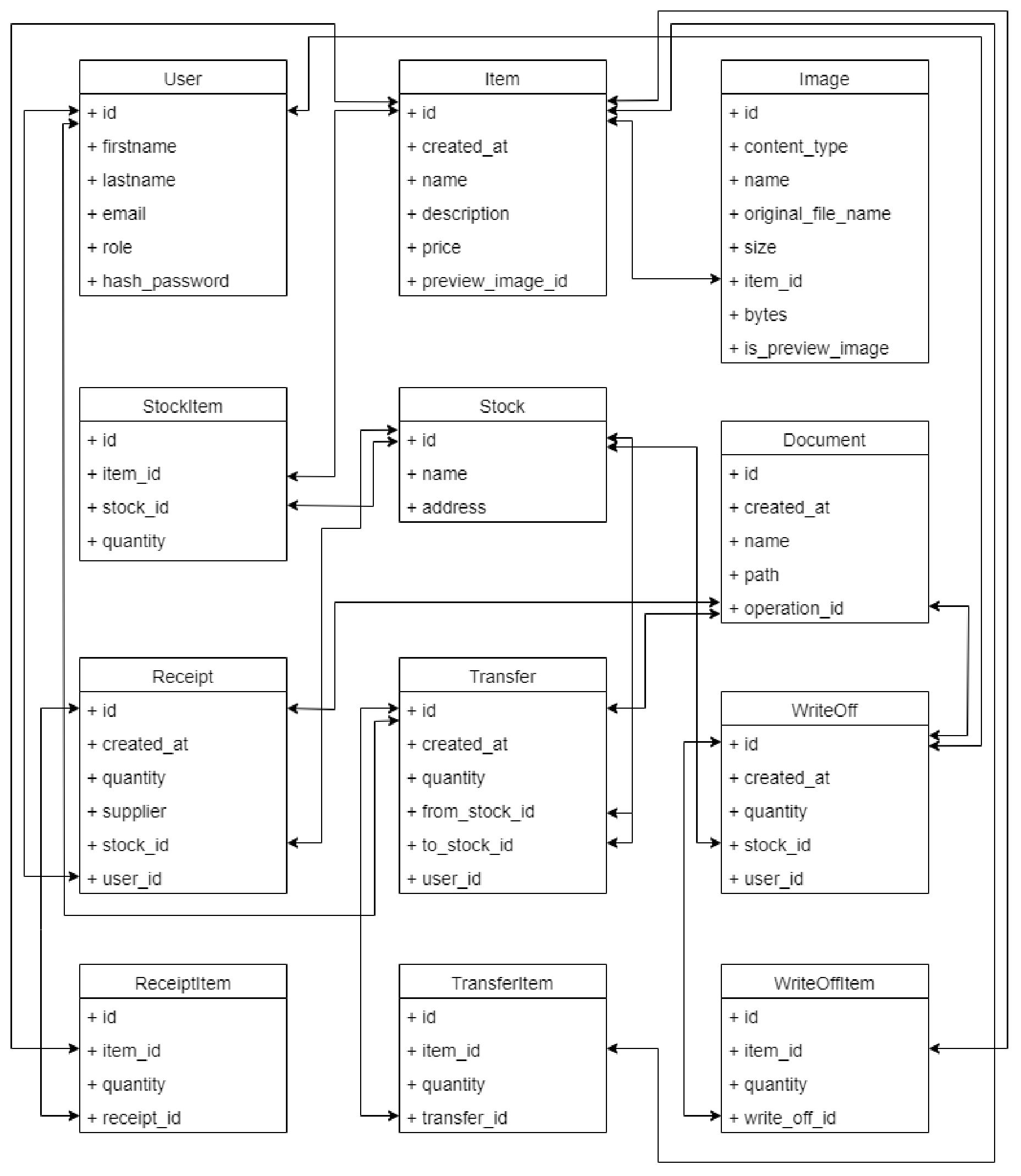


Рисунок 2.9 - Структура базы данных

* 1. Выводы по главе

В данной главе была описана архитектура веб-приложения на фреймворке Spring Boot, разработана функциональная модель системы, где мы определили входные данные, функции и выходные данные. Далее разработаны IDEF-диаграммы, куда вошли контекстная диаграмма учета товаров на складах, диаграмма первого уровня декомпозиции и диаграмма декомпозиции для функции приемки. Также разработаны алгоритмы системы для функции приемка, списания и перемещения.

Таким образом были выполнены следующие задачи:

1. Рассмотрена архитектура веб-приложения.
2. Разработана функциональная модель системы.
3. Разработаны алгоритмы системы.
4. Рассмотрена структура базы данных приложения.

3 Реализация информационной системы

* 1. Реализация веб-приложения

Как упоминалось ранее, для реализации информационной системы учета товаров на складах был выбран фреймворк Spring Boot, который обладает мощным набором инструментов и библиотек для реализации предъявленных требований к нашей информационной системе.

Рассмотрим поток работы приложения на основе Spring Boot по шагам:

1. Клиент отправляет HTTP-запрос на сервер (GET, POST, PUT и т.д.).
2. Запрос попадает в контроллер, который обрабатывает запрос, вызывает соответствующий метод сервисного слоя и передает необходимые данные.
3. Сервисный слой содержит бизнес-логику приложения. Здесь выполняются необходимые операции, такие как вычисления, преобразование данных, проверка прав доступа и другие бизнес- специфичные операции. Сервисный слой может использовать другие компоненты или сервисы для выполнения определенных задач.
4. При необходимости сервисный слой может взаимодействовать с репозиторием, который выполняет операции чтения, записи, обновления или удаления данных из базы данных. Сервисный слой может использовать результаты из репозитория для дальнейшей обработки или возвратить данные клиенту.
5. Полученные данные возвращаются обратно в контроллер,

который формирует HTTP-ответ для клиента.

1. Ответ отправляется обратно клиенту, который получает

результат и может обрабатывать его соответствующим образом.

Данный процесс представлен на рисунке 3.1:

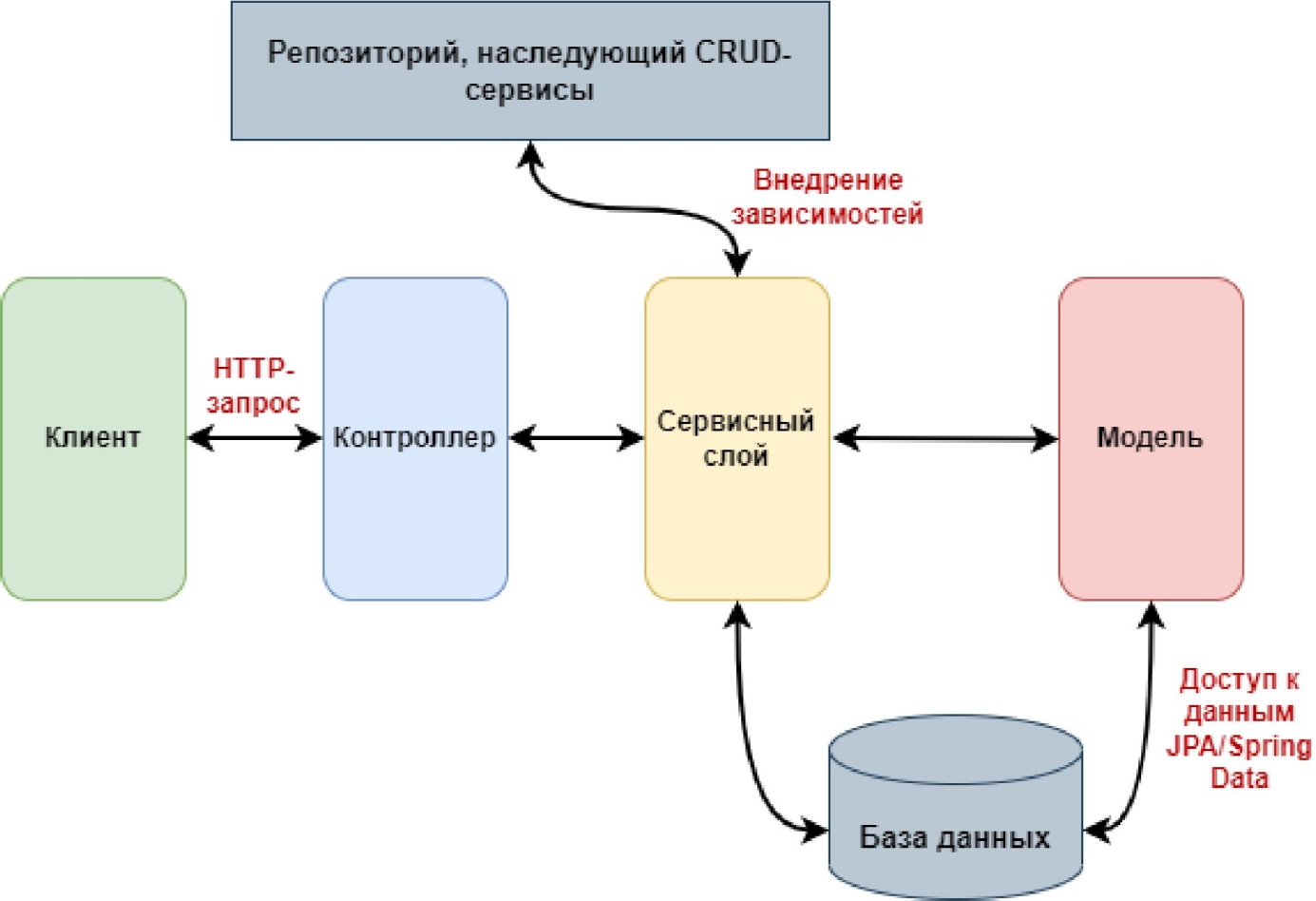


Рисунок 3.1 - Поток работы приложения на Spring Boot

В качестве СУБД был выбран PostgreSQL. Она обеспечивает эффективное хранение и доступ к данным, а также имеет широкий спектр функциональных возможностей, которые полностью закрывает наши потребности [9].

Spring Boot и PostgreSQL хорошо интегрируются друг с другом. Особое внимание стоит уделить поддержке JPA в Spring Boot, которая позволит нам легко выполнять операции создания, чтения, обновления и удаления данных в таблицах PostgreSQL, используя объектно-ориентированный подход. Для этого нужно определить методы в интерфейсе репозитория, а Spring Data JPA автоматически реализует их.

* + 1. **Описание методов сервера**
       1. **Регистрация**

Метод регистрации выполняет создание новой учетной записи пользователя в системе. Для этого пользователь должен заполнить в форму регистрации имя, фамилию, почту и пароль.

Таблица 3.1 - Описание метода «Регистрация».

|  |  |
| --- | --- |
| **Название метода** | Регистрация пользователя |
| **HTTP-запрос** | GET |
| **Описание** | Отправляет пользователю страницу с формой для заполнения |
| **Входные данные** | нет |
| **Выходные данные** | HTML страница |

Таблица 3.2 - Описание метода «Регистрация».

|  |  |
| --- | --- |
| **Название метода** | Регистрация пользователя |
| **HTTP-запрос** | POST |
| **Описание** | Выполняет создание новой учетной записи пользователя в системе. |
| **Входные данные** | Объект SignUpDto, содержащий данные  пользователя, такие как имя, фамилия, электронная почта и пароль |
| **Выходные данные** | HTML страница |

* + - 1. Вход в систему

Данный метод используется для аутентификации пользователя в системе. Пользователь должен предоставить правильные учетные данные для успешного входа в систему.

Таблица 3.3 - Описание метода «Вход в систему».

|  |  |
| --- | --- |
| **Название метода** | Вход в систему |
| **HTTP-запрос** | GET |
| **Описание** | Отправляет пользователю страницу с формой для заполнения |
| **Входные данные** | нет |
| **Выходные данные** | HTML страница |

Таблица 3.4 - Описание метода «Вход в систему».

|  |  |
| --- | --- |
| **Название метода** | Вход в систему |

|  |  |
| --- | --- |
| **HTTP-запрос** | POST |
| **Описание** | Аутентификация и авторизация пользователя в системе |
| **Входные данные** | Логин и пароль |
| **Выходные данные** | Авторизованный пользователь |

* + - 1. Работа с товарами

Здесь описаны методы для работы с товарами.

Таблица 3.5 - Описание метода «Список товаров».

|  |  |
| --- | --- |
| **Название метода** | Список товаров |
| **HTTP-запрос** | GET |
| **Описание** | Выводит список товаров |
| **Входные данные** | Ключ |
| **Выходные данные** | Если ключ непустой, то выводятся товары, название которых совпадает с ключом. Если ключ пустой, но выводятся все товары |

Таблица 3.6 - Описание метода «Получение информации о товаре».

|  |  |
| --- | --- |
| **Название метода** | Получение информации о товаре |
| **HTTP-запрос** | GET |
| **Описание** | Выводит информацию о конкретном товаре по его идентификатору |
| **Входные данные** | ID товара |
| **Выходные данные** | Данные о товаре |

Таблица 3.7 - Описание метода «Создание товара».

|  |  |
| --- | --- |
| **Название метода** | Создание товара |
| **HTTP-запрос** | GET |
| **Описание** | Отправляет пользователю страницу с формой для заполнения |
| **Входные данные** | нет |
| **Выходные данные** | HTML страница |

Таблица 3.8 - Описание метода «Создание товара».

|  |  |
| --- | --- |
| **Название метода** | Создание товара |
| **HTTP-запрос** | POST |
| **Описание** | Добавляет новый товар в систему |
| **Входные данные** | Данные о товаре, фотографии |
| **Выходные данные** | Добавленный в базу данных товар |

Таблица 3.9 - Описание метода «Редактирование товара».

|  |  |
| --- | --- |
| **Название метода** | Редактирование товара |
| **HTTP-запрос** | GET |
| **Описание** | Отправляет пользователю страницу с данными товара |
| **Входные данные** | ID товара |
| **Выходные данные** | HTML страница |

Таблица 3.10 - Описание метода «Редактирование товара».

|  |  |
| --- | --- |
| **Название метода** | Редактирование товара |
| **HTTP-запрос** | POST |
| **Описание** | Обновляет информацию о товаре |
| **Входные данные** | Данные о товаре |
| **Выходные данные** | Отредактированный товар |

Таблица 3.11 - Описание метода «Удаление товара».

|  |  |
| --- | --- |
| **Название метода** | Удаление товара |
| **HTTP-запрос** | DELETE |
| **Описание** | Удаляет товар из системы |
| **Входные данные** | ID товара |
| **Выходные данные** | Обновленная страница со списком товаров |

* + - 1. Работа со складами

Здесь описаны методы для работы со складами.

Таблица 3.12 - Описание метода «Список складов».

|  |  |
| --- | --- |
| **Название метода** | Список складов |

Продолжение таблицы 3.12

|  |  |
| --- | --- |
| **HTTP-запрос** | GET |
| **Описание** | Выводит список складов |
| **Входные данные** | нет |
| **Выходные данные** | Список со всеми складами |

Таблица 3.13 - Описание метода «Получение информации о складе».

|  |  |
| --- | --- |
| **Название метода** | Получение информации о складе |
| **HTTP-запрос** | GET |
| **Описание** | Выводит информацию о конкретном складе по его идентификатору |
| **Входные данные** | ID склада |
| **Выходные данные** | Данные о складе |

Таблица 3.14 - Описание метода «Создание склада».

|  |  |
| --- | --- |
| **Название метода** | Создание склада |
| **HTTP-запрос** | GET |
| **Описание** | Отправляет пользователю страницу с формой для заполнения |
| **Входные данные** | нет |
| **Выходные данные** | HTML страница |

Таблица 3.15 - Описание метода «Создание склада».

|  |  |
| --- | --- |
| **Название метода** | Создание склада |
| **HTTP-запрос** | POST |
| **Описание** | Добавляет новый склад в систему |
| **Входные данные** | Данные о складе |
| **Выходные данные** | Добавленный в базу данных склад |

Таблица 3.16 - Описание метода «Редактирование склада».

|  |  |
| --- | --- |
| **Название метода** | Редактирование склада |
| **HTTP-запрос** | GET |
| **Описание** | Отправляет пользователю страницу с данными склада |
| **Входные данные** | ID склада |
| **Выходные данные** | HTML страница |

Таблица 3.17 - Описание метода «Редактирование склада».

|  |  |
| --- | --- |
| **Название метода** | Редактирование склада |
| **HTTP-запрос** | POST |
| **Описание** | Обновляет информацию о складе |
| **Входные данные** | Данные о складе |
| **Выходные данные** | Отредактированный склад |

Таблица 3.18 - Описание метода «Удаление склада».

|  |  |
| --- | --- |
| **Название метода** | Удаление склада |
| **HTTP-запрос** | DELETE |
| **Описание** | Удаляет склад из системы |
| **Входные данные** | ID склада |
| **Выходные данные** | Обновленная страница со списком складов |

* + - 1. Добавление товара на склад

Таблица 3.19 - Описание метода «Добавление товара на склад»

|  |  |
| --- | --- |
| **Название метода** | Добавление товара на склад |
| **HTTP-запрос** | GET |
| **Описание** | Отправляет пользователю страницу с формой для заполнения |
| **Входные данные** | нет |
| **Выходные данные** | HTML страница |

Таблица 3.20 - Описание метода «Добавление товара на склад».

|  |  |
| --- | --- |
| **Название метода** | Добавление товара на склад |
| **HTTP-запрос** | POST |
| **Описание** | Добавляет товар на склад |
| **Входные данные** | ID склада, ID товара и его количество |
| **Выходные данные** | Обновленная страница со списком товаров на складах |

3.1.1.6 Приемка

Таблица 3.21 - Описание метода «Список приемок».

|  |  |
| --- | --- |
| **Название метода** | Список приемок |
| **HTTP-запрос** | GET |
| **Описание** | Выводит список приемок |

|  |  |
| --- | --- |
| **Входные данные** | Нет |
| **Выходные данные** | Список со всеми приемками |

Таблица 3.22 - Описание метода «Получение информации о приемке».

|  |  |
| --- | --- |
| **Название метода** | Получение информации о приемке |
| **HTTP-запрос** | GET |
| **Описание** | Выводит информацию о конкретной приемке по его идентификатору |
| **Входные данные** | ID приемки |
| **Выходные данные** | Данные о приемке |

Таблица 3.23 - Описание метода «Создание приемки».

|  |  |
| --- | --- |
| **Название метода** | Создание приемки |
| **HTTP-запрос** | GET |
| **Описание** | Отправляет пользователю страницу с формой для заполнения |
| **Входные данные** | нет |
| **Выходные данные** | HTML страница |

Таблица 3.24 - Описание метода «Создание приемки».

|  |  |
| --- | --- |
| **Название метода** | Создание склада |
| **HTTP-запрос** | POST |
| **Описание** | Добавляет новую приемку в систему |
| **Входные данные** | Данные о приемке |
| **Выходные данные** | Добавленная в базу данных приемка |

3.1.1.7 Перемещение

Таблица 3.25 - Описание метода «Список перемещений».

|  |  |
| --- | --- |
| **Название метода** | Список перемещений |
| **HTTP-запрос** | GET |
| **Описание** | Выводит список перемещений |
| **Входные данные** | нет |
| **Выходные данные** | Список со всеми перемещениями |

Таблица 3.26 - Описание метода «Получение информации о перемещении».

|  |  |
| --- | --- |
| **Название метода** | Получение информации о перемещении |
| **HTTP-запрос** | GET |
| **Описание** | Выводит информацию о конкретном  перемещении по его идентификатору |
| **Входные данные** | ID перемещения |
| **Выходные данные** | Данные о перемещении |

Таблица 3.27 - Описание метода «Создание перемещения».

|  |  |
| --- | --- |
| **Название метода** | Создание перемещения |
| **HTTP-запрос** | GET |
| **Описание** | Отправляет пользователю страницу с формой для заполнения |
| **Входные данные** | нет |
| **Выходные данные** | HTML страница |

Таблица 3.28 - Описание метода «Создание перемещения».

|  |  |
| --- | --- |
| **Название метода** | Создание склада |
| **HTTP-запрос** | POST |
| **Описание** | Добавляет новое перемещение в систему |
| **Входные данные** | Данные о перемещении |
| **Выходные данные** | Добавленное в базу данных перемещение |

3.1.1.8 Списание

Таблица 3.29 - Описание метода «Список списаний».

|  |  |
| --- | --- |
| **Название метода** | Список списаний |
| **HTTP-запрос** | GET |
| **Описание** | Выводит список списаний |
| **Входные данные** | нет |
| **Выходные данные** | Список со всеми списаниями |

Таблица 3.30 - Описание метода «Получение информации о списании».

|  |  |
| --- | --- |
| **Название метода** | Получение информации о списании |
| **HTTP-запрос** | GET |
| **Описание** | Выводит информацию о конкретном списании по его идентификатору |
| **Входные данные** | ID списания |
| **Выходные данные** | Данные о списании |

Таблица 3.31 - Описание метода «Создание списания».

|  |  |
| --- | --- |
| **Название метода** | Создание списания |
| **HTTP-запрос** | GET |
| **Описание** | Отправляет пользователю страницу с формой для заполнения |
| **Входные данные** | нет |
| **Выходные данные** | HTML страница |

Таблица 3.32 - Описание метода «Создание списания».

|  |  |
| --- | --- |
| **Название метода** | Создание списания |
| **HTTP-запрос** | POST |
| **Описание** | Добавляет новое списание в систему |
| **Входные данные** | Данные о списании |
| **Выходные данные** | Добавленное в базу данных списание |

3.1.1.9 Работа с документами

Таблица 3.33 - Описание метода «Генерация документов».

|  |  |
| --- | --- |
| **Название метода** | Г енерация документов |
| **HTTP-запрос** | GET |
| **Описание** | Генерирует и сохраняет документ по данным операции |
| **Входные данные** | Объект операции |
| **Выходные данные** | HTML страница |

Таблица 3.34 - Описание метода «Скачивание документов».

|  |  |
| --- | --- |
| **Название метода** | Скачивание документов |
| **HTTP-запрос** | GET |
| **Описание** | Скачивает документ из системы |
| **Входные данные** | Название документа |
| **Выходные данные** | Файл документа |

* 1. Формирование документов

Для операций приемки, списания и перемещения на складе требуется формирование соответствующих складских документов. В нашей системе пользователи должны иметь возможность скачивать эти документы на основе введенных данных.

Для реализации этой функциональности было принято решение генерировать документы по шаблону при необходимости, а не хранить их отдельно в базе данных. Это позволяет избежать загрузки базы данных большим объемом документов, которые могут не потребоваться пользователям в данный момент. Таким образом, в базе данных будет храниться только название, дата создания и путь к файлу.

Для работы с генерацией и обработкой документов будет использоваться Apache POI. Apache POI является популярной и широко используемой библиотекой для работы с документами Microsoft Office, которая позволяет программно создавать и редактировать документы, добавлять текст, таблицы, форматирование и многое другое. Данная библиотека является надежным и мощным инструментом, который поможет нам реализовать необходимую функциональность по работе с документами в нашем проекте.

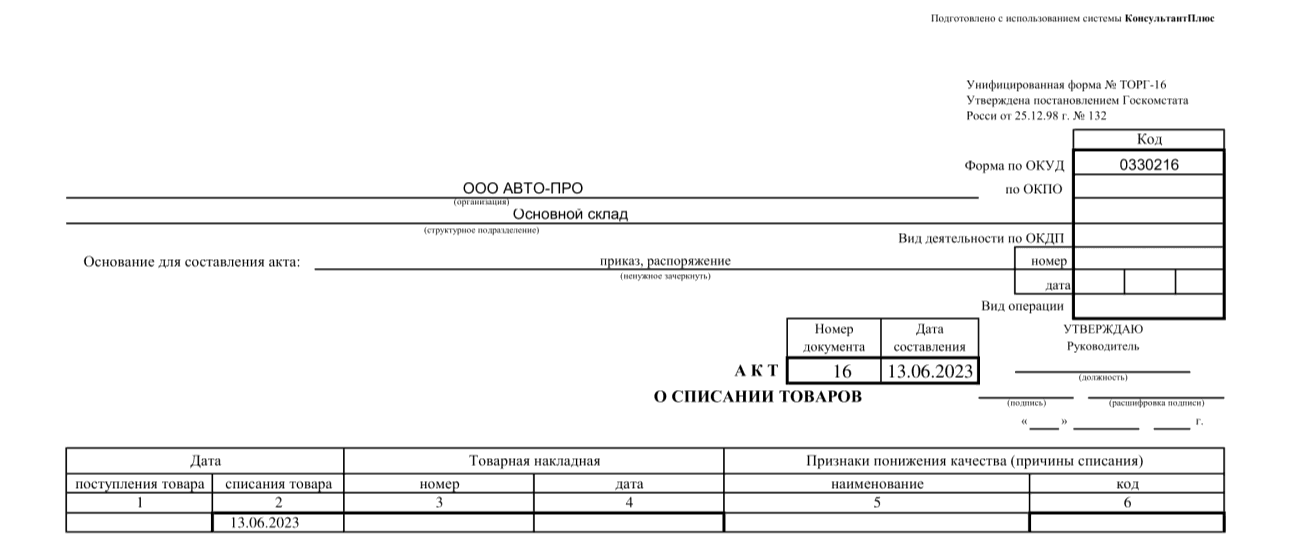
* + 1. Заполнение шаблона документов

Для того чтобы не создавать документ с нуля, используется предварительно созданный шаблон с заданным форматированием и структурой. Задача состоит в то, чтобы найти нужные ячейки и сохранить в них полученные данные. Для этого используются классы библиотеки, которые позволяют сначала получить доступ к листам документа, а затем к строкам и столбцам.

Проблема возникает с изменяющимися данными, например, со списком товаров. Для таких случаев используется Apache POI в сочетании с циклом, чтобы динамически заполнять ячейки в шаблоне для каждого товара.

* + 1. Сохранение документов

После добавления всех данных, генерируется файл с уникальным именем, куда записывается содержимое документа. Далее документ сохраняется в папке с документами. На рисунке 3.2 представлен пример сформированного документа:



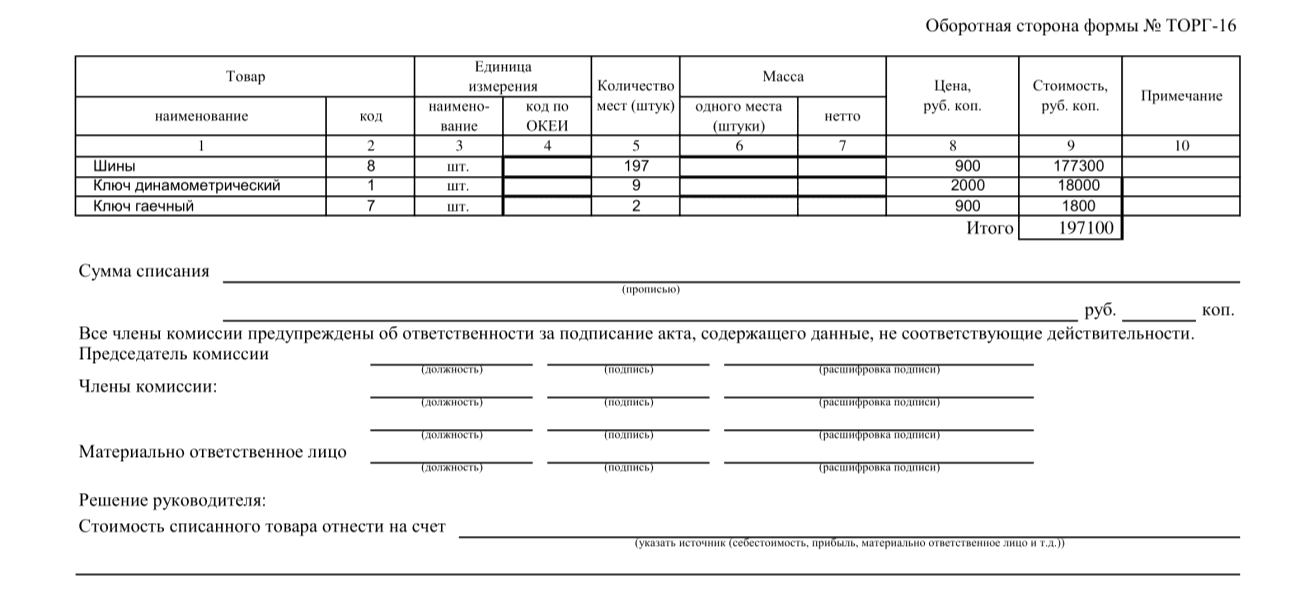


Рисунок 3.2 - Заполненный шаблон документа «ТОРГ-16» Описание интерфейса веб-приложения

В базе данных сохраняется история о том, какие документы были уже сформированы. Каждый документ доступен для скачивания на устройство пользователя.

Интерфейс веб-приложения должен быть интуитивно понятным и удобным для пользователей. Он должен обеспечивать легкую навигацию и эффективное взаимодействие с функциональностью приложения.

* + 1. Страница «Главная»

Главная страница веб-приложения является основной точкой входа для пользователей. Она предоставляет пользователю основную информацию о функциональности приложения. Пользовательский интерфейс страницы «Главная» представлен на рисунке 3.3:

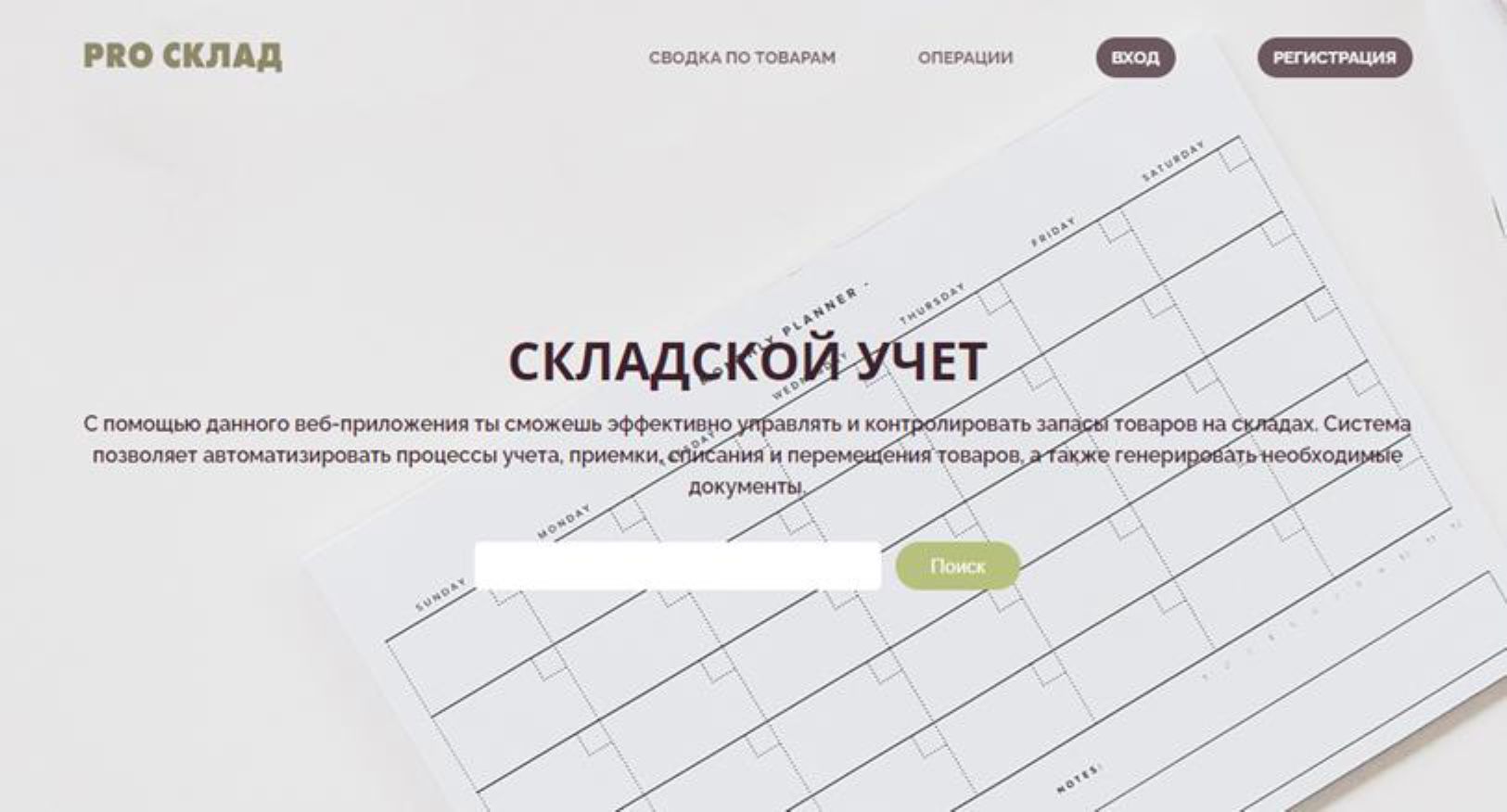


Рисунок 3.3 - Страница «Главная»

На данной странице пользователь может найти шапку сайта с кнопками для входа и регистрации, просмотром сводки товаров и операциями. В середине страницы располагается поисковая строка, которая позволяет пользователям искать товары.

* + 1. **Страница «Регистрация»**

Данная страница предназначена для регистрации в системе. От пользователя требуется ввести имя, фамилию, почту и пароль. Если веденные данные корректны, система зарегистрирует пользователя и по умолчанию присвоит ему роль "User", по которому доступен режим чтения. Далее при необходимости администратор может назначить пользователю роль "Admin", которая предоставляет полный доступ к операциям системы.

На рисунке 3.4 представлена страница регистрации пользователей.

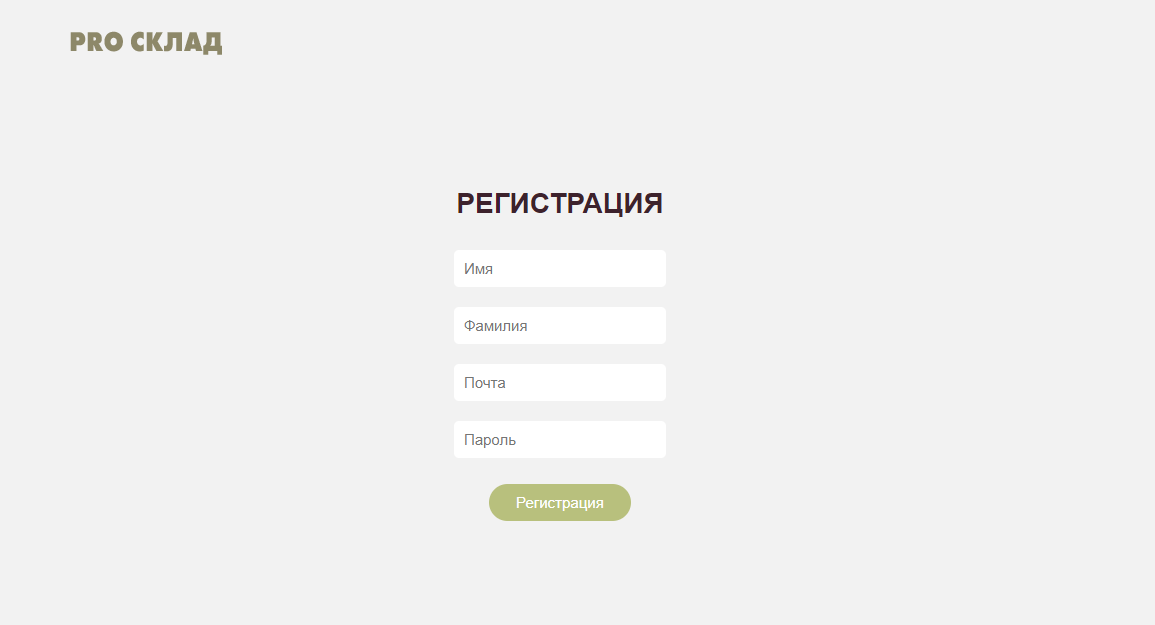


Рисунок 3.4 - Страница «Регистрация»

* + 1. Страница «Вход в систему»

Для входа в систему пользователь должен указать свою почту и пароль. Введенный пароль сравнивается с хэшем пароля из базы данных. Если данные совпадают, то пользователь получает доступ к системе. Кроме того, пользователь может отметить опцию "Запомнить меня" для автоматического входа в следующий раз.

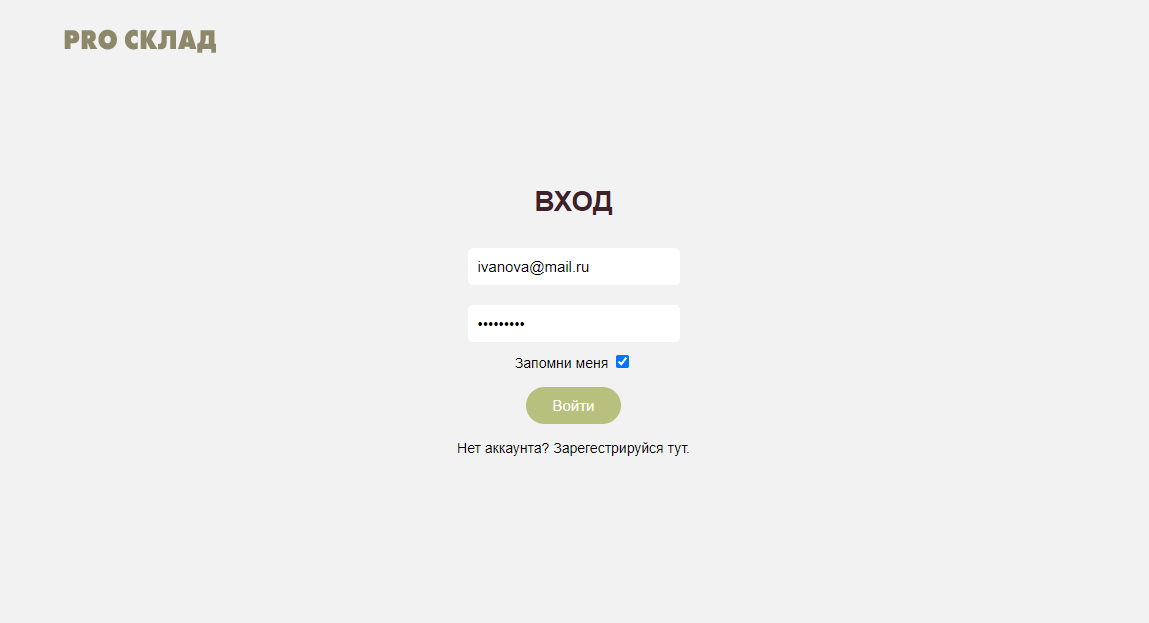


Рисунок 3.5 - Страница «Вход в систему»

* + 1. Страница «Товары»

Страница «Товары» отображает список всех товаров, которые находятся в базе данных. В таблице приведены названия товаров и их цены. Ниже таблицы есть кнопка для добавления нового товара.

Также на странице присутствует поисковая строка, которая позволяет пользователю быстро находить нужные товары. Пользователь может ввести часть или полное название товара, и система найдет соответствующие результаты и отобразит их на странице.

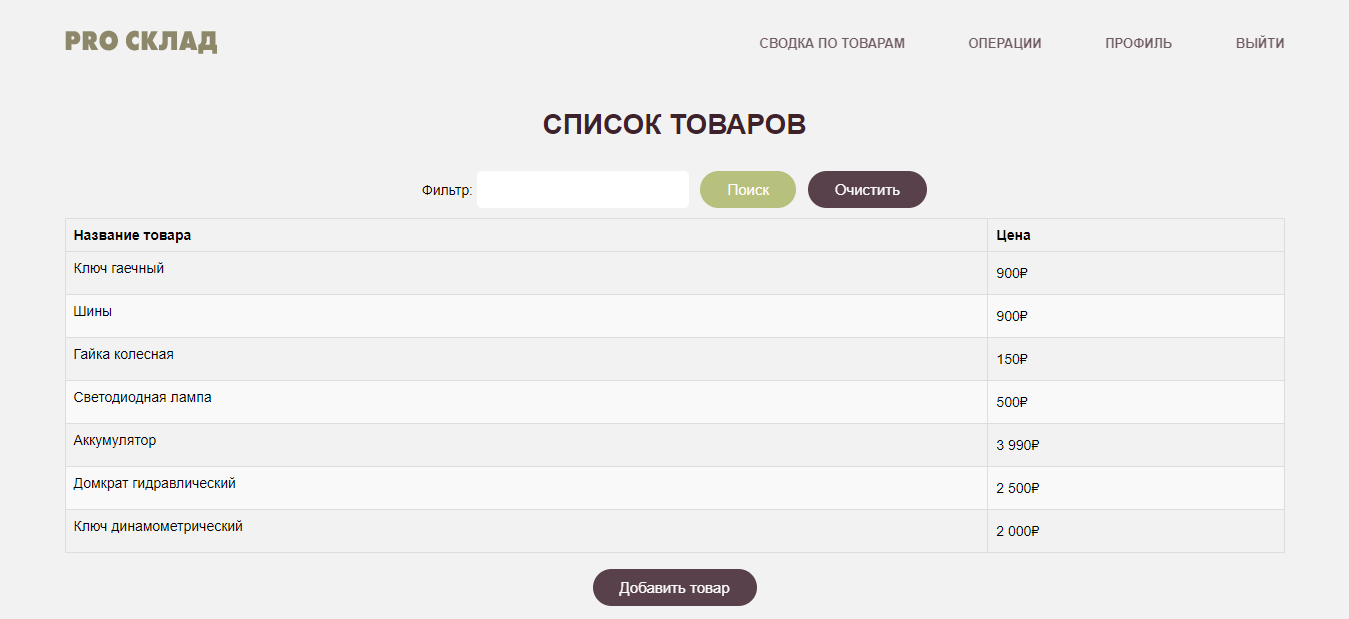


Рисунок 3.6 - Страница «Товары»

Для того чтобы получить более детальную информацию о товаре, можно кликнуть на его название, и откроется отдельная страница «Товар» со всеми сведениями о товаре, включая номер, название, цену, фотографии и описание товара.

На странице с товаром предусмотрены кнопки редактирования и удаления товара из системы. При нажатии на редактирование откроется новая страница с формой для редактирования. Операция удаления товара выполняется без создания отдельной страницы. Пользовательский интерфейс страницы «Товар» представлен на рисунке 3.7:



Рисунок 3.7 - Страница «Товар»

Страница «Редактирования» представлена на рисунке 3.8:

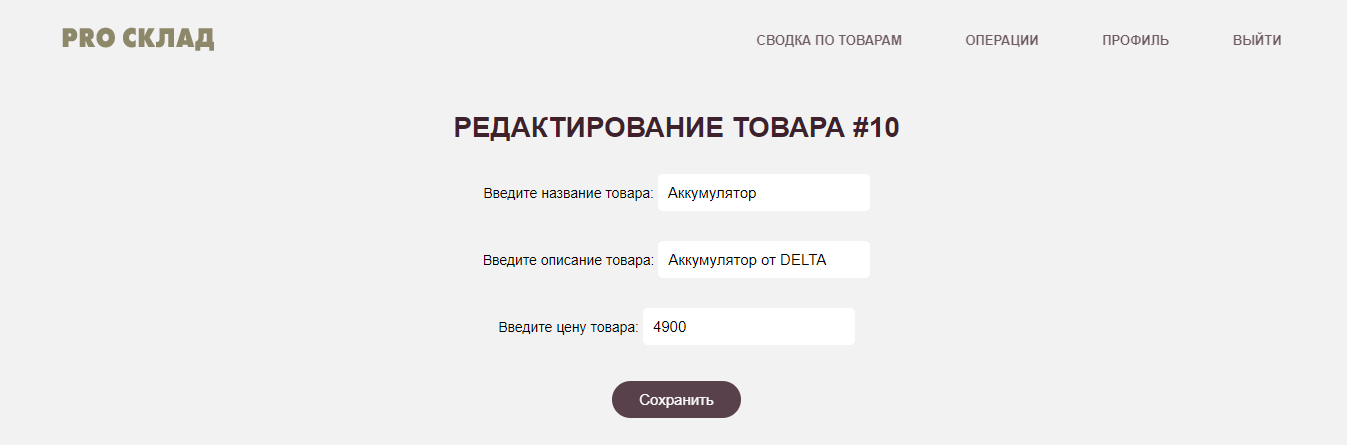


Рисунок 3.8 - Страница «Редактирование товара»

Также на странице «Товары» есть кнопка для добавления нового товара, который открывает страницу «Добавление товара». Данная страница содержит форму для создания нового товара. От пользователя требуется ввести название, описание, цену и прикрепить до трех фотографий и сохранить товар. Если введенные данные корректны, то система занесет товар в базу данных и отобразит его в списке с товарами. Товары в списке отображаются по порядку их добавления. Интерфейс страницы «Добавление товара» представлен на рисунке 3.9:

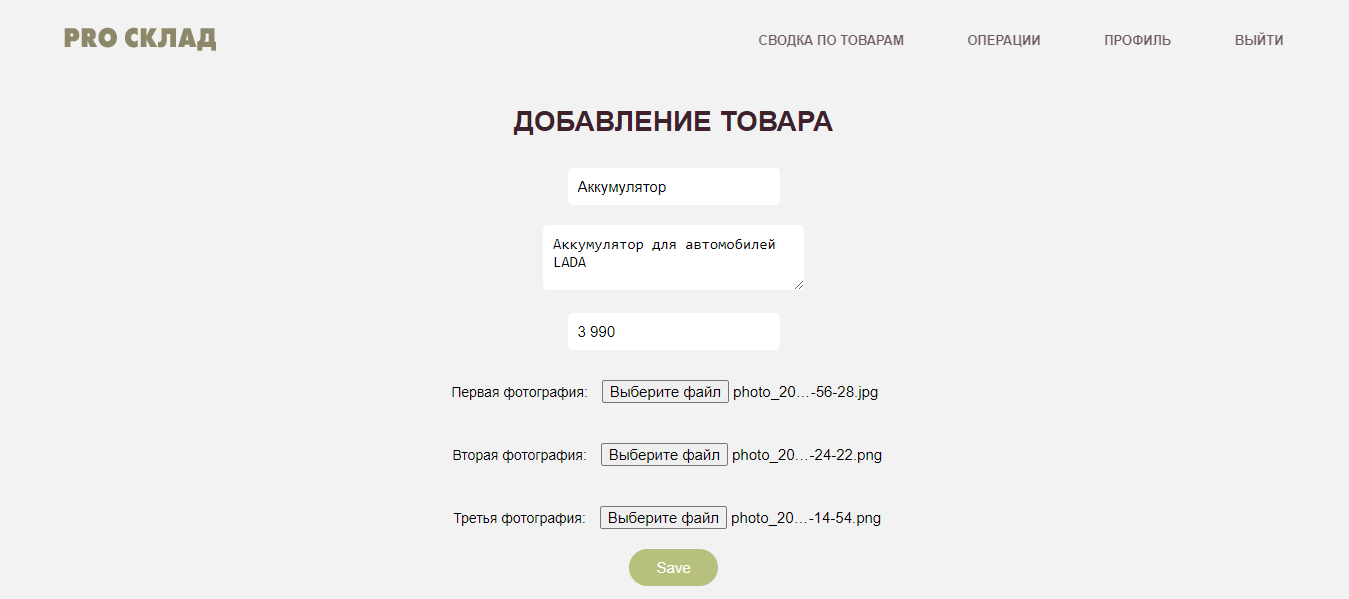


Рисунок 3.9 - Страница «Добавление товара»

Добавлять, редактировать и удалять товары могут только пользователи с ролью администратора.

* + 1. Страница «Склады»

Данная страница отображает список всех складов, которые есть в базе данных. Аналогично товарам склады можно добавлять, редактировать и удалять. Страница «Склады» показана на рисунке 3.10:

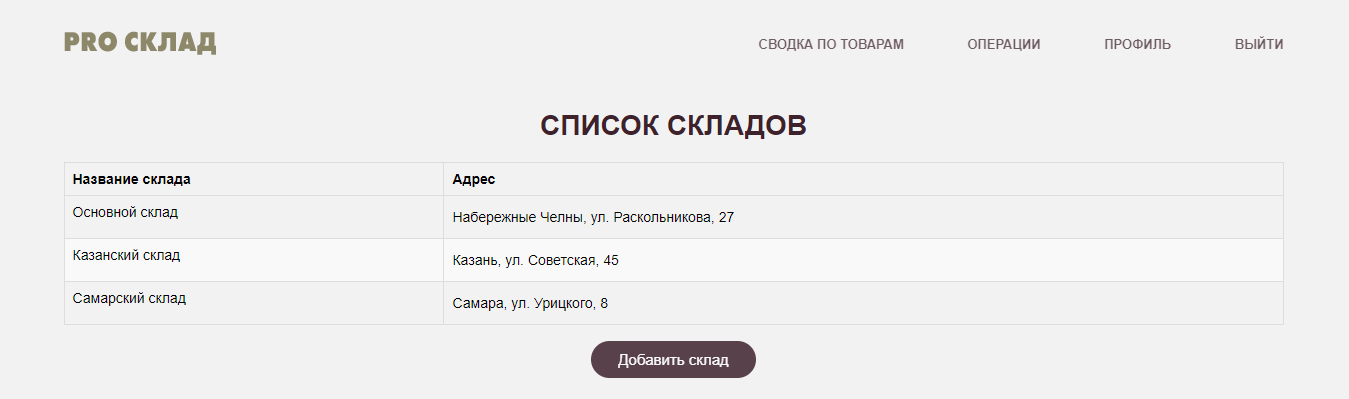


Рисунок 3.10 - Страница «Склады»

* + 1. Страница «Сводка»

Страница «Сводка» показывает запасы товаров на складах. В таблице представлено название товара, его количество и склад, в котором товар находится. Чтобы добавить товар на склад, нужно нажать на кнопку «Добавить товар на склад». На рисунке 3.11 представлен интерфейс страницы «Сводка».

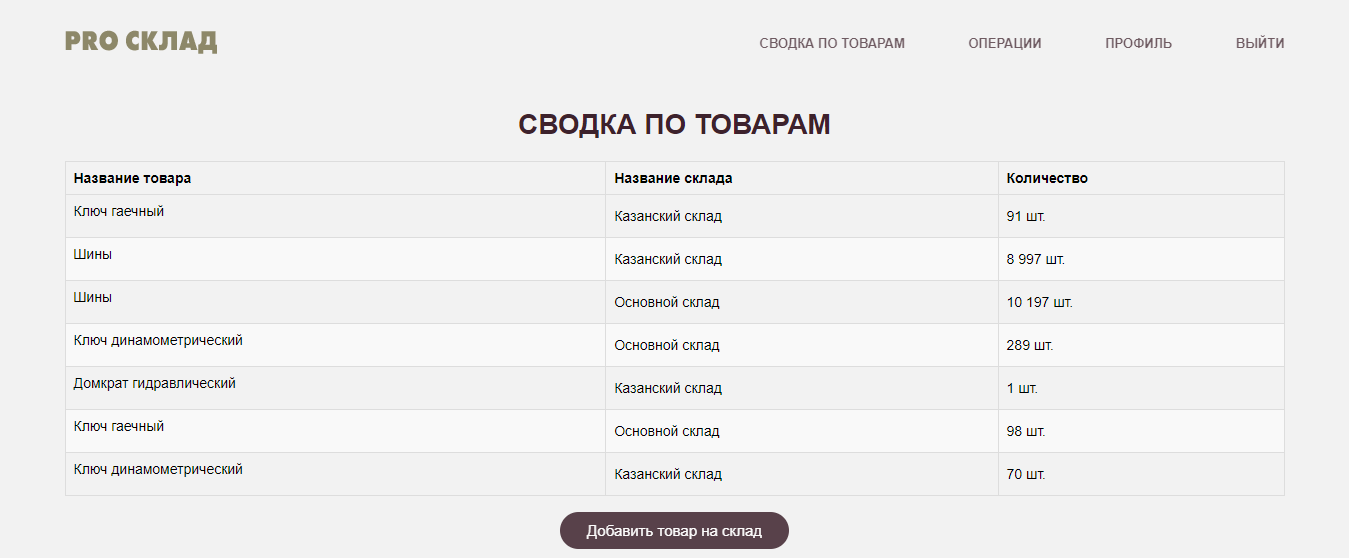


Рисунок 3.11 - Страница «Сводка по товарам»

Если пользователь хочет добавить товар на склад, он открывает страницу «Добавление товара на склад». Здесь необходимо заполнить идентификаторы склада и товара и указать количество для добавления. Система проверяет введение данные, если товара с данным идентификаторомнет на складе, то создается новая запись. Если товар уже есть на складе, то система увеличивает количество товара. Интерфейс страницы показан на рисунке 3.12:

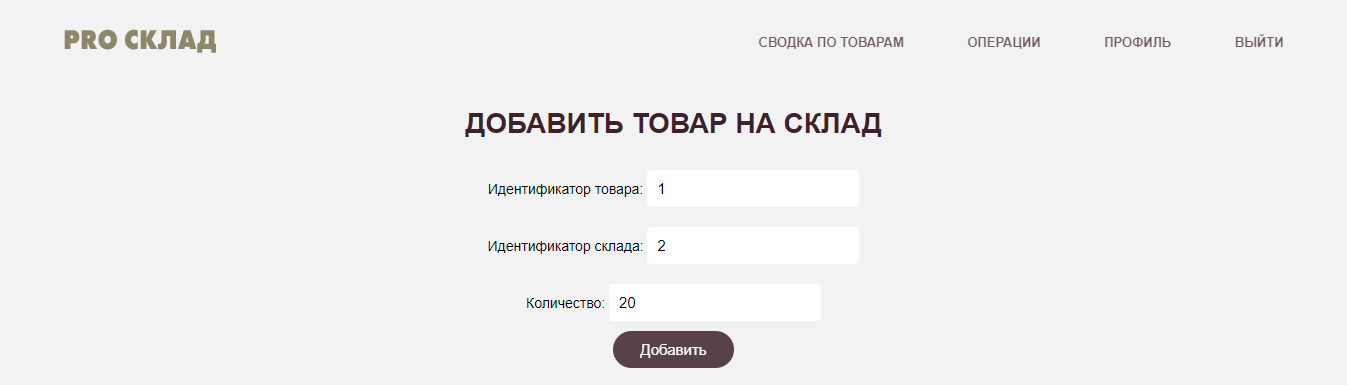


Рисунок 3.12 - Страница «Добавление товара на склад»

* + 1. Страница «Приемка»

Для выполнения операции приемки требуется открыть страницу «Приемка», ввести идентификатор склада, поставщика, товары и их количество. В список можно добавлять несколько товаров, при

необходимости убрать из списка товар нажатием на кнопку «Удалить». После добавления всех поступивших товаров, пользователь нажимаем на кнопку «Создать приемку». В первую очередь система получает имя аутентифицированного пользователя и выполняет поиск этого пользователя в базе данных по его электронной почте. Если пользователь найден, то записывает в базе данных, что операция была совершена данным пользователем. Далее, если остальные данные корректны, система сохранит приемку в базе данных. Иначе выйдет страница об ошибке. Интерфейс страницы «Приемка» представлен на рисунке 3.13:

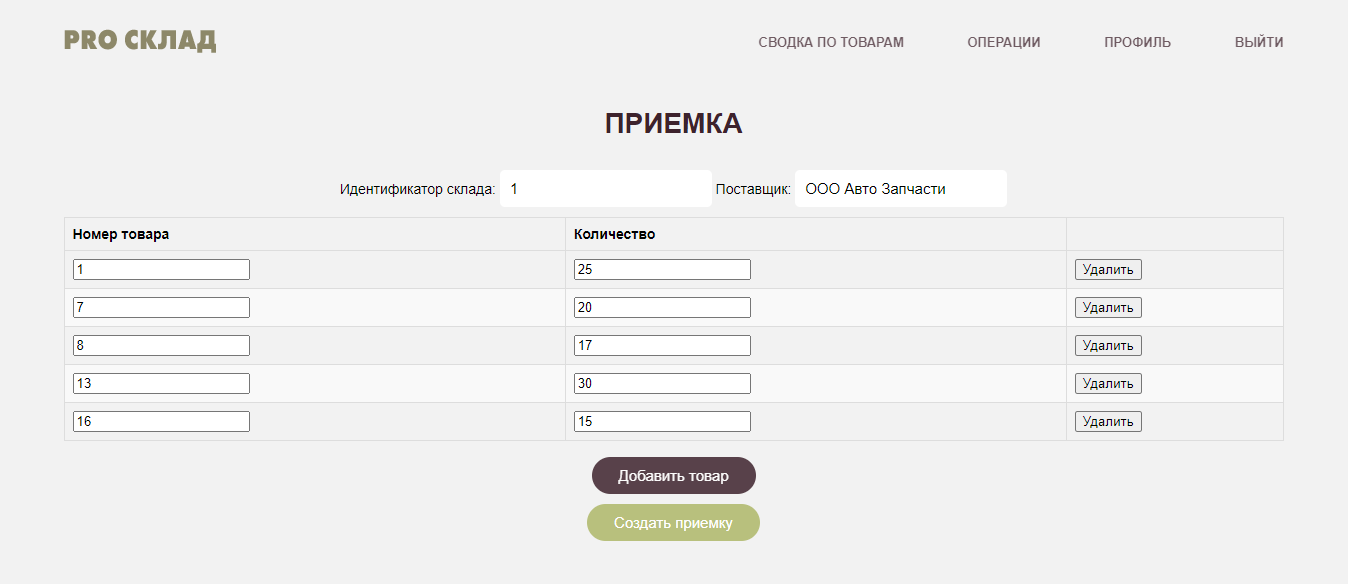


Рисунок 3.13 - Страница «Приемка»

* + 1. Страница «Перемещение»

Данная страница предназначена для выполнения перемещений товаров между складами. От пользователя требуется заполнить идентификаторы складов назначения и убытия. Далее указать товары и их количество. После завершения заполнения пользователь нажимает на кнопку «Переместить». Аналогично приемке, если данные корректны, система добавляет перемещение в базу данных и делает перерасчет количества товаров на складах. На рисунке 3.14 изображен интерфейс страницы «Перемещение».

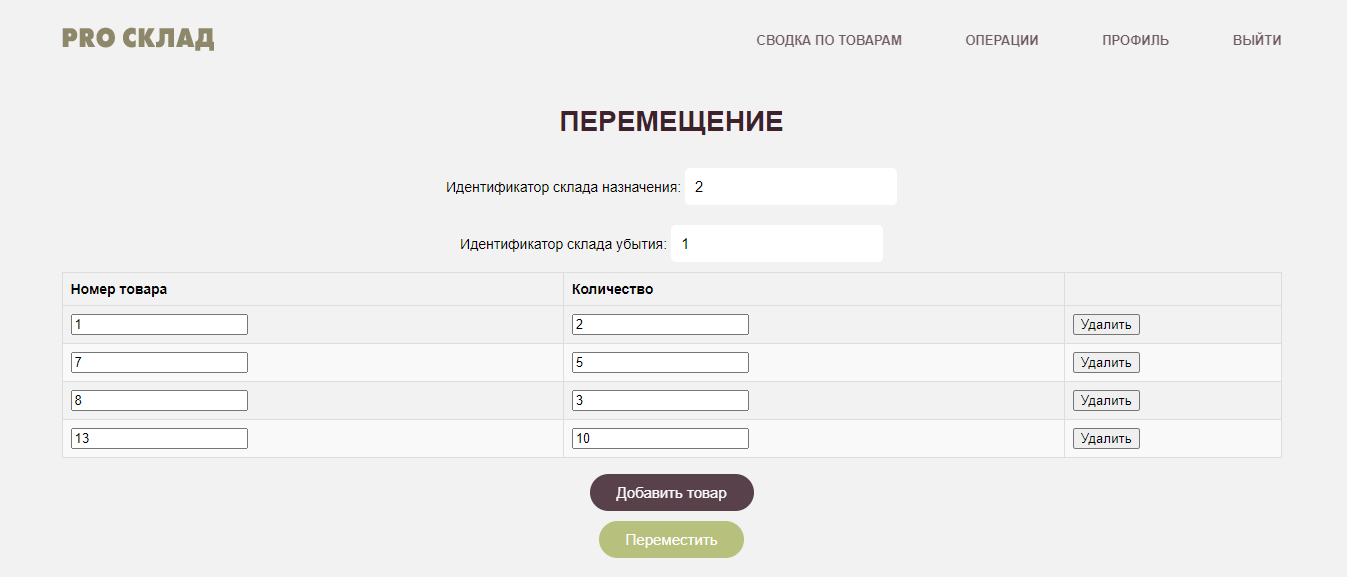


Рисунок 3.14 - Страница «Перемещение»

3.2.9 Страница «Списание»

Данная страница предназначена для выполнения списания товаров на складах. Пользователю необходимо заполнить идентификатор склада, указать товары и их количество для списания. Списать можно сразу несколько товаров. После заполнения пользователь нажимает на кнопку «Списать». Система проверяет пользователя и веденные данные и, если они корректны, вносит запись о списании в базу данных и пересчитывает остатки товаров. При некорректных данных выйдет страница об ошибке. Интерфейс страницы представлен на рисунке 3.15:

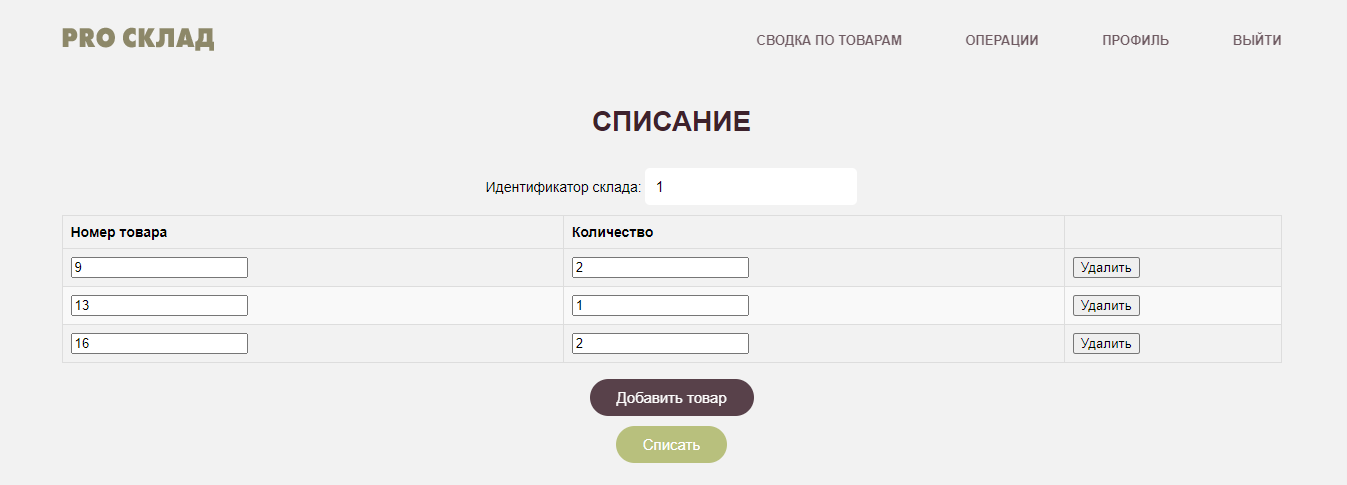


Рисунок 3.15 - Страница «Списание»

3.3 Выводы по главе

В ходе третьей главы, посвященной реализации информационной системы учета товаров на складах, были выполнены следующие задачи:

1. Рассмотрена структура веб-приложения.
2. Описаны методы сервера.
3. Описан принцип формирования документов.
4. Продемонстрирован пользовательский интерфейс.

Таким образом задача по реализации системы выполнена в полном объеме.

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломной работы была разработана информационная система для учета товаров на складах для компании ООО «Авто-Про». Ранее компания вела ручной учет, который занимал много времени и имел высокие риски недочетов и потери товаров. Данная система решает эти проблемы, предоставляя надежный способ хранения данных о товарах, а также удобный функционал для проведения операций приемки, списания, перемещения и формирование соответствующих складских документов.

В первой главе был проведен анализ требований на разработку, где выполнены следующие задачи:

* проведение анализа процесса учета товаров на складах;
* разработка модели процессов;
* разработка пользовательских требований;
* разработка функциональных требований;
* разработка нефункциональных требований.

Вторая глава по проектированию веб-приложения решены задачи:

* рассмотрение архитектуры веб-приложения;
* разработку функциональной модели системы;
* разработку алгоритмов системы;
* рассмотрение структуры базы данных приложения.

В третьей главе выполнены задачи:

* рассмотрение структуры веб-приложения;
* описание методов сервера;
* описание принципов формирования документов;
* описание пользовательского интерфейса.

В результате выполнения всех поставленных задач, цель работы была достигнута. Разработанная информационная система для учета товаров на складах компании ООО «Авто-Про» позволяет эффективно управлять складскими операциями, обеспечивает точность и достоверность данных, а также сокращает время выполнения задач, повышая производительность и качество работы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Хамадеев Ш.А. Методология моделирования бизнес-процессов BPMN2. Учебно-методическое пособие по дисциплине «Проектирование АСОИУ». - Набережные Челны: ИПЦ НЧИ К(П)ФУ, 2017. - 36 с.
2. Хамадеев Ш.А. Методология описания пользовательских требований. Учебно-методическое пособие по дисциплине «Проектирование АСОИУ». - Набережные Челны: ИПЦ НЧИ К(П)ФУ, 2017. - 28 с.
3. Вигерс К., Битти Д. Разработка требований к программному обеспечению. 3-е изд., дополненное / Пер. с англ. — М.: Издательство «Русская редакция»; СПб.: БХВ-Петербург, 2014. — 736 стр.: ил.
4. Вендров, А. М. Практикум по проектированию программного обеспечения экономических информационных систем / А.М. Вендров. - М.: Финансы и статистика, 2018. - 192 c.
5. Архитектура информационных систем / Б.Я. Советов и др. - М.: Academia, 2012. - 288 c.
6. Сьерра К., Бейтс Б. Изучаем Java. 2-е издание / Пер. с англ. - Москва: Эксмо, 2018. - 720 с.: ил.
7. Уоллс К. Spring в действии. 6-е изд./ Gер. с англ. А. Н. Киселева. - М.: ДМКПресс, 2022. - 544 с.: ил.
8. Арлоу Д., Нейштадт И. UML 2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование, 3-е издание. - Пер. с англ. - СПб: Символ-Плюс, 2017. - 624 с., ил.
9. Моргунов, Е. П. PostgreSQL. Основы языка SQL: учеб. Пособие / Е. П. Моргунов; под ред. Е. В. Рогова, П. В. Лузанова. — СПб.: БХВ- Петербург, 2018. — 336 с.: ил.
10. Что такое складской учет и как его вести [Электронный ресурс] // МойСклад.ру [сайт] URL: [https://www.moysklad.ru/poleznoe/shkola-](https://www.moysklad.ru/poleznoe/shkola-torgovli/organizaciya-skladskogo-ucheta/)

[torgovli/organizaciya-skladskogo-ucheta/](https://www.moysklad.ru/poleznoe/shkola-torgovli/organizaciya-skladskogo-ucheta/) (Дата обращения: 20.05.23 г.)

1. ER-диаграмма [Электронный ресурс] // Lucidchart [сайт] URL:

[https://www.lucidchart.com/pages/ru/erd-%D0%B4%D0%B8%D0%B0%D0%B3](https://www.lucidchart.com/pages/ru/erd-%d0%b4%d0%b8%d0%b0%d0%b3%20%d1%80%d0%b0%d0%bc%d0%bc%d0%b0) [%D 1 %80%D0%B0%D0%BC%D0%BC%D0%B0](https://www.lucidchart.com/pages/ru/erd-%d0%b4%d0%b8%d0%b0%d0%b3%20%d1%80%d0%b0%d0%bc%d0%bc%d0%b0) (Дата обращения: 20.05.23 г.)

1. Образцы документов [Электронный ресурс] // КонсультантПлюс [сайт] URL:<https://www.consultant.ru/document/>(Дата обращения: 22.05.23 г.)
2. BPMN для «чайников» [Электронный ресурс] // BPMN2 URL: [https://bpmn2.ru/e-mail-course/bpmu-dlya-chainikov-bazovie-elementy](https://bpmn2.ru/e-mail-course/bpmn-dlya-chainikov-bazovie-elementy) (Дата

обращения: 22.05.23 г.)

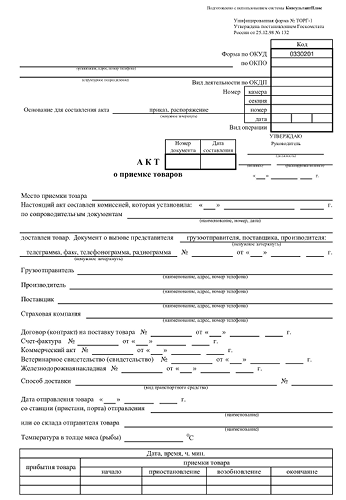
1. Диаграмма вариантов использования [Электронный ресурс] // Flexberry [сайт] URL: [https://flexberrv.github.io/ru/fd use-case-diagram.html](https://flexberry.github.io/ru/fd_use-case-diagram.html)

(Дата обращения: 25.05.23 г.)

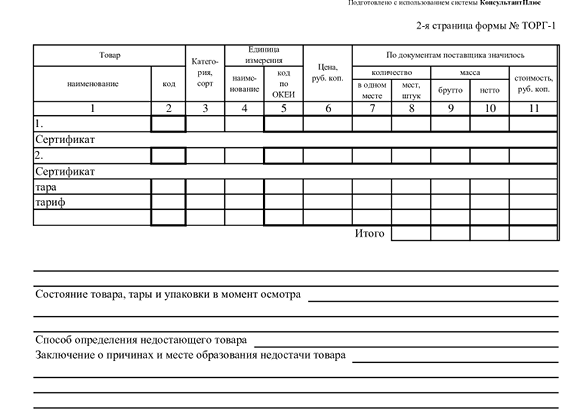
1. Назначение и состав методологии IDEF0 [Электронный ресурс] // Блог о бизнес-процессах и BPMN [сайт] URL: [https://bpmu.pro/process/idef0](https://bpmn.pro/process/idef0) (Дата обращения: 25.05.23 г.)

**ПРИЛОЖЕНИЕ А**

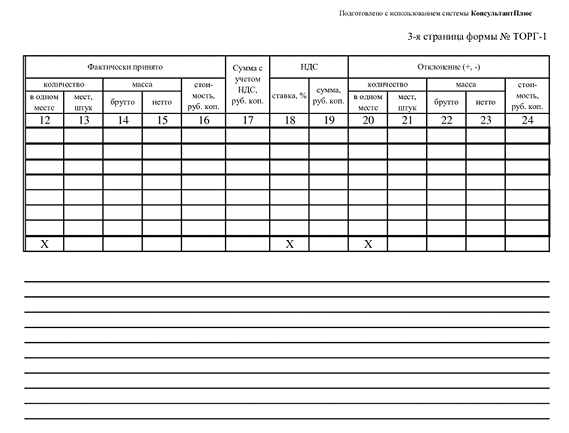
**Бланки документов**

****

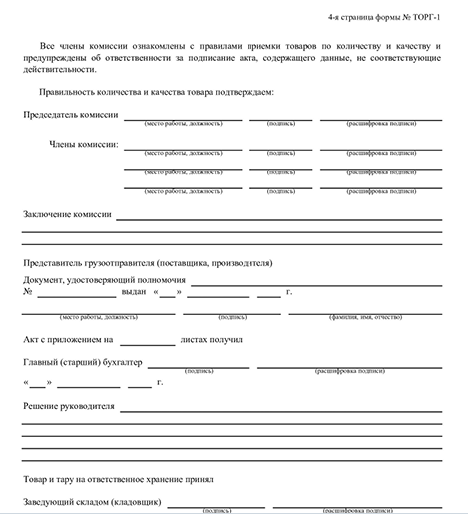
Бланк документа «ТОРГ-1» - 1 страница



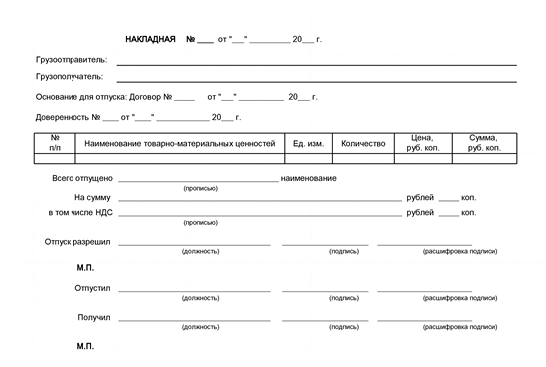
Бланк документа «ТОРГ-1» – 2 страница

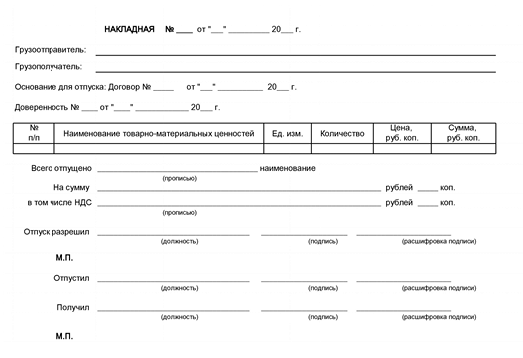


Бланк документа «ТОРГ-1» - 3 страница

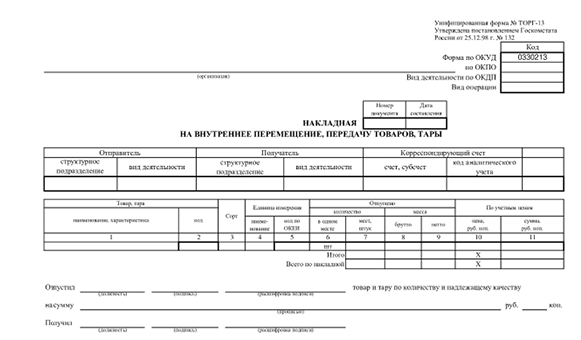


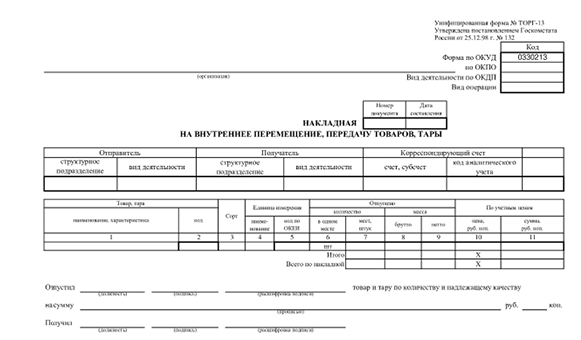
Бланк документа «ТОРГ-1» - 4 страница



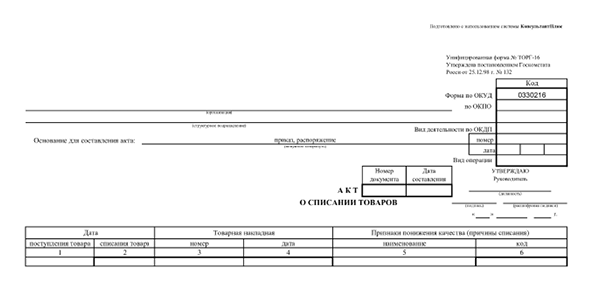


Бланк документа «Приходная накладная»

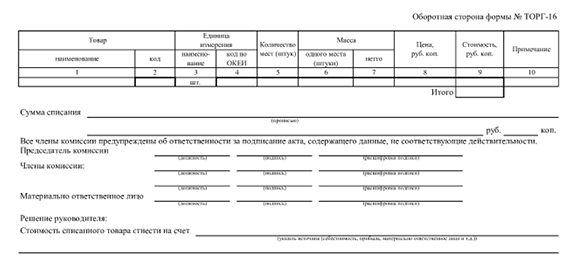




Бланк документа «ТОРГ-13»



Бланк документа «ТОРГ-16» - 1



Бланк документа «ТОРГ-16» - 2 страница

ПРИЛОЖЕНИЕ Б

Листинг программы

Класс контроллера регистрации SignUpController.java

package com.example.website.controllers;

import com.example.website.dto.SignUpDto;

import com.example.website.services.SignUpService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.validation.BindingResult;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.ModelAttribute;

import org.springframework.web.bind.annotation.PostMapping;

import javax.validation.Valid;

@Controller

public class SignUpController {

@Autowired

private SignUpService service;

@GetMapping("/signUp")

public String getSignUpPage() { return "sign\_up";

} ”

@PostMapping("/signUp")

public String signUp(@Valid @ModelAttribute("form") SignUpDto form, BindingResult bindingResult) {

if (bindingResult.hasErrors()) { return "sign\_up";

} ”

service.signUp(form);

return "redirect:/signIn";

}

}

Класс контроллера входа SignInController.java

package com.example.website.controllers;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PostMapping;

@Controller

public class SignInController {

@GetMapping("/signIn")

public String getSignInPage() { return "sign\_in";

} ”

}

Класс контроллера пользователей UsersController.java

package com.example.website.controllers;

import com.example.website.dto.UserDto;

import com.example.website.services.UsersService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.security.access.prepost.PreAuthorize;

import org.springframework.ui.Model;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import java.util.List;

@Controller

public class UsersController {

@Autowired

private UsersService usersService;

@PreAuthorize("hasAuthority('ADMIN')")

@GetMapping("/users")

public String getUsersPage(Model model) {

List<UserDto> users = usersService.getAllUsers();

model.addAttribute("users", users); return "users";

}

@PreAuthorize("hasAuthority('ADMIN')")

@PostMapping("/users/{user-id}/delete")

public String deleteUser(@PathVariable("user-id") Long userId) { usersService.deleteUser(userId);

return "redirect:/users";

}

}

Класс конфигурации безопасности WebSecurityConfig.java

package com.example.website.security.config;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.beans.factory.annotation.Qualifier;

import

org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuild er;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import

org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.security.web.util.matcher.AntPathRequestMatcher;

import java.util.concurrent.TimeUnit;

@EnableWebSecurity

public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

@Autowired

@Qualifier(value = "CustomUserDetailsService")

private UserDetailsService userDetailsService;

@Autowired private PasswordEncoder passwordEncoder;

@Autowired

@Override

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder);

}

@Override

protected void configure(HttpSecurity http) throws Exception { http.csrf().disable();

http.authorizeRequests()

.antMatchers("/", "/signUp").permitAll()

.antMatchers("/users", "/newReceipt", "/newTransfer", "/newWriteOff", "/download/\*\*", "/generateTorg16/\*\*", "/generateTorg13/\*\*", "/generateTorg1/\*\*", "/generateNakladnaya/\*\*", "/newItem", "/editItem/\*\*", "/deleteItem/\*\*", "/newStock", "/editStock/\*\*", "/deleteStock/\*\*", "/addItem").hasAnyAuthority("ADMIN")

.antMatchers("/receipts", "/transfers", "/writeOffs", "/documents", "/items", "/stocks", "/stockItems").authenticated();

http.formLogin()

.loginPage("/signIn")

.usernameParameter("email")

.defaultSuccessUrl("/")

.failureUrl("/signIn?error")

.permitAll()

.and()

.rememberMe()

.tokenValiditySeconds((int)TimeUnit.*DAYS* .toSeconds(21));

http.logout()

.logoutUrl("/logout")

.logoutRequestMatcher(new AntPathRequestMatcher("/logout", "GET"))

.clearAuthentication(true)

.invalidateHttpSession(true)

.deleteCookies("JSESSIONID", "remember-me")

.logoutSuccessUrl("/");

}

}

Класс контроллера товаров ItemController.java

package com.example.website.controllers;

import com.example.website.models.Item;

import com.example.website.services.ItemsService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.data.repository.query.Param;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.\*;

import org.springframework.web.multipart.MultipartFile;

import java.io.IOException;

import java.time.LocalDateTime;

import java.util.List;

import java.util.Optional;

@Controller

public class ItemController {

@Autowired

private ItemsService itemsService;

@GetMapping("/items")

public String getAllItems(Model model, @Param("keyword") String keyword) {

List<Item> items = itemsService.findAll(keyword);

model.addAttribute("items", items);

return "items";

}

@GetMapping("/items/{id}")

public String getPost(@PathVariable("id") Long id, Model model) {

Optional<Item> optionalItem = itemsService.getById(id);

if (optionalItem.isPresent()) {

Item item = optionalItem.get();

model.addAttribute("item", item);

model.addAttribute("images", item.getImages());

return "item";

} else { return "error";

}

}

@GetMapping("/newItem")

public String getNewItemPage() { return "itemForm";

}

@PostMapping("/newItem")

public String createItem(Item item, @RequestParam("file1") MultipartFile file1, @RequestParam("file2") MultipartFile file2, @RequestParam("file3") MultipartFile file3) throws IOException {

item.setCreatedAt(LocalDateTime.*now* ());

*// item.setQuantity(O); // дефолтное значение идет 0*

itemsService.save(item, file1, file2, file3);

return "redirect:/items/" + item.getId();

}

@GetMapping("/editItem/{id}")

public String editItem(@PathVariable Long id, Model model) { Optional<Item> optionalItem = itemsService.getById(id);

if (optionalItem.isPresent()) {

Item item = optionalItem.get();

model.addAttribute("item", item);

return "editItemForm";

} else {

System.*err*.println("Could not find a item #" + id);

return "error";

}

}

@PostMapping("/editItem/{id}")

public String editPost(@PathVariable Long id, Item item) { Optional<Item> optionalItem = itemsService.getById(id);

if (optionalItem.isPresent()) {

Item editedItem = optionalItem.get();

editedItem.setName(item.getName());

editedItem.setDescription(item.getDescription());

editedItem.setPrice(item.getPrice());

itemsService.save(editedItem);

return "redirect:/items/" + editedItem.getId();

} else { return "error";

}

}

@DeleteMapping("/deleteItem/{id}")

public String deleteItem(@PathVariable Long id) {

Optional<Item> optionalItem = itemsService.getById(id);

if (optionalItem.isPresent()) {

itemsService.deleteById(id);

return "redirect:/items";

} else { return "error";

}

}

}

Класс контроллера складов StockController.java

package com.example.website.controllers;

import com.example.website.models.Stock;

import com.example.website.models.StockItem;

import com.example.website.services.ItemsService;

import com.example.website.services.StockItemsService;

import com.example.website.services.StocksService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import java.util.List;

import java.util.Optional;

@Controller

public class StockController {

@Autowired

private StocksService stocksService;

@Autowired

private StockItemsService stockItemsService;

@GetMapping("/stocks")

public String getAllStocks(Model model){

List<Stock> stocks = stocksService.findAll();

model.addAttribute("stocks", stocks); return "stocks";

}

@GetMapping("/stocks/{id}")

public String getStock(@PathVariable("id") Long id, Model model){ Optional<Stock> optionalStock = stocksService.getById(id);

if (optionalStock.isPresent()) {

Stock stock = optionalStock.get();

List<StockItem> stockitems = stockItemsService.findByStockId(stock.getId());

model.addAttribute("stock", stock);

model.addAttribute("stockitems", stockitems);

return "stock";

} else {

return "error";

}

}

@GetMapping("/newStock")

public String getNewStockPage() {

return "stockForm";

}

@PostMapping("/newStock")

public String createStock(Stock stock) {

stocksService.save(stock);

return "redirect:/stocks/" + stock.getid();

}

@GetMapping("/editStock/{id}")

public String editStock(@PathVariable Long id, Model model){

Optional<Stock> optionalStock = stocksService.getByid(id);

if (optionalStock.isPresent()) {

Stock stock = optionalStock.get();

model.addAttribute("stock", stock);

return "editStockForm";

} else {

System.*err*.println("Could not find a stock #" + id);

return "error";

}

}

@PostMapping ("/editStock/{id}")

public String editStock(@PathVariable Long id, Stock stock){

Optional<Stock> optionalStock = stocksService.getByid(id);

if (optionalStock.isPresent()) {

Stock editedStock = optionalStock.get();

editedStock.setName(stock.getName());

editedStock.setAddress(stock.getAddress());

stocksService.save(editedStock);

return "redirect:/stocks/" + editedStock.getid();

} else {

return "error";

}

}

@DeleteMapping("/deleteStock/{id}")

public String deleteStock(@PathVariable Long id){

Optional<Stock> optionalStock = stocksService.getByid(id);

if (optionalStock.isPresent()){

stocksService.deleteByid(id);

return "redirect:/stocks";

} else {

return "error";

}

}

}

Класс контроллера товаров на складах StockItemsController.java

package com.example.website.controllers;

import com.example.website.models.Item;

import com.example.website.models.Stock;

import com.example.website.models.Stockitem;

import com.example.website.services.ItemsService;

import com.example.website.services.StockitemsService;

import com.example.website.services.StocksService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestParam;

import java.util.List;

import java.util.Optional;

@Controller

public class StockItemsController {

@Autowired

private StockItemsService stockItemsService;

@Autowired

private StocksService stocksService;

@Autowired

private ItemsService itemsService;

@GetMapping("/stockItems")

public String getAllStockItems(Model model){

List<StockItem> stockItems = stockItemsService.findAll();

model.addAttribute("stockItems", stockItems); return "stockItems";

}

@GetMapping("/addItem")

public String getAddItemPage() { return "addItem";

}

@PostMapping("/addItem")

private String addItemToStockItems(@RequestParam("itemId") Long itemId, @RequestParam("stockId") Long stockId, Integer quantity){

Optional<Item> optionalItem = itemsService.getById(itemId);

Optional<Stock> optionalStock = stocksService.getById(stockId);

if (optionalStock.isPresent() && optionalItem.isPresent()){

Stock stock = optionalStock.get();

Item item = optionalItem.get();

List<StockItem> stockItemList = stockItemsService.findByItemId(item.getId());

if (!stockItemList.isEmpty()){

for (int i = 0; i < stockItemList.size(); i++) {

if (stockItemList.get(i).getStock().equals(stock)) {

stockItemList.get(i).setQuantity(stockItemList.get(i).getQuantity() + quantity);

stockItemsService.save(stockItemList.get(i)); return "redirect:/stocks/" + stockId;

} } StockItem newStockItem = new StockItem(); newStockItem.setItem(item);

newStockItem.setStock(stock);

newStockItem.setQuantity(quantity);

stockItemsService.save(newStockItem); } else {

StockItem stockItem = new StockItem();

stockItem.setItem(item);

stockItem.setStock(stock);

stockItem.setQuantity(quantity);

stockItemsService.save(stockItem); } return "redirect:/stocks/" + stockId; } else { return "error";

}

}

}

Класс контроллера приемок ReceiptController.java

package com.example.website.controllers;

import com.example.website.models.\*;

import com.example.website.services.\*;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestParam;

import java.time.LocalDateTime;

import java.util.List;

import java.util.Optional;

@Controller

public class Receiptcontroller {

@Autowired

private UsersService usersService;

@Autowired

private StockItemsService stockItemsService;

@Autowired

private StocksService stocksService;

@Autowired

private ReceiptsService receiptsService;

@Autowired

private ReceiptItemsService receiptItemsService;

@GetMapping("/receipts")

public String getAllReceipts(Model model) {

List<Receipt> receipts = receiptsService.findAll();

model.addAttribute("receipts", receipts);

return "receipts";

}

@GetMapping("/receipts/{id}")

public String getReceipt(@PathVariable("id") Long id, Model model) { Optional<Receipt> optionalReceipt = receiptsService.getById(id); if (optionalReceipt.isPresent()) {

Receipt receipt = optionalReceipt.get();

model.addAttribute("receipt", receipt); return "receipt";

} else { return "error";

}

}

@GetMapping("/newReceipt")

public String getNewReceiptPage() { return "receiptForm";

}

@PostMapping("/newReceipt")

public String createReceipt(@RequestParam("stockId") Long stockId, @RequestParam("supplier") String supplier, @RequestParam("item\_id[]") Long[] itemId,

@RequestParam("item\_qty[]") Integer[] quantity) {

String auth = SecurityContextHolder.*getContext*().getAuthentication().getName();

Optional<User> optionalUser = usersService.findUserByEmail(auth);

if (optionalUser.isPresent()) {

Optional<Stock> optionalStock = stocksService.getById(stockId);

if (optionalStock.isPresent()) {

Receipt receipt = new Receipt();

receipt.setUser(optionalUser.get());

receipt.setStock(optionalStock.get());

receipt.setSupplier(supplier);

receipt.setCreatedAt(LocalDateTime. *now*());

receiptsService.save(receipt);

*// если товара с id нет, то вывести ошибку добавьте товар с id, после этого выполните приемку*

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

if (!stockItemsService.findByItemId(itemId[i]).isEmpty()) {

if (stockItemsService.existsStockItemByItemIdAndStockId(itemId[i], stockId)) {

System.*out*.println("moeap и склад уже есть");

*// найти данный товар и добавить количество*

StockItem stockItem =

stockItemsService.findByItemIdAndStockId(itemId[i], stockId);

stockItem.setQuantity(stockItem.getQuantity() + quantity[i]);

stockItemsService.save(stockItem);

ReceiptItem receiptItem = new ReceiptItem();

receiptItem.setItem(stockItem.getItem());

receiptItem.setQuantity(quantity[i]);

receiptltem.setReceipt(receipt);

receiptltemsService.save(receiptltem);

receipt.addReceiptltem(receiptltem);

} else {

System.*out*.println("moeap есть, но в др складе");

*// создать новую запись, скопировав товарное id*

List<StockItem> stockItemList = stockItemsService.findByItemId(itemId[i]);

Stockitem newStockItem = new Stockitem();

newStockItem.setStock(optionalStock.get());

newStockItem.setItem(stockItemList.get(0).getItem());

newStockitem.setQuantity(quantity[i]); stockitemsService.save(newStockitem);

Receiptitem receiptitem = new Receiptitem();

receiptItem.setItem(newStockItem.getitem());

receiptitem.setQuantity(newStockitem.getQuantity());

receiptitem.setReceipt(receipt);

receiptitemsService.save(receiptitem);

receipt.addReceiptitem(receiptitem); }

} else {

System.*out*.println("Toeapa с id #" + itemid[i] + " нет в базе данных.

Прошу добавьте товар в базу данных и после повторите функцию");

}

}

receipt.setQuantity(receipt.getListSize());

receiptsService.save(receipt);

return "redirect:/receipts";

} else { return "error";

}

}

else { return "error";

}

}

}

Класс контроллера списаний WriteOffController.java

package com.example.website.controllers;

import com.example.website.models.\*;

import com.example.website.services.\*;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestParam;

import java.time.LocalDateTime;

import java.util.List;

import java.util.Optional;

@Controller

public class WriteOffController {

@Autowired

private DocumentsService documentsService;

@Autowired

private UsersService usersService;

@Autowired

private ExcelGenerationService excelGenerationService;

@Autowired

private WriteOffService writeOffService;

@Autowired

private WriteOffitemsService writeOffitemsService;

@Autowired

private StockItemsService stockItemsService;

@Autowired

private StocksService stocksService;

@Autowired

private ItemsService itemsService;

@GetMapping("/writeOffs")

public String getAllWriteOffs(Model model){

List<WriteOff> writeoffs = writeOffService.findAll();

model.addAttribute("writeOffs", writeOffs);

return "write\_offs";

} ”

@GetMapping("/writeOffs/{id}")

public String getWriteOff(@PathVariable("id") Long id, Model model){

Optional<WriteOff> optionalWriteOff = writeOffService.getById(id);

if (optionalWriteOff.isPresent()) {

WriteOff writeOff = optionalWriteOff.get();

model.addAttribute("writeOff", writeOff); return "write\_off";

} else { return "error";

}

}

@GetMapping("/newWriteOff")

public String getNewWriteOffPage() { return "writeOffForm";

}

@PostMapping("/newWriteOff")

public String createWriteOff(@RequestParam("item\_id[]") Long[] itemId,

@RequestParam("item\_qty[]") Integer[] quantity,

@RequestParam("stockId") Long stockId) {

String auth = SecurityContextHolder.*getContext*().getAuthentication().getName();

Optional<User> optionalUser = usersService.findUserByEmail(auth);

if (optionalUser.isPresent()) {

Optional<Stock> optionalStock = stocksService.getById(stockId);

if (optionalStock.isPresent()) {

Stock stock = optionalStock.get();

WriteOff writeOff = new WriteOff();

writeOff.setUser(optionalUser.get());

writeOff.setStock(stock);

writeOff.setCreatedAt(LocalDateTime.*now* ());

writeOffService.save(writeOff);

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

Optional<Item> optionalItem = itemsService.getById(itemId[i]);

if (optionalItem.isPresent()) {

*// проверить что этот товар есть на данном складе и его кол-во больше чем quantity*

if

(stockItemsService.existsStockItemByItemIdAndStockIdAndQuantityGreaterThan(itemId[i], stock.getId(), quantity[i])) {

stockItemsService.writeOff(itemId[i], stock.getId(), quantity[i]);

WriteOffItem writeOffItem = new WriteOffItem();

writeOffItem.setWriteOff(writeOff);

writeOffItem.setItem(optionalItem.get());

writeOffItem.setQuantity(quantity[i]);

writeOffItemsService.save(writeOffItem);

writeOff.addWriteOffItem(writeOffItem);

} else {

System.*out*.println("Toeapa нет на складе, либо его количество меньше quantity");

}

} else {

System. *out* .printlnC’^eap с id #" + itemId[i] + " не существует.");

}

}

writeOff.setQuantity(writeOff.getListSize());

writeOffService.save(writeOff);

return "redirect:/writeOffs/" + writeOff.getId();

} else {

System. *out* .println("C^ag не найден.");

}

} else {

return "error";

}

return "error";

}

}

Класс контроллера перемещений TransferController.java

package com.example.website.controllers;

import com.example.website.models.\*;

import com.example.website.services.\*;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestParam;

import java.time.LocalDateTime;

import java.util.List;

import java.util.Optional;

@Controller

public class Transfercontroller {

@Autowired

private TransfersService transfersService;

@Autowired

private UsersService usersService;

@Autowired

private TransferItemsService transferItemsService;

@Autowired

private StockItemsService stockItemsService;

@Autowired

private StocksService stocksService;

@Autowired

private ItemsService itemsService;

@Autowired

private ExcelGenerationService excelGenerationService;

@GetMapping("/transfers")

public String getTransfers(Model model){

List<Transfer> transfers = transfersService.findAll();

model.addAttribute("transfers", transfers);

return "transfers";

}

@GetMapping("/transfers/{id}")

public String getTransfer(@PathVariable("id") Long id, Model model){

Optional<Transfer> optionalTransfer = transfersService.getById(id);

if (optionalTransfer.isPresent()) {

Transfer transfer = optionalTransfer.get();

model.addAttribute("transfer", transfer);

return "transfer";

} else { return "error";

}

}

@GetMapping("/newTransfer")

public String getTransferPage() {

return "transferForm";

}

@PostMapping("/newTransfer")

public String createTransfer(@RequestParam("fromStockId") Long fromStockId, @RequestParam("toStockId") Long toStockId,

@RequestParam("item\_id[]") Long[] itemId, @RequestParam("item\_qty[]") Integer[] quantity) {

String auth = SecurityContextHolder.*getContext*().getAuthentication().getName();

Optional<User> optionalUser = usersService.findUserByEmail(auth);

if (optionalUser.isPresent()) {

Optional<Stock> optionalFromStock = stocksService.getById(fromStockId);

Optional<Stock> optionalToStock = stocksService.getById(toStockId);

if (optionalFromStock.isPresent() && optionalToStock.isPresent()){

Stock fromStock = optionalFromStock.get();

Stock toStock = optionalToStock.get();

Transfer transfer = new Transfer();

transfer.setUser(optionalUser.get());

transfer.setFromStock(fromStock);

transfer.setToStock(toStock);

transfer.setCreatedAt(LocalDateTime.*now* ());

transfersService.save(transfer);

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

Optional<Item> optionalItem = itemsService.getById(itemId[i]);

if (optionalItem.isPresent()){

*// проверить что этот товар есть на from складе и его кол-во больше чем quantity*

if (!stockItemsService.findByItemId(itemId[i]).isEmpty()){

if

(stockItemsService.existsStockItemByItemIdAndStockIdAndQuantityGreaterThan(itemId[i], fromStock.getId(), quantity[i])){

*// проверить есть ли товар в новом складе, если нет то создать его*

if

(!stockItemsService.existsStockItemByItemIdAndStockId(itemId[i], toStock.getId())){

*// если товара нет в данном складе, то создаем его*

StockItem stockItem = new StockItem();

stockItem.setStock(toStock);

stockItem.setItem(optionalItem.get());

stockItem.setQuantity(0);

stockItemsService.save(stockItem);

}

stockItemsService.transfer(itemId[i], fromStock.getId(), toStock.getId(), quantity[i]);

TransferItem transferItem = new TransferItem();

transferItem.setTransfer(transfer);

transferItem.setItem(optionalItem.get());

transferItem.setQuantity(quantity[i]);

transfer.addTransferItem(transferItem);

transferItemsService.save(transferItem);

} else {

System. *out* .println("Ko,n-BO товара с id #" + itemId[i] + " на from

складе меньше, чем " + quantity[i] + " или же нет в наличии");

}

} else {

System.*out*.println("Товар с id #" + itemId[i] + " нет в наличии.");

}

} else {

System.*out*.println("Товара с id #" + itemId[i] + " нет в базе данных.");

}

}

transfer.setQuantity(transfer.getListSize());

transfersService.save(transfer);

return "redirect:/transfers/" + transfer.getId();

} else {

return "error";

}

} else {

return "error";

}

}

}

Класс контроллера документов DocumentController.java

package com.example.website.controllers;

import com.example.website.models.Document;

import com.example.website.models.Receipt;

import com.example.website.models.Transfer;

import com.example.website.models.WriteOff;

import com.example.website.services.\*;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.core.io.ClassPathResource;

import org.springframework.core.io.Resource;

import org.springframework.[http.HttpHeaders](http://http.HttpHeaders);

import org.springframework.[http.MediaType](http://http.MediaType);

import org.springframework.[http.ResponseEntity](http://http.ResponseEntity);

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import java.io.\*;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.time.LocalDateTime;

import java.util.List;

import java.util.Optional;

@Controller

public class Documentcontroller {

@Autowired

private ReceiptsService receiptsService;

@Autowired

private TransfersService transfersService;

@Autowired

private WriteOffService writeOffService;

@Autowired

private ExcelGenerationService excelGenerationService;

@Autowired

private DocumentsService documentsService;

@GetMapping("/documents")

public String getAllDocuments(Model model){

List<Document> documents = documentsService.findAll();

model.addAttribute("documents", documents); return "docs";

}

@GetMapping("/documents/{id}")

public String getDocument(@PathVariable("id") Long id, Model model){ Optional<Document> optionalDocument = documentsService.getById(id); if (optionalDocument.isPresent()) {

Document document = optionalDocument.get();

model.addAttribute("document", document); return "doc";

} else { return "error";

}

}

@GetMapping("/download/{fileName}")

public ResponseEntity<Resource> downloadFile(@PathVariable String fileName) throws IOException {

String filePath = "static/documents/" + fileName;

Resource resource = new ClassPathResource(filePath);

if (resource.exists()) {

Path file = Paths.*get*(resource.getURI());

byte[] fileContent = Files.*readAllBytes*(file);

HttpHeaders headers = new HttpHeaders();

headers.setContentDispositionFormData("attachment", fileName);

headers.setContentType(MediaType. *APPLICATION\_OCTET\_STRE^);*

return ResponseEntity.*ok*() .headers(headers) .body(resource);

} else { return ResponseEntity.*notFound*().build();

}

}

@GetMapping("/generateTorg16/{id}")

public String generateTorg16(@PathVariable("id") Long id, Model model){ Optional<WriteOff> optionalWriteOff = writeOffService.getById(id); if (optionalWriteOff.isPresent()) {

Writeoff writeoff = optionalWriteOff.get();

Document document = new Document();

document.setName("T0Pr-16");

document.setCreatedAt(LocalDateTime.*now());*

document.setOperation\_id(writeOff.getId());

document.setPath(excelGenerationService.generateTorg16(writeOff)); documentsService.save(document);

model.addAttribute("writeOff", writeoff); return "write\_off";

} else { return "error";

}

}

@GetMapping("/generateTorg13/{id}")

public String generateTorg13(@PathVariable("id") Long id, Model model){ Optional<Transfer> optionalTransfer = transfersService.getById(id);

if (optionalTransfer.isPresent()) {

Transfer transfer = optionalTransfer.get();

Document document = new Document();

document.setName("T0Pr-13");

document.setCreatedAt(LocalDateTime.*now*());

document.setOperation\_id(transfer.getId());

document.setPath(excelGenerationService.generateTorg13(transfer)); documentsService.save(document);

model.addAttribute("transfer", transfer); return "transfer";

} else { return "error";

}

}

@GetMapping("/generateNakladnaya/{id}")

public String generateNakladnaya(@PathVariable("id") Long id, Model model){

Optional<Receipt> optionalReceipt = receiptsService.getById(id);

if (optionalReceipt.isPresent()) {

Receipt receipt = optionalReceipt.get();

Document document = new Document();

document.setName("Накладная");

document.setCreatedAt(LocalDateTime.*now(*) );

document.setOperation\_id(receipt.getId());

document.setPath(excelGenerationService.generateNakladnaya(receipt));

documentsService.save(document);

model.addAttribute("receipt", receipt); return "receipt";

} else { return "error";

}

}

@GetMapping("/generateTorg1/{id}")

public String generateTorg1(@PathVariable("id") Long id, Model model){

Optional<Receipt> optionalReceipt = receiptsService.getById(id);

if (optionalReceipt.isPresent()) {

Receipt receipt = optionalReceipt.get();

Document document = new Document();

document.setName("T0Pr-1");

document.setCreatedAt(LocalDateTime.*now*());

document.setOperation\_id(receipt.getId());

document.setPath(excelGenerationService.generateTorg1(receipt)); documentsService.save(document);

model.addAttribute("receipt", receipt); return "receipt";

} else { return "error";

}

}

Класс сервиса ExcelGenerationServiceImpl.java

package com.example.website.services;

import com.example.website.models.Receipt;

import com.example.website.models.Transfer;

import com.example.website.models.WriteOff;

import org.apache.poi.ss.usermodel.\*;

import org.apache.poi.ss.util.CellRangeAddress;

import org.apache.poi.ss.util.RegionUtil;

import org.apache.poi.xssf.usermodel.XSSFWorkbook;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.core.io.ClassPathResource;

import org.springframework.core.io.Resource;

import org.springframework.stereotype.Service;

import org.springframework.util.ResourceUtils;

import java.io.\*;

import java.time.LocalDateTime;

import java.time.Month;

@Service

public class ExcelGenerationServicelmpl implements ExcelGenerationService {

public String generateTorg16(WriteOff writeoff) { try {

Resource resource = new ClassPathResource("static/documents/blank-torg-16.xlsx");

File file = resource.getFile();

InputStream inputStream = new FileInputStream(file);

Workbook workbook = new XSSFWorkbook(inputStream);

Sheet sheet = workbook.getSheetAt(0);

Row row = sheet.getRow(6); *// организация*

CellStyle cellStyle = workbook.createCellStyle(); cellStyle.setAlignment(HorizontalAlignment.*CENTER*) ;

for (int column = 0; column <= 55; column++) {

Cell cell = row.createCell(column);

cell.setCellValue("OOO АВТО-ПРО");

cell.setCellStyle(cellStyle); }

Row row2 = sheet.getRow(8); *// склад как подразделение*

CellStyle cellStyle2 = workbook.createCellStyle();

cellStyle2.setAlignment(HorizontalAlignment.*CENTER*) ;

for (int column = 0; column <= 67; column++) {

Cell cell2 = row2.createCell(column);

cell2.setCellValue(writeOff.getStock().getName());

cell2.setCellStyle(cellStyle2); }

Row row3 = sheet.getRow(17);

Cell cell3 = row3.getCell(51);

cell3.setCellValue(writeOff.getId());

LocalDateTime today = LocalDateTime.*now();*

Month month = today.getMonth();

int monthNumber = month.getValue();

String date = today.getDayOfMonth() + "." + String.*format*("%02d", monthNumber) + "."

+ today.getYear();

Cell cell4 = row3.getCell(57);

cell4.setCellValue(date);

Row row4 = sheet.getRow(25);

Cell cell5 = row4.getCell(11);

cell5.setCellValue(date);

Sheet sheet2 = workbook.getSheetAt(1);

int n = 0;

int price = 0;

for (int i = 5; i < writeOff.getListSize() + 5; i++){

Row row5 = sheet2.getRow(i);

Cell cell6 = row5.createCell(1);

cell6.setCellValue(writeOff.getWriteOffItems().get(n).getItem().getName());

Cell cell7 = row5.createCell(18);

cell7.setCellValue(writeOff.getWriteOffItems().get(n).getItem().getId());

Cell cell8 = row5.createCell(37);

cell8.setCellValue(writeOff.getWriteOffItems().get(n).getQuantity());

Cell cell9 = row5.createCell(57);

cell9.setCellValue(writeOff.getWriteOffItems().get(n).getItem().getPrice());

Cell cell10 = row5.createCell(66);

price += writeOff.getWriteOffItems().get(n).getQuantity() \* writeOff.getWriteOffItems().get(n).getItem().getPrice();

cell10.setCellValue(writeOff.getWriteOffItems().get(n).getQuantity() \* writeOff.getWriteOffItems().get(n).getItem().getPrice());

n++;

if (i < 5 + writeOff.getListSize() - 1){

int insertRowIndex = i + 1;

sheet2.shiftRows(insertRowIndex, sheet2.getLastRowNum(), 1);

Row newRow = sheet2.createRow(insertRowIndex);

copyRow(row5, newRow, sheet);

}

if (i == 5 + writeOff.getListSize() - 1){

Row row6 = sheet2.getRow(i + 1);

Cell cell11 = row6.getCell(66);

cell11.setCellValue(price);

} }

File outputDirectory = ResourceUtils.*getFile*("classpath:static/documents");

if (!outputDirectory.exists()) {

outputDirectory.mkdirs();

}

String timestamp = String.*valueOf*(System.*currentTimeMillis*());

String newFileName = "generated\_torg\_16\_" + timestamp + ".xlsx";

File outputFile = new File(outputDirectory, newFileName);

String filePath = outputFile.getAbsolutePath();

OutputStream outputStream = new FileOutputStream(outputFile);

workbook.write(outputStream);

inputStream.close();

outputStream.close();

System.*out*.println("Файл ТОРГ-16 сгенерирован и сохранен");

return filePath;

} catch (Exception e) {

e.printStackTrace();

} return "error";

}

public String generateTorg1(Receipt receipt) { try {

Resource resource = new ClassPathResource("static/documents/blank-torg-1.xlsx");

File file = resource.getFile();

InputStream inputStream = new FileInputStream(file);

Workbook workbook = new XSSFWorkbook(inputStream);

Sheet sheet = workbook.getSheetAt(0);

Row row = sheet.getRow(6);

Cell cell = row.getCell(4);

cell.setCellValue("OOO АВТО-ПРО");

Row row2 = sheet.getRow(8);

Cell cell2 = row2.getCell(4);

cell2.setCellValue(receipt.getStock().getName());

Row row3 = sheet.getRow(19);

Cell cell3 = row3.getCell(53);

cell3.setCellValue(receipt.getId());

LocalDateTime today = LocalDateTime.*now(); // дата в нужном формате*

Month month = today.getMonth();

int monthNumber = month.getValue();

String date = today.getDayOfMonth() + "." + String.*format*("%02d", monthNumber) + "."

+ today.getYear();

Row row4 = sheet.getRow(19);

Cell cell4 = row4.getCell(64);

cell4.setCellValue(date);

Row row5 = sheet.getRow(23);

Cell cell5 = row5.getCell(26);

cell5.setCellValue(receipt.getStock().getName());

Row row6 = sheet.getRow(24);

Cell cell6 = row6.getCell(64);

cell6.setCellValue(today.getDayOfMonth());

Cell cell7 = row6.getCell(74);

cell7.setCellValue(String.*format*("%02d", monthNumber));

Cell cell8 = row6.getCell(97);

cell8.setCellValue(today.getYear());

Row row7 = sheet.getRow(25);

Cell cell9 = row7.getCell(39);

cell9.setCellValue("накладная № " + receipt.getId() + " от " +

receipt.getCreatedAt().getDayOfMonth() + "." + String.*format*("%02d", monthNumber) + "." +

receipt.getCreatedAt().getYear());

Row row8 = sheet.getRow(33);

Cell cell10 = row8.getCell(22);

cell10.setCellValue(receipt.getSupplier());

Row row9 = sheet.getRow(61);

Cell cellll = row9.getCell(3);

cell11.setCellValue(date);

Cell cell12 = row9.getCell(24);

cell12.setCellValue(date);

Cell cell13 = row9.getCell(89);

cell13.setCellValue(date);

Sheet sheet2 = workbook.getSheetAt(l);

int price = 0;

if (receipt.getListSize() == 1){

Row row10 = sheet2.getRow(6);

Cell cell14 = row10.getCell(2);

cell14.setCellValue(receipt.getReceiptItems().get(0).getItem().getName());

Cell cell15 = row10.getCell(24);

cell15.setCellValue(receipt.getReceiptItems().get(0).getItem().getId());

Cell cell16 = row10.getCell(55);

cell16.setCellValue(receipt.getReceiptItems().get(0).getItem().getPrice());

Cell cell18= row10.getCell(73);

cell18.setCellValue(receipt.getReceiptItems().get(0).getQuantity());

Cell cell19 = row10.getCell(97);

price +=

receipt.getReceiptItems().get(0).getQuantity()\*receipt.getReceiptItems().get(0).getItem().getPric e();

cell19.setCellValue(price);

Row row11 = sheet2.getRow(8);

Cell cell20 = row11.getCell(97);

cell20.setCellValue(price);

} else {

int k = 1;

int n = 0;

for (int i = 6; i < 6 + receipt.getListSize() \* 2; i = i + 2) {

Row row10 = sheet2.getRow(i);

Cell number = row10.getCell(0);

number.setCellValue(k + ".");

k++;

Cell cell14 = row10.getCell(2);

cell14.setCellValue(receipt.getReceiptItems().get(n).getItem().getName());

Cell cell15 = row10.getCell(24);

cell15.setCellValue(receipt.getReceiptItems().get(n).getItem().getId());

Cell cell16 = row10.getCell(55);

cell16.setCellValue(receipt.getReceiptItems().get(n).getItem().getPrice());

Cell cell18= row10.getCell(73);

cell18.setCellValue(receipt.getReceiptItems().get(n).getQuantity());

Cell cell19 = row10.getCell(97);

price += receipt.getReceiptItems().get(n).getQuantity() \* receipt.getReceiptItems().get(0).getItem().getPrice();

cell19.setCellValue(receipt.getReceiptItems().get(n).getQuantity() \* receipt.getReceiptItems().get(O).getItem().getPrice());

n++;

Row row11 = sheet2.getRow(i + 1);

if (i < 6 + receipt.getListSize() \* 2 - 2){

int insertRowIndexl = i + 2;

int insertRowIndex2 = i + 3;

sheet2.shiftRows(insertRowIndex1, sheet2.getLastRowNum(), 1);

sheet2.shiftRows(insertRowIndex2, sheet2.getLastRowNum(), 1);

Row newRowl = sheet2.createRow(insertRowIndex1);

Row newRow2 = sheet2.createRow(insertRowIndex2);

copyRow(row10, newRowl, sheet2);

copyRow(row11, newRow2, sheet2);

}

if (i == 6 + receipt.getListSize() \* 2 - 2){

Row row12 = sheet2.getRow(i + 2);

Cell cell20 = row12.getCell(97);

cell20.setCellValue(price);

}

}

}

Sheet sheet3 = workbook.getSheetAt(2);

Row row13 = sheet3.getRow(27);

Cell cell21 = row13.getCell(2);

cell21.setCellValue(receipt.getSupplier());

Row row14 = sheet3.getRow(34);

Cell cell22 = row14.getCell(2);

cell22.setCellValue(today.getDayOfMonth());

Cell cell23 = row14.getCell(9);

cell23.setCellValue(String.*format*("%02d", monthNumber));

Cell cell24 = row14.getCell(27);

cell24.setCellValue(today.getYear());

File outputDirectory = ResourceUtils.*getFile*("classpath:static/documents");

if (!outputDirectory.exists()) {

outputDirectory.mkdirs();

}

String timestamp = String.*valueOf*(System.*currentTimeMillis*());

String newFileName = ,,generated\_blank-torg-1\_" + timestamp + ".xlsx";

File outputFile = new File(outputDirectory, newFileName);

String filePath = outputFile.getAbsolutePath();

OutputStream outputStream = new FileOutputStream(outputFile);

workbook.write(outputStream);

inputStream.close();

outputStream.close();

System.*out*.println("Файл ТОРГ-1 сгенерирован и сохранен");

return filePath;

} catch (Exception e) {

e.printStackTrace();

} return "error";

}

private void copyRow(Row sourceRow, Row targetRow, Sheet sheet) { targetRow.setHeight(sourceRow.getHeight());

for (int i = sourceRow.getFirstCellNum(); i < sourceRow.getLastCellNum(); i++) {

Cell sourceCell = sourceRow.getCell(i);

if (sourceCell != null) {

Cell targetCell = targetRow.createCell(i);

targetCell.setCellStyle(sourceCell.getCellStyle());

targetCell.setCellType(sourceCell.getCellType());

if (sourceCell.getCellType() == CellType.*STRING*) {

targetCell.setCellValue(sourceCell.getStringCellValue());

} else if (sourceCell.getCellType() == CellType.*NUMERIC*) {

targetCell.setCellValue(sourceCell.getNumericCellValue());

} else if (sourceCell.getCellType() == CellType.*BOOLEAN*) {

targetCell.setCellValue(sourceCell.getBooleanCellValue());

} else if (sourceCell.getCellType() == CellType.*ERROR*) { targetCell.setCellErrorValue(sourceCell.getErrorCellValue());

} else if (sourceCell.getCellType() == CellType.*FORMULA)* {

targetCell.setCellFormula(sourceCell.getCellFormula());

} }

}

for (int i = 0; i < sheet.getNumMergedRegions(); i++) {

CellRangeAddress mergedRegion = sheet.getMergedRegion(i);

if (mergedRegion.isInRange(sourceRow.getRowNum(), sourceRow.getRowNum())) { CellRangeAddress newMergedRegion = new CellRangeAddress( targetRow.getRowNum(), targetRow.getRowNum() + mergedRegion.getLastRow() -

mergedRegion.getFirstRow(),

mergedRegion.getFirstColumn(), mergedRegion.getLastColumn() );

sheet.addMergedRegion(newMergedRegion); }

}

}

@Override

public String generateTorg13(Transfer transfer) { try {

Resource resource = new ClassPathResource("static/documents/blank-torg-13.xlsx"); File file = resource.getFile();

InputStream inputStream = new FileInputStream(file);

Workbook workbook = new XSSFWorkbook(inputStream);

Sheet sheet = workbook.getSheetAt(0);

Row row = sheet.getRow(5);

CellStyle cellStyle = workbook.createCellStyle();

cellStyle.setAlignment(HorizontalAlignment.*CENTER*) ;

String value = "ООО АВТО-ПРО";

for (int column = 0; column <= 19; column++) {

Cell cell = row.createCell(column);

cell.setCellValue(value); cell.setCellStyle(cellStyle);

}

Row row2 = sheet.getRow(10);

Cell cell2 = row2.createCell(15);

cell2.setCellValue(transfer.getId());

LocalDateTime today = LocalDateTime.*now();*

Month month = today.getMonth();

int monthNumber = month.getValue();

String date = today.getDayOfMonth() + "." + String.*format*("%02d", monthNumber) + "."

+ today.getYear();

Row row3 = sheet.getRow(10);

Cell cell3 = row3.createCell(17);

cell3.setCellValue(date);

Row row4 = sheet.getRow(15);

Cell cell4 = row4.createCell(0);

cell4.setCellValue(transfer.getFromStock().getName());

Row row5 = sheet.getRow(15);

Cell cell5 = row5.createCell(6);

cell5.setCellValue(transfer.getToStock().getName());

int n = 0;

int price = 0;

for (int i = 21; i < transfer.getListSize() + 21; i++){

Row row6 = sheet.getRow(i);

Cell cell6 = row6.createCell(0);

cell6.setCellValue(transfer.getTransferItems().get(n).getItem().getName());

Row row6\_1 = sheet.getRow(i);

Cell cell6\_1 = row6\_1.createCell(5);

cell6\_1.setCellValue(transfer.getTransferItems().get(n).getItem().getId());

Cell cell7 = row6.createCell(15);

cell7.setCellValue(transfer.getTransferItems().get(n).getQuantity());

Cell cell8 = row6.createCell(20);

cell8.setCellValue(transfer.getTransferItems().get(n).getItem().getPrice());

Cell cell9 = row6.createCell(22);

price += transfer.getTransferItems().get(n).getQuantity() \* transfer.getTransferItems().get(n).getItem().getPrice();

cell9.setCellValue(transfer.getTransferItems().get(n).getQuantity() \* transfer.getTransferItems().get(n).getItem().getPrice());

n++;

if (i < 21 + transfer.getListSize() - 1){

int insertRowIndex = i + 1;

sheet.shiftRows(insertRowIndex, sheet.getLastRowNum(), 1);

Row newRow = sheet.createRow(insertRowIndex);

copyRow(row6, newRow, sheet);

}

if (i == 21 + transfer.getListSize() - 1){

Row row7 = sheet.getRow(i + 1);

Cell cell10 = row7.getCell(22);

cell10.setCellValue(price);

Row row8 = sheet.getRow(i + 2);

Cell cell11 = row8.getCell(22);

cell11.setCellValue(price);

}

}

File outputDirectory = ResourceUtils.*getFile*("classpath:static/documents");

if (!outputDirectory.exists()) { outputDirectory.mkdirs();

}

String timestamp = String.*valueOf*(System.*currentTimeMillis*());

String newFileName = "generated\_torg\_13\_" + timestamp + ".xlsx";

File outputFile = new File(outputDirectory, newFileName);

String filePath = outputFile.getAbsolutePath();

OutputStream outputStream = new FileOutputStream(outputFile);

workbook.write(outputStream);

inputStream.close();

outputStream.close();

System.*out*.println("Файл ТОРГ-13 сгенерирован и сохранен");

return filePath;

} catch (Exception e) {

e.printStackTrace();

} return "error";

}

@Override

public String generateNakladnaya(Receipt receipt) { try {

Resource resource = new ClassPathResource("static/documents/blank\_nakladnaya.xlsx");

File file = resource.getFile();

InputStream inputStream = new FileInputStream(file);

Workbook workbook = new XSSFWorkbook(inputStream);

Sheet sheet = workbook.getSheetAt(0);

Row row = sheet.getRow(2);

Cell cell = row.getCell(4);

cell.setCellValue("№"+ receipt.getId() + " от " +

receipt.getCreatedAt().getDayOfMonth() + " " + receipt.getCreatedAt().getMonth() + " " +

receipt.getCreatedAt().getYear());

Row row2 = sheet.getRow(4);

Cell cell2 = row2.getCell(3);

cell2.setCellValue(receipt.getSupplier());

Row row3 = sheet.getRow(6);

Cell cell3 = row3.getCell(3);

cell3.setCellValue("000 АВТО-ПРО");

if(receipt.getListSize() == 1){

Row row4 = sheet.getRow(14);

Cell cell4 = row4.getCell(1);

cell4.setCellValue(1);

Cell cell5 = row4.getCell(2);

cell5.setCellValue(receipt.getReceiptItems().get(0).getItem().getName());

Cell cell6 = row4.getCell(9);

cell6.setCellValue(receipt.getReceiptItems().get(0).getQuantity());

Cell cell7 = row4.getCell(11);

cell7.setCellValue(receipt.getReceiptItems().get(0).getItem().getPrice());

} else {

int n = 0;

int k = 1;

for (int i = 14; i < 14 + receipt.getListSize(); i++) {

Row row4 = sheet.getRow(i);

Cell cell4 = row4.getCell(1);

cell4.setCellValue(k);

Cell cell5 = row4.getCell(2);

cell5.setCellValue(receipt.getReceiptItems().get(n).getItem().getName());

Cell cell6 = row4.getCell(9);

cell6.setCellValue(receipt.getReceiptItems().get(n).getQuantity());

Cell cell7 = row4.getCell(11);

cell7.setCellValue(receipt.getReceiptItems().get(n).getItem().getPrice()); k++;

n++;

if (i < 14 + receipt.getListSize() - 1){

int insertRowIndex = i + 1;

sheet.shiftRows(insertRowIndex, sheet.getLastRowNum(), 1);

Row newRow = sheet.createRow(insertRowIndex);

copyRow(row4, newRow, sheet);

}

}

}

File outputDirectory = ResourceUtils.*getFile*("classpath:static/documents");

if (!outputDirectory.exists()) {

outputDirectory.mkdirs();

}

String timestamp = String.*valueOf*(System.*currentTimeMillis*());

String newFileName = "generated\_nakladnaya\_" + timestamp + ".xlsx";

File outputFile = new File(outputDirectory, newFileName);

String filePath = outputFile.getAbsolutePath();

OutputStream outputStream = new FileOutputStream(outputFile);

workbook.write(outputStream);

inputStream.close();

outputStream.close();

System.*out*.println("Файл Приходная накладная сгенерирован и сохранен");

return filePath;

} catch (Exception e) {

e.printStackTrace();

}

return "error";

}

}