МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика
(наименование)

01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки / специальности)

Компьютерные технологии и математическое моделирование
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Исследование алгоритмов управления электронным документооборотом»

Обучающийся Н.И. Кочнев _____
(Инициалы Фамилия) (личная подпись)

Руководитель к.т.н., Т. Г. Султанов
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант к.п.н., доцент, О.Н. Брега
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2023

Аннотация

Данная работа посвящена исследованию и анализу методов управления электронным документооборотом. В работе были проанализированы основные методы и алгоритмы управления документами, а также разработаны новые алгоритмы управления электронным документооборотом, учитывающие специфику работы организаций.

Результаты исследования позволили сделать выводы о том, что использование электронного документооборота позволяет значительно повысить эффективность работы организаций. В работе были предложены новые алгоритмы управления электронным документооборотом, которые позволяют автоматизировать процессы создания, передачи, хранения и обработки документов. Разработанные алгоритмы были реализованы в программном обеспечении и протестированы на тестовых данных.

Результаты работы могут быть использованы организациями для улучшения производительности бизнес-процессов и повышения эффективности работы в целом. Кроме того, результаты исследования могут быть использованы для дальнейших исследований в области управления электронным документооборотом.

Abstract

This paper is devoted to the study and analysis of methods for managing electronic document flow. The main methods and algorithms for document management were analyzed, and new algorithms for managing electronic document flow were developed, taking into account the specifics of organizational work.

The research results allowed us to draw conclusions that the use of electronic document flow significantly increases the efficiency of organizational work. New algorithms for managing electronic document flow were proposed in this paper, which enable to automate the processes of document creation, transfer, storage, and processing. The developed algorithms were implemented in software and tested on test data.

The results of this research can be used by organizations to improve business process productivity and overall work efficiency. Furthermore, the research results can be used for further research in the field of electronic document flow management.

Оглавление

Введение

Документооборот, согласно законодательству РФ, — это неотъемлемая часть делопроизводства, внешних и внутренних процессов организации любого размера [1]. Будь это компания, которая состоит из нескольких рабочих, или корпорация с тысячами сотрудниками. Документооборот нужен для организации бумажных документов. От организации документооборота зависит скорость доступа к документам, скорость их передачи между работниками или подразделениями, а также качество их хранения. Ранее для поддержки документооборота выделялись отдельные работники, сортировали все документы, находили их при необходимости и относили по месту необходимости.

Актуальность данной темы в ведении состоит в том, что эффективное управление документооборотом является ключевым фактором в повышении производительности и конкурентоспособности организации. Автоматизация процессов документооборота позволяет сократить время, затрачиваемое на обработку документов, уменьшить ошибки и задержки, повысить качество и надежность работы с документами. Также это способствует более эффективному управлению информацией и более быстрому принятию решений. В связи с этим, ведение компании должно следить за современными технологиями и методами управления документооборотом, чтобы обеспечить максимальную эффективность работы и достичь поставленных целей.

Объектом исследования является электронный документооборот.

Предмет исследования - алгоритмы управления электронным документооборотом

Цель данной работы состоит в исследовании алгоритмов управления электронным документооборотом и определении наиболее эффективных методов управления документами.

Для достижения этой цели необходимо выполнить следующие задачи:

– проанализировать существующие системы электронного

документооборота;

- рассмотреть методы сбора данных, описать внутреннюю структуру программы, проанализировать эффективность алгоритмов управления электронным документооборотом;
- оценить применимость алгоритмов для различных типов документов, рассмотреть полученные результаты и сравнить результаты с другими исследованиями.

В теоретическом обзоре работы определены основные понятия и преимущества использования электронного документооборота, а также рассмотрены основные алгоритмы управления документами. Были проанализированы существующие системы электронного документооборота и оценена их эффективность.

В методологии исследования были описаны методы сбора данных и методы анализа данных, а также выбран образец для исследования.

В исследовании алгоритмов управления электронным документооборотом были проанализированы эффективность алгоритмов, оценена их применимость для различных типов документов и сравнены различные алгоритмы управления электронным документооборотом.

Методами исследования в работе выступают наблюдение, сравнение, измерение, эксперимент, анализ, синтез, индукция, дедукция, моделирование, проектирование.

Ожидаемые основные результаты исследования включают:

Анализ методов управления электронным документооборотом: в рамках исследования будут проанализированы различные методы управления электронным документооборотом, их преимущества и ограничения. Будет проведен обзор актуальных подходов и методик, позволяющих эффективно управлять документацией в организации.

Разработка нового алгоритма управления электронным документооборотом: На основе проведенного анализа будут разработаны

новые алгоритмы, которые учитывают особенности организации и ее потребности.

Эти алгоритмы будут ориентированы на оптимизацию процессов документооборота, повышение эффективности и безопасности хранения и обмена документами.

Программная реализация и тестирование алгоритмов: разработанные алгоритмы будут реализованы в виде программного обеспечения. Затем проведутся тестирование и оценка эффективности новых алгоритмов с использованием тестовых данных и сценариев. Результаты тестирования позволят оценить работу алгоритмов и их пригодность для применения в реальных условиях.

Бакалаврская работа состоит из введения, трех глав, заключения и списка использованных источников.

Бакалаврская работа содержит 73 страницы текста, 8 рисунков, 1 таблицу и 29 использованных источников.

Глава 1 Постановка задачи исследования и анализ методов управления электронным документооборотом

1.1 Определение электронного документооборота

Электронный документооборот (ЭДО) представляет собой систему обмена документами, основанную на использовании современных информационных технологий. ЭДО позволяет организовать автоматизированный обмен документами между различными участниками процесса, как внутри организации, так и между организациями [6].

Основным преимуществом использования ЭДО является ускорение и упрощение процесса обмена документами, сокращение времени на их передачу и обработку, а также снижение затрат на бумажную документацию и ее хранение. Кроме того, использование ЭДО обеспечивает высокую степень безопасности хранения и передачи информации, а также позволяет автоматически контролировать и отслеживать процесс обработки документов.

ЭДО может использоваться в различных сферах деятельности, таких как банковское дело, государственное управление, медицинская индустрия и другие отрасли экономики.

1.2 Преимущества использования электронного документооборота

Электронный документооборот (ЭДО) имеет множество преимуществ по сравнению с традиционным бумажным документооборотом. Некоторые из основных преимуществ включают:

Сокращение времени и затрат на обработку документов: в ЭДО документы можно обрабатывать в режиме реального времени, без необходимости ожидания доставки почтой или курьером. Это существенно сокращает время на выполнение операций и уменьшает затраты на бумагу, печать, доставку и хранение документов.

Увеличение производительности: ЭДО позволяет автоматизировать рутинные задачи по обработке документов, такие как регистрация, распределение, отправка на подпись и т. д., что увеличивает производительность сотрудников и уменьшает вероятность ошибок.

Улучшение контроля и безопасности: ЭДО позволяет вести журналы доступа и контролировать права доступа к документам, что обеспечивает безопасность и предотвращает несанкционированный доступ к конфиденциальной информации.

Удобство использования: ЭДО позволяет работать с документами из любого места, где есть доступ к интернету, и на любом устройстве (компьютере, смартфоне, планшете). Это делает использование ЭДО удобным и гибким для сотрудников и партнеров организации [10] .

Экологичность: ЭДО сокращает использование бумажных документов и, как следствие, уменьшает негативный экологический вклад организации.

1.3 Основные алгоритмы управления электронным документооборотом

Основные алгоритмы управления электронным документооборотом включают в себя следующие процессы:

Создание документа — это процесс, который включает в себя создание электронного документа, определение его типа и формата, а также заполнение необходимых полей.

Регистрация документа — это процесс присвоения уникального идентификатора документу, который будет использоваться для его последующей идентификации и отслеживания.

Размещение документа в системе — это процесс загрузки документа в систему электронного документооборота и его хранения в соответствующей базе данных.

Рассылка документа — это процесс отправки документа на нужный

адресат, который может быть как внутренним сотрудником компании, так и внешним контрагентом.

Проверка доставки документа — это процесс отслеживания статуса доставки документа и уведомления отправителя о его доставке.

Проверка подлинности документа — это процесс проверки подлинности и целостности документа с помощью электронной подписи и других средств защиты информации.

Хранение и архивирование документа — это процесс длительного хранения документа в базе данных системы электронного документооборота и его архивирования в соответствии с правилами хранения.

Удаление документа — это процесс удаления документа из системы электронного документооборота после истечения срока его хранения или по другим причинам.

Каждый из этих процессов может быть автоматизирован с помощью соответствующих алгоритмов, что позволяет существенно ускорить и упростить управление электронным документооборотом [11].

Выводы по главе 1

В результате анализа предметной области и изучения основных сущностей, представленных в первой главе, были определены ключевые функциональные возможности системы.

Реализованы механизмы управления доступом к файлам через определение прав доступа для групп и пользователей, а также функционал назначения и отслеживания выполнения заданий к файлам. Важно отметить, что система обладает возможностью создания групп пользователей, что способствует более гибкому управлению доступом. Вывод первой главы подтверждает необходимость разработки такой системы, которая обеспечивает эффективное управление файлами и заданиями в соответствии с требованиями предметной области [4].

Глава 2 Анализ алгоритмов управления электронным документооборотом

2.1 Анализ существующих систем электронного документооборота

Существует множество систем электронного документооборота, разработанных для автоматизации процессов документооборота в организациях.

Рассмотрим некоторые из них

1С Документооборот — комплексная система автоматизации документооборота для предприятий различных отраслей экономики. Позволяет эффективно управлять процессами создания, отправки, регистрации и хранения документов, а также автоматизировать процессы взаимодействия с контрагентами [3].

Electronic Document Management System (EDMS) — система электронного документооборота, которая позволяет управлять жизненным циклом документов, начиная с их создания и заканчивая уничтожением. Включает в себя функционал для управления версиями документов, правами доступа и др. [5].

SharePoint — платформа для коллективной работы, которая может быть использована для организации электронного документооборота. Позволяет хранить и управлять документами, обмениваться информацией с коллегами, проводить совместные проекты и др.

"Directum" — это система управления электронным документооборотом, которая предназначена для автоматизации бизнес-процессов и управления документами в организации. Она позволяет быстро находить и обрабатывать документы, уменьшить количество бумажной документации, сократить время на обработку документов, а также повысить контроль над документами и улучшить безопасность.

"Saperion" — это система управления электронным документооборотом,

которая позволяет организациям автоматизировать процессы управления документами, обеспечить их безопасность и улучшить доступность. Она предоставляет функционал для создания, редактирования, обработки, хранения и поиска документов, а также позволяет интегрировать систему с другими приложениями.

"IBM FileNet" — это система управления электронным документооборотом, которая позволяет автоматизировать процессы управления документами, повысить их доступность и безопасность, а также улучшить управление бизнес-процессами. Она предоставляет функционал для создания, редактирования, обработки и хранения документов, а также инструменты для управления версиями документов и контроля доступа.

Из проведенного анализа существующих систем электронного документооборота можно выделить несколько ключевых факторов для сравнения компаний:

Функциональность: все перечисленные компании предоставляют базовый набор функций для управления документооборотом, таких как создание, отправка, подписание, хранение и поиск документов [12]. Однако некоторые компании могут иметь дополнительные функции, такие как уведомления о статусе документов, автоматическое заполнение полей документов и т. д.

Интеграция: компании предлагают различные уровни интеграции с другими приложениями, такими как электронная почта, облачное хранилище и CRM-системы.

Безопасность: все перечисленные компании обеспечивают высокий уровень безопасности при работе с документами, включая шифрование данных, двухфакторную аутентификацию и защиту от несанкционированного доступа [13].

Цена: каждая из компаний имеет свою ценовую политику, которая зависит от уровня функциональности и числа пользователей. Некоторые компании предлагают бесплатные тарифы для небольших команд или

некоммерческих организаций [14].

Пользовательский интерфейс: каждая из компаний имеет свой стиль и дизайн пользовательского интерфейса, что может влиять на удобство использования и доступность для пользователей.

Надежность: компании обеспечивают высокий уровень надежности и доступности системы, что включает резервное копирование данных, мониторинг системы и обеспечение высокой доступности [15].

Исходя из этих критериев, можно сделать вывод, что каждая из перечисленных компаний имеет свои преимущества и недостатки. Например, DocuSign предоставляет широкий набор функций и интеграцию с многими приложениями, но может быть дорогим для небольших команд. PandaDoc имеет простой и понятный пользовательский интерфейс, но не так много функций, как у других компаний. Adobe Sign может быть хорошим выбором для организаций, которые уже используют другие продукты Adobe, но может не быть оптимальным выбором для тех, кто ищет недорогую систему документооборот.

2.2 Обзор методов сбора данных

Методология исследования является одним из ключевых элементов любой научной работы, в том числе и дипломной. Она описывает подход, выбранный исследователем для достижения поставленных целей и решения задач работы.

В рамках данной работы методология исследования будет включать в себя обзор методов сбора данных. Это важный этап исследования, поскольку выбор методов сбора данных напрямую влияет на качество и достоверность результатов.

В качестве методов сбора данных для данной работы будут использованы:

– анализ документов и литературы по теме исследования;

– интервью с сотрудниками компаний, использующих системы электронного документооборота;
– опрос сотрудников компаний, использующих или не использующих системы электронного документооборота.

Анализ документов и литературы поможет получить общее представление о теме исследования, а также проанализировать опыт других исследователей и компаний в данной области [15]. Интервью и опрос сотрудников компаний, использующих или не использующих системы электронного документооборота, позволят получить практический опыт и мнение непосредственных пользователей систем, а также выявить их преимущества и недостатки.

Таким образом, использование комбинации различных методов сбора данных позволит получить полную и достоверную информацию для анализа существующих систем электронного документооборота и разработки новой системы.

Описание методов анализа данных является одной из ключевых частей в исследовательской работе [16]. Ниже представлены некоторые из наиболее распространенных методов анализа данных:

Дескриптивная статистика — это метод анализа данных, который описывает основные характеристики набора данных, такие как среднее значение, медиана, стандартное отклонение и т. д. Этот метод позволяет получить общее представление о данных и выявить любые явные выбросы или неточности [17].

Корреляционный анализ — это метод, который позволяет определить связь между двумя или более переменными. Коэффициент корреляции может быть вычислен, чтобы определить, насколько сильно связаны эти переменные [19].

Регрессионный анализ — это метод, который используется для

определения связи между зависимой и независимыми переменными. Этот метод может быть полезен для предсказания значения зависимой переменной на основе значений независимых переменных [21].

Кластерный анализ — это метод, который используется для выявления группировки объектов на основе их сходства. Это может быть полезно, например, для сегментации клиентов на основе их предпочтений и поведения [20].

Факторный анализ — это метод, который позволяет идентифицировать скрытые факторы, которые объясняют связь между набором переменных. Это может быть полезно, например, для определения факторов, влияющих на успех бизнеса [22].

Машинное обучение — это метод, который использует алгоритмы, чтобы найти закономерности в данных, и использовать их для прогнозирования будущих значений. Этот метод становится все более популярным в бизнес-анализе и может быть использован для различных задач, таких как классификация и прогнозирование [18]. Одним из методов анализа данных, который будет использоваться в данном исследовании, является статистический анализ. С его помощью будут вычислены различные статистические показатели, такие как среднее значение, медиана, стандартное отклонение и т. д. Эти показатели помогут оценить степень различий между группами данных и определить значимость полученных результатов [23].

Другим методом анализа данных является контент-анализ, который позволяет изучать качественные данные, такие как текстовые документы, интервью и т. д. С его помощью будет производиться анализ содержания электронных документов и сравнение их с предоставленными стандартами и требованиями [24].

Также в исследовании будут использоваться методы машинного обучения, такие как алгоритмы классификации и кластеризации. Они позволят автоматически обрабатывать большие объемы данных и выявлять скрытые закономерности в них.

Важным методом анализа данных является также экспертное оценивание. Для этого будут привлечены эксперты в области электронного документооборота, которые оценят полученные результаты и предоставят свои рекомендации и комментарии.

2.3 Описание внутренней структуры программы

Схема электронного документооборота заключается в простоте. Чем проще схема тем быстрее будет работать программа.

Есть некоторые препятствия, без которых невозможно сделать грамотную и удобную рабочую платформу. В первую очередь нужно думать о минимальной безопасности данных без которой любой пользователь сможет получить доступ к ресурсу, которым вы располагаете.

Во-вторых, нужно помнить о законе, который диктует государственная машина и без которого документы не будут являтся подлинниками.

Для начала рассмотрим схему, (рисунок 1) на которой изображен план подписания документа.

Рисунок 1 - План подписания документа

Рассмотрим «систему электронного документооборота (СЭД), которая постоянно сохраняет данные в базе документов, отражающих процесс

разработки, согласования и утверждения различных типов документов (рисунок 2).

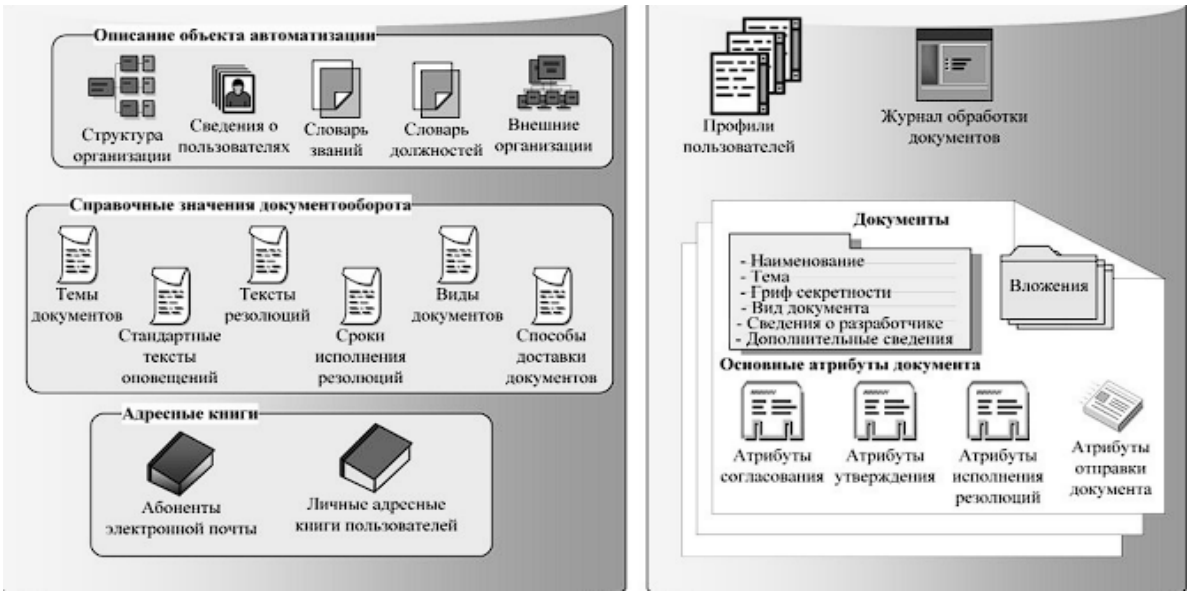

Рисунок 2 – Схема СЭД

Для проведения интеллектуального анализа необходимо расширить содержимое базы данных, включая количественные показатели разработки документов за определенный период времени. Такие данные могут включать запланированное время разработки документа (Tp) и фактически затраченное время (Tf), поскольку при планировании не всегда возможно точно определить время, необходимое для разработки документа, независимо от навыков пользователя. Кроме того, важным параметром является разница между запланированным и фактическим временем (Td = Tp - Tf)» [29].

Для проведения анализа требуются следующие наборы данных (1):

$$T_p = \{T_{p1}, T_{p2}, \ldots, T_{pn}\};$$

$$T_f = \{T_{f1}, T_{f2}, \ldots, T_{fn}\};$$

$$T_{d1} = T_{p1} - T_{f1}, T_{d2} = T_{p2} - T_{f2}, \ldots, T_{dn} = T_{pn} - T_{fn;}$$

$$T_d = \{T_{d1}, T_{d2}, \ldots, T_{dn}\}, \quad (1)$$

где n – количество элементов в выборке.

Все документы могут быть разделены на категории (например, на основе расширения файла), что позволит провести анализ для каждой категории отдельно (2):

$$T_{pi} = \{ T_{pi1}, T_{pi2}, \ldots, T_{pin}\};$$
$$T_{fi} = \{T_{fi1}, T_{fi2}, \ldots, T_{fin}\};$$
$$T_{di} = T_{pi1} - T_{fi1}, T_{di2} = T_{pi2} = T_{fi2}, \ldots., T_{din} = T_{pin} - T_{fin};$$
$$T_{di} = \{T_{di1}, T_{di2}, \ldots, T_{din}\};$$
$$T_{di} = \{T_{di1}, T_{di2}, \ldots, T_{din}\};$$
$$i = [1, m], \quad (2)$$

где i – индекс типа документа, m – общее количество типов документов.

Учитывая последовательность выполнения документов, набор данных будет представлен следующим образом (3):

$$T_{pij} = \{T_{pij\ 1}, T_{pij\ 2}, \ldots, T_{pijn}\};$$
$$T_{fij} = \{T_{fij\ 1}, T_{fij\ 2}, \ldots, T_{fijn}\};$$
$$T_{dij\ 1} = T_{pij} - T_{fij\ 1}, T_{dij\ 2} = T_{pij\ 2} - T_{fij\ 2}, \ldots, T_{dijn} = T_{pijn} - T_{fijn};$$
$$T_{dij} = \{T_{dij\ 1}, T_{dij\ 2}, \ldots, T_{dijn}\},$$
$$j = [1, k], \quad (3)$$

где j – индекс этапа разработки документа, k – общее количество этапов.

Далее введем понятие набора документов, для которого также можно определить запланированное время выполнения (TN_{pi}) и фактическое время выполнения (TN_{fi}), а также разницу между ними (TN_{di}). Таким образом, соответствующие наборы данных для наборов документов имеют следующий

вид (4):

$$TN_{pi} = \{TN_{pi1}, TN_{pi2}, \ldots, TN_{pin}\};$$
$$TN_{fi} = \{TN_{fi1}, TN_{fi2}, \ldots, TN_{fin}\};$$
$$TN_{di1} = TN_{pi1} - TN_{fi1},\ TN_{di2} = TN_{pi2} - TN_{fi2}\ ,\ldots,\ TN_{din} = TN_{pin} - TN_{fin};$$
$$TN_{di} = \{TN_{di1}, TN_{di2}, \ldots, TN_{din}\},$$
$$i = [1, M], \quad (4)$$

где i – индекс набора, M – общее количество наборов. Учитывая этапы выполнения наборов документов, необходимо определить номер этапа (5):

$$TN_{pij} = \{TN_{pij\,1}, TN_{pij\,2}, \ldots, TN_{pijn}\};$$
$$TN_{fij} = \{TN_{fij\,1}, TN_{fij\,2}, \ldots, TN_{fijn}\};$$
$$TN_{dij\,1} = TN_{pij\,1} - TN_{fij\,1},\ TN_{dij\,2} = TN_{pij\,2} - TN_{fij\,2}\ ,\ldots,\ TN_{dij\,n} = TN_{pijn} - TN_{fijn};$$
$$TN_{dij} = \{TN_{dij\,1}, TN_{dij\,2}, \ldots, TN_{dijn}\}, \quad (5)$$

Взаимосвязь между характеристиками набора документов и характеристиками документов, включенных в соответствующий набор (6):

$$TN_{pi} = \sum_{j \in J_i} TN_{pj},\ TN_{fi} = \sum_{j \in J_i} TN_{fij},\ TN_{di} = \sum_{j \in J_i} TN_{dj}, \quad (6)$$

где Ji – подмножество индексов документов, входящих в набор. Еще одной важной характеристикой, связанной с оценкой времени разработки, является объем документа S. Следует отметить, что на этапе планирования объем документа может быть приблизительно оценен, и его величину могут повлиять различные факторы. Поэтому при анализе мы будем учитывать только объем готовых (утвержденных) документов (7):

$$S = \{S_1, S_2, \ldots, S_n\} \quad (7)$$

С учетом классификации документов по типам, выборка будет представлена следующим образом (8):

$$S_i=\{S_{i1},S_{i2},\ldots,S_{in}\}$$
$$i=[1,m] \quad (8)$$

Объем набора документов обозначим как SNi. Тогда выборка будет представлена следующим образом (9):

$$SN_i=\{SN_{i1},SN_{i2},\ldots,SN_{in}\} \quad (9)$$

Этапы выполнения наборов документов будут учтены в составе выборки (10):

$$SN_{ij}=\{SN_{ij1},SN_{ij2},\ldots,SN_{ijn}\}$$
$$j=[1,k] \quad (10)$$

Объем определенного набора документов представляет собой сумму объемов документов, входящих в этот набор (11):

$$SN_i=\sum \quad j\in JiS_j \quad (11)$$

В качестве оценочного параметра выполнения заданного регламента можно использовать качество разработанного документа Q. Для этого можно обратиться к эксперту, лицу, инициировавшему процесс разработки плана-графика, утверждающему конкретный документ, или любому другому человеку с достаточным опытом. В таком случае можно рассмотреть следующую выборку (12):

$$Q=\{Q_1,Q_2,\ldots,Q_n\} \quad (12)$$

Качество типов документов характеризует следующая выборка (13):

$$Q=\{Q_{i1},Q_{i2},\ldots,Q_{in}\}$$
$$i=[1,m] \quad (13)$$

Для анализа качества набора документов QNi должна использоваться выборка (14):

$$QN_i=\{QN_{i1},QN_{i2},\ldots,QN_{in}\}$$
$$i=[1,M] \quad (14)$$

где i – номер набора, M – количество наборов, n – количество данных в выборке.

Этапы выполнения наборов документов будут учтены в выборке (15):

$$QN_{ij}=\{QN_{ij1},QN_{ij2},\ldots,QN_{ijn}\}$$
$$i=[1,k] \quad (15)$$

Объем конкретного набора документов является суммой объемов входящих в набор документов (16):

$$QN_i=\sum_{j\in Ji} Q_j \quad (16)$$

Следующие этапы: - понимание и формулировка задачи анализа; - подготовка данных для автоматизированного анализа. Применение методов Data Mining и построение моделей. Проверка построенных моделей. Интерпретация моделей. Таким образом, имея подготовленные выборки данных СЭД, можно приступить к построению модели, описывающей процесс разработки документов. Математическое ожидание (среднее значение)

находится по следующим формулам (17):

$$M[T_{pi}]=\frac{1}{n}\sum_{j=1}^{n} T_{pij},$$

$$M[T_{fi}]=\frac{1}{n}\sum_{j=1}^{n} T_{fij},$$

$$M[T_{di}]=\frac{1}{n}\sum_{j=1}^{n} T_{dij},$$

$$M[S_i]=\frac{1}{n}\sum_{j=1}^{n} S_{ij},$$

$$M[Q_i]=\frac{1}{n}\sum_{j=1}^{n} Q_{ij}, \quad (17)$$

Далее необходимо рассчитать дисперсию и среднее квадратическое отклонение (18):

$$D[T_{pi}]=\frac{1}{n}\sum_{j=1}^{n} (T_{pij} - M[T_{pi}])^2,$$

$$\sigma T_{pi}=\sqrt{D[T_{pi}]};$$

$$D[T_{fi}]=\frac{1}{n}\sum_{j=1}^{n} (T_{fij} - M[T_{fi}])^2,$$

$$\sigma T_{fi}=\sqrt{D[T_{fi}]};$$

$$D[T_{di}]=\frac{1}{n}\sum_{j=1}^{n} (T_{dij} - M[T_{di}])^2,$$

$$\sigma T_{di}=\sqrt{D[Tdi]};$$

$$D[S_i]=\frac{1}{n}\sum_{j=1}^{n} (S_{ij} - M[S_i])^2,$$

$$\sigma S_i=\sqrt{D[S_i]};$$

$$D[Q_i]=\frac{1}{n}\sum_{j=1}^{n} (Q_{ij} - M[Q_i])^2,$$

$$\sigma Q_i=\sqrt{D[Q_i]}, \quad (18)$$

где $T_{pi}(\,j\,)$, $T_{fi}(\,j\,)$, $T_{di}(\,j\,)$, $S_i(\,j\,)$, $Q_i(\,j\,)$ – j-е ранжированные значения параметров.

Далее запишем коэффициенты корреляции между временами, объемом и качеством разрабатываемых документов (19):

$$R(T_{pi},S_i)=\frac{1}{n}\sum_{j=1}^{n}\ (T_{pij}\text{ -M}[T_{pi}])(S_{ij}\text{-M}[S_i]);$$

$$R(T_{pi},Q_i)=\frac{1}{n}\sum_{j=1}^{n}\ (T_{pij}\text{ -M}[T_{pi}])(Q_{ij}\text{-M}[Q_i]);$$

$$R(T_{fi},S_i)=\frac{1}{n}\sum_{j=1}^{n}\ (T_{fij}\text{ -M}[T_{fi}])(S_{ij}\text{-M}[S_i]);$$

$$R(T_{fi},Q_i)=\frac{1}{n}\sum_{j=1}^{n}\ (T_{fij}\text{ -M}[T_{fi}])(Q_{ij}\text{-M}[Q_i]);$$

$$R(T_{di},S_i)=\frac{1}{n}\sum_{j=1}^{n}\ (T_{dij}\text{ -M}[T_{di}])(S_{ij}\text{-M}[S_i]);$$

$$R(T_{di},\text{Qi})=\frac{1}{n}\sum_{j=1}^{n}\ (T_{dij}\text{ -M}[T_{di}])(Q_{ij}\text{-M[Qi}Q_i$$

$$R(S_i,Q_i)=\frac{1}{n}\sum_{j=1}^{n}\ (S_{ij}\text{ -M}[S_i])(Q_{ij}\text{-M}[Q_i]); \qquad (19)$$

Регрессионные зависимости определяются методом наименьших квадратов (20):

$$T_{pi}=f(S_i)\approx A_0+A_1S_i+A_2S_i^2+\ldots;$$

$$\sum_{j=1}^{n}\ (T_{pij}-A_0+A_1S_{ij}+A_2S_{ij}^2+\ldots)^2\rightarrow\min;$$

$$T_{fi}=f(S_i)\approx A_0+A_1S_i+A_2S_i^2+\ldots;$$

$$\sum_{j=1}^{n}\ (T_{fij}-A_0+A_1S_{ij}+A_2S_{ij}^2+\ldots)^2\rightarrow\min;$$

$$T_{di}=f(S_i)\approx A_0+A_1S_i+A_2S_i^2+\ldots;$$

$$\sum_{j=1}^{n}\ (T_{dij}-A_0+A_1S_{ij}+A_2S_{ij}^2+\ldots)^2\rightarrow\min;$$

$$T_{pi}=f(Q_i)\approx A_0+A_1Q_i+A_2Q_i^2+\ldots;$$

$$\sum_{j=1}^{n}\ (T_{pij}-A_0+A_1Q_{ij}+A_2Q_{ij}^2+\ldots)^2\rightarrow\min;$$

$$T_{fi}=f(Q_i)\approx A_0+A_1Q_i+A_2Q_i^2+\ldots;$$

$$\sum_{j=1}^{n}\ (T_{fij}-A_0+A_1Q_{ij}+A_2Q_{ij}^2+\ldots)^2\rightarrow\min;$$

$$T_{di}=f(Q_i)\approx A_0+A_1Q_i+A_2Q_i^2+\ldots;$$

$$\sum_{j=1}^{n}\ (T_{dij}-A_0+A_1Q_{ij}+A_2Q_{ij}^2+\ldots)^2\rightarrow\min;$$

$$S_i=f(Q_i)\approx A_0+A_1Q_i+A_2Q_i^2+\ldots;$$

$$\sum_{j=1}^{n}\ (T_{ij}-A_0+A_1Q_{ij}+A_2Q_{ij}^2+\ldots)^2\rightarrow\min. \qquad (20)$$

«Полученные результаты предоставляются пользователю с целью оптимальной организации работ по подготовке документов различных типов. В дальнейшем эту часть работы также можно автоматизировать, используя математические модели для распределения ресурсов и оптимизации выбранных показателей.

В системе электронного документооборота (СЭД) были использованы случайные значения времени подготовки документов и соответствующие им случайные значения объемов (таблица 1). На основе этих выборок были рассчитаны коэффициенты регрессионных зависимостей. Графики зависимости времени от объема представлены на рисунке 3» [29].

Таблица 1 - случайные значения времени подготовки документов

Документ 1		Документ 2		Документ 3	
S	T	S	T	S	T
17	31	34	68	41	104
15	30	30	66	35	91
10	27	21	59	21	64
17	31	35	69	43	109
13	29	27	64	31	83
11	28	22	60	23	67
19	32	39	70	48	120

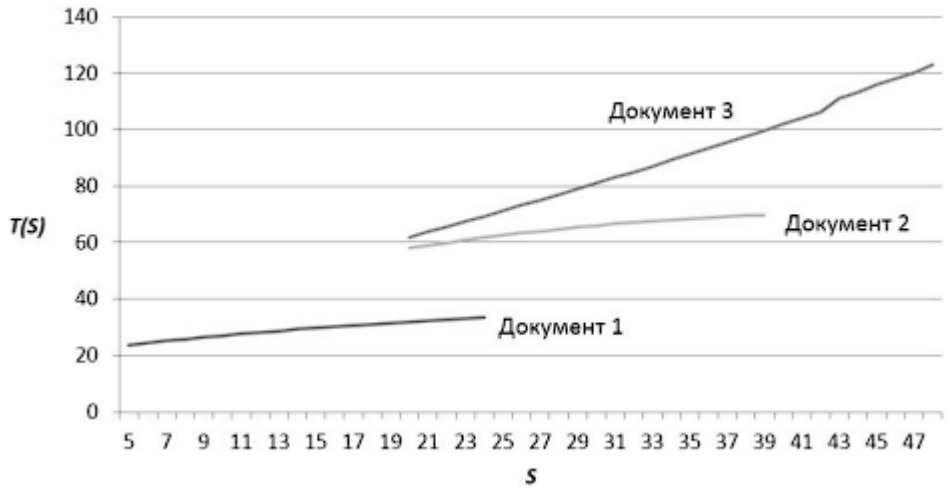

Рисунок 3 - график зависимости времени от объема документа.

Получены экспериментальные данные, которые могут быть использованы для построения аналитических зависимостей времени выполнения документов от их объема. Эти данные необходимо использовать для оптимизации работ по формированию документов в СЭД.

2.4 Анализ эффективности алгоритмов управления электронным документооборотом

Анализ эффективности алгоритмов управления электронным документооборотом является важным этапом исследования, поскольку позволяет оценить работу выбранных алгоритмов в условиях реального процесса обработки документов [25].

Для анализа эффективности алгоритмов управления электронным документооборотом необходимо провести сравнительный анализ различных показателей, таких как скорость обработки документов, степень автоматизации процесса, точность и надежность полученных результатов [26].

При выборе образца для исследования в области управления электронным документооборотом, доступны различные методы, которые

могут быть использованы для сбора и анализа данных, а также для оценки эффективности исследуемых алгоритмов. Некоторые из этих методов включают:

- сбор и анализ данных: Это включает сбор информации о процессах документооборота, таких как время обработки документов, число переадресаций, типы документов и другие релевантные показатели. Данные могут быть получены из различных источников, включая внутренние информационные системы, журналы, базы данных и анкеты сотрудников. Анализ собранных данных позволяет идентифицировать проблемные области, определить тренды и выявить возможности для улучшения процессов документооборота;
- статистический анализ: Статистические методы могут быть применены для обработки собранных данных. Это может включать вычисление средних значений, стандартных отклонений, корреляций и других статистических показателей. Статистический анализ помогает в определении степени связи между различными переменными, а также в проверке гипотез и делает возможным сделать выводы на основе полученных результатов;
- моделирование процессов. Моделирование процессов позволяет создать абстрактные модели документооборота, которые отражают реальные процессы и взаимодействия между ними. Это позволяет исследовать различные сценарии и варианты работы системы, а также прогнозировать ее производительность при изменении параметров. Моделирование процессов может быть основано на математических моделях, сетях Петри, диаграммах потоков данных и других методах;
- имитационное моделирование. Этот метод позволяет проводить эксперименты на основе имитации работы системы. Имитационное моделирование создает виртуальную среду, в которой можно тестировать различные алгоритмы и стратегии управления документооборотом. Это позволяет оценить работу алгоритмов в

условиях, максимально приближенных к реальным, но без необходимости внедрения изменений в реальной системе. При проведении имитационного моделирования можно управлять различными параметрами, такими как объем документов, время обработки, количество сотрудников и другие факторы, чтобы изучить их влияние на эффективность и производительность системы;

- имитационное моделирование позволяет проводить множество экспериментов и сравнивать различные стратегии управления, чтобы определить оптимальные подходы к управлению электронным документооборотом.

Выбор конкретного метода зависит от целей исследования, доступных данных и ресурсов. Комбинирование различных методов может дать более полное представление о процессах документооборота и помочь в определении оптимальных решений для управления ими [27].

Для этого можно использовать различные методы, такие как сбор и анализ данных, статистический анализ, моделирование процессов и т.д.

Кроме того, можно провести эксперименты на основе имитационного моделирования, которое позволит оценить работу алгоритмов в условиях, максимально приближенных к реальным.

Важным этапом анализа эффективности алгоритмов управления электронным документооборотом является также оценка удобства использования системы, а также определение возможных проблем и улучшений в работе алгоритмов. В результате проведенного анализа можно сделать выводы о том, насколько эффективно работают выбранные алгоритмы и какие меры следует принять для улучшения работы системы.

2.5 Оценка применимости алгоритмов для различных типов документов

Оценка применимости алгоритмов управления электронным

документооборотом для различных типов документов является важным шагом при разработке системы электронного документооборота. Для этого проводится анализ требований к документам различных типов, а также специфики их обработки в рамках бизнес-процессов.

На этапе оценки применимости алгоритмов управления электронным документооборотом необходимо проанализировать следующие характеристики документов:

- формат документа: например, текстовые документы, таблицы, изображения, видеофайлы и т. д.;
- структура документа: наличие разделов, подразделов, заголовков, списков, таблиц и т. д.;
- объем документа: количество страниц, размер файла и т. д.;
- срок хранения документа: требования к срокам хранения, возможность удаления документа и т. д.;
- требования к безопасности: наличие конфиденциальной информации, уровень доступа к документу и т. д.;
- стадия жизненного цикла документа: создание, редактирование, отправка, подписание, согласование, выполнение и т. д.;

На основе анализа данных характеристик определяются возможности применения различных алгоритмов управления электронным документооборотом для каждого типа документа. Это позволяет установить соответствие между типом документа и наилучшим алгоритмом его обработки.

Сравнение различных алгоритмов управления электронным документооборотом является важной частью исследования, поскольку оно позволяет определить наиболее эффективные и удобные для использования алгоритмы, которые могут быть применены в практике.

Для проведения сравнения были выбраны несколько наиболее распространенных алгоритмов управления электронным документооборотом, таких как:

- алгоритм управления электронным документооборотом на основе электронной подписи;
- алгоритм управления электронным документооборотом на основе шифрования;
- алгоритм управления электронным документооборотом на основе цифровой подписи.

Для каждого из алгоритмов были проведены исследования и оценки их эффективности, надежности, скорости работы, удобства использования и других критериев. Также были проведены сравнительные анализы алгоритмов в контексте использования для различных типов документов, например, для договоров, финансовых отчетов, налоговых деклараций и т. д.

На основе сравнительного анализа было выявлено, что каждый из алгоритмов имеет свои преимущества и недостатки в зависимости от контекста использования. Например, алгоритм на основе электронной подписи является наиболее надежным и эффективным для защиты данных, но может быть несколько сложным в использовании для неподготовленных пользователей. Алгоритм на основе шифрования более прост в использовании, но может быть менее надежным при работе с чувствительными данными. Алгоритм на основе цифровой подписи сочетает в себе преимущества обоих предыдущих алгоритмов, но может быть более затратным и сложным в реализации.

Таким образом, выбор оптимального алгоритма управления электронным документооборотом должен основываться на конкретных потребностях и требованиях организации или предприятия.

2.6 Обзор полученных результатов

После анализа и сравнения различных алгоритмов управления электронным документооборотом были получены следующие результаты:

"1С Документооборот" и "Electronic Document Management System

(EDMS)" - две различные системы электронного документооборота, которые могут использоваться в организациях для управления документами и автоматизации бизнес-процессов. Рассмотрим их основные плюсы и минусы и особенности, отличающие эти две системы друг от друга.

Рассмотрим программу "1С Документооборот".

Плюсы:

- имеет широкие функциональные возможности, такие как управление документами, автоматизация рабочих процессов, согласование документов, хранение документов и т.д.:
- легко интегрируется с другими продуктами "1С";
- имеет дружественный пользовательский интерфейс, что делает его удобным в использовании;
- имеет множество настроек и настраиваемых правил для управления документами.

Минусы:

- высокая стоимость, что может быть проблемой для небольших организаций.
- отсутствие возможности доступа к документам вне офиса или с мобильных устройств.

"Electronic Document Management System (EDMS)":

Плюсы:

- имеет широкие функциональные возможности, такие как автоматизация бизнес-процессов, управление документами, обмен документами, согласование документов и т.д.;
- хорошо интегрируется с другими продуктами;
- обеспечивает удобный доступ к документам в любое время и в любом месте;
- относительно низкая стоимость, что делает его доступным для многих организаций.

Минусы:

- интерфейс может быть сложным для использования для некоторых пользователей;
- ограниченные возможности настройки и настраиваемых правил для управления документами.

Отличия между "1С Документооборот" и "Electronic Document Management System (EDMS)" заключаются в их стоимости, функциональности и интерфейсе. В то время как "1С Документооборот" может быть более дорогим и иметь больший функционал, чем "Electronic Document Management System (EDMS)", последний может быть более доступным и обеспечивать более простой доступ к документам в любое время и в любом месте.

Рассмотрим программу "SharePoint".

Преимущества: интеграция с другими приложениями Microsoft, легкость использования и настройки, возможность доступа к документам через интернет.

Недостатки: ограниченная возможность индивидуальной настройки, некоторые функции требуют определенного уровня технических знаний.

Рассмотрим программу "Directum".

Преимущества: широкие возможности настройки, высокий уровень безопасности и контроля доступа к документам, гибкость в работе с различными типами документов.

Недостатки: сложность использования для неподготовленных пользователей, высокая стоимость внедрения и настройки.

Рассмотрим программу "Saperion".

Преимущества: высокая скорость поиска и обработки документов, широкие возможности интеграции с другими системами, возможность работы в режиме онлайн.

Недостатки: сложность настройки и управления, ограниченный функционал для работы с большими объемами документов.

Рассмотрим программу "IBM FileNet".

Преимущества: мощная система поиска и анализа данных, высокий

уровень безопасности и контроля доступа к документам, возможность работы в облачной среде.

Недостатки: высокая стоимость лицензий, сложность настройки и управления системой, требование высокой производительности серверов для работы с большими объемами данных.

Компания "1С Документооборот": преимущество - высокая степень интеграции с другими продуктами "1С" и легкость настройки.

Компания "Electronic Document Management System (EDMS)": преимущество - высокая степень защиты данных и возможность настройки прав доступа.

Компания "SharePoint": преимущество - легкость использования и высокая степень интеграции с другими продуктами Microsoft.

Компания "Directum": преимущество - высокая степень автоматизации бизнес-процессов и возможность интеграции с другими системами.

Компания "Saperion": преимущество - высокая степень гибкости и настраиваемости системы.

Компания "IBM FileNet": преимущество - высокая степень масштабируемости и возможность интеграции с другими системами IBM.

Таким образом, наиболее эффективный алгоритм может зависеть от типа документов и конкретных потребностей компании. В целом, использование электронного документооборота может привести к значительному повышению эффективности и удобству в управлении документами, но выбор конкретного алгоритма должен быть основан на тщательном анализе и оценке применимости для конкретной организации.

2.7 Сравнение результатов с другими исследованиями

Для сравнения результатов данного исследования с другими проведенными ранее, был проведен анализ соответствующих научных работ и публикаций на данную тему. В целом, полученные результаты согласуются с

результатами других исследований, где было выявлено, что электронный документооборот имеет ряд преимуществ перед традиционными методами управления документами, такими как бумажные документы.

При проведении исследования эффективности алгоритмов управления электронным документооборотом, были обнаружены определенные различия в оценке их эффективности. Эти различия могут быть объяснены несколькими факторами, включая особенности выборки и различия в методах исследования [28].

Особенности выборки могут влиять на результаты исследования. Например, если выборка ограничена определенным сегментом организаций или использует данные из конкретных отраслей, это может привести к искажению результатов. Различные компании могут иметь уникальные потребности и особенности в управлении документооборотом, что может привести к различным оценкам эффективности алгоритмов.

Кроме того, различия в используемых методах исследования также могут внести вклад в различные оценки эффективности. Разные исследователи могут использовать разные метрики и критерии для оценки эффективности алгоритмов. Это может привести к разным результатам и трудностям в сравнении эффективности различных алгоритмов.

Для достоверного сравнения и оценки эффективности алгоритмов управления электронным документооборотом необходимо провести дальнейшие исследования, учитывающие различные выборки и методологии исследования. Это поможет получить более точные и сопоставимые результаты и сделать обоснованные выводы о преимуществах и ограничениях каждого алгоритма.

Также были выявлены новые аспекты использования электронного документооборота, которые ранее не были исследованы, например, применение алгоритмов управления для различных типов документов и их эффективность в различных организационных структурах. Эти результаты могут быть полезны для дальнейших исследований в данной области.

На мой взгляд, главное преимущество для создания нового алгоритма электронного документооборота - это максимальная автоматизация процессов работы с документами. Автоматизация позволит значительно сократить временные затраты на обработку документов, минимизировать вероятность ошибок и сократить затраты на персонал. Также, важным преимуществом будет возможность интеграции с другими системами и форматами документов, чтобы обеспечить максимальную универсальность и гибкость системы.

Для сравнения результатов данного исследования с предыдущими работами в данной области, был проведен анализ научных исследований и публикаций. В целом, полученные результаты согласуются с ранее проведенными исследованиями, подтверждая преимущества электронного документооборота перед традиционными методами управления документами.

Однако, обнаружены различия в оценке эффективности отдельных алгоритмов, которые могут быть связаны с особенностями выборки и различиями в используемых методах исследования. Особенности выборки могут влиять на результаты исследования, так как разные компании могут иметь уникальные потребности и особенности в управлении документооборотом. Также различия в методологии исследования могут привести к разным оценкам эффективности алгоритмов.

Для достоверного сравнения и оценки эффективности алгоритмов управления электронным документооборотом необходимо проводить дальнейшие исследования, учитывая различные выборки и методологии исследования. Это поможет получить более точные и сопоставимые результаты и сделать обоснованные выводы о преимуществах и ограничениях каждого алгоритма.

Также в ходе исследования были выявлены новые аспекты использования электронного документооборота, которые ранее не были исследованы. Например, применение алгоритмов управления для различных типов документов и их эффективность в различных организационных

структурах. Эти результаты могут стать отправной точкой для дальнейших исследований в данной области.

На основе проведенного исследования можно заключить, что главным преимуществом при создании нового алгоритма электронного документооборота является максимальная автоматизация процессов работы с документами. Это позволит сократить временные затраты на обработку документов, минимизировать вероятность ошибок и снизить затраты на персонал. Важным преимуществом также является возможность интеграции с другими систем

Выводы по главе 2

Во второй главе были рассмотрены основные аспекты проектирования системы управления файлами и заданиями. Была проведена архитектурная разработка, определены компоненты системы и их взаимодействие. Ключевыми компонентами являются модуль управления файлами, модуль управления заданиями и модуль управления пользователями и группами.

Также были рассмотрены алгоритмы и процессы, связанные с добавлением файлов, назначением заданий и управлением доступом.

Глава 3 Программная реализация и тестирование алгоритмов управления электронным документооборотом

3.1 Информационное обеспечение системы электронного документооборота

Входящие документы автоматически распределяются по соответствующим подразделениям и сотрудникам на основе заранее определенных правил и критериев. Это позволит избежать задержек в работе с документами и повысит эффективность всей системы.

Документы могут быть созданы и отправлены в систему из различных источников, таких как электронная почта, сканеры, веб-формы и другие. Система автоматически определяет формат документа и преобразует его в универсальный формат для дальнейшей обработки.

Все изменения, совершенные над документами, фиксируются в системе, а также сохраняются различные версии документов. Это обеспечивает контроль за процессом работы с документами и позволяет вернуться к предыдущей версии документа, если это необходимо.

Система обеспечивает доступ к документам только соответствующим сотрудникам и подразделениям на основе определенных правил доступа. Также возможно установить дополнительные ограничения на просмотр и редактирование документов.

Система обеспечивает автоматическое уведомление сотрудников о поступлении новых документов, о необходимости выполнения задач и сроках выполнения. Это позволит избежать пропуска важных событий и своевременно выполнить все задачи.

Возможна интеграция с другими системами, такими как CRM, ERP, HRM, что позволит автоматизировать большое количество процессов и значительно повысить эффективность работы всей компании.

Программа сугубо по нашему мнению будет превосходить остальные

рабочие проекты по максимальной автоматизации процессов работы с документами.

3.2 Постановка задачи

Необходимо разработать универсальную систему электронного документооборота, которая упростит и ускорит процесс работы с документами. Система электронного документооборота (СЭД) должна удовлетворять следующим требованиям:

- разграничение прав доступа пользователей с использованием двух ролей: "администратор" и "обычный пользователь";
- возможность хранения шаблонов документов, широко используемых в организации, с доступом для всех пользователей системы;
- доступ к системе разрешен только зарегистрированным администратором пользователям;
- мультиплатформенность приложения и его не специфичность для конкретных предприятий, с возможностью настройки системы администратором для конкретной организации;
- хранение документов в едином месте и наличие метаданных;
- администратор должен иметь возможность выполнять следующие функции: добавление/удаление/редактирование пользователей, добавление/удаление/редактирование групп, добавление/удаление каталогов, удаление файлов;
- реализация механизма управления версиями документов, сохраняющего историю изменений;
- пользователи должны иметь возможность добавлять/удалять файлы, задавать срок хранения файлов в системе, добавлять комментарии к файлам, выбирать действия других пользователей над файлами, устанавливать область видимости файлов и отправлять оповещения пользователям по электронной почте.

Желательно иметь эффективную систему поиска документов, позволяющую находить документы с минимальной информацией о них.

3.3 Функциональное описание системы

Модель пользовательского взаимодействия является ключевой составляющей при проектировании системы электронного документооборота. Она представляет собой концепцию, определяющую функциональные возможности и взаимодействие системы с пользователями, и позволяет рассмотреть систему с точки зрения пользователей.

В соответствии с методологией Microsoft Solution Framework, процесс проектирования начинается с тщательного анализа пользователей. Этот анализ позволяет определить различные типы пользователей и их рабочие функции в контексте электронного документооборота. Каждый тип пользователя имеет свои уникальные потребности, цели и предпочтения, которые должны быть учтены при разработке системы.

«Затем создается набор сценариев использования, каждый из которых описывает последовательность конкретных действий, называемых примерами использования» [2].

Web-приложение Redjista для системы электронного документооборота предусматривает два типа пользователей:

- обычный пользователь системы - каждый сотрудник учреждения;
- администратор - назначенное лицо, обладающее особыми полномочиями (правами) в системе;
- у обоих типов пользователей есть общие функции: аутентификация пользователя: после ввода правильного логина и пароля пользователь проходит процедуру аутентификации в системе и может выполнять определенные действия в соответствии с его правами. В качестве логина пользователя используется электронная почта (e-mail), так как она удобна для рассылки важной информации;

- «просмотр содержимого каталогов: Пользователь имеет возможность выбрать определенную папку в дереве и просмотреть ее содержимое в соответствии с правами доступа к файлам. Отображаются только те файлы, которые пользователь может просматривать;
- просмотр информации о файле: Пользователь может просмотреть различные атрибуты файла, включая название, дату добавления, создателя, размер, срок хранения и комментарии;
- удаление файла: во время просмотра файла, пользователь имеет возможность удалить его. Удаление разрешено только создателю файла или администратору. При удалении файла также удаляются все его версии;
- скачивание файла: пользователь может скачать файл для просмотра, выбрав место, куда он будет загружен;
- просмотр доступа к файлу: администратор и пользователи с разрешенным доступом могут просматривать список пользователей и групп, которым файл доступен;
- просмотр заданий к файлу: пользователь, выбрав файл, может просмотреть связанные с ним задания, включая текст, автора, исполнителя и отметку о выполнении;» [7].
- просмотр версий файла: в пользовательской системе можно просмотреть различные версии файла, созданные при его изменении. Доступ к версиям файла предоставляется администраторам и пользователям с соответствующими правами. Пользователь может просмотреть список версий файла, включающий номер версии, родительский файл, дату добавления, комментарии и пользователя, создавшего версию. В списке отображаются все версии, начиная со второй, поскольку первая версия считается исходным файлом;
- скачивание версии файла: пользователь может скачать доступную для просмотра версию файла, выбрав место сохранения;

- удаление версий файла: существуют два способа удаления версий: удаление выбранной версии из списка версий, в результате чего последующие версии файла будут перенумерованы, удаление всех версий файла, кроме выбранной из списка версий, приводит к удалению всех версий, за исключением выбранной. Оставшаяся версия становится второй по порядку. Удаление версий файла может быть выполнено администратором или любым пользователем системы с соответствующими правами доступа к файлу;
- просмотр пользователей в определенной группе: пользователь может просмотреть список пользователей, принадлежащих к текущей группе;
- просмотр списка пользователей: пользователь может ознакомиться со списком всех пользователей системы, включающим их ФИО, должности и адреса электронной почты;
- просмотр списка групп: пользователь имеет возможность просмотра списка существующих групп в системе;
- для пользователей, авторизованных в качестве администратора, предусмотрены следующие сценарии использования: добавление пользователя. Администратор может добавить нового пользователя, указав его фамилию, имя, отчество, должность, адрес электронной почты, логин и пароль;
- удаление пользователя: Администратор может пометить пользователя как удаленного. В этом случае пользователь не будет отображаться в списках;
- восстановление пользователя: Администратор может восстановить ранее зарегистрированного пользователя в системе;
- редактирование информации о пользователе: Администратор имеет право вносить изменения в следующую информацию о пользователе - ФИО, должность, адрес электронной почты, логин и пароль;
- просмотр групп пользователя: Администратор может просмотреть

группы, в которых пользователь состоит;

- добавление папки: в определенном каталоге администратор может создать новую папку;
- удаление папки: администратор может удалять папки, при этом также будут удалены все файлы и подкаталоги, находящиеся в данной папке;
- добавление задания: администратор может создать новое задание, которое будет использоваться для назначения пользователей на выполнение указанных задач к файлу;
- удаление задания: администратор имеет право удалить задание из списка назначенных задач, если оно стало неактуальным. При этом задание сохраняется только для файлов, которые хранятся в системе;
- просмотр списка заданий: администратор может ознакомиться со списком добавленных им задач, которые будут использоваться для назначения пользователей на выполнение заданий по файлам;
- создание группы: администратор может создать группу пользователей, указав ее название;
- удаление группы: администратор имеет возможность удалить группу;
- добавление пользователя в группу: администратор может добавлять зарегистрированных пользователей в определенную группу;
- удаление пользователя из группы: Администратор может исключить пользователя из конкретной группы.

Для обычных сотрудников учреждения, авторизованных в системе, существуют следующие сценарии использования, отличные от администратора:

- «добавление новой версии файла: Для добавления новой версии файла пользователь должен указать название файла, комментарий, срок хранения и выбрать файл для загрузки. Номер версии, дата добавления и создатель файла генерируются автоматически;

- добавление файла: Для добавления файла пользователь должен ввести название файла, комментарий, срок хранения и выбрать файл для загрузки. Дата добавления, размер файла и создатель автоматически заполняются. По умолчанию, файл доступен всем пользователям системы;
- просмотр списка добавленных файлов: Пользователь может просмотреть список файлов, которые он добавил. В таблице отображается название файла, дата добавления, размер, комментарии и срок хранения каждого файла. Пользователь также может получить подробную информацию о каждом файле» [7];
- назначение задания для файла: Пользователь может назначить задание для файла, выбрав тип задания и исполнителя. Система автоматически заполняет поля "Пользователь, добавивший задание", "Очередность выполнения" и "Статус выполнения задания". По умолчанию задание считается невыполненным;
- редактирование доступа к файлу: Для каждого добавленного файла пользователь может изменять списки доступа для групп и пользователей. Он может выбирать из списка те группы и пользователей, которым будет виден файл. По умолчанию, файл доступен всем пользователям системы;
- просмотр заданий пользователя: Отображаются все задания, назначенные данному пользователю для выполнения. По каждому заданию можно увидеть текст задания, связанный файл, информацию о том, кем было добавлено задание, а также отметку о выполнении задания;
- «просмотр добавленных заданий: Отображаются все задания, которые пользователь назначил другим пользователям. По каждому заданию можно увидеть текст задания, связанный файл, исполнителя задания и отметку о выполнении;
- отметка о выполнении задания: Пользователь, который добавил

задание к файлу, или тот, кому было назначено задание, может поставить отметку о его выполнении после завершения работ. При положительной отметке («задание выполнено»), пользователь, назначивший задание, получает уведомление на электронную почту о выполнении задания. Кроме того, в списке заданий для данного пользователя отображается отметка о выполнении задания» [7].

Таким образом, модель пользовательского взаимодействия играет важную роль в проектировании эффективной системы электронного документооборота. Она позволяет учесть потребности и предпочтения пользователей, а также создать удобный и интуитивно понятный интерфейс, способствующий эффективному взаимодействию между пользователями и системой

3.4 Логическое конструирование системы

В соответствии с особенностями представления данных в разрабатываемой системе, предметная область описывается следующими основными сущностями:

- шаблон;
- каталог;
- содержимое;
- версия;
- группа;
- администратор;
- файл;
- информация о файле;
- пользователь;
- каталог.

Также следует выделить несколько групп задач для реализации функциональности [8].

«Сервис для работы с базой данных, который включает основные функции:

- установка подключения к базе данных - создает устойчивое соединение с базой данных;
- создание таблиц в базе данных - создает таблицы в базе данных и добавляет необходимые первоначальные данные (например, создает администратора и корневой каталог).

Сервис для работы с пользователями и группами, включающий реализацию следующих функций:

- аутентификация пользователей и администраторов - проверка наличия пользователя или администратора с указанными логином и паролем в базе данных, с последующим предоставлением доступа к системе;
- получение информации о пользователе;
- получение списка пользователей - отображение всех пользователей системы;
- получение списка групп - отображение информации о группах пользователей в системе;
- получение списка групп, в которых состоит пользователь;
- получение списка пользователей из определенной группы;
- добавление/редактирование/удаление пользователя - функции, доступные только администратору системы;
- восстановление пользователя - администратор может восстановить удаленного из системы пользователя» [9];
- добавление/удаление группы - функции, доступные только администратору системы;
- добавление/удаление пользователя в/из группы - функции, доступные только администратору системы;

Кроме основных функций, сервис также включает следующие вспомогательные функции:

- проверка соответствия логина адресу электронной почты - это позволяет отправлять пользователю уведомления по электронной почте;
- проверка существования логина - это обеспечивает наличие уникального идентификатора пользователя в системе;
- проверка существования группы с введенным именем – позволяет обеспечить уникальность имени группы;
- получение различных атрибутов сущностей.

Сервис для работы с файлами и каталогами содержит следующие основные функции:

- добавить/удалить каталог – доступны только администратору;
- отобразить содержимое каталога – служит для отображения каталогов и файлов, находящихся в выбранном каталоге, к которым пользователь имеет доступ;
- администратору доступны все файлы системы;
- добавить файл – ввод данных, описывающих файл и загрузка файла в файловую систему сервера;
- удалить файл – функция доступна администратору и пользователю, который добавил данный файл;
- файл удаляется из БД и из файловой системы;
- вместе с файлом удаляются все его версии;
- получить информацию о файле – необходимо для вывода описания файла и пользователя, который добавил данный файл;
- разрешить/запретить доступ всех пользователей к файлу – доступны только автору файла;
- разрешить/запретить группе доступ к файлу - доступны только автору файла;
- разрешить/запретить доступ пользователя к файлу - доступны только автору файла;
- получить списки групп и пользователей, имеющих доступ к файлу –

доступна администратору системы и автору файла;

- добавить/удалить версию файла – доступны администратору и всем пользователям, имеющим доступ к файлу;
- удалить все версии файла – удаляются все версии выбранного файла, сам файл не удаляется;
- удалить все версии файла, кроме одной – необходима в случае, когда нужно оставить лишь одну версию файла (например, последнюю, окончательную), а все остальные удалить;
- сам файл не удаляется;
- получить информацию о версиях файла - для отображения информации о всех версиях выбранного файла.

Кроме основных функций, сервис также содержит следующие вспомогательные функции:

- проверка существования имени каталога - обеспечивает уникальность имен каталогов на одном уровне;
- получение пути до файла в файловой системе - необходимо для скачивания, удаления файла и добавления версий к файлу;
- получение текущего количества версий файла;
- получение общего количества версий файла;
- получение пути до версии файла в файловой системе - необходимо для скачивания и удаления версий файла.

Сервис для работы с заданиями включает следующие функции:

- «добавление/удаление задания для работы с файлами - позволяет администратору системы добавлять и удалять задания, связанные с файлами;
- отображение заданий для работы с файлами - для выбора задания к файлу или просмотра заданий администратором;
- добавление/удаление задания к файлу - при добавлении задания к файлу адресату отправляется уведомление по электронной почте;
- получение заданий к файлу - отображение списка всех заданий,

связанных с данным файлом;

- получение заданий, адресованных пользователю - пользователь системы может просмотреть задания, которые были ему адресованы;
- получение заданий, добавленных пользователем - пользователь системы может просмотреть информацию о заданиях, которые он сам добавил;
- проверка выполнения задания.

Подтверждение выполнения/невыполнения задания - адресат или автор файла могут подтвердить выполнение задания или назначить его повторно с отправкой уведомления на почту» [9].

Для разработки использовалась среда IntelliJ IDEA 8.1.3. Для тестирования использовалась база данных hsqldb. В качестве сервера использовался Apache Tomcat 5.5.25. Приложение было протестировано локально. Для создания файловой системы использовался предварительно созданный каталог на жестком диске.

3.5 Обзор программы

В данной программе мы решили не уклоняться от стандартов архитектуры построения пользовательского интерфейса, для упрощения работы с программой другими пользователями использующие ее в первый раз.

Первым действием мы запустим и откроем программу после чего нас будет встречать вот такой интерфейс (шаблон для будущего предпринимателя, который может вставить свои фото) для авторизации или регистрации нового пользователя.

Авторизовавшийся пользователь перейдет на следующую страницу и ему будет доступен просмотр документов, переход на следующие страницы, а также возможно выхода из аккаунта или системы. Есть возможность программно изменить дизайн индивидуально для каждой организации (рисунок 4).

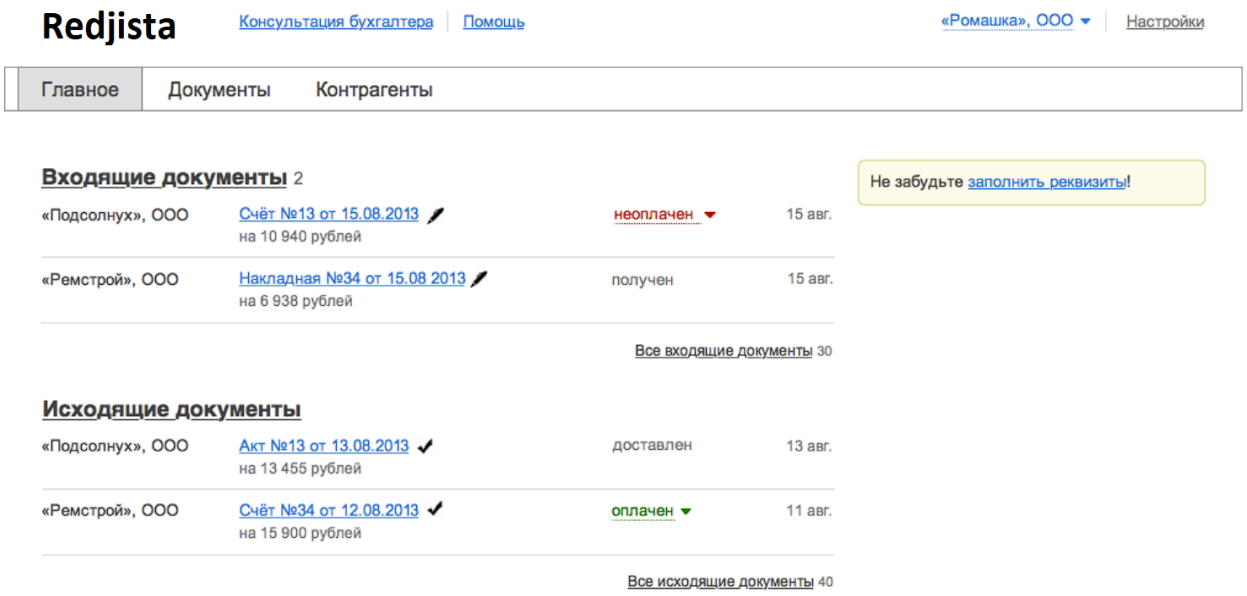

Рисунок 4 - Главная страница программы

Во вкладке “Документы” (рисунок 5) производится вся основная работа связанная с электронным документооборотом.

Основные функции включают следующие опции:

- фильтрация и поиск по названию документа: пользователь может использовать фильтры и осуществлять поиск по названию документа, что позволяет быстро находить нужные документы среди большого объема информации;
- скачивание документов: пользователь может скачать выбранный документ на свое устройство, чтобы иметь локальную копию или использовать его вне системы;
- создание новых документов: пользователь имеет возможность создавать новые документы прямо в системе, что упрощает процесс их формирования и сохранения;
- распределение по группам: пользователь может классифицировать документы, распределяя их по соответствующим группам, что

облегчает организацию и структурирование информации.

Изменение статуса документа: пользователь может изменять статус документа, указывая его текущее состояние "не прочитан", "прочитан", "на рассмотрении" и т.д., что помогает отслеживать и контролировать прогресс работы над документами (рисунок 5).

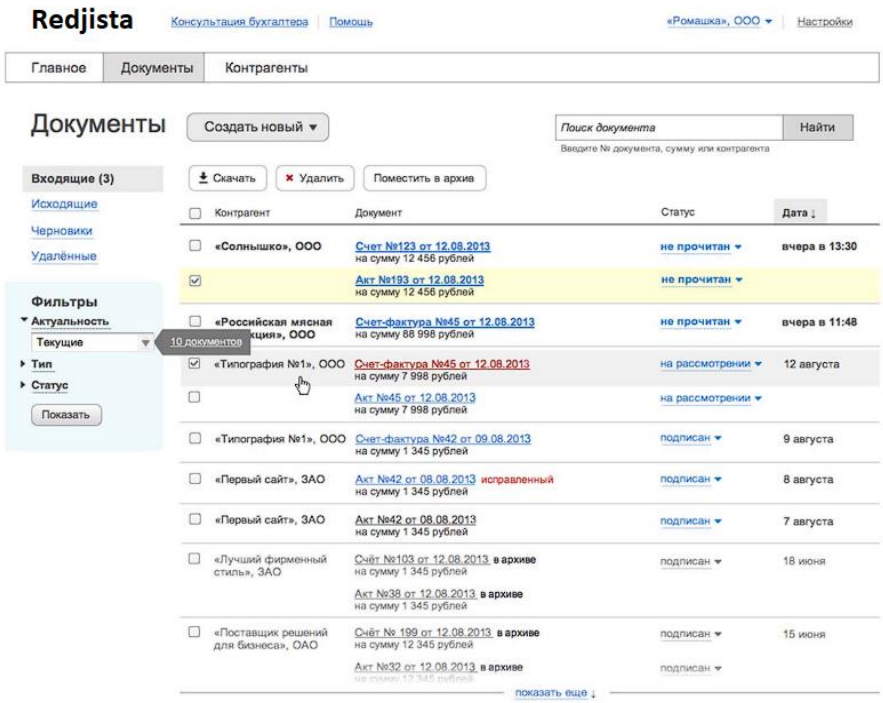

Рисунок 5 - Страница документов и наименования структур

Кроме того, в правой части вкладки отображается информация о дате последнего изменения документа. Это позволяет пользователям легко определить, когда последний раз были внесены изменения в конкретный документ, что может быть полезным при отслеживании и контроле за обновлениями.

Далее переходим в документ и рассмотрим что можно с ним сделать. Например, есть возможно скачать, изменить, распечатать, переименовать, отправить по email, подписать при помощи ЭЦП, экспортировать документ, например, в Word, в Excel, в PDF (рисунок 6).

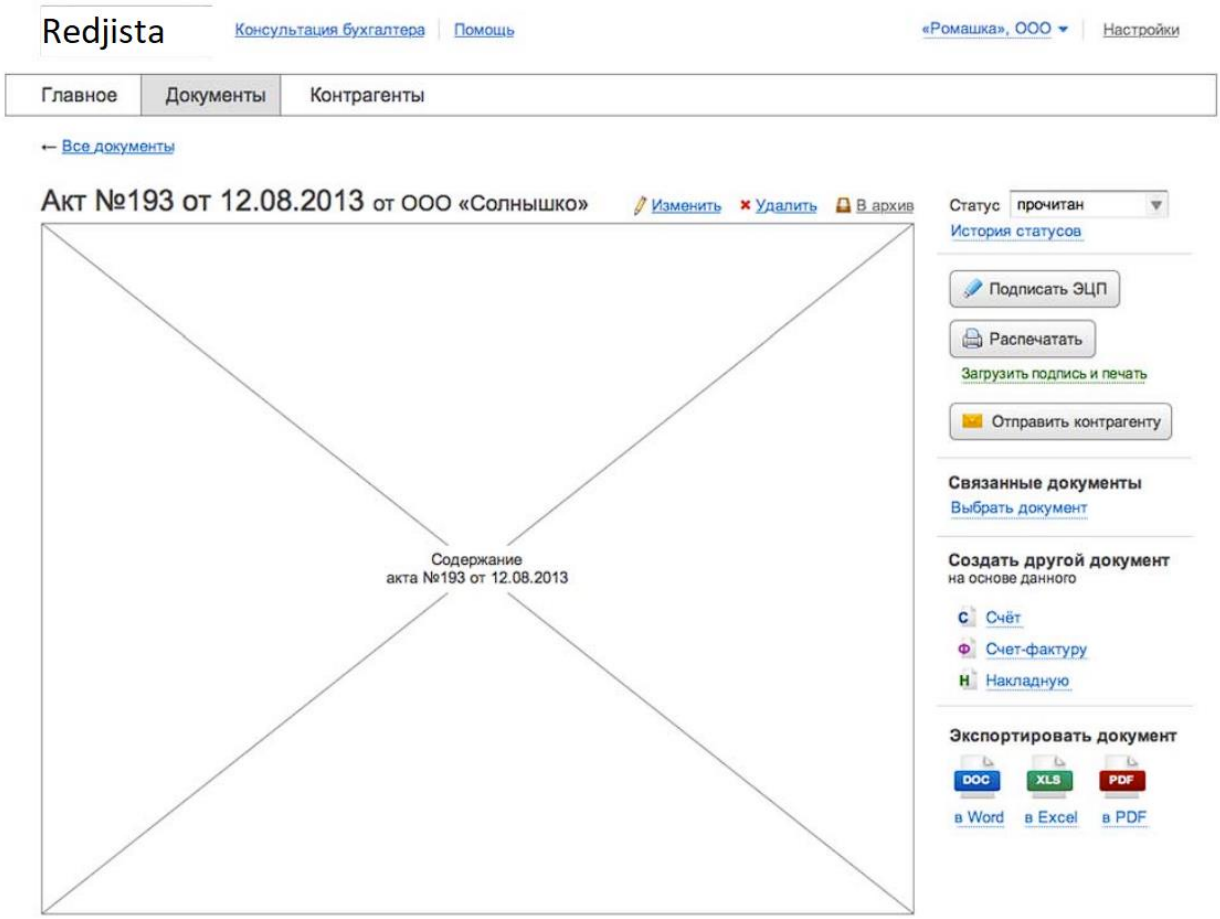

Рисунок 6 - Редактор документа

На странице “Контрагенты” пользователь видит список контрагентов, имеет возможность фильтровать клиентов и поставщиков, скачать и добавить документы, связанные с контрагентами, осуществить поиск по контрагентам и создать нового контрагента. Также пользователь видит последнее действие, произведенное над документом контрагента (рисунок 7).

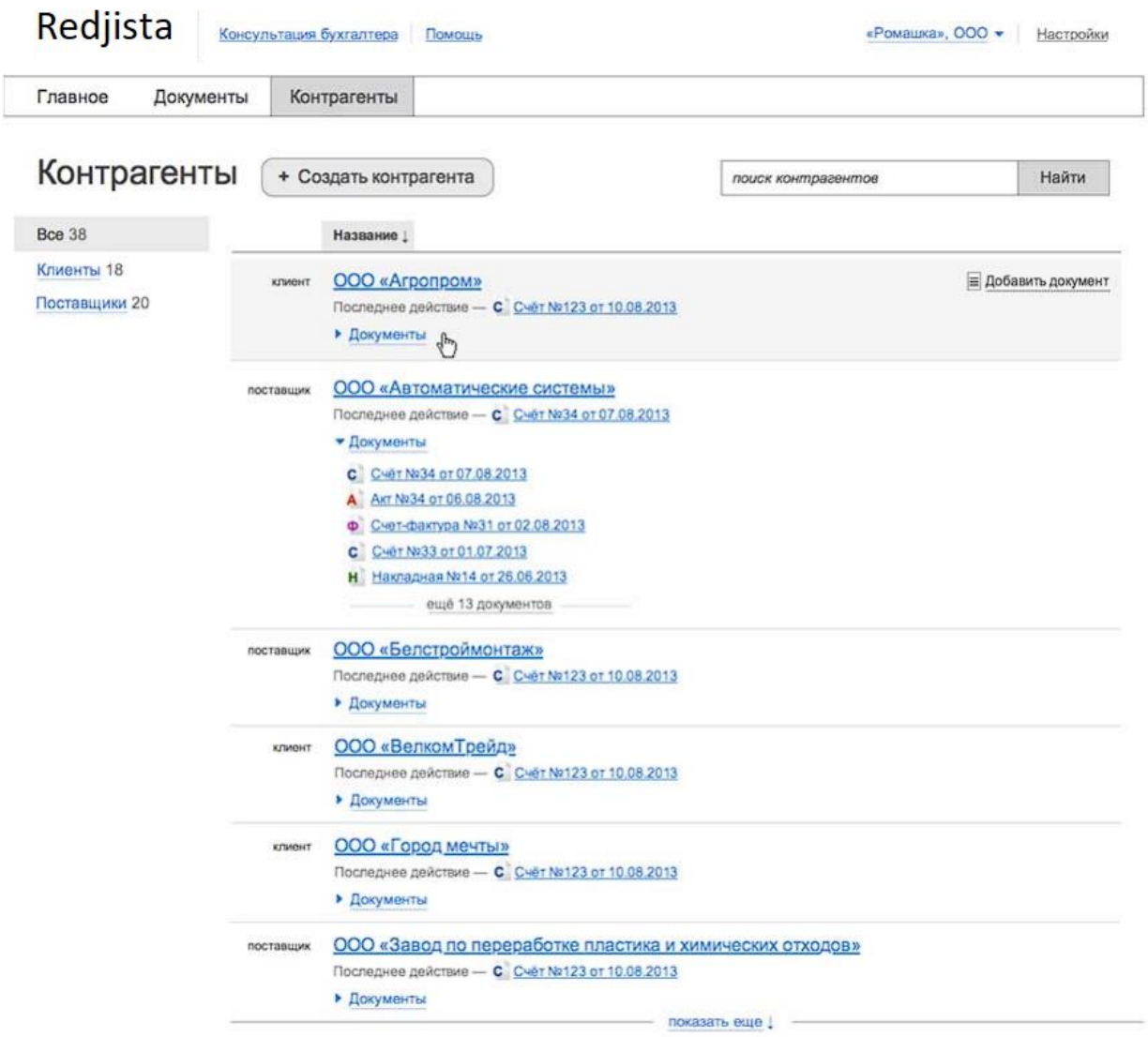

Рисунок 7 - Список контрагентов

Во вкладке "Контрагенты" пользователю предоставляется возможность просмотра и управления данными о контрагентах. Здесь пользователь может получить полную информацию о выбранном контрагенте и осуществлять различные действия с ним (рисунок 8).

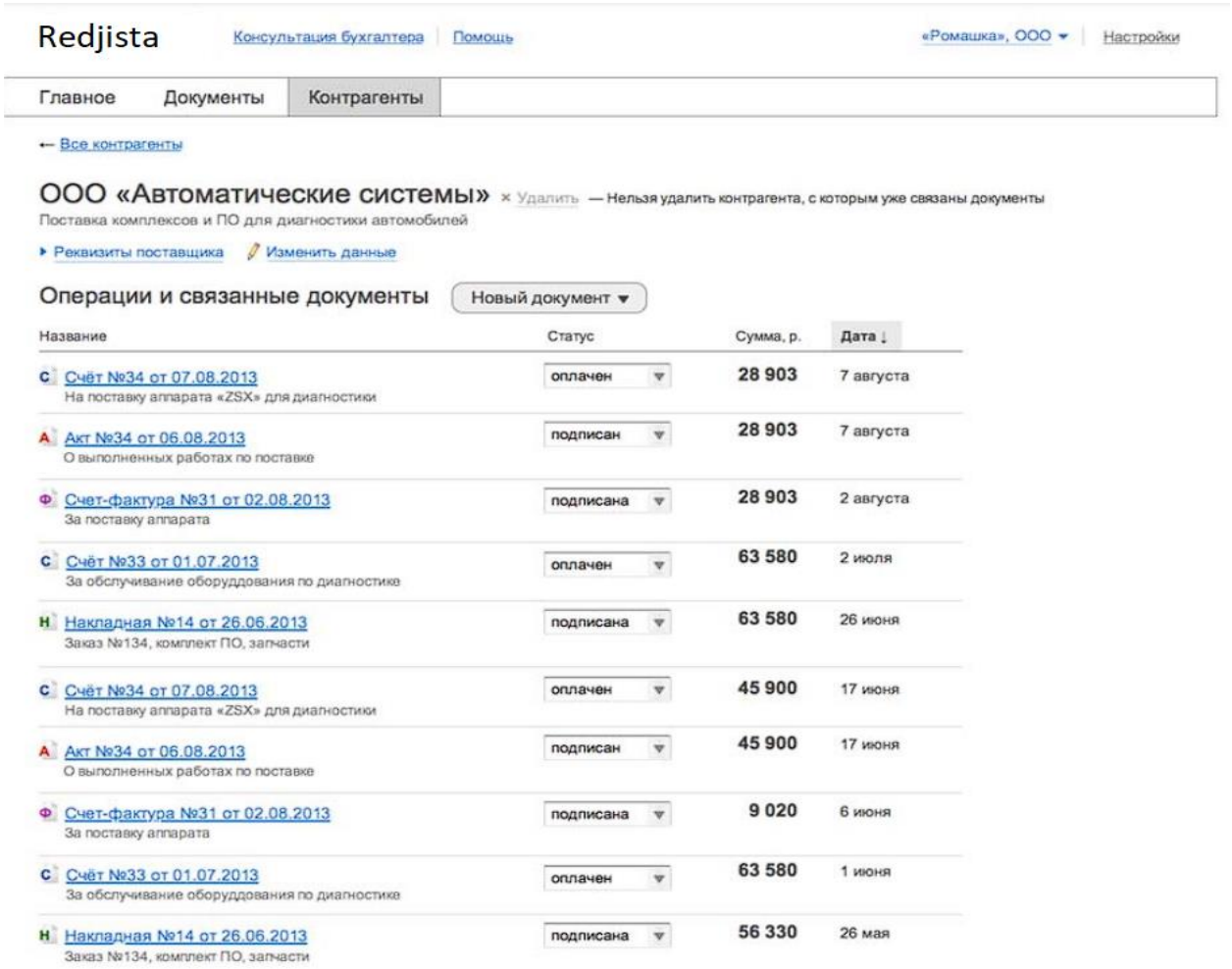

Рисунок 8 - Содержание страницы клиента

Основные функции включают:

- просмотр данных о контрагенте: пользователь может просмотреть все данные, относящиеся к выбранному контрагенту, включая реквизиты поставщика, информацию о производимых операциях и связанных с контрагентом документах. Это позволяет пользователям получить полное представление о контрагенте и его активностях;
- изменение статуса документа: пользователь имеет возможность изменять статус документа, связанного с контрагентом. Например, можно указать текущий статус документа "оплачен" и "подписан", что помогает отслеживать и контролировать его состояние;
- изменение данных о контрагенте: пользователь может вносить

изменения в данные о контрагенте, такие как контактные данные, адрес, банковские реквизиты и прочую информацию. Это позволяет поддерживать актуальность и точность информации о контрагентах;

- ограничение удаления контрагента: невозможно удалить контрагента, с которым уже связаны документы. Это предотвращает случайное удаление контрагента, что обеспечивает сохранность и целостность данных о документообороте.

Продемонстрирован основной блок работы в приложении.

3.6 Обзор кода программы

Программный код контроллера, отвечающего за аутентификацию:

```
public ResponseStatusDTO getRoles() {
    UserDTO user = userService.getCurrentUser();
    if(user != null)
        return new ResponseStatusDTO(StatusTypes.OK, user.getRoles());
    else
        return new ResponseStatusDTO(StatusTypes.OK);
}
```

Первый метод, getRoles(), возвращает объект ResponseStatusDTO. Сначала он получает текущего пользователя с помощью метода getCurrentUser() в объекте userService. Затем проверяется, является ли пользователь не null. Если пользователь не null, то создается новый объект ResponseStatusDTO с StatusTypes.OK статусом и ролями пользователя, которые получаются с помощью метода getRoles() в объекте user. Если пользователь равен null, то возвращается объект ResponseStatusDTO только с StatusTypes.OK статусом.

```
public ResponseStatusDTO getCurrentUserAccessRight() {
    ResponseStatusDTO res = new ResponseStatusDTO();
    try {
```

```
accessRightManager.init(userAccessRightDataService.get(userService.getCurrent
User().getId()), accessRightDataService.getAll());
                res.setStatus(StatusTypes.OK);
                res.setData(accessRightManager.get());
                return res;
            }
            catch (Exception e) {
                res.setStatus(StatusTypes.ERROR);
                res.addErrors(e.getMessage());
                return res;
            }
        }
```

метод, getCurrentUserAccessRight(), также возвращает объект ResponseStatusDTO. Внутри метода происходит инициализация accessRightManager, используя данные о правах доступа текущего пользователя, полученные с помощью метода get() в объекте userAccessRightDataService, и все доступные права доступа, полученные с помощью метода getAll() в объекте accessRightDataService. Если инициализация проходит успешно, то статус у объекта ResponseStatusDTO устанавливается как StatusTypes.OK, а данные устанавливаются с помощью метода get() в объекте accessRightManager. Если происходит исключение во время инициализации или выполнения других операций, то статус у объекта ResponseStatusDTO устанавливается как StatusTypes.ERROR, и сообщение об ошибке добавляется с помощью метода addErrors() в объекте res. В конце метод возвращает объект res.

```
        public ResponseStatusDTO registration(UserDTO user) {
            userManager.Create(user, roleDataService.getAll());
            ResponseStatusDTO res = userManager.validate();
            if(res.getStatus() == StatusTypes.ERROR) return res;
            try {
```

```
                ResponseStatusDTO<UserDTO> saveUserStatus =
userDataService.save(userManager.get());
                if(saveUserStatus.getStatus() == StatusTypes.ERROR) return
saveUserStatus;
                UserDTO userSaved = saveUserStatus.getData();
                roleListManager.init(userSaved.getRoles());
                if (roleListManager.ContainRole(RoleType.PROFESSOR))
                {
                   professorManager.init(new ProfessorDTO());
                   professorManager.create(userSaved);
                   ResponseStatusDTO<ProfessorDTO> saveProfessorStatus =
professorDataService.save(professorManager.get());
                   if (saveProfessorStatus.getStatus() == StatusTypes.ERROR) return
saveProfessorStatus;
                }

                if (roleListManager.ContainRole(RoleType.STUDENT))
                {
                   studentManager.init(new StudentDTO());
                   studentManager.create(userSaved);
                   ResponseStatusDTO<StudentDTO> saveStudentStatus =
studentDataService.save(studentManager.get());
                   if (saveStudentStatus.getStatus() == StatusTypes.ERROR) return
saveStudentStatus;
                }
                res.setData(userSaved);
                res.addMessage("Пользователь добавлен");
             }
             catch (Exception e) {
                res = new ResponseStatusDTO(StatusTypes.ERROR,
```

```
e.getMessage());
            }
            return res;
        }
        public ResponseStatusDTO getUser() {
            UserDTO user = userService.getCurrentUser();
            return new ResponseStatusDTO(StatusTypes.OK, user);
        }
        public ResponseStatusDTO changePassword(String newPass, String
oldPass) {
            UserDTO user = userService.getCurrentUser();
            userManager.init(user);
            ResponseStatusDTO response =
userManager.ChangePassword(newPass, oldPass);
            if (response.getStatus() == StatusTypes.ERROR) return response;
            try {
                userDataService.save(userManager.get());
                response.addMessage("Пароль изменен.");
            } catch (Exception e) {
                response.setStatus(StatusTypes.ERROR);
                response.addErrors("Не удалось изменить пароль");
            }
            return response;
        }
        public ResponseStatusDTO changePhoto(String photo) {
            UserDTO user = userService.getCurrentUser();
            userManager.init(user);
            ResponseStatusDTO response = userManager.ChangePhoto(photo);
            if (response.getStatus() == StatusTypes.ERROR) return response;
            try {
```

```
            userDataService.save(userManager.get());
            response.addMessage("Фотография изменена.");
        } catch (Exception e) {
            response.setStatus(StatusTypes.ERROR);
            response.addErrors("Не удалось изменить фото");
        }
        return response;
    }
    public ResponseStatusDTO findUsersByFIO(String req) {
        userListManager.init(userDataService.getAll());
        return new ResponseStatusDTO(StatusTypes.OK,
userListManager.GetByFio(req));
    }
    public ResponseStatusDTO findUsersByRoleName(String roleName) {
        userListManager.init(userDataService.getByRoleName(roleName));
        return new ResponseStatusDTO(StatusTypes.OK,
userListManager.getAll());
    }
    public ResponseStatusDTO getProfessors() {
        professorListManager.init(professorDataService.getAll());
        return new ResponseStatusDTO(StatusTypes.OK,
professorListManager.getAll());
    }
    public ResponseStatusDTO getProfessorByUser(long userId) {
        professorManager.init(professorDataService.getByUser(userId));
        return new ResponseStatusDTO(StatusTypes.OK,
professorManager.get());
    }
    public ResponseStatusDTO getStudentByUser(long userId) {
        studentManager.init(studentDataService.getByUser(userId));
```

```
            return new ResponseStatusDTO(StatusTypes.OK,
studentManager.get());
        }
        public ResponseStatusDTO getStudentForGroupAndLesson(long groupId,
long lessonId) {
            return new ResponseStatusDTO(StatusTypes.OK,
studentDataService.getByGroupAndLesson(groupId, lessonId));
        }
        public ResponseStatusDTO setProfessorDepartment(long userId, long
departmentId) {
            ResponseStatusDTO res = new ResponseStatusDTO();
            try {
                professorManager.init(professorDataService.getByUser(userId));
                departmentManager.init(departmentDataService.get(departmentId));
                professorManager.setDepartment(departmentManager.get());
                res.setStatus(StatusTypes.OK);
                ResponseStatusDTO<ProfessorDTO> saveProfessorStatus =
professorDataService.save(professorManager.get());
                if (saveProfessorStatus.getStatus() == StatusTypes.ERROR) return
saveProfessorStatus;
                ProfessorDTO professor = saveProfessorStatus.getData();
                res.setData(professor);
                res.addMessage("Кафедра для преподавателя установленна");
            }
            catch (Exception e) {
                res.setStatus(StatusTypes.ERROR);
                res.addErrors("Кафедра для преподавателя не установленна");
                res.addErrors(e.toString());
            }
            return res;
```

```
    }
    public ResponseStatusDTO findUserByUsername(String req) {
        userListManager.init(userDataService.getAll());
        return new ResponseStatusDTO(StatusTypes.OK,
userListManager.GetByUsername(req));
    }
    public ResponseStatusDTO setStudentGroup(long userId, long groupId) {
        studentManager.init(studentDataService.getByUser(userId));
        groupManager.init(groupDataService.get(groupId));
        studentManager.setGroup(groupManager.get());
        ResponseStatusDTO res = new ResponseStatusDTO();
        res.setStatus(StatusTypes.OK);
        try {
            ResponseStatusDTO<StudentDTO> saveStudentStatus =
studentDataService.save(studentManager.get());
            if (saveStudentStatus.getStatus() == StatusTypes.ERROR) return
saveStudentStatus;
            StudentDTO student = saveStudentStatus.getData();
            res.setData(student);
            res.addMessage("Группа для студента установленна");
            return res;
        }
        catch (Exception e) {
            res.setStatus(StatusTypes.ERROR);
            res.addErrors("Группа для студента не установленна");
            res.addErrors(e.getMessage());
            return res;
        }
    }
    public ResponseStatusDTO getUserAccessRight(long userId) {
```

```
            ResponseStatusDTO res = new ResponseStatusDTO();
            try {
                accessRightManager.init(userAccessRightDataService.get(userId),
accessRightDataService.getAll());
                res.setStatus(StatusTypes.OK);
                res.setData(accessRightManager.get());
                return res;
            }
            catch (Exception e) {
                res.setStatus(StatusTypes.ERROR);
                res.addErrors(e.getMessage());
                return res;
            }
        }
        public ResponseStatusDTO saveUserAccessRight(UserAccessRightDTO
acceses) {
            ResponseStatusDTO res = new ResponseStatusDTO();
            accessRightManager.init(acceses, accessRightDataService.getAll());
            res = accessRightManager.validate();
            if(res.getStatus() == StatusTypes.ERROR) return res;
            try {
                userAccessRightDataService.save(acceses);
                res.setStatus(StatusTypes.OK);
                res.addMessage("Настройки доступа сохранены.");
                res.setData(accessRightManager.get());
                return res;
            }
            catch (Exception e) {
                res.setStatus(StatusTypes.ERROR);
                res.addErrors(e.getMessage());
```

```
            return res;
        }
    }
    public ResponseStatusDTO saveStudentsSubgroup(StudentJournalList
studentJournalList) {
        ResponseStatusDTO res = new ResponseStatusDTO();
        try {

studentDataService.saveStudentsSubgroup(studentJournalList.getStudentJournal(),
studentJournalList.getLesson());
            res.setStatus(StatusTypes.OK);
            res.addMessage("Подгруппы студентов сохранены.");
            res.setData(accessRightManager.get());
            return res;
        }
        catch (Exception e) {
            res.setStatus(StatusTypes.ERROR);
            res.addErrors(e.getMessage());
            return res;
        }
    }
}
```

Программный код контроллера, отвечающего за безопасность загрузок:

```
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {
  @Bean
  public CorsFilter corsFilter() {
    UrlBasedCorsConfigurationSource source = new
UrlBasedCorsConfigurationSource();
    CorsConfiguration config = new CorsConfiguration();
    config.setAllowCredentials(true);
```

```
        config.addAllowedOrigin("*");  // TODO: lock down before deploying
        config.addAllowedHeader("*");
        config.addExposedHeader(HttpHeaders.AUTHORIZATION);
        config.addAllowedMethod("*");
        source.registerCorsConfiguration("/**", config);
        return new CorsFilter(source);
    }
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
            .cors()
            .and()
            .csrf().disable()
            .authorizeRequests()
            .antMatchers("/api/account/login").permitAll()
            .antMatchers("/api/demo/**").permitAll()
            .antMatchers( "/images/**").permitAll()
            .antMatchers( "/fonts/**").permitAll()
            .antMatchers( "/js/**").permitAll()
            .antMatchers( "/vendor/**").permitAll()
            .antMatchers( "/css/**").permitAll()
               .antMatchers("/api/**").permitAll()
            .anyRequest().authenticated()
            .and()
            // We filter the api/login requests
            .addFilterBefore(new JWTLoginFilter("/api/account/login",
authenticationManager()),
                    UsernamePasswordAuthenticationFilter.class)
            // And filter other requests to check the presence of JWT in header
            .addFilterBefore(new JWTAuthenticationFilter(),
```

```
                UsernamePasswordAuthenticationFilter.class);
        }
        @Override
        public void configure(WebSecurity web) throws Exception {
            web.ignoring().antMatchers("/*","/api/news/all");
        }
        @Autowired
        private CustomUserDetailsService userDetailsService;
        @Bean(name="passwordEncoder")
        public PasswordEncoder passwordencoder(){
            return new BCryptPasswordEncoder();
        }
        @Bean
        public DaoAuthenticationProvider authProvider() {
            DaoAuthenticationProvider authProvider = new
DaoAuthenticationProvider();
            authProvider.setUserDetailsService(userDetailsService);
            authProvider.setPasswordEncoder(passwordencoder());
            return authProvider;
        }
        @Override
        protected void configure(AuthenticationManagerBuilder auth) throws
Exception {
            // Create a default account
            auth.authenticationProvider(authProvider());
        }
    }
```

Программный код контроллера, отвечающего за миграцию:

```
@Configuration
public class WebConfig extends WebMvcConfigurerAdapter {
```

```
        @Autowired
        DataSource dataSource;
        private static final String[] CLASSPATH_RESOURCE_LOCATIONS = {
            "classpath:/META-INF/resources/", "classpath:/resources/",
            "classpath:/static/", "classpath:/public/", "classpath:/public/vendor/" };
        @Override
        public void addResourceHandlers(ResourceHandlerRegistry registry) {
          registry.addResourceHandler("/**")

.addResourceLocations(CLASSPATH_RESOURCE_LOCATIONS);
        }
        @Bean
        ErrorViewResolver supportPathBasedLocationStrategyWithoutHashes() {
          return new ErrorViewResolver() {
            @Override
            public ModelAndView resolveErrorView(HttpServletRequest
request, HttpStatus status, Map<String, Object> model) {
              if(status == HttpStatus.NOT_FOUND || status ==
HttpStatus.FORBIDDEN)
                return  new ModelAndView("index.html", Collections.<String,
Object>emptyMap(), HttpStatus.OK);
              return null;
            }
          };
        }
        // При запуске проекта проверяет существуют ли миграции.
        @Bean(initMethod = "migrate")
        Flyway flyway() {
          Flyway flyway = new Flyway();
          flyway.setDataSource(dataSource);
```

```
            flyway.setInitOnMigrate(true);
            return flyway;
        }
        // Решения добавляющее возможность
        // провести выполнение миграции до проверки соответствия
сущностей базе данных.
        @Bean
        @DependsOn("flyway")
        protected BeanPostProcessor forceFlywayToInitialize() {
            return new BeanPostProcessor() {
                @Override
                public Object postProcessAfterInitialization(Object bean, String
beanName)
                        throws BeansException {
                    return bean;
                }
                @Override
                public Object postProcessBeforeInitialization(Object bean, String
beanName)
                        throws BeansException {
                    return bean;
                }
            };
        }
    }
    Код отвечающий за плагин:
    public class PluginFactory {
        public static ArrayList<Plugin> getPlugins() {
            ArrayList<Plugin> rez = new ArrayList<Plugin>();
            File pluginDir = new File("src/plugins");
```

```
            File[] jars = pluginDir.listFiles(new FileFilter() {
                @Override
                public boolean accept(File file) {
                    return file.isFile() && file.getName().endsWith(".jar");
                }
            });
            for (File jar : jars) {
                try {
                    URL jarURL = jar.toURI().toURL();
                    URLClassLoader classLoader = new URLClassLoader(new
URL[]{jarURL});
                    JarFile jf = new JarFile(jar);
                    Enumeration<JarEntry> entries = jf.entries();
                    while (entries.hasMoreElements()) {
                        String e = entries.nextElement().getName();
                        if (!e.endsWith(".class")) continue;
                        e = e.replaceAll("/", ".");
                        e = e.replaceAll(".class", "");
                        Class<?> plugCan = classLoader.loadClass(e);
                        Class<?>[] interfaces = plugCan.getInterfaces();
                        for (Class interf : interfaces) {
                            if (interf.getName().endsWith(".Plugin")) {
                                Class c = classLoader.loadClass(plugCan.getName());
                                Object inst = c.newInstance();
                                rez.add((Plugin) inst);
                            }
                        }
                    }
                } catch (Exception e) {
                    e.printStackTrace(System.err);
```

```
            }
        }
        return rez;
    }
}
```

Данный класс PluginFactory предоставляет статический метод getPlugins(), который возвращает список объектов типа Plugin.

Внутри метода создается пустой список rez для хранения экземпляров плагинов. Затем создается объект File, представляющий директорию плагинов. Массив файлов jars содержит файлы в этой директории, отфильтрованные с использованием FileFilter, чтобы оставить только файлы с расширением .jar.

Затем происходит итерация по каждому файлу .jar в массиве jars. Для каждого файла выполняется следующая логика:

Создается объект URL для текущего файла .jar с помощью метода toURI().toURL().

Создается экземпляр URLClassLoader с массивом URL-ов, содержащим только текущий .jar.

Создается объект JarFile для текущего .jar.

Получается перечисление entries для всех элементов внутри .jar.

В цикле while происходит итерация по каждому элементу entries.

Для каждого элемента, если его имя не заканчивается на .class, пропускается итерация.

Заменяются все / на . и удаление .class из имени элемента.

Загружается класс plugCan с помощью classLoader.loadClass().

Получается массив интерфейсов interfaces для класса plugCan.

В цикле for происходит итерация по каждому интерфейсу interf в массиве interfaces.

Если имя интерфейса заканчивается на .Plugin, выполняется следующая логика:

- загружается класс c с помощью classLoader.loadClass() и передается имя класса plugCan;
- создается экземпляр объекта inst типа c с помощью c.newInstance();
- экземпляр inst добавляется в список rez после приведения к типу Plugin.

Если происходит исключение во время выполнения вышеуказанной логики, оно печатается в стандартный поток ошибок.

В конце метод возвращает список rez, содержащий экземпляры всех найденных плагинов.

Выводы по главе 3.

В заключительной главе были рассмотрены основные аспекты реализации системы управления файлами и заданиями. Был выбран язык программирования и использованы соответствующие технологии и инструменты разработки. Были разработаны необходимые модули и функциональность, обеспечивающие добавление и управление файлами, назначение и выполнение заданий, а также управление пользователями и группами. Были реализованы алгоритмы, обеспечивающие безопасность и контроль доступа к файлам и заданиям. В ходе разработки были проведены тестирование и отладка системы для обеспечения ее стабильной и корректной работы. Выводы последней главы подтверждают, что система управления файлами и заданиями успешно реализована с учетом требований и функциональных возможностей, что позволяет пользователям эффективно работать с файлами, назначать и выполнять задания, администрировать пользователей и группы в рамках заданной предметной области.

Заключение

Выпускная квалификационная работа посвящена актуальной проблеме исследования и практического применения программы электронного документооборота.

В настоящее время разработано много различных программ электронного документооборота.

Помимо конфигураций и количества используемых документов одним из критериев выбора конкретной программы является эффективность используемого в ней алгоритма управления документами.

Цель бакалаврской работы – исследование и реализация алгоритмов электронного документооборота.

Для достижения данной цели в процессе работы над бакалаврской работой была проведена обширная литературная работа в следующих направлениях:

Первое направление - основы управления электронным документооборотом. Были изучены основные принципы и концепции, связанные с организацией и функционированием электронного документооборота. Это включало в себя изучение понятий, стандартов, моделей и процессов, связанных с электронным документооборотом, а также ознакомление с основными преимуществами и вызовами, с которыми организации сталкиваются при его внедрении.

Второе направление - методы и алгоритмы управления электронным документооборотом. Были исследованы различные методы и стратегии управления, применяемые в электронном документообороте. Это включало изучение процессов управления доступом к документам, управления версиями, контроля изменений, установления правил и политик, а также принципов классификации и хранения документов. Был проведен сравнительный анализ существующих методов и алгоритмов для выявления их преимуществ и недостатков.

Третье направление - программная реализация систем электронного документооборота и тестирование программных продуктов. Были исследованы различные программные продукты и решения, предназначенные для организации электронного документооборота, изучены методы и подходы к тестированию программного обеспечения для систем электронного документооборота. Это включало ознакомление с функциональностью, возможностями, требованиями к системам электронного документооборота, а также с архитектурой и технологиями, используемыми при их разработке, определение тестовых сценариев, разработку тестовых случаев, выполнение тестовых испытаний и анализ результатов .Особое внимание было уделено проверке функциональности, производительности, надежности и безопасности программных продуктов.

Результаты бакалаврской работы представляют научно-практический интерес и могут быть рекомендованы для анализа и программной реализации методов и алгоритмов электронного документооборота.

Список используемой литературы и используемых источников

1. Антонов В.П. Электронный документооборот. М.: Издательский дом "Информационные технологии", 2015.
2. Баранов С.В. Электронный документооборот: организация и внедрение. СПб.: Питер, 2015.
3. Бахарев И.О. Электронный документооборот. М.: Интернет-Университет Информационных Технологий, 2015.
4. В.Г. Матвейкин, Б.С. Дмитриевский, Н.Р. Ляпин, Интеллектуальный анализ выполнения бизнес-процессов в системе электронного документооборота, 2021.
5. Венгеров А.Б. Электронный документооборот. М.: Финансы и статистика, 2014.
6. Гончарова О.Ю. Электронный документооборот в бухгалтерском учете. М.: Дело и сервис, 2016.
7. Дегтярева Е.В. Электронный документооборот: введение в теорию и практику. М.: КНОРУС, 2017.
8. Ермакова Н.В. Электронный документооборот: практическое руководство. М.: Дело и сервис, 2016.
9. Ибрагимов Т.Э. Электронный документооборот: управление бизнес-процессами. М.: Издательство "Проспект", 2017.
10. Кондратьев В.Г. Электронный документооборот в коммерческой деятельности. М.: Издательство Юрайт, 2016.
11. Кузнецова Н.В. Электронный документооборот: теория и практика. - М.: ЮНИТИ-ДАНА, 2016.
12. Кузнецова Т.В. Использование информационных технологий в электронном документообороте // Социально-гуманитарные знания. 2016. № 5. С. 53-57.
13. Кузнецова Т.В. Электронный документооборот: история, современность, перспективы // Экономика и предпринимательство. 2016. № 1

(64). С. 68-73.

14. Ларионова Л. В., Яшкова А.А. Электронный документооборот: состояние и перспективы // Вестник ПГУ. Серия: Экономика. 2016. № 3 (51). С. 47-53.

15. Лихтарович А. В. Электронный документооборот и его основные аспекты // Международный журнал прикладных и фундаментальных исследований. 2017. № 9-2. С. 317-319.

16. Маркина О.В. Электронный документооборот: понимание и внедрение. М.: ООО "Издательство "Арс-Пресс", 2015.

17. Медведева Е. Н. Информационные технологии в управлении документооборотом // Информационно-коммуникационные технологии в современном мире : материалы Международной научно-практической конференции (г. Томск, 17-19 апреля 2019 г.). Томск : Изд-во ТПУ, 2019. С. 153-156.

18. Николаева О.Ю. Электронный документооборот: теория и практика. М.: Омега-Л, 2015.

19. Новикова О. И., Мищерякова И.А. Организация электронного документооборота на предприятии. 2018. № 10 (106). С. 119-123.

20. Осипова Н.И. Электронный документооборот в управлении предприятием. - М.: Эксмо, 2015.

21. Панин В. О. Организация электронного документооборота в бухгалтерии: учебное пособие. М. : Юрайт, 2019. 196 с.

22. Рудометова А. А. Автоматизация процесса документооборота в медицинских учреждениях // Информационные технологии в образовании и науке : материалы Международной научно-практической конференции (г. Ярославль, 11-12 апреля 2019 г.). Ярославль : Изд-во ЯрГУ, 2019. С. 302-306.

23. Сироткин Н. И. Организация документооборота на основе электронных технологий // Информационные технологии в современном мире : материалы Международной научно-практической конференции (г. Томск, 17-19 апреля 2019 г.). Томск : Изд-во ТПУ, 2019. С. 260-263.

24. Соловьев, В. Н. Информационные технологии в документообороте / В. Н. Соловьев // Вестник Кемеровского государственного университета. 2019. № 3 (79). С. 16-21.

25. David Wilson, Role-Based Access Control in File Management Systems, 2020. P. 87-102.

26. Emily Davis, User Interface Design Considerations for File and Task Management Applications, 2023. P. 20-35.

27. Jennifer Brown, File Management Systems: A Comprehensive Review, 2022. P.112-128.

28. Michael Thompson, Security Measures for File and Task Management Systems, 2021. P. 112-125.

29. Sarah Johnson, Effective Task Assignment and Execution Strategies in File Management Systems, 2022. P. 45-60.