

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
GRADUATION THESIS

Исследование и разработка системы автоматизированной загрузки графиков
платежей

Обучающийся / Student Авраменко Антон Дмитриевич

Факультет/институт/кластер/ Faculty/Institute/Cluster факультет программной инженерии и компьютерной техники

Группа/Group P42141

Направление подготовки/ Subject area 09.04.04 Программная инженерия

Образовательная программа / Educational program Системное и прикладное программное обеспечение 2021

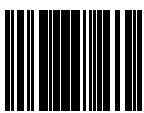
Язык реализации ОП / Language of the educational program Русский

Статус ОП / Status of educational program

Квалификация/ Degree level Магистр

Руководитель ВКР/ Thesis supervisor Балакшин Павел Валерьевич, кандидат технических наук, Университет ИТМО, факультет программной инженерии и компьютерной техники, доцент (квалификационная категория "ординарный доцент")

Обучающийся/Student

Документ подписан	
Авраменко Антон Дмитриевич	
28.05.2023	

(эл. подпись/ signature)

Авраменко
Антон
Дмитриевич

(Фамилия И.О./ name
and surname)

Руководитель ВКР/
Thesis supervisor

Документ подписан	
Балакшин Павел Валерьевич	
28.05.2023	

(эл. подпись/ signature)

Балакшин
Павел
Валерьевич

(Фамилия И.О./ name
and surname)

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University**

**ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ /
OBJECTIVES FOR A GRADUATION THESIS**

Обучающийся / Student Авраменко Антон Дмитриевич
Факультет/институт/кластер/ Faculty/Institute/Cluster факультет программной инженерии и компьютерной техники
Группа/Group P42141
Направление подготовки/ Subject area 09.04.04 Программная инженерия
Образовательная программа / Educational program Системное и прикладное программное обеспечение 2021
Язык реализации ОП / Language of the educational program Русский
Статус ОП / Status of educational program
Квалификация/ Degree level Магистр
Тема ВКР/ Thesis topic Исследование и разработка системы автоматизированной загрузки графиков платежей
Руководитель ВКР/ Thesis supervisor Балакшин Павел Валерьевич, кандидат технических наук, Университет ИТМО, факультет программной инженерии и компьютерной техники, доцент (квалификационная категория "ординарный доцент")

Основные вопросы, подлежащие разработке / Key issues to be analyzed

Техническое задание.

Разработать систему автоматизированной загрузки графиков платежей в приложение регламентированного учета. Для этого необходимо предпринять следующие шаги:

1. Проанализировать существующие программные средства, позволяющие взаимодействовать с пользовательским интерфейсом приложений.
2. Изучить процесс ручной загрузки графиков в приложение регламентированного учета, выявить необходимые входные параметры, а также логику поведения системы в случае возникновения ошибок.
3. Разработать систему автоматизированной загрузки графиков.
4. Разработать способ взаимодействия пользователей с разрабатываемой системой загрузки графиков.
5. Провести апробацию разработанной системы и определить, насколько разработанная система позволяет ускорить процесс загрузки графиков.

Исходные данные к работе.

1. Алгоритм ручной загрузки графиков в систему регламентированного учета «1С: Бухгалтерия».

Цель и задачи работы.

Целью ВКР является сокращение времени загрузки графиков лизинговых платежей в систему регламентированного учета за счёт использования разработанной системы

автоматизированной загрузки. Для достижения поставленной цели решаются следующие задачи:

1. Определить способ взаимодействия разрабатываемого программного комплекса с пользовательским интерфейсом целевых систем.
2. Исследовать алгоритм загрузки графиков в систему регламентированного учета «1С: Бухгалтерия», выявления необходимы данных для загрузки и способов обработки ошибок системы регламентированного учета.
3. Разработать систему автоматизированной загрузки графиков, а также пользовательский интерфейс для взаимодействия с разрабатываемой системой.
4. Провести пользовательское тестирование разработанной системы.

Перечень подлежащих разработке вопросов.

1. Определение особенностей программного взаимодействия с пользовательским интерфейсом.
2. Анализ существующих способов запуска не сопровождаемых программных роботов.
3. Выделение дополнительных способов запуска не сопровождаемых программных роботов.
4. Исследование существующего программного интерфейса центра управления роботами на предмет возможности использования его в качестве источника данных для пользовательского интерфейса запуска и мониторинга разрабатываемой системы.
5. Разработка и тестирование системы автоматизированной загрузки графиков платежей.

Рекомендуемые материалы и пособия для выполнения работы.

1. Балакшин П.В. Программное обеспечение для роботизированной автоматизации процессов // Альманах научных работ молодых ученых Университета ИТМО -2018. - Т. 2. - С. 245-248. URL: <https://elibrary.ru/item.asp?id=41375484/>.
2. Беломытцев И.О. Основные проблемы внедрения решений, основанных на роботизированной автоматизации процессов (RPA) // Инновационная наука -№4/2019. URL: <https://cyberleninka.ru/article/n/osnovnye-problemy-vnedreniya-resheniy-osnovannyh-na-robotizirovannoy-avtomatizatsii-protsessov-rpa>.

Форма представления материалов ВКР / Format(s) of thesis materials:

Схемы архитектуры системы; макеты интерфейса; снимки экрана приложения; презентация, оформленная в Microsoft Power Point

Дата выдачи задания / Assignment issued on: 10.10.2022

Срок представления готовой ВКР / Deadline for final edition of the thesis 30.05.2023

Характеристика темы ВКР / Description of thesis subject (topic)

Тема в области фундаментальных исследований / Subject of fundamental research: нет / not

Тема в области прикладных исследований / Subject of applied research: да / yes

СОГЛАСОВАНО / AGREED:

Руководитель ВКР/
Thesis supervisor

Документ подписан	
Балакшин Павел Валерьевич	

Балакшин
Павел

17.05.2023

(эл. подпись)

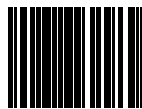
Валерьевич

Задание принял к
исполнению/ Objectives
assumed BY

Документ
подписан

Авраменко
Антон
Дмитриевич

17.05.2023



Авраменко
Антон
Дмитриевич

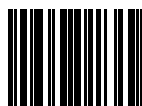
(эл. подпись)

Руководитель ОП/ Head
of educational program

Документ
подписан

Бессмертный
Игорь
Александрович

02.06.2023



Бессмертный
Игорь
Александрович

(эл. подпись)

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University**

**АННОТАЦИЯ
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ
SUMMARY OF A GRADUATION THESIS**

Обучающийся / Student Авраменко Антон Дмитриевич

Факультет/институт/кластер/ Faculty/Institute/Cluster факультет программной инженерии и компьютерной техники

Группа/Group P42141

Направление подготовки/ Subject area 09.04.04 Программная инженерия

Образовательная программа / Educational program Системное и прикладное программное обеспечение 2021

Язык реализации ОП / Language of the educational program Русский

Статус ОП / Status of educational program

Квалификация/ Degree level Магистр

Тема ВКР/ Thesis topic Исследование и разработка системы автоматизированной загрузки графиков платежей

Руководитель ВКР/ Thesis supervisor Балакшин Павел Валерьевич, кандидат технических наук, Университет ИТМО, факультет программной инженерии и компьютерной техники, доцент (квалификационная категория "ординарный доцент")

**ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ
DESCRIPTION OF THE GRADUATION THESIS**

Цель исследования / Research goal

Сокращение времени загрузки графиков лизинговых платежей в систему регламентированного учета за счёт использования разработанной системы автоматизированной загрузки

Задачи, решаемые в ВКР / Research tasks

1. Определение способа взаимодействия разрабатываемого программного комплекса с пользовательским интерфейсом целевых систем. 2. Исследование алгоритма загрузки графиков в систему регламентированного учета «1С: Бухгалтерия», выявления необходимых данных для загрузки и способов обработки ошибок системы регламентированного учета. 3. Разработка системы автоматизированной загрузки графиков, а также пользовательский интерфейс для взаимодействия с разрабатываемой системой. 4. Проведение пользовательского тестирования разработанной системы.

Краткая характеристика полученных результатов / Short summary of results/findings

В результате выполнения работы были исследованы системы, с которыми требуется взаимодействие, проанализирован и формализован алгоритм автоматизации, выявлены требуемые выходные и входные данные, возможные исключения и способы их обработки. Основываясь на алгоритме был разработан программный робот взаимодействующий с целевыми системами. Проанализированы существующие способы запуска программного

робота, а также предложен собственный способ запуска нивелирующий недостатки существующих подходов. Реализован предлагаемый способ запуска программных роботов и его пользовательский интерфейс. Реализованная система прошла пользовательское тестирование и была успешно внедрена в промышленную эксплуатацию. Разработанная система обеспечила сокращение времени загрузки графиков лизинговых платежей в 4.2 раза.

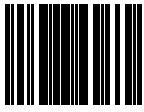
Наличие публикаций по теме выпускной работы / Publications on the topic of the thesis

1. Авраменко А.Д. Исследование и разработка способов запуска не сопровождаемых программных роботов//Сборник тезисов докладов конгресса молодых ученых. Электронное издание - 2023 (Тезисы)

Наличие выступлений на конференциях по теме выпускной работы / Conference reports on the topic of the thesis

1. XII Конгресс молодых ученых ИТМО, 03.04.2023 - 06.04.2023 (Конференция, статус - всероссийский)

Обучающийся/Student

Документ подписан	
Авраменко Антон Дмитриевич	
28.05.2023	

(эл. подпись/ signature)

Авраменко
Антон
Дмитриевич

(Фамилия И.О./ name
and surname)

Руководитель ВКР/
Thesis supervisor

Документ подписан	
Балакшин Павел Валерьевич	
28.05.2023	

(эл. подпись/ signature)

Балакшин
Павел
Валерьевич

(Фамилия И.О./ name
and surname)

Оглавление

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ.....	9
ВВЕДЕНИЕ.....	11
1. АНАЛИЗ ЦЕЛЕВЫХ СИСТЕМ И СПОСОБОВ ВЗАИМОДЕЙСТВИЯ С НИМИ.....	13
1.1. Обзор систем	13
1.2. Сравнительный анализ способов интеграции.....	16
1.3. Выводы к главе 1.....	22
2. ИССЛЕДОВАНИЕ АЛГОРИТМА ЗАГРУЗКИ ГРАФИКОВ В ЦЕЛЕВУЮ СИСТЕМУ УЧЁТА.....	23
2.1. Исследования ручного процесса загрузки графиков	23
2.2. Выявление возможных исключений и способов их обработки	32
2.3. Выявление входных значений и потоков данных	34
2.4. Финальный алгоритм автоматизации	35
2.5. Выводы к главе 2.....	37
3. РАЗРАБОТКА СИСТЕМЫ АВТОМАТИЗАЦИИ.....	38
3.1. Выработка требований к системе.....	38
3.2. Выбор инструментов	40
3.2.1. Определение вида взаимодействия с пользовательским интерфейсом	40
3.2.2. Определение способа запуска автоматизированного процесса	43
3.3. Реализация программного робота	46
3.4. Реализация пользовательского приложения	52
3.4.1. Общие сведения.....	52
3.4.2. Реализация взаимодействия с источником данных.....	53
3.4.3. Авторизация пользователей и определение уровня доступа к приложению.....	58
3.4.4. Реализация пользовательского интерфейса.....	60
3.5. Тестирование системы.....	64
3.6. Выводы к главе 3.....	65
4. АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ.....	66
ЗАКЛЮЧЕНИЕ	68

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	69
ПРИЛОЖЕНИЕ А (Листинг сервиса RPAAuthService)	72

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

ПО – программное обеспечение.

ОС – операционная система, программное обеспечение выполняющая управлением компьютером, позволяющая запускать программы решающие прикладные задачи.

UI (от англ. User Interface) – визуальный интерфейс приложения, отображающийся пользователю при взаимодействии с приложениями.

Горизонтальная масштабируемость – возможность увеличения производительности программной системы за счет увеличения количества машин, на которых данная система запускается.

API (от англ. Application Programming Interface) – программный интерфейс приложения, позволяющий взаимодействовать программным продуктам между собой и описывающий способ этого взаимодействия.

Валидация – в разработке UI проверка вводимых пользователем значений на соответствие ожидаемым шаблонам.

LowCode – подход к разработке программного обеспечения, характеризующийся минимальным задействованиям конструкций различных языков программирования.

Селектор – набор атрибутов и их параметров определяющий элемент пользовательского интерфейса, хороший селектор однозначно определяет элемент интерфейса среди других.

Двухфакторная аутентификация – подход к идентификации пользователя, при котором у пользователя запрашиваются

аутентификационные данные двух разных типов, например пароль и одноразовый код аутентификации.

TSL (от англ. transport layer security) – протокол защиты транспортного уровня, является криптографическим протоколом использует асимметричное шифрование, обеспечивает защищенную передачу данных между узлами сети.

Веб-хук – способ оповещения дочерней системы о произошедшем в родительской системе событии путем отправки HTTP/HTTPS запроса от родительской системы к дочерней.

HttpContext – объект ASP.NET MVC хранящий все данные о HTTP-запросе.

Web Socket – протокол позволяющий установить между серверной и клиентской частями приложений двухстороннюю связь, что в свою очередь обеспечивает возможность передачи сообщений как от клиента к серверу, так и от сервера к клиенту по инициативе сервера.

ВВЕДЕНИЕ

В современном мире электронный документооборот становится всё активнее внедряется в нашу жизнь. Так электронные документы используются во многих сферах: медицина, образование, продажи, финансовый и бухгалтерский учет, подача заявлений в государственные организации и многое другое. Так согласно исследованию LogiRus [1] объем электронного документа оборота в России за 2022 год вырос на 22% в сравнении с 2021 годом, а темпы внедрения систем электронного документа оборота среди юридических лиц и индивидуальных предпринимателей выросли на 26%.

Электронный документооборот, несомненно, имеет множество преимуществ, среди которых: экономия бумаги, возможность подписывать документы удаленно, что особенно ярко проявляется во время ограничительных мер, связанных с карантином.

Однако следует упомянуть, что если мы говорим о электронном документообороте, то он включает в себя не только возможность подписания документов электронной подписью, но и необходимость хранения всей документации в электронной форме, а иногда даже в нескольких никак не связанных с собой электронных системах. И именно это является одним из недостатков электронного документооборота, а именно необходимость переноса данных из одной системы в другую, что является монотонной работой, которая не требует от сотрудника организации никакой аналитики, однако занимает очень много времени сотрудника.

С вышеописанной проблемой столкнулась и компания заказчик разрабатываемого решения (крупная российская автолизинговая компания – далее в тексте ВКР будет обозначаться как «заказчик»), сотрудники которой

тратят много рабочего времени на перенос данных между системами заведения заявок, создания графиков лизинговых платежей и системой регламентированного учета графиков. Необходимость сокращения времени, затрачиваемого сотрудниками лизинговой компании на загрузку графиков в систему регламентированного учета, обуславливает **актуальность данной работы.**

Целью работы является сокращение времени загрузки графиков лизинговых платежей в систему регламентированного учета за счёт использования разработанной системы автоматизированной загрузки.

Для достижения цели работы были определены следующие задачи:

1. Определить способ взаимодействия разрабатываемого программного комплекса с пользовательским интерфейсом целевых систем.
2. Исследовать алгоритм загрузки графиков в систему регламентированного учета «1С: Бухгалтерия», выявления необходимы данных для загрузки и способов обработки ошибок системы регламентированного учета.
3. Разработать систему автоматизированной загрузки графиков, а также пользовательский интерфейс для взаимодействия с разрабатываемой системой.
4. Провести пользовательское тестирование разработанной системы.

1. АНАЛИЗ ЦЕЛЕВЫХ СИСТЕМ И СПОСОБОВ ВЗАИМОДЕЙСТВИЯ С НИМИ

1.1. Обзор систем

Для выявления наиболее подходящего способа взаимодействия с целевыми системами необходимо провести исследование целевых систем. Основными и единственными критерием, который будет оцениваться при анализе приложений, будет являться возможность и способы интеграции целевых приложений между собой, а также со сторонними приложениями и системами.

В перечень целевых систем, с которыми требуется интеграция, входят:

- Кредитный конвейер «БалансПлатформа»
- Система генерации целевых графиков «1С: Калькулятор»
- Целевая система регламентированного учета «1С: Бухгалтерия»

Ниже представлены более подробные описания целевых систем. Следует учесть, что все крупные коммерческие системы разрабатываются индивидуально под каждого целевого потребителя, поэтому описанные системы будут рассматриваться на примере конкретной системы, которая имеется у заказчика.

Кредитный конвейер «БалансПлатформа» [2] – является веб-приложением позволяющим заводить и отслеживать на всех этапах сделку автомобильного лизинга. Параметры сделки, хранимые в системе заказчика:

- Номер заявки на заключение договора лизинга, либо номер договора в случае успешного заключения

- Статус договора или заявки – текущее состояние договора лизинга, в зависимости от статуса договора/заявки отображаются остальные параметры.
- Даты договора/заявки, например: дата заключения, даты передачи транспортного средства, а также другие.
- Все данные договора, а также сам договор в электронном виде.

Ввиду того что изначально заказчиком не предполагалась интеграция данной системы с другими системами имеющаяся версия системы «БалансПлатформа» не предоставляет программного интерфейса, ввиду чего интеграция с данной системой возможна только с использованием пользовательского (визуального) интерфейса системы.

Система генерации целевых графиков «1С: Калькулятор» – является специально доработанной для заказчика версией приложения «1С: Предприятие 8.3». Данное решение является системой расчета лизинговых проектов компании, которые основываются на наборе параметров лизингового договора.

Результатом работы данной системы являются финансовые показатели договора лизинга, а также интересующие нас графики платежей и начислений. Ввиду уникальности системы она не имеет возможности интеграции с помощью программного интерфейса с другими программными продуктами, артефакты системы могут быть экспортированы из нее в виде текстовых файлов, либо в виде файлов в формате .xlsx/.xls.

Целевая система регламентированного учета «1С: Бухгалтерия» – представляет собой доработанную версию приложения «1С: Предприятие

8.3». Система используется внутри компании заказчика для ведения регламентированного учета.

Регламентированным учетом – является учет правела ведения которого закреплены на законодательном уровне и четко обозначены в учетной политике предприятия.

Под регламентированный учет компании заказчика попадает четкое ведение состояния лизинговой сделки, а именно фиксирование ключевых статусов, а также изменений условий оплаты сделки после установления одно из ключевых статусов договора. К перечню ключевых статусов и их изменений относятся следующие:

- Заключение договора;
- Досрочный выкуп (полный);
- Досрочный выкуп (частичный);
- Расторжение с изъятием;
- Изменение вида договора;
- Аннулирование;

При наступлении каждого из вышеперечисленных статусов в системе регламентированного учета создается изменение договора, а также к данному изменению прикрепляется графики платежей и начислений.

Если говорить о возможности программной интеграции данной системы с другими программными системами, то данная система имеет возможность импорта данных посредством загрузки в неё файлов в формате .scv. Однако стоит упомянуть, что для четкого соблюдения регламента предприятия в целевой системе предусмотрены ряд проверок

прикрепляемых графиков, а также то-что при импорте данных в систему данные проверки выполнятся не будет, что приводит невозможности использования инструмента импорта данных в целевую систему.

В результате анализа целевых приложений удалось выяснить, что ни одно из приложений не имеет возможность интеграции посредством программного интерфейса. Ввиду этого фактора единственным способом интеграции с целевыми приложениям остается способ использование пользовательского (визуального) интерфейса системы, а также использование файловой системы ОС.

1.2. Сравнительный анализ способов интеграции

Для корректного сравнения предлагаемого решения интеграции между приложениями были выделены следующие критерии:

- Время и объем работы необходимые на интеграцию
- Возможность централизованного управления и ведения логирования о работе
- Возможность горизонтального масштабирования

Ниже представлено подробное описание критериев:

Время и объем работы необходимый на интеграцию – количество времени, которое необходимо затратить разработчику на реализацию интеграции с целевыми приложениями. Данный критерий важен для коммерческой разработки поскольку время, затраченное на разработку системы прямо пропорционально затратам компании заказчика на

реализацию проекта. Кроме того, немаловажным фактором при коммерческой разработке является не только бюджет проекта, но и само время реализации поскольку в крупных организациях существуют планы реализации решений, которые четко ограничены по срокам. Нарушение сроков может привести к нарушению плана, что ведет к серьезным финансовым и в некоторых случаях репутационным потерям.

Возможность централизованного управления и ведения логирования о работе – является важным критерием поскольку именно централизованное управление и ведение логирования является лучшим инструментом для максимально быстрого решения нештатных ситуаций при эксплуатации целевой системы автоматизации. Что в свою очередь позволяет избавиться от нежелательных простоев системы автоматизации и сократить потери времени на исправление выявленных недоработок системы.

Горизонтальная масштабируемость – свойство системы, которое при увеличении количества запросов к системе дает возможность быстро увеличить производительность системы посредством увеличения количества узлов, обрабатывающих запросы. Применительно к целевой системе автоматизации предполагается, что при увеличении интенсивности загрузок графиков будет иметься возможность быстро увеличить производительность системы автоматизации чтобы обработать возросшую нагрузку.

Для проведения сравнения были выбраны следующие способы программного взаимодействия с пользовательским интерфейсом:

- Программный интерфейс взаимодействия на примере Microsoft UI Automation [3]. Данный способ предоставляется компанией Microsoft как универсальный способ взаимодействия с пользовательским интерфейсом доступный во всех

операционных системах, поддерживающих Windows Presentation Foundation (WPF)[4]. Данный способ взаимодействия дает возможность интерактивно взаимодействовать с пользовательским интерфейсом, а именно программными средствами симулировать пользовательское взаимодействие с интерфейсом. Однако, следует упомянуть, что для определения элементов графического интерфейса взаимодействие с которыми необходимо произвести взаимодействие необходимо знать дерево компонентов целевого пользовательского интерфейса (что является невозможным в рамках текущей задачи так как целевые приложения были разработаны третьими лицами), либо же использовать сторонние приложения (например Inspect [5]), позволяющее сканировать дерево UI компонентов приложения и выбирать интересующие разработчика элементы интерфейса. Данный способ поиска элементов интерфейса накладывает на разработчика обязанность переносить целевые селекторы между приложением сканирования UI компонентов и сходным кодом программы, что увеличивает вероятность ошибки, а также ведет к увеличению сроков разработки. При данном способе взаимодействия с элементами пользовательского интерфейса присутствует возможность реализации центрального управления средства автоматизации и логирование действий, однако реализация вышеперечисленных особенностей требует дополнительной разработки, причем разработка средства центрального управления является более трудозатратой, чем разработка целевого решения. Если говорить о возможности горизонтального масштабирования, то данная особенность также требует дополнительной разработки, а именно разработки системы очередности выполнения заявок, а также системы

балансировки нагрузки между узлами, выполняющими целевую автоматизацию.

- Роботизированная автоматизация процессов (RPA) – это цифровая технология, которая преимущественно использует сочетание пользовательского интерфейса и поверхностных функций для создания сценариев, которые автоматизируют рутинную работу. Иными словами, автоматизация бизнес-процессов с применением RPA позволяют проводить предсказуемую работу по переносу данных без участия человека. Инструменты RPA связывают приложения, устраняя ошибки ввода, ускоряя процессы и сокращая расходы. На сегодняшний день объем рынка RPA систем достигает 2.9 миллиарда долларов США [6], рост 19.5% год к году, что говорит обо всё большей заинтересованности бизнеса в данной технологии. На рисунке 1 представлен результат исследования производителей RPA платформ от исследовательской и консалтинговой компания Gartner [7], из него видно, что UiPath является несомненным лидером рынка.



Рисунок 1 – Magic Quadrant RPA

Ввиду лидирующей позиции UiPath, будет лучшим вариантом рассматривать функциональность RPA продуктов на примере продуктов именно этого бренда. Для взаимодействия с пользовательским интерфейсом и определения селекторов интересующих разработчика элементов используется утилита UiPath UIExplorer [8], которая позволяет по одному клику определять селектор искомого элемента и автоматически переносить его в программный код. Центральное управление компонентами UiPath, в также логирование является базовой функциональностью продукта и реализовано в UiPath

Orchestrator [9]. Благодаря UiPath Orchestrator также реализована масштабируемость решений, а именно этот инструмент выполняет роль балансировщика нагрузки, а также внутри него реализована система очередей заявок.

Ниже приведена сравнительная обобщающая сравнение способов взаимодействия с графическим интерфейсом, сравнение производится по ранее выработанным критериям.

Таблица 1 – Сравнительная таблица способов взаимодействия с пользовательским интерфейсом

Критерий	UI Automation	RPA
Время и объем работы	Высокий, требуется использование дополнительного ПО	Средний
Возможность централизованного управления и ведения логирования о работе	Требуется дополнительная разработка	+
Возможность горизонтального масштабирования	Требуется дополнительная разработка	+

В результате проведенного сравнительного анализа можно сделать вывод, что взаимодействие с графическим интерфейсом с помощью технологии RPA является предпочтительным ввиду наличия предварительно реализованной функциональности центрального управления и возможности масштабирования, а также более удобного для разработчика способа выбора селекторов элементов графического интерфейса. Ещё стоит заметить, что компания заказчик имеет приобретенные лицензии UiPath RPA на момент начала реализации решения, что также склоняет к выбору технологии RPA.

1.3. Выводы к главе 1

Основной целью данной главы был выбор наиболее подходящего способа взаимодействия с целевыми приложения, интеграцию с которыми необходимо обеспечить в рамках автоматизации целевого процесса. Для этого был проведен анализ целевых приложений, выявлено, что ни одно из рассматриваемых приложений не имеет программного интерфейса взаимодействия, ввиду этого интеграция с рассматриваемыми приложениями доступна только с использованием графического интерфейса этих приложений. Ввиду ограничения на способ взаимодействия с целевыми приложениями был проведен сравнительный анализ двух средств позволяющих взаимодействовать с графическим интерфейсом. Предварительно были выделены критерии относительно которых оценивались способы взаимодействия. В результате сравнения, а также принимая во внимание доступность ПО для заказчика, было выявлено, что наиболее предпочтительным способом взаимодействия с целевыми приложениями является технология RPA, а конкретно RPA продукт компании UiPath.

2. ИССЛЕДОВАНИЕ АЛГОРИТМА ЗАГРУЗКИ ГРАФИКОВ В ЦЕЛЕВУЮ СИСТЕМУ УЧЁТА

2.1. Исследования ручного процесса загрузки графиков

Процесс ручной загрузки графиков является последовательностью шагов, выполняемых сотрудником заказчика для загрузки графиков платежей в целевую систему регламентированного учета «1С: Бухгалтерия». Данная последовательность шагов была представлена заказчиком в устной форме с использованием визуальной демонстрации процесса сотрудником заказчика.

Ниже представлены шаги, продемонстрированные заказчиком, часть информации на представленных пояснительных изображениях скрыта с целью соблюдения соглашения о неразглашении.

Шаги процесса загрузки графиков:

Шаг 1. Открытие целевого договора лизинга в системе кредитного конвейера «Баланс Платформа».

Для выполнения этого шага необходимо открыть в браузере кредитный конвейер и выполнить авторизацию пользователя. После этого необходимо перейти во вкладку «Договоры», выполнить поиск и открытие искомого договора лизинга, открытие искомого договора лизинга происходит путем двойного клика на строчку, содержащую искомый договор. На рисунке 2 приведён пример демонстрирующий интерфейс кредитного калькулятора, а также поле поиска, требуемое для заполнения.

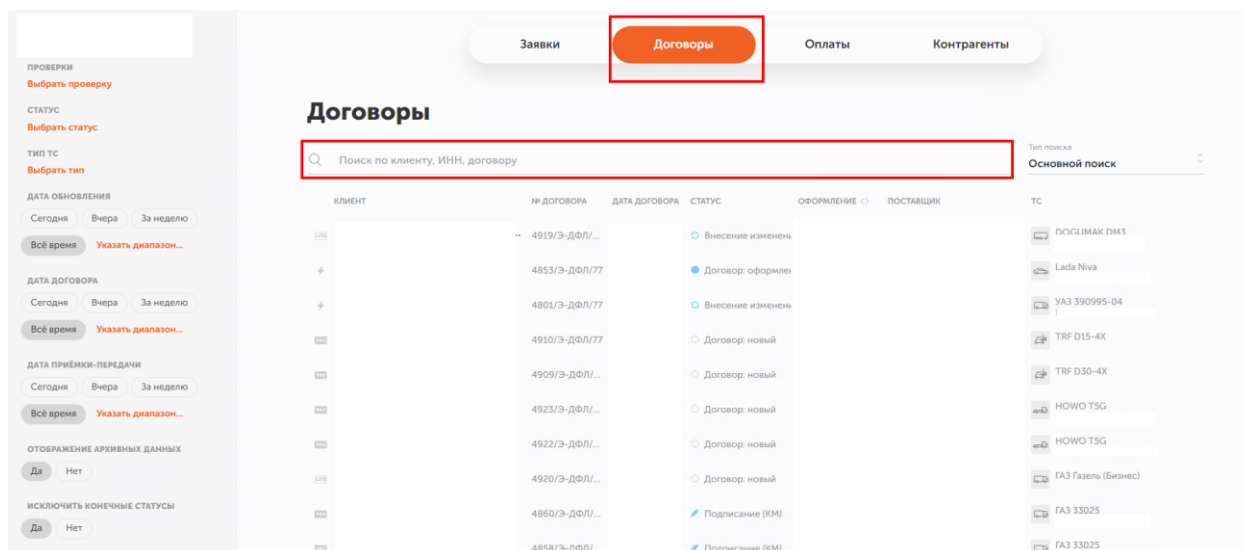


Рисунок 2 – Интерфейса поиска договора в кредитном конвейере

Шаг 2. Проверка статуса искомого договора и получение даты приемки передачи.

В данном шаге необходимо перейти на вкладку приемка «Приёмка-передача» договора (1), проверить статус (2, должна быть совершена фиксация – зеленая галочка напротив вкладки фиксация), также необходимо получить дату приемки передачи транспортного средства. На рисунке 3 приведён интерфейс вкладки «Приёмка-передача». Данный шаг происходит только если видом изменение указанным пользователем является «Приёмка-передача».

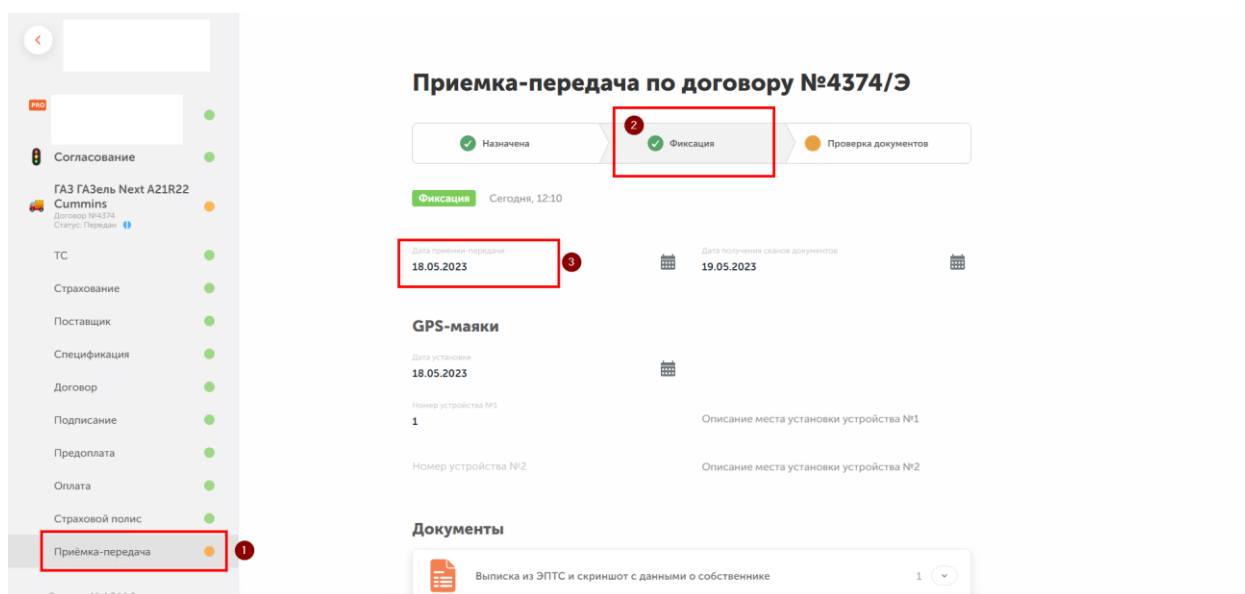


Рисунок 3 – Интерфейс договора вкладки «Приёмка-передача»

Шаг 3. Получение номера расчета договора и особенностей договора.

На данном шаге необходимо перейти во вкладку «ТС» и получить значения флага «Валютная сделка» (1) и флага «Субсидия» (2). На рисунке 4 приведён интерфейс вкладки «ТС» и отображены интересующие флаги.

Рисунок 4 – Интерфейс вкладки «ТС»

Далее на этом шаге необходимо на вкладке «ТС» получить номер расчета, который содержит в себе искомый график лизинговых платежей. На рисунке 5 приведён интерфейс вкладки «ТС», отображено поле номер расчета. Стоит отметить, что заказчик пожелал иметь возможность ручного ввода номера расчета пользователем системы автоматизации.

Рисунок 5 – Интерфейс вкладки «ТС», поле номера расчета

Шаг 4. Поиск расчета в системе генерации целевых графиков «1С: Калькулятор»

Для выполнения данного шага необходимо открыть базу «1С: Калькулятор». И выполнить поиск расчета по номеру, полученному на шаге 3. После поиска необходимо открыть расчет двойным кликом по строке искомого расчета. Интерфейс приложения «1С: Калькулятор», а также поле поиска представлено на рисунке 6.

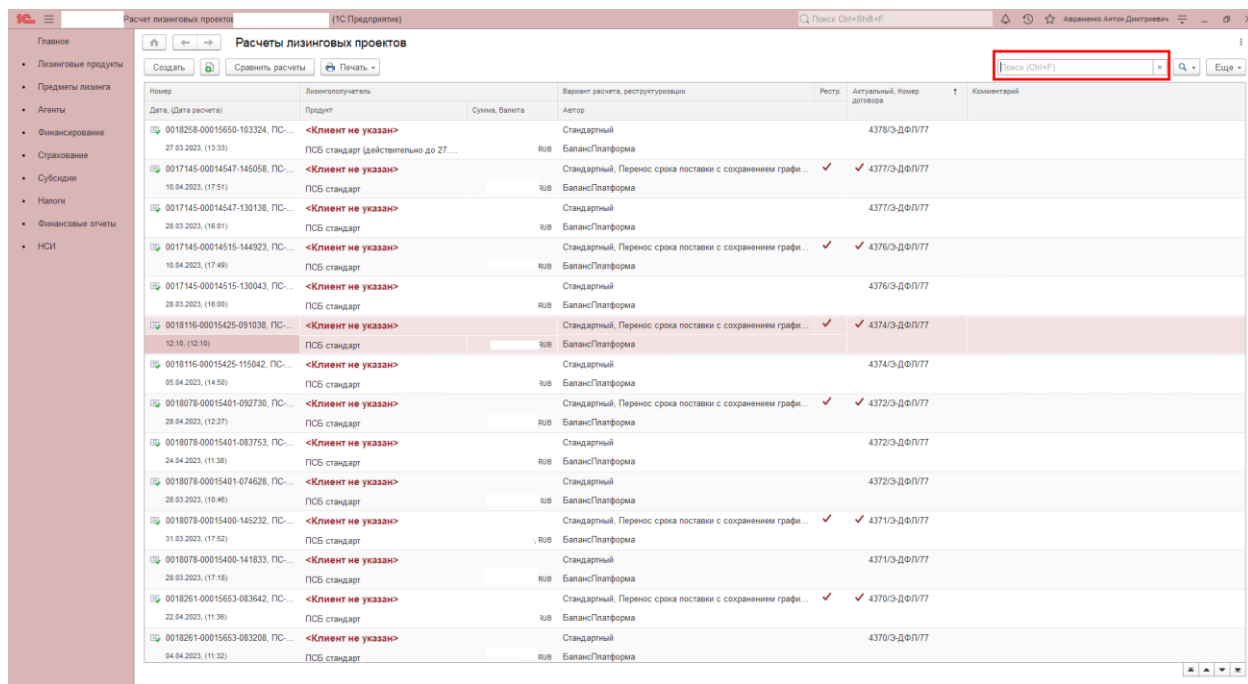


Рисунок 6 – Интерфейс системы генерации целевых графиков «1С: Калькулятор»

Шаг 5. Открытие целевых графиков

Этот шаг предусматривает последовательность действий в интерфейсе системы «1С: Калькулятор», а именно:

1. Открытие раздела «Финансовые отчеты» (рисунок 7)
2. Открытие раздела «ФСБУ_тест» (рисунок 8)
3. Сохранение целевого графика нажатием по кнопки сохранить и выбором целевого каталога сохранения (рисунок 9)

Рисунок 7 – Основной интерфейс искомого расчета

Рисунок 8 – Интерфейс вкладки «Финансовые отчеты»

Рисунок 9 – Интерфейс раздела «ФСБУ_тест», кнопка сохранить

Шаг 6. Редактирование целевого графика платежей

На данном шаге необходимо произвести редактирование и очистку целевого графика платежей для соблюдения регламентов компании заказчика. В редактирование графика входят следующие пункты:

- Удаление пустых строк графика. Строка считается пустой если в ней пусты все ячейки, либо же все ячейки кроме ячейки с датой.
- Проверка, есть ли в разных строках даты, которые полностью совпадают, если есть, складываются ячейки в этих строках, для ячейки даты используется поздняя из представленных дат;
- Проверка, есть ли в разных строках поля с датами, где совпадают месяц и год, если есть, использует самую позднюю из представленных дат;

Шаг 7. Сравнение итоговых сумм графика и договора лизинга

На этом шаге необходимо сравнить «Сумму лизинговых платежей», «Авансовый платёж» и «Выкупной платёж», представленные в целевом графике, полученном на шаге 4, с теми же величинами, представленными в договоре финансового лизинга. Если же на шаге 3 было выяснено, что договор является валютным или же он был субсидирован то данный шаг не выполняется.

Шаг 8. Открытие договора лизинга в целевой системе регламентированного учета.

На этом шаге необходимо открыть целевую систему регламентированного учета «1С: Бухгалтерия» и выполнить поиск искомого договора финансового лизинга по его номеру. Интерфейс целевой системы и поле поиска представлено на рисунке 10. Открытие найденного договора производится двойным кликом по строке, содержащей искомый договор.

Наименование	Код	Организация	Контрагент	Вид договора	Номер договора	Дата	Валюта
№4361/З-Дел/77 с	000004373	ООО	И	Лизинг	4361/З-Дел/77		руб.
№4362/З-Дел/77 с	000004374	ООО	И	Лизинг	4362/З-Дел/77		руб.
№4363/З-Дел/77 с	000004375	ООО	И	Лизинг	4363/З-Дел/77		руб.
№4364/З-Дел/77 с	000004558	ООО	И	Лизинг	4364/З-Дел/77		руб.
№4365/З-Дел/77 с	000004348	ООО	И	Лизинг	4365/З-Дел/77		руб.
№4366/З-Дел/77 с	000004347	ООО	И	Лизинг	4366/З-Дел/77		руб.
№4367/З-Дел/77 с	000004346	ООО	И	Лизинг	4367/З-Дел/77		руб.
№4368/З-Дел/77 с	000004345	ООО	И	Лизинг	4368/З-Дел/77		руб.
№4369/З-Дел/77 с	000004230	ООО	И	Лизинг	4369/З-Дел/77		руб.
№437/З-Дел	000000579	ООО	И	Лизинг	437/З-Дел		руб.
№4370/З-Дел/77	000004344	ООО	И	Лизинг	4370/З-Дел/77		руб.
№4371/З-Дел/77	000004244	ООО	И	Лизинг	4371/З-Дел/77		руб.
№4372/З-Дел/77	000004243	ООО	И	Лизинг	4372/З-Дел/77		руб.
№4374/З-Дел/77	000004362	ООО	И	Лизинг	4374/З-Дел/77		руб.
№4376/З-Дел/77	000004314	ООО	И	Лизинг	4376/З-Дел/77		руб.
№4377/З-Дел/77	000004313	ООО	И	Лизинг	4377/З-Дел/77		руб.
№4378/З-Дел/77	000004325	ООО	И	Лизинг	4378/З-Дел/77		руб.
№4379/З-Дел/77	000004363	ООО	И	Лизинг	4379/З-Дел/77		руб.
№438/З-Дел от	000000580	ООО	И	Лизинг	438/З-Дел		руб.
№438/З-Дел от	000000999	ООО	И	Лизинг	438/З-Дел		руб.
№439/З-Дел от	000000581	ООО	И	Лизинг	439/З-Дел		руб.
№4390/З-Дел/77	000004480	ООО	И	Лизинг	4390/З-Дел/77		руб.
№4392/З-Дел/77	000004236	ООО	И	Лизинг	4392/З-Дел/77		руб.
№4393/З-Дел/77	000004235	ООО	И	Лизинг	4393/З-Дел/77		руб.
№4394/З-Дел/77	000004234	ООО	И	Лизинг	4394/З-Дел/77		руб.
№4395/З-Дел/77	000004233	ООО	И	Лизинг	4395/З-Дел/77		руб.
№4396/З-Дел/77	000004251	ООО	И	Лизинг	4396/З-Дел/77		руб.
№4397/З-Дел/77 от 29.03.2023	000004242	ООО "ПСБ ЛЕЗИНГ"	ВЕЛЕС ООО ИФН 774...	Лизинг	4397/З-Дел/77	29.03.2023	руб.
№440/З-Дел от 26.04.2021	000000220	ООО "ПСБ ЛЕЗИНГ"	НР ЛОГИСТИК	Лизинг	440/З-Дел	26.04.2021	руб.
№440/З-Дел от 11.08.2021	000000593	ООО "ПСБ ЛЕЗИНГ"	КОМПЛЕКСНЫЙ ПОД...	Лизинг	440/З-Дел	11.08.2021	руб.
№4400/З-Дел/77 от 26.03.2023	000004348	ООО "ПСБ ЛЕЗИНГ"	ИП "СЕРГЕЙ ПИЩЕ"	Лизинг	4400/З-Дел/77	26.03.2023	руб.

Рисунок 10 – Интерфейс основной страницы системы регламентированного учета «1С: Бухгалтерия»

Шаг 9. Добавление изменения в договор лизинга

На данном шаге на основной странице договора лизинга необходимо перейти во вкладку история изменений (1), нажать кнопку «Добавить» (2) и выбрать целевой вид изменений договора лизинга (3). На рисунке 11 представлен интерфейс окна договора лизинга с пронумерованными действиями необходимыми для выполнения описываемого шага.

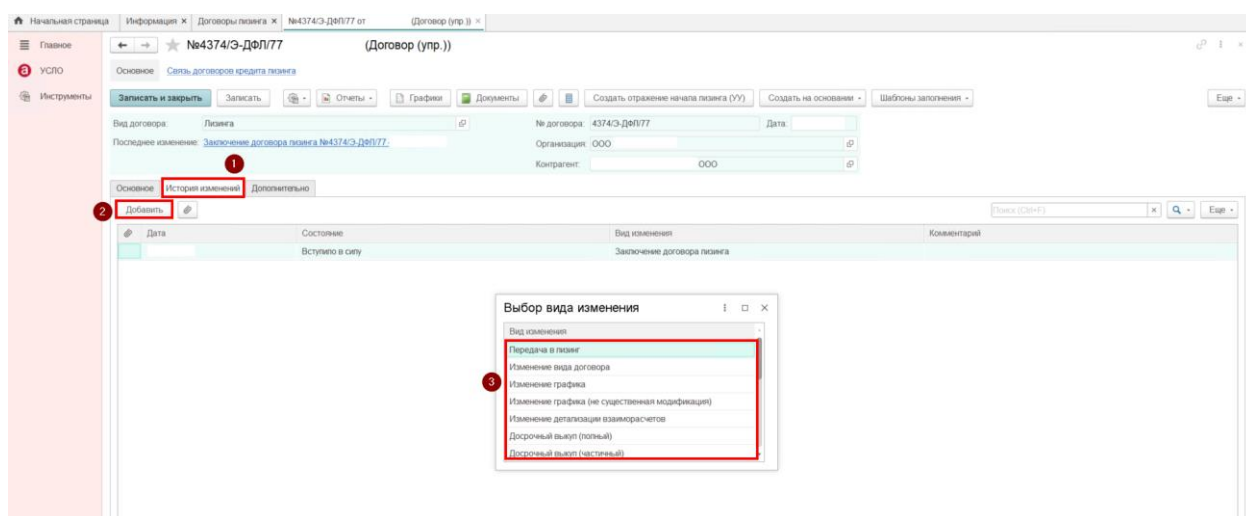


Рисунок 11 – Интерфейс страницы договора и добавления нового изменения

Шаг 10. Заполнения данных изменения

На данном шаге необходимо проверить, что созданное изменение является черновиком (1), заполнить дату изменения договора (2), проверить указан ли предмет графика в поле «Предмет лизинга» (3), если предмет графика не указан, указать предмет графика с наименованием «Основное приложение». Нажать кнопку «Загрузить графики» (4). На рисунке 12 приведен пример интерфейса изменения договора, а также обозначена последовательность действий в рамках шага 9.

The screenshot shows a web application interface for 'Вариант изменения договора контрагента (создание)'. It includes a top navigation bar, a left sidebar with 'Главное' and 'Инструменты', and a main content area. The main area contains several sections: 'Записать и закрыть', 'Дополнительно', 'Вид изменения', 'Договор', 'Статус процесса', 'Основное', 'Предмет лизинга', 'Графики', and 'Комментарий'. Red boxes and numbers highlight specific elements: (1) 'Черновик' status, (2) date field '10.05.2022, 0:00:00', (3) 'Основное приложение' in the 'Предмет лизинга' table, and (4) 'Загрузить графики' button.

N	Предмет графика	Номенклатура	% НДС	Сумма	НДС	Всего
1	Основное приложение		20%			

Рисунок 12 – Интерфейс окна «Изменение договора»

Шаг 11. Заполнение параметров графика

На данном шаге необходимо заполнить параметры графика. Заполнение параметров «Номер дополнительного соглашения» (1) и «Предмет графика» (2) изображены на рисунке 13. На рисунке 14 отмечены параметры «Имя выбранного файла» (1), «Имя страницы» (2), необходимые для заполнения, а также кнопка «Загрузить данные» (3), которую необходимо нажать для продолжения загрузки.

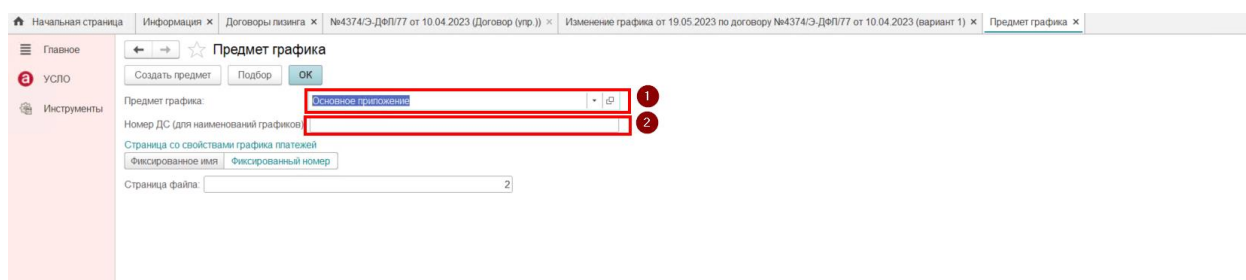


Рисунок 13 – Интерфейс параметров график

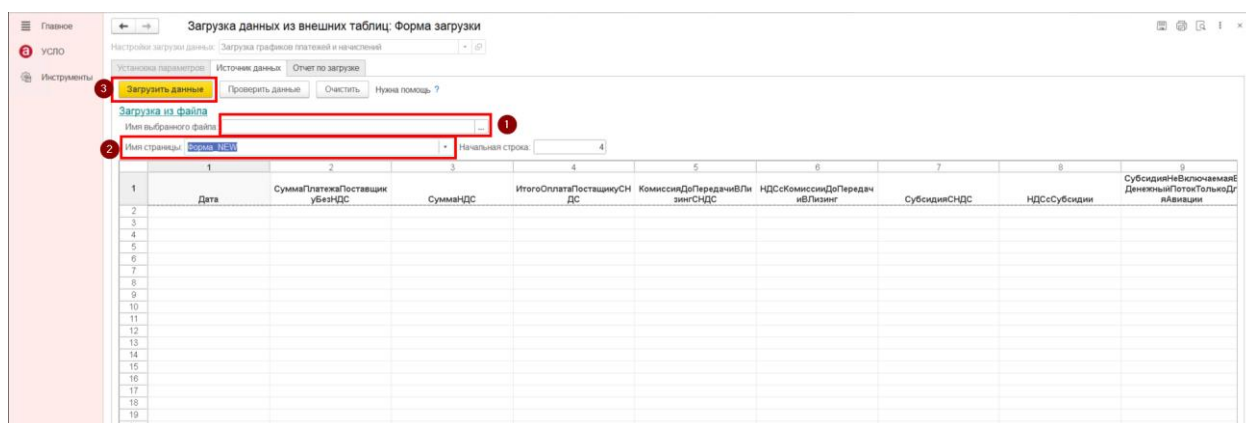


Рисунок 14 – Интерфейс выбора файла графика

Шаг 12. Получение результата загрузки графиков

После нажатия кнопки «Загрузить данные» система регламентированного учета производит проверку загружаемого графика и отображает результат проверки. Результатом проверки графика может быть 3 статуса:

- Ошибки отсутствуют – в загружаемом графике отсутствуют ошибки и график успешно загружен в систему регламентированного учета.
- Есть некритически ошибки – в графике обнаружены некритические ошибки, сотрудник может загрузить график невзирая на ошибки
- Есть критически ошибки – в графике обнаружены критически ошибки, загрузка графика невозможна

На рисунке 15 приведен пример обнаружения в графике критических ошибок.

1	2	3	4	5	6	7
Название проверки	Результат проверки	Описание проверки	Расшифровка проверки			
Дубли дат	Не пройдена [--]	Проверка считается успешно пройденной, если дубли дат в колонке 1/A «Дата» не найдено	Найдены строки с дублированием дат: 127, 128			
Даты по возрастанию	Не пройдена [--]	Проверка считается успешно пройденной, если: 1) Даты расположены по возрастанию; 2) Разница между датами двух соседних строк для каждой из колонок: 14N «Личный платеж с НДС», 15/O «НДС с личного платежа», 20T «Выкупной платеж с НДС», 21/U «НДС с выкупного платежа», 46/AZ «Начисление личного платежа с НДС», 51/AZ «Начисление выкупного платежа с НДС» меньше или равно 2 месяцам для «Автоплатежа», меньше или равно 3 месяцам для других личностных продуктов.	Нарушена хронология дат в строках: 129			
ЧЕКСК 8	Не пройдена [--]	Проверка считается успешно пройденной, если итоговая колонка 27/AA «Итого к начислению с НДС» = 0 (не допускается отклонение)	Выявлено отклонение: 4,27 - 4,08 = 0,19			

Рисунок 15 – Интерфейс отображения результата проверки загружаемого графика.

Шаг 13. Сохранение изменений договора.

На этом шаге необходимо провести сохранение всех внесенных в базу регламентированного учета изменений, для этого необходимо нажать кнопку «Записать и закрыть» в окнах: изменение договора, страница договора.

Все вышеприведённые шаги выполняются пользователем вручную, в связи с этим пользователь интерактивно реагирует на все возникающие в процессе загрузки графика нештатные ситуации и ошибки, получаемые от информационных систем. Поскольку механизм программной автоматизации не может анализировать ошибки, генерируемые целевыми системами, необходимо вывить все возможные исключения и способы их обработки.

2.2. Выявление возможных исключений и способов их обработки

Выявление возможных исключений происходило путем ручной загрузки графиков платежей в целевую систему разработчиком, а также дополнительными исследовательскими сессиями с заказчиком продукта. Выявленных исключения относятся к двум категориям технические ошибки

и ошибки бизнес-процесса. По согласованию с заказчиком продукта все выявленные исключения обрабатываются единообразно: информированием пользователя системы автоматизации, который в свою очередь информирует службу технической поддержки в случае возникновения технических ошибок, либо изменяет данные передаваемые системе автоматизации в случае возникновения ошибок, связанных с бизнес-процессом. Ниже приведен перечень возможных исключений, которые могут возникнуть в процессе выполнения автоматизации.

Технические исключения:

- Неверные учетные данные для авторизации в системе «Кредитный конвейер»;
- База «1С: Калькулятор» недоступна;
- База «1С: Бухгалтерия» недоступна;
- Созданное изменение договора находится в статусе «Актуальный»

Исключения, связанные с бизнес-процессом:

- В системе «Кредитный конвейер» не найден искомый договор
- В системе «Кредитный конвейер» не произведена фиксация статуса «Приёмка-передача»
- В системе «1С: Калькулятор» найдено более одного расчета с искомым номером
- В системе «1С: Калькулятор» не найдено расчета с искомым номером
- В системе «1С: Калькулятор» в указанном расчете представлен пустой график
- Несовпадение итоговых сумм в договоре финансового лизинга и в целевом графике

- В системе «1С: Бухгалтерия» найдено более одного расчета с искомым номером
- В системе «1С: Бухгалтерия» в искомом договоре не указан контрагент (без указания контрагента система не дает сохранить изменения договора)
- Системе «1С: Бухгалтерия» не распознала график в загружаемом файле.

2.3. Выявление входных значений и потоков данных

Для успешной автоматизации любого ручного процесса необходимо определить набор входных значений и других параметров необходимых к передаче между автоматизируемыми системами, так на основе раздела 2.1. данной работы был проведен анализ необходимых данных для совершения каждого шага, перечень необходимых данных для каждого шага приведены в таблице 2, шаги для которых не требуются, были пропущены.

Таблица 2 – перечень входных и выходных данных для каждого шага

Номер шага	Входные данные	Выходные данные
1	Номер договора лизинга;	-
2	Вид изменения;	Дата «Приёмки-передачи»
3	-	Номер расчета; Является ли договор валютной сделкой; Является ли договор сделкой с субсидией;
4	Номер расчета;	-
5	-	Целевой график
6	Целевой график	Целевой график
7	Целевой график	Целевой график
8	Номер договора лизинга;	-
9	Вид изменения договора лизинга;	-
10	Дата изменения договора лизинга	

11	Номер дополнительного соглашения; Целевой график;	
13	-	Результат обработки графиков системой регламентированного учета;

Опираясь на таблицу 2 и на шаги, выявленные в пункте 2.1. данной работы была создана диаграмма потоков данных, представленная на рисунке 16. Данная диаграмма отображает потоки данных, передаваемые между системами, задействованными в автоматизации, а также данные передаваемые и получаемые пользователем системы.

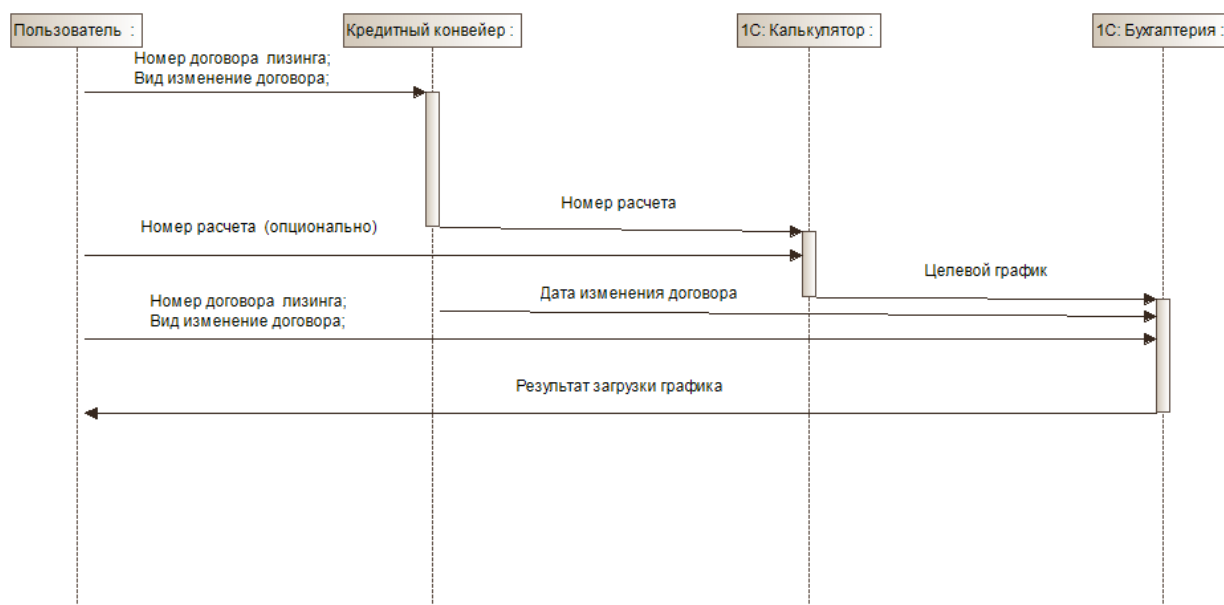


Рисунок 16 – диаграмма потока данных между целевыми системами и пользователем.

2.4. Финальный алгоритм автоматизации

На основании предыдущих разделов данной главы был построен финальный алгоритм, согласно которому требуется реализовать систему автоматизации. Блок-схема полученного алгоритма представлена на рисунке 17.

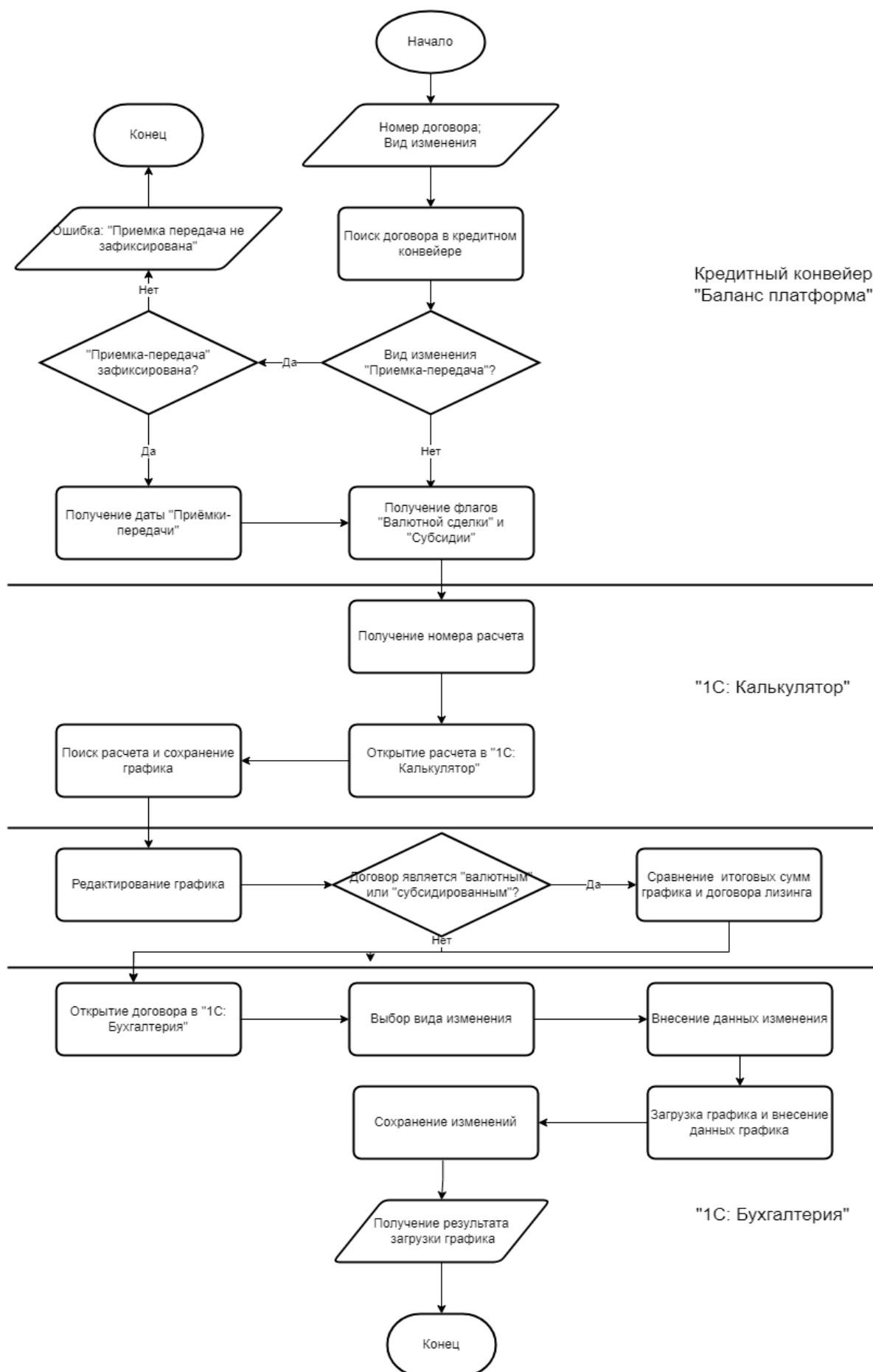


Рисунок 17 – Блок-схема алгоритма загрузки графиков.

2.5. Выводы к главе 2

Основной целью данной главы было разработать алгоритм, согласно которому должна работать система автоматизации. Для достижения данной цели была исследована последовательность ручных шагов сотрудников компании заказчика, выполняемых для загрузки графиков в целевую систему регламентированного учета. На основе предоставленным заказчиком ручных шагов были выявлены данные, передаваемые между интегрируемыми системами, а также получаемы от пользователя отдаваемые ему. Проанализировав ручные шаги, а также проконсультировавшись с заказчиком, были выявлены возможные исключения, которые могут произойти во время работы с целевыми системами, а также способы их обработки. На основе всех полученных данных был составлен финальный алгоритм взаимодействия с интегрируемыми системами.

3. РАЗРАБОТКА СИСТЕМЫ АВТОМАТИЗАЦИИ

3.1. Выработка требований к системе

Любое программное обеспечение, а рассматриваемая система автоматизации несомненно является программным обеспечением разрабатывается на основе требований, которые должны выполняться программным обеспечением. Требования к программному обеспечению вырабатываются разработчиком и заказчиком совместно, однако утверждает требования всегда заказчик продукта. Ниже представлены требования к системе автоматизации разработанные совместно с заказчиком и утвержденные им:

1. Система должна освободить пользователя от выполнения части его ручной работы, во время работы системы пользователь должен иметь возможность выполнять другие работы.
2. Система должна получать следующие данные: номер договора, финальные суммы договора лизинга непосредственно из файла договора лизинга.
3. Система должна выполнять валидацию вводимых пользователем данных, отображать некорректно введенные данные, не позволять запустить процесс автоматизации с недостаточным количеством данных или с некорректно введенными данными.

4. Система должна отображать пользователю текущий статус заявки отправленной системе автоматизации
5. Система должна отображать пользователю информацию о состоянии очереди заявок. А также все заявки всех пользователей, находящиеся в очереди, обрабатываемые и уже завершённые – это позволит пользователем системы исключить ситуации, когда несколько пользователей отправили на загрузку одинаковые заявки.
6. Система должна начинать работу по заявке пользователя сразу после поступления заявки, в случае если нет выполняющихся заявок, либо же добавлять заявку в очередь, если в данный момент выполняется другая заявка.
7. Система должна предоставлять пользователю возможность удалять из очереди заявки, обработка которых ещё не началась.
8. Система должна предоставлять пользователю полную информацию о результатах загрузки графиков платежей в целевую систему регламентированного учета, а также предоставлять пользователю информацию обо всех типах ошибок, которые произошли в процессе загрузки графика.
9. Система должна иметь сквозную авторизацию пользователей.

3.2. Выбор инструментов

Основываясь на выше представленных требованиях, будут выбираться технологии и продукты с помощью которых будут производиться разработка системы автоматизации.

3.2.1. Определение вида взаимодействия с пользовательским интерфейсом

Как было сказано в главе 1 способом взаимодействия с графическим интерфейсом интегрируемых приложений будет программные роботы от производителя UiPath.

UiPath предоставляет два вида программных роботов [10]:

- Сопровождаемые программные роботы
- Несопровождаемые программные роботы

Сопровождаемые программные роботы выступают в качестве личных помощников конечных пользователей, таких как персонал отдела кадров, операторы колл-центра и т. д. Поскольку автоматизация с участием должна работать под наблюдением человека, она лучше всего подходит для небольших, более фрагментированных задач.

Поскольку всегда присутствует пользователь-человек, данный вид роботов не должен использоваться для выполнения задач, которые сам пользователь не может выполнить (недостаточно прав пользователя для использования какого-либо ресурса). Любые учетные данные, необходимые во время выполнения контролируемого процесса, всегда должны быть

учетными данными, которые пользователь, запускающий автоматизацию, знает и предоставляет сам.

Не сопровождаемые программные роботы – это автономные роботы, которым не требуется надзор человека для выполнения задач. Они предназначены для сложных и часто повторяющихся задач, обычно требующих выполнения в пакетном режиме, которые можно решить на основе predetermined набора правил.

Основываясь на книге Alok Mani Tripathi *earning Robotic Process Automation* [11], были выявлены преимущества и недостатки каждого типа роботов.

Достоинствами сопровождаемых роботов являются:

- Интерактивность выполнения задачи – робот приступит к выполнению задачи сразу после отправки задачи сотрудником;
- Возможность наблюдения за ходом выполнения задачи – поскольку робот выполняет задачу непосредственно на машине сотрудника.

К недостаткам можно отнести:

- Невозможность сотрудником выполнять другие рабочие задачи вовремя выполнения задачи роботом;
- Необходимость лицензирования робота для каждого рабочего места, то есть для каждого сотрудника должна быть выделена лицензия.

Достоинствами не сопровождаемых роботов являются:

- Одним роботом могут пользоваться несколько сотрудников;
- Во время выполнения задачи роботом сотрудник может выполнять другие задачи.

К недостаткам можно отнести:

- Невозможность отследить статус задачи, отправленной на работа, до конца ее выполнения;
- Между отправкой задачи на работа и началом ее выполнения может пройти определенный период времени ввиду наличия очереди задач.

Стоит заметить, что все описанные преимущества и недостатки обоих видов программных роботов относятся к стандартным роботам, которые предоставляются производителем продукта.

Беря во внимание требование к системе автоматизации, выработанные в разделе 3.1, для создания целевой системы могут быть использованы только не сопровождаемые программные роботы. Данный вывод сделан исходя из того, что вместе с работой сопровождаемого программного робота параллельно на одной машине сотрудник не может выполнять другие задачи, что нарушает первое и самого главное требование к системе.

3.2.2. Определение способа запуска автоматизированного процесса

Исходя из документации по запуску не сопровождаемых роботов UiPath, робот может быть запущен следующими способами:

- В ручном режиме
- По расписанию
- После добавления заявки в очередь на обработку

Запуск несопровождаемых программных роботов в ручном режиме доступен только из интерфейса системы центрального администрирования роботов UiPath Orchestrator [9]. Поскольку рядовой пользователь разрабатываемого решения не должен иметь доступ к системе администрирования, данный способ запуска является неприменимым для решения поставленной задачи.

Запуск несопровождаемых программных роботов по расписанию определяет, что разрабатываемая автоматизация будет запускаться в раз один раз в определенный промежуток времени, что не соответствует требованию к системе о незамедлительном начале выполнения заявки после её добавления в очередь в случае отсутствия обрабатываемых заявок. Кроме того, рассматриваемый процесс требует постоянного изменения входных параметров обрабатываемых заявок, что невозможно ввиду отсутствия взаимодействия пользователя с механизмом запуска автоматизации.

Запуск несопровождаемых программных роботов по добавлению заявки в очередь на обработку предполагает использование «триггеров очереди» [12], которые срабатывают, когда в очереди появляется новая заявка, запущенный триггер запускает сконфигурированный в нем процесс, также данный вид триггера позволяет настроить количество одновременно

запускаемых роботов для задач одного типа, там самым одной настройкой администратор системы может обеспечить масштабируемость этой системы. Стоит заметить, что для срабатывания этого триггера, заявка в очередь должна быть добавлена какой-либо третьей стороной. Стандартные средства UiPath предполагают добавление и использование UiPath Apps [13] для добавления заявок в очередь, одна это решение не подходит для компании заказчика поскольку требует приобретения дополнительных лицензий. Также одной из рекомендаций UiPath является следование шаблону проектирования «Производитель-потребитель», в которой производителем является ещё один несопровождаемый программный робот. С данным роботом производителем взаимодействует пользователь посредством ресурса посредника (электронная почта, файловая система), информацию о результате выполнения автоматизации пользователь также получает средствами электронной почты. На рисунке 18 представлена высокоуровневая архитектура данного способа запуска целевого процесса автоматизации.

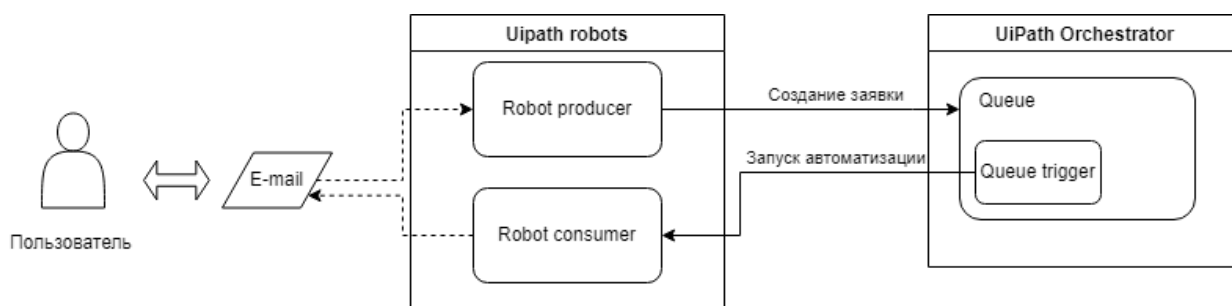


Рисунок 18 – Архитектура системы с несопровождаемым роботом производителем

Такой подход имеет перечень недостатков:

- Излишнее использование лицензий несопровождаемых роботов.
- Возможность взаимодействия пользователя с роботом производителем не напрямую, а через ресурсы посредники

(электронная почта, файловая система), что приводит к невозможности валидации входных параметров для задач, а также невозможности отображения, как текущего статуса заявки, так и всей очереди заявок, что является несоблюдением требований к разрабатываемой системе.

Предлагаемым вариантом запуска несопровождаемых роботов является использование API Orchestrator дополнительным веб-приложением, которое будет выполнять функции:

- Создания заявок
- Отображение статуса заявок
- Отображение результата загрузки графиков в целевую систему регламентированного
- Отображение общей очереди заявок и исторических данных

Высокоуровневая архитектура предлагаемого решения представлена на рисунке 19.

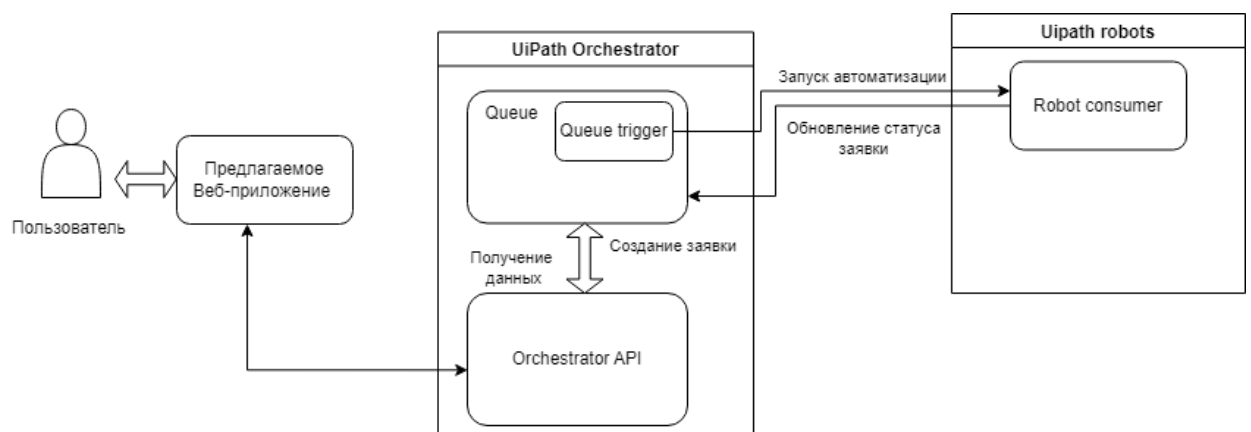


Рисунок 19 – Высокоуровневая архитектура предлагаемой системы

Как видно из архитектуры предлагаемого решения пользователь напрямую взаимодействует с интерфейсом системы автоматизации. Что

позволяет предоставить пользователю более удобный вариант взаимодействия с системой, позволяет сразу валидировать входные данные, введенные пользователем. Для данных, которые предполагают выбор из нескольких вариантов, дают возможность пользователю произвести выбор из заранее введенных полей, что сокращает вероятность пользовательской ошибки, кроме того, взаимодействие с оркестратором посредством API позволяет динамически отображать статусы и состояния заявок. Как видно из описания выше предлагаемый способ запуска автоматизации позволяет избавиться от всех недостатков стандартных способов запуска, предоставляемых производителем UiPath.

3.3. Реализация программного робота

Как было выявлено в предыдущих разделах программный робот будет реализоваться с использованием платформы UiPath. Платформа UiPath предоставляет из себя композит из трех неотъемлемых составляющих:

- UiPath Orchestrator – средство администрирования.
- UiPath Robot – среда выполнения программных роботов.
- UiPath Studio – средство создания программных роботов.

UiPath Studio [13] представляет из себя LowCode платформу с использованием которой создаются алгоритмы автоматизации. Данная платформа использует перечень активностей [14] как изначально предоставляемых производителем, так и есть возможность добавлять активности разработанные самостоятельно и прикрепленные к проекту в виде программных библиотек в формате .nupkg. Пример использования

активностей для шага 1 (Поиск договора в системе «Кредитный конвейер Баланс Платформа») приведен на рисунке 20.

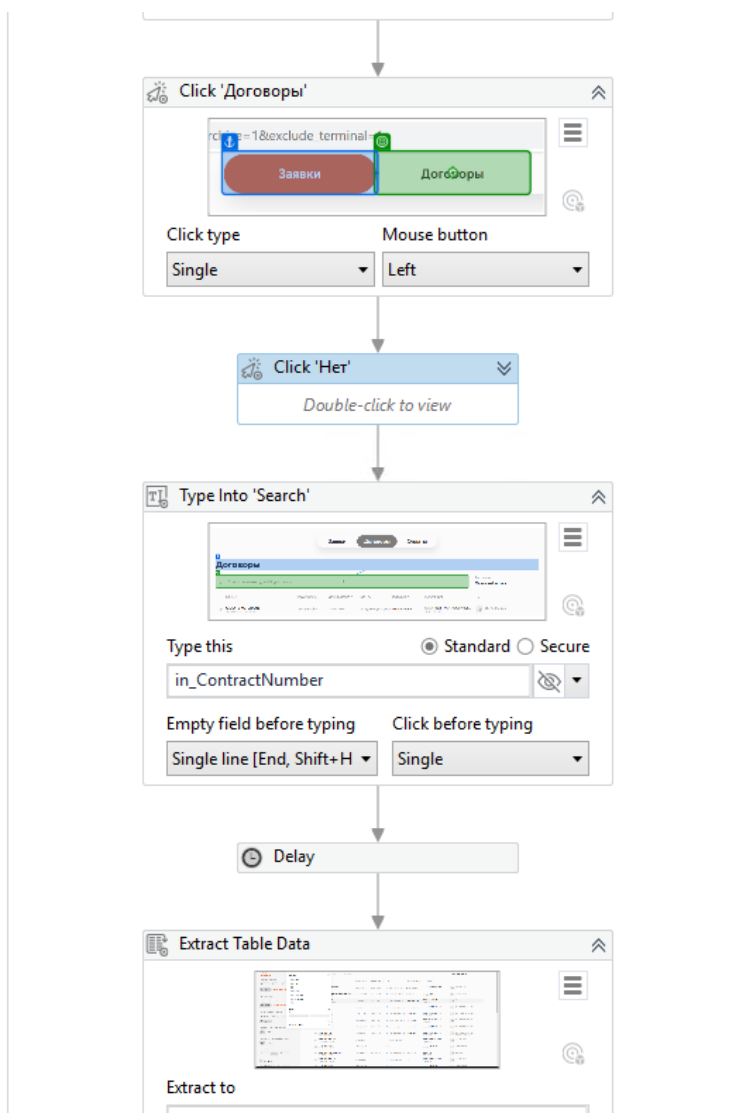


Рисунок 20 – пример использования активностей для введения номер договора

Как было сказано в разделе 3.2.2 разрабатываемый алгоритм автоматизации является «Потребителем» при использовании шаблона проектирования «Производитель-потребитель», ввиду этого для одинаковой обработки каждой заявки, добавленной в очередь, был использован конечный автомат представленный на рисунке 21.

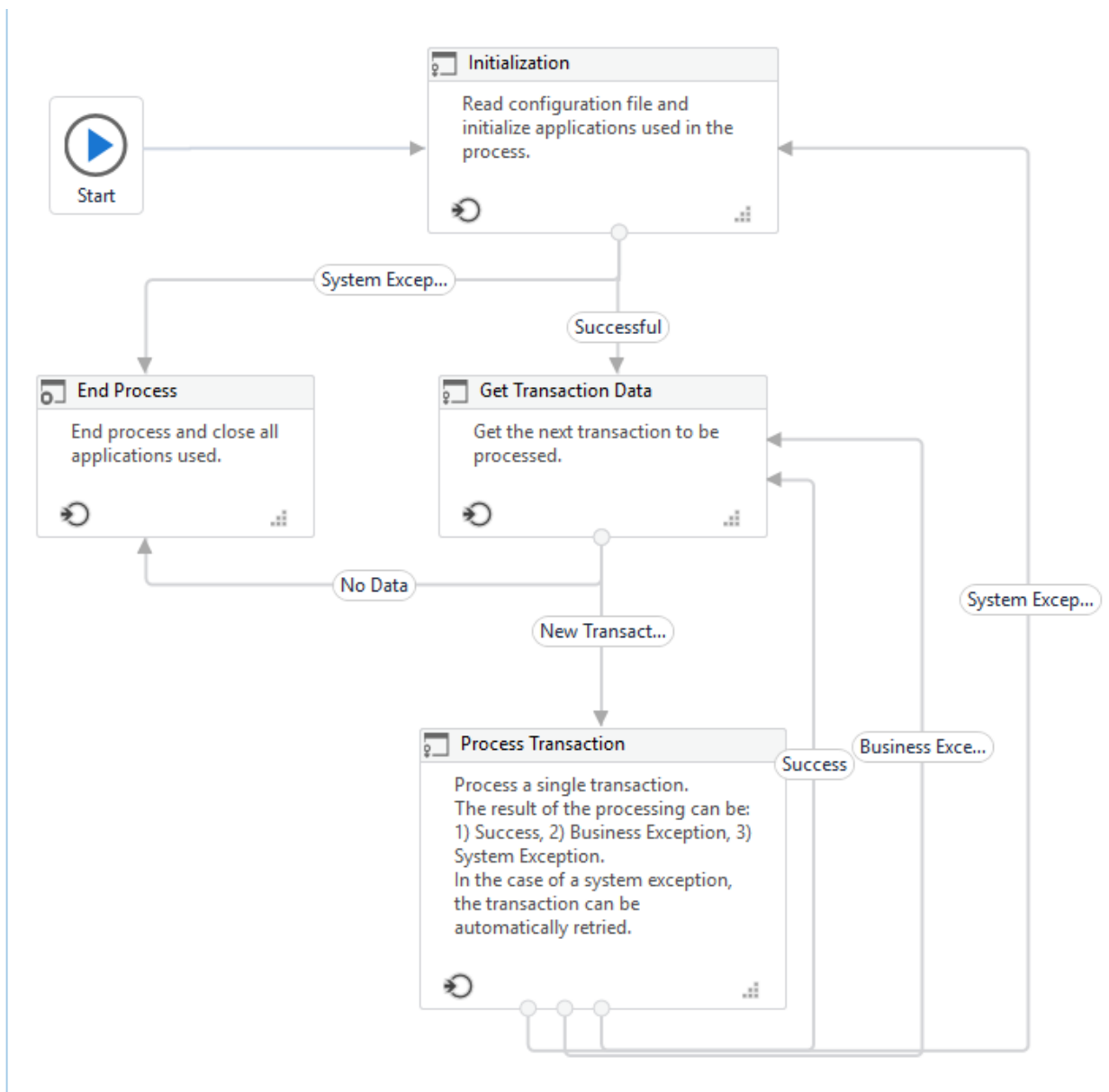


Рисунок 21 – конечный автомат, разработанный для процесса потребителя

Благодаря конечному автомату мы можем быть уверены, что при обработке каждой заявки начальные состояния приложений, активные приложение, внутренние переменные будут инициализированы одинаковым образом. Кроме того, такая архитектура процесса позволяет удобным для разработчика образом считывать все необходимые учетные данные, а также необходимые ресурсы (assets)[15] в блоке инициализации, также в случае необходимости добавления будет необходимо поменять только один файл, а для изменения данных нет необходимости менять код автоматизации, можно изменить данный из интерфейса администрирования. Все учетные

данные пользователей для доступа к автоматизированным системам хранятся в UiPath Orchestrator в зашифрованном виде, для получения данных из Orchestrator существует специальная активность (рисунок 22), кроме того, даже после получения данных из хранилища внутри процесса они хранятся в виде зашифрованной строки [16].

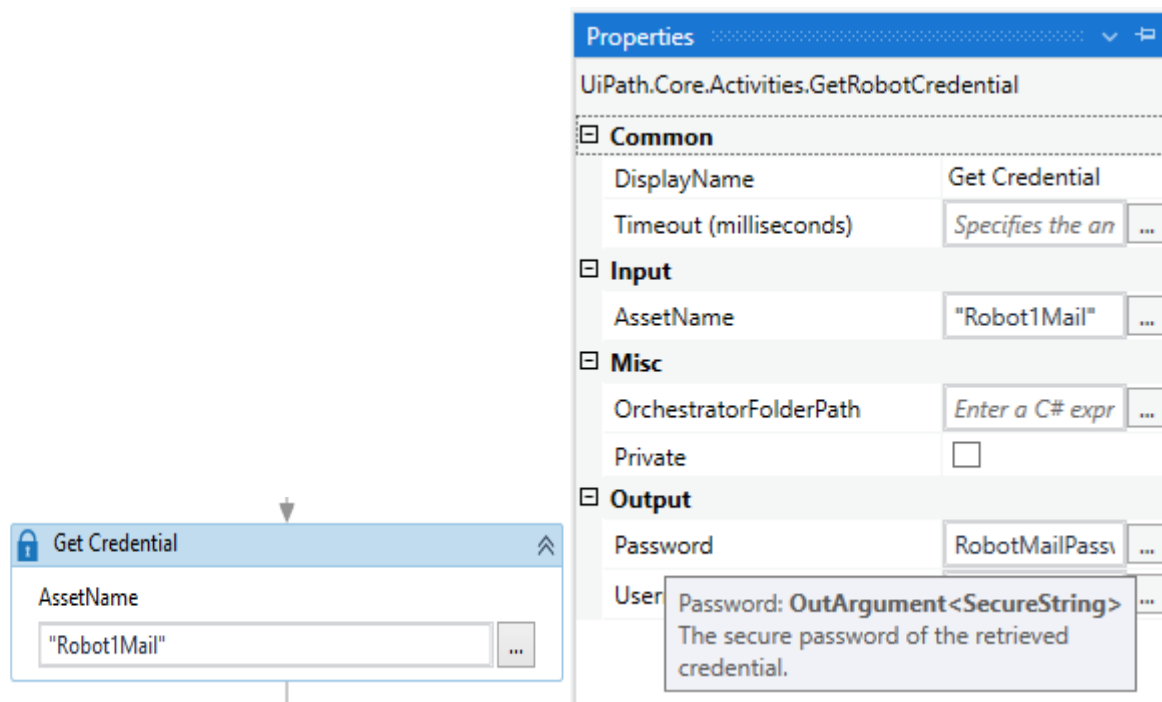


Рисунок 22 – активность получения учетных данных и ее параметры

В ходе разработки автоматизации, а именно интеграции с пользовательским интерфейсом были встречены некоторые особенности, о которых стоит упомянуть.

Заголовки окон приложений, построенных с использованием платформы «1С», не являются статическими и могут меняться как с выходом новой версии интегрируемой базы данных «1С», так и разработчиками этой базы вне зависимости от её версии. Ввиду этого в селекторы всех элементов необходимо было предоставить актуальны заголовок окна получаемый из ассетов, пример селектора с динамическим окна приведен в листинге 1.

```

<wnd app='1cv8c.exe' cls='V8TopLevelFrameSDI' title='{ {
in_AccountingWindowTitle } }' />
<uia name='История изменений' role='tab' />
<uia name='История изменений' role='tab' />
<uia role='tool bar' />
<uia name='Добавить' role='button' />

```

Листинг 1 – пример селектора с динамическим заголовком окна

Добавление предмета графика (Шаг 11) – при поиске искомого предмета среди существующих была встречена ситуация при которой при заведении предмета графика сотрудники заказчика допускали ошибки в наименовании предмета графика (лишние/недостающие пробелы, неверный регистр), для решение этой проблемы были применены нечувствительные к регистру селекторы, пример такого селектора приведен в Листинге 2, а также поиск наименования предмета графика получаемое роботом был проведен с помощью регулярных выражений, предварительно преобразовав наименование предмета графика к синтаксису регулярных выражение (Листинг 3). Данное преобразование позволяет сделать необязательными все пробелы, что нивелирует человеческий фактор во время создания предметов графика

```

<wnd app='1cv8c.exe' cls='V8DropWindowExWnd' idx='*' />
<uia role='list' />
<uia name='.{ {SubjectNameWithoutSpaces} }.?' role='list item'
matching:name='regex' casesensitive:name='false' />

```

Листинг 2 – селектор нечувствительный к регистру символов с использованием регулярных выражений

```

SubjectNameWithoutSpaces =
in_SubjectName.Replace("\", "\\").Replace("?", "\?").Replace(" ", "
?").Replace("(", "\(").Replace(")", "\)").Replace(".", "\.").Repla
ce("+", "\+").Replace("-", "\-

```

```
").Replace("[", "\\[").Replace("]", "\\]").Replace("{", "\\{").Replace("}", "\\}").Replace(" ", "\\ ").Replace("|", "\\|").Replace("№", " ?№ ?")
```

Листинг 3 – преобразование предмета графика к синтаксису регулярных выражений

Еще одной сложностью для автоматизации была авторизация пользователя робота в систему «Кредитный конвейер Баланс Платформа» ввиду использования системой двухфакторной аутентификацию. Для разрешения этой проблемы было установлено расширение браузера Authenticator Extension [17], которое позволяет получать код второго уровня доступа прямо из браузера, в котором была проведена попытка входа. Пример последовательности активностей, выполняющих получение кода второго уровня приведён на рисунке 23.

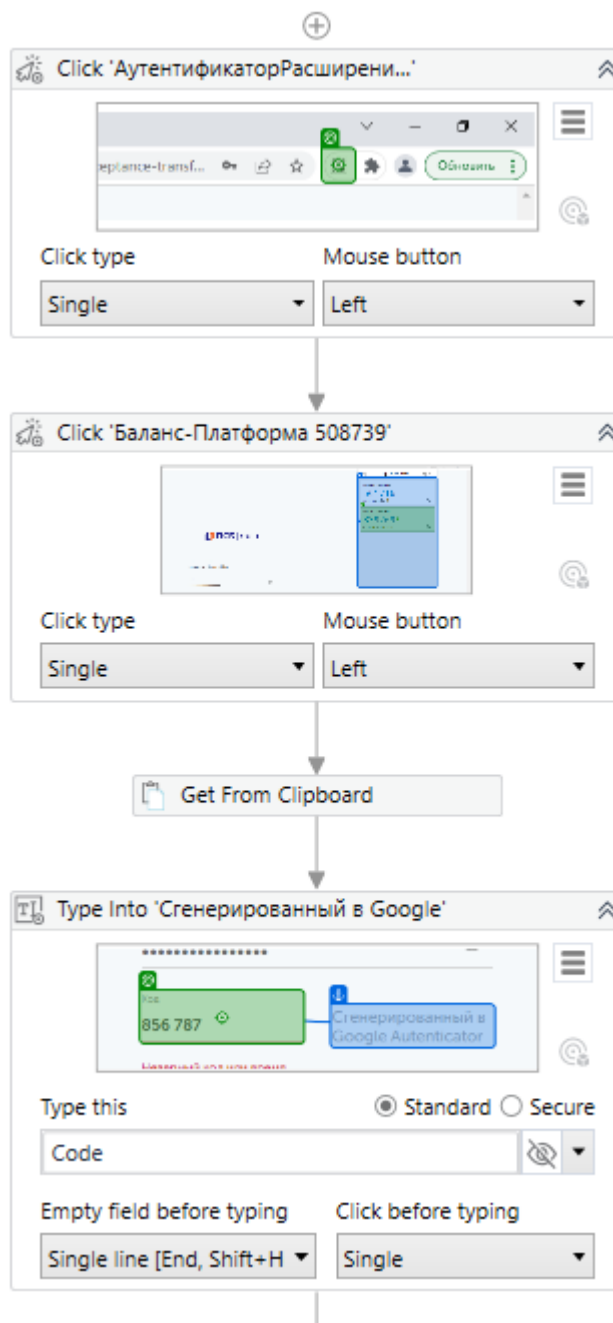


Рисунок 23 – последовательность активностей для получения кода второго уровня аутентификации

3.4. Реализация пользовательского приложения

3.4.1. Общие сведения

Пользовательское приложение является веб приложение, которое имеет клиент-серверную архитектуру, при такой архитектуре клиентская (она же интерфейс) часть реализует отображение визуальной составляющей,

а серверная часть производит непосредственно обработку запросов пользователя и выдачу данных из базы данных или другого аналогичного хранилища. Взаимодействие между частями приложения происходит с использованием протоколов HTTP [18] или HTTPS, который является расширением протокола HTTP за счет использования протокола.

Для реализации пользовательского интерфейса был использована платформа для создания одностраничных веб приложений Angular [19], выбор данной платформы разработки обусловлен нативной поддержкой платформой языка программирования TypeScript. TypeScript [20] является расширение языка JavaScript, ключевым преимуществом TypeScript является явное статическое указание типов данных, а также полноценная поддержка классов, по аналогии с объектно-ориентированными языками программирования. Поддержка полноценных классов позволяет создавать строгие модели данных, что удобно для соблюдения моделей данных предоставляемых API Orchestrator.

Для реализации серверной части разрабатываемого пользовательского приложения использовался фреймворк ASP .NET Core [20], который является инструментом для создания серверных частей веб приложений. Кроме того, данный фреймворк поддерживает нативное развёртывание с использованием веб-сервера IIS [21], что является преимуществом так как окружение заказчика уже имеет этот веб-сервер.

3.4.2. Реализация взаимодействия с источником данных

Как было сказано выше, веб-приложение состоит из двух компонентов, серверного клиентского и серверного компонента. Серверный компонент в свою очередь выполняет обязанности по получению данных из источника данных. В случае разрабатываемого приложения источником данных является база данных UiPath Orchestrator доступ к которой

осуществляется путем запросов к API. Взаимодействие с API сводится к нескольким основным запросам:

- Запрос на авторизацию пользователя API, данный запрос позволяет получить уникальную метку представителя для пользователя API, с помощью этой метки осуществляется авторизованный к остальным эндпоинтам API. В Листинге 4 представлена реализация запроса на авторизацию пользователя,

```
public async override Task<RPAAuth> Authenticate(Uri url, string
tenancyName, string login, string pswd)
{
    OrchAuth result = null;

    HttpClient client = _httpClientFactory.CreateClient();

    client.BaseAddress = url;

    JObject jbody = new JObject();
    jbody.Add("tenancyName", tenancyName);
    jbody.Add("usernameOrEmailAddress", login);
    jbody.Add("password", pswd);

    string body = jbody.ToString();

    var content = new StringContent
    (
        body,
        Encoding.UTF8,
        "application/json"
    );

    client.DefaultRequestHeaders.Accept.Add
    (
        new MediaTypeWithQualityHeaderValue("application/json")
    );
    var response = await client.PostAsync("/api/account/authenticate",
content);
    if (!response.IsSuccessStatusCode)
    {
        var res = await response.Content.ReadAsStringAsync();
        throw new Exception(res);
    }
    await response.Content.ReadAsStringAsync().ContinueWith
    (
        (Task<string> x) =>
        {
```

```

        if (x.IsFaulted)
            throw x.Exception;
        result = JsonConvert.DeserializeObject<OrchAuth>(x.Result);
    }
    );
    return (RPAAuth) result;
}

```

Листинг 4 – реализация запроса авторизации пользователя API

Методы представленные в Листинге 4 осуществляется из RPAAuthService, параметры для осуществления запроса сервис получает из конфигурационного файла приложения, полный листинг упомянутого сервиса представлен в Приложении А.

- Запрос на добавление элемента в очередь, позволяет добавить заявку, полученную от пользователя в очередь на обработку, после добавления заявки в очередь программный робот приступает к выполнению целевой автоматизации. Реализация запроса на добавление элемента в очередь представлена в Листинге 5.

```

public override async Task<string> AddQueueItem(string procName, IRequest
data)
{
    string result = "";

    string token = await _authService.GetTokenByProc(procName);
    RpaServiceEnum service =
    _config.FindProcess(procName).serviceName;

    HttpClient client = _httpClientFactory.CreateClient();

    client.BaseAddress = _config.GetBaseUrl(service);

    FolderService folderService =
    (FolderService)_servicesFactory.Resolve(service, ServiceType.FOLDERS);
    string folderResponse = await folderService.GetFolder(procName);

    JObject jobject = JObject.Parse(folderResponse);

    client.DefaultRequestHeaders.Add("X-UIPATH-OrganizationUnitId",
jobject["value"][0]["Id"].ToString());
    client.DefaultRequestHeaders.Add("Authorization", "Bearer " + token);
}

```

```

JObject body = new JObject();
JObject itemData = new JObject();
JObject specificContent = JObject.FromObject(data);
itemData.Add("Name", _config.GetQueueName(procName));
itemData.Add("Priority", "Normal");
itemData.Add("SpecificContent", specificContent);
itemData.Add("DeferDate", null);
itemData.Add("DueDate", null);
body.Add("itemData", itemData);

var content = new StringContent(
    body.ToString(),
    Encoding.UTF8,
    "application/json"
);

var response = await
client.PostAsync("/odata/Queues/UiPathODataSvc.AddQueueItem()",
content);
if (!response.IsSuccessStatusCode)
{
    var res = await response.Content.ReadAsStringAsync();
    throw new Exception(res);
}

await response.Content.ReadAsStringAsync().ContinueWith(
(
    (Task<string> x) =>
    {
        if (x.IsFaulted)
            throw x.Exception;

        result = x.Result;

    }
));

return result;
}

```

Листинг 5 – Запрос на добавление элемента в очередь

- Запрос на получение всех элементов очереди, позволяет получить актуальное состояние заявок в очереди при перезагрузке страницы пользователем, реализация запроса приведена в Листинге 6.


```

public override async Task<HistoryResponse>
GetAllQueueItemsInQueue(string procName)
{
    OrchestratorHistoryResponse result;

    string queueResponse = await GetQueueByName(procName);
    JObject jobject = JObject.Parse(queueResponse);

    var query = HttpUtility.ParseQueryString(string.Empty);
    query["$filter"] = "QueueDefinitionId eq " +
jobject["value"][0]["Id"].ToString();
    string queryString = query.ToString();

    result = (OrchestratorHistoryResponse) await
GetQueueItemsQuery(procName, queryString);

    if(result.TotalCount!= result.Items.Count)
    {
        int returnedPerRequest = 1000;
        int requestsToGo = (int)Math.Ceiling( (decimal)result.TotalCount /
returnedPerRequest) - 1;

        for( int i = 0; i < requestsToGo; i++)
        {
            query["$skip"] = (returnedPerRequest * (i + 1)).ToString();
            queryString = query.ToString();
            result.Items.AddRange( ((OrchestratorHistoryResponse) await
GetQueueItemsQuery(procName, queryString)).Items);
        }
    }
    return result;
}

```

Листинг 6 – Реализация запроса на получение всех элементов очереди

Также для взаимодействия с базой данных UiPath Orchestrator, а именно для получения актуальной информации о статусе обрабатываемой заявки, использовались веб-хуки. Веб-хуки в Orchestrator позволяют отправить информацию на указанный сетевой адрес о изменение состояния элемента очереди. Контроллер принимает запросы в серверном компоненте приложения и рассылает информацию о изменения состояния элемента очереди клиентским частям, для соединения серверной и клиентской части используется протокол WebSocket, в качестве идентификатора клиентской

части используется уникальный идентификатор пользователя, который также храниться в каждом элементе очереди.

3.4.3. Авторизация пользователей и определение уровня доступа к приложению

Авторизация пользователей, согласно требованиям к системе, должна происходить бесшовно, то есть пользователь не должен вводить свои учетные данные для доступа к системе. Ввиду этого требования авторизация пользователей производилась с использованием проверки подлинности NTLM. Данный вид проверки подлинности используется для сквозной аутентификации пользователя в домене Active Directory [22], таким образом если устройств, с которого осуществляется попытка аутентификации, находится в домене, автоматически осуществляется разрешается доступ при запросе к веб-приложению развернутому в IIS. Кроме того, при таком виде авторизации пользователя в HttpContext помещается доменное имя пользователя, по которому при обращении к домен-контроллеру можно получить принадлежность пользователя к подразделениям компании, что в свою очередь позволяет ограничить уровень доступа пользователя к приложению. В Листинге 7 приведён пример методов получения пользователя по его доменному имени, получение его адреса электронной почты, и получение подразделений предприятия, к которому относится рассматриваемый пользователь.

```
public UserModel GetUser(string userName)
{
    UserModel user = new UserModel(userName);
    user.Email = GetEmail(userName);
    getADGroups(ref user);

    return user;
}

public string GetEmail(string userName)
```

```

{
    string account = userName.Replace(@"####\", "");
    var entry = new System.DirectoryServices.DirectoryEntry();

    try
    {
        // get a DirectorySearcher object
        DirectorySearcher search = new DirectorySearcher(entry);

        // specify the search filter
        search.Filter = "(&(objectClass=user)(anr=" + account + "))";

        // specify which property values to return in the search
        search.PropertiesToLoad.Add("givenName"); // first name
        search.PropertiesToLoad.Add("sn");        // last name
        search.PropertiesToLoad.Add("mail");       // smtp mail address

        // perform the search
        SearchResult result = search.FindOne();

        if (result != null)
        {
            return result.Properties["mail"][0].ToString();
        }
        else
        {
            return "Unknown User";
        }
    }
    catch (Exception ex)
    {
        return ex.Message;
    }
}

private void getADGroups(ref UserModel userMod)
{
    List<string> result = new List<string>();

    // establish domain context
    PrincipalContext yourDomain = new PrincipalContext(ContextType.Domain);

    // find your user
    UserPrincipal user = UserPrincipal.FindByIdentity(yourDomain, userMod.Username);

    if (user == null) return;
    // if found - grab its groups
    userMod.DistinguishedName = user.DistinguishedName;

    PrincipalSearchResult<Principal> groups = user.GetAuthorizationGroups();

    // iterate over all groups

```

```
foreach (Principal p in groups)
{
    // make sure to add only group principals
    if (p is GroupPrincipal)
        result.Add(((GroupPrincipal)p).ToString());
}

userMod.Groups = result;
//return result;
}
```

Листинг 7 – определение пользователя и его подразделений

3.4.4. Реализация пользовательского интерфейса

Клиентская часть приложения реализована с использованием архитектуры одностраничного приложения (SPA, от англ. Single Page Application), что обеспечивает бесшовную работу интерфейса (нет необходимости каждый раз загружать страницу при переходе между страницами). Основываясь на требованиях к разрабатываемой системе, был разработан пользовательский интерфейс приложения.

На рисунке 24 приведен интерфейс стартовой страницы приложения, на котором отображается адрес электронной почты пользователя, от имени которого будут произведены все действия, а также область для выбора файла договора, загрузку которого необходимо произвести.

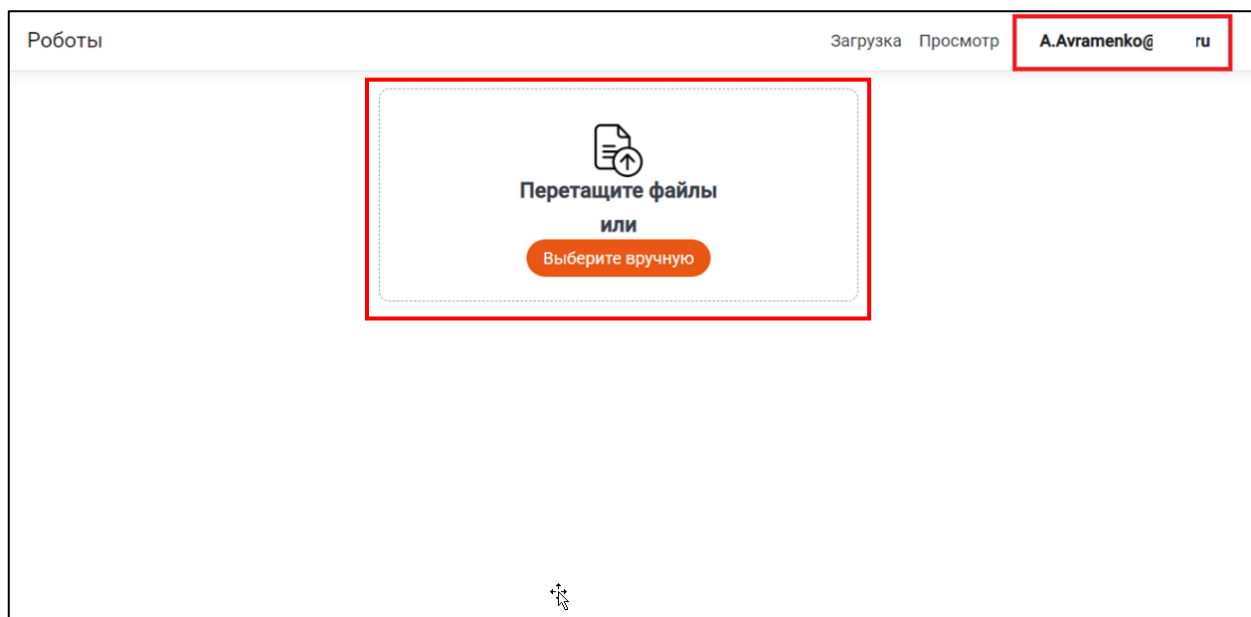


Рисунок 24 – Начальная страница приложения

На рисунке 25 изображен интерфейс заявки с отображением параметров, которые требуются от пользователя.

Рисунок 25 – Создание заявки на обработку

После заполнения параметров заявки и отправки ее на обработку заявка переходит в статус «В очереди», в течение обработки заявки меняется и ее статус. На рисунке 25 и рисунке 26 продемонстрированы статусы заявки «В очереди» и «Обрабатывается» соответственно.

В случае если во время обработки заявки произошла ошибка одного из трех представленных видов, пользователь имеет возможность кликнуть на статус заявки и ему отобразится подробная информация по произошедшей ошибке. На рисунке 28 представлено отображение «Критических ошибок» полученных после выполнения контрольных процедур в целевой системе регламентированного учета.

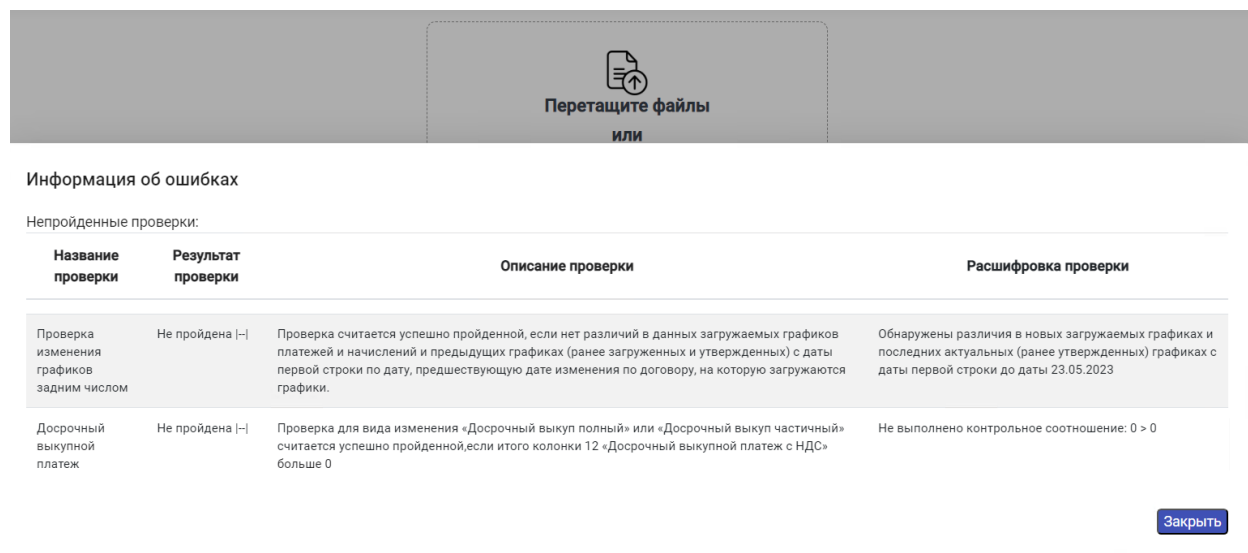


Рисунок 28 – Подробное описание критических ошибок

Также было реализовано отображение общей очереди заявок всех пользователей. Интерфейс общей очереди поддерживает полнотекстовый поиск по параметрам отправленной заявки, а также фильтрацию заявок по статусам и датам отправки. Реализованный интерфейс представлен на рисунке 28. Также на рисунке 29 можно увидеть кнопки скачивания отчета, при нажатии на кнопку «Скачать отчет» происходит формирование файла отчета в формате .xlsx, в сформированный отчет входят заявки, удовлетворяющие указанным фильтрам.

Фильтры ^

Дата отправки заявки

От До

Статус заявки

☒ Все
 ☐ Успешно
 ☐ Ошибка
 ☐ Критические ошибки
 ☐ Некритические ошибки

Процесс

Процесс
Auto

[Скачать отчет](#)
[Скачать отчет по всем заявкам](#)

Поиск...

Имя файла	Номер расчета	Дата	Вид изменения	Создатель	Дата создания	Статус
48593-ДФЛ.docx	-	-	Передача	PSBL\Anna.Vasilieva	25.05.2023	Успешно
48583-ДФЛ.docx	-	-	Передача	PSBL\Anna.Vasilieva	25.05.2023	Успешно
48603-ДФЛ.docx	-	-	Передача	PSBL\Anna.Vasilieva	25.05.2023	Успешно
48573-ДФЛ.docx	-	-	Передача	PSBL\Anna.Vasilieva	25.05.2023	Успешно
48563-ДФЛ.docx	-	-	Передача	PSBL\Anna.Vasilieva	25.05.2023	Успешно
ДС 1 к 47613-ДФЛ_пересчет.docx	ПС-000116391	17.05.2023	Передача	PSBL\Anna.Vasilieva	24.05.2023	Успешно
48513-ДФЛ.docx	-	-	Передача	PSBL\Anna.Vasilieva	24.05.2023	Успешно
48273-ДФЛ.docx	-	-	Передача	PSBL\Anna.Vasilieva	24.05.2023	Успешно
48283-ДФЛ.docx	-	-	Передача	PSBL\Anna.Vasilieva	24.05.2023	Успешно
ДС 1 к 38733-ДФЛ_пересчет.docx	ПС-000112903	16.05.2023	Передача	PSBL\Anna.Vasilieva	24.05.2023	Успешно

Рисунок 29 – Интерфейс общей очереди заявок

3.5. Тестирование системы

Тестирование разработанной системы производилось сотрудниками компании заказчика, которые будут пользоваться системой при промышленной эксплуатации. В ходе тестирования разработанной системы были сотрудниками заказчика было запущено 538 заявок, на основании которых были выявлены следующие недостатки и требуемые улучшения системы:

- Требуется добавить дополнительный необязательный входной параметр «Дата изменения», ввиду того что в системе

«Кредитный конвейер Баланс платформа» может быть неактуальная дата «Приемки-передачи».

- В программном роботе добавить обработку исключения целевой системы регламентированного учета, при котором дата изменения договора входит в запрещенный период, в результате чего невозможно сохранить внесенные изменения.

Выявленные недостатки системы были исправлены, после повторного тестирования разработанный продукт был выведен в промышленную эксплуатацию.

3.6. Выводы к главе 3

Основной целью данной главы была разработка системы автоматизации. Для достижения данной цели были выработаны требования к разрабатываемому продукту совместно с компанией заказчиком, были определены средства разработки, разработана система автоматизации. После разработки система автоматизации была протестирована, совместно с представителями заказчика были выявлены, а в последствие и исправлены недостатки системы. В результате проведения работ, описанных в данной главе, разрабатываемая система была выведена в промышленную эксплуатацию.

4. АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ

Как было сказано в главе 3 разработанная система была выведена в промышленную эксплуатацию. За всё время промышленной эксплуатации системы сотрудниками заказчика было отправлено 1497 заявок из которых 1179 были обработаны успешно. На рисунке 29 представлена диаграмма распределения статусов, с которыми были завершены загрузки заявок. Данные получены из отчета, сформированного системой.

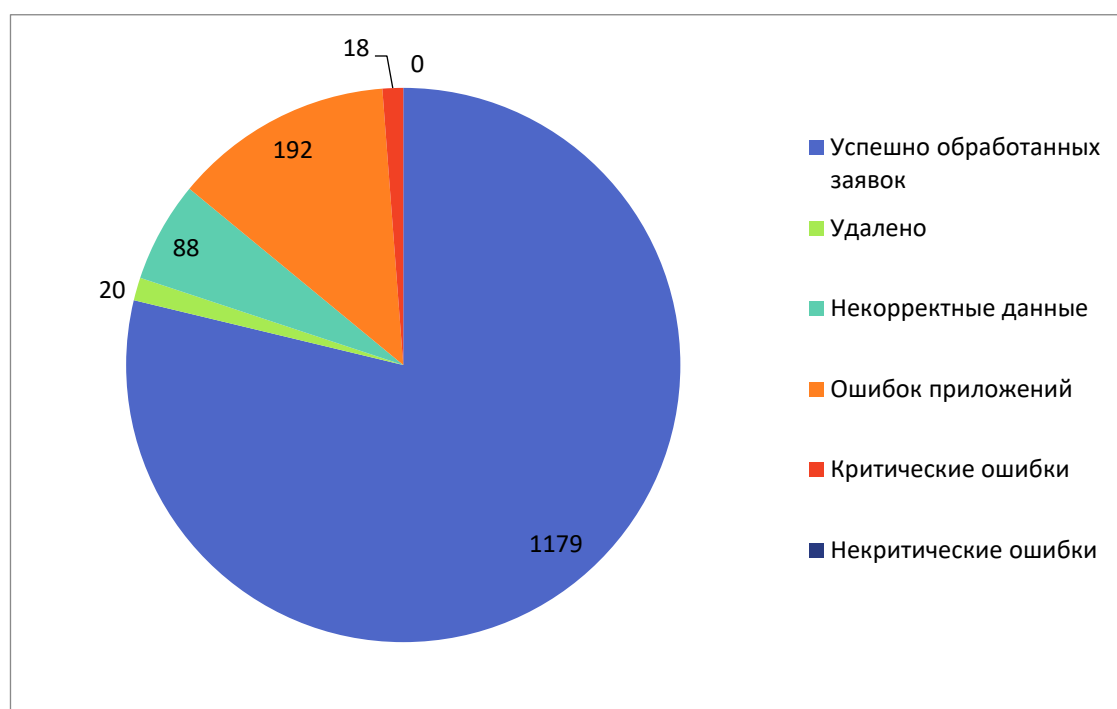


Рисунок 29 – Диаграмма распределения статусов завершения загрузок

Из диаграммы видно, что 79% загруженных заявок было обработано успешно, основная часть возникших ошибок (13%) связана с работой интегрируемых приложений, а именно их временной недоступностью. На ошибки связанный с соблюдением регламентов компании приходится 1% (критические ошибки) и 0% (некритические ошибки). Причем если загружать полученные из системы «1С: Калькулятор» графики вручную некритические ошибки проявляются достаточно часто, из чего можно сделать вывод, что разработанный алгоритм обработки графиков полностью разрешает некритические ошибки.

Согласно заявлению сотрудников компании заказчика на ручную загрузку одного графика лизинговых платежей уходит от 20 до 35 минут времени в зависимости от сложности графика, сгенерированного системой «1С: Калькулятор». Если рассматривать разработанную систему, то согласно отчету на успешную загрузку одного графика уходит 6 минут 34 секунды, что является ускорением загрузки в 4.2 раза. Однако во время загрузки графиков разработанной системой сотрудник компании может выполнять другие рабочие задачи, а на отправку заявки и анализ результата работы системы сотрудник тратит в среднем 90 секунд. Исходя из выше представленных данных можно сделать вывод, что внедрение разработанной системы на данный момент позволило сократить трудозатраты сотрудников компании заказчика приблизительно на 636 человеко-часов.

ЗАКЛЮЧЕНИЕ

В рамках данной работы была разработана система автоматизации загрузки графиков лизинговых платежей в систему регламентированного учета «1С: Бухгалтерия». Разработанная система успешно прошла приемочное тестирование заказчиком и была выпущена в промышленную эксплуатацию. Система используется компанией заказчиком и позволяет значительно сократить трудозатраты сотрудников заказчика, что свидетельствует о выполнении цели, поставленной перед данной работой.

По итогам внедрения системы можно сделать вывод о целесообразности использования программных роботов (RPA) для автоматизации комплексных бизнес-процессов требующих интеграции между собой нескольких информационных систем. Кроме того, предложенный способ запуска программных роботов прошел проверку промышленной эксплуатаций, что свидетельствует о его работоспособности.

Об успешности реализации работы также сообщает заинтересованность заказчика в расширении функциональности разработанной системы, а именно реализация загрузки дополнительных соглашений по договору лизинга и реализация загрузки графиков полученных непосредственно от пользователей, а также заинтересованность в автоматизации других бизнес-процессов, например: формирование дополнительных соглашений к договору лизинга и генерация отчета о контрагенте для сотрудников службы безопасности заказчика.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Объем электронного документа в РФ вырос почти на четверть [Электронный ресурс]. – 2023. – URL: https://logirus.ru/news/transport/obem_elektronnogo_dokumenta_v_rf_vyros_pochti_na_chetvert.html (дата обращения – 01.04.2023)
2. Кредитный конвейер [Электронный ресурс]. – 2022. – URL: <https://balance-pl.ru/conveer> (дата обращения – 02.02.2023)
3. UI Automation Overview [Электронный ресурс]. – 2021. – URL: <https://learn.microsoft.com/en-us/dotnet/framework/ui-automation/ui-automation-overview> (дата обращения – 11.10.2021)
4. Desktop Guide (WPF .NET) [Электронный ресурс]. – 2023. – URL: <https://learn.microsoft.com/en-us/dotnet/desktop/wpf/overview/?view=netdesktop-7.0> (дата обращения – 13.03.2023)
5. Accessibility tools – Inspect [Электронный ресурс]. – 2023. – URL: <https://learn.microsoft.com/en-us/windows/win32/winauto/inspect-objects> (дата обращения – 13.03.2023)
6. Gartner Says Worldwide RPA Software Spending to Reach \$2.9 Billion in 2022 [Электронные ресурс]. – 2022. – URL: <https://www.gartner.com/en/newsroom/press-releases/2022-08-1-rpa-forecast-2022-2q22-press-release> (дата обращения: 07.01.2023).
7. Derek Miers, Marc Kerremans, Saikat Ray, Cathy Tornbohm, Magic Quadrant for Robotic Process Automation Software [Электронный ресурс]. – 2019. –

URL: <https://nextgenmas.com/wp-content/uploads/2020/04/Gartner-Reprint.pdf> (дата обращения: 17.10.2022).

8. UIExplorer [Электронный ресурс]. – 2021. – URL: <https://docs.uipath.com/studio/standalone/2023.4/user-guide/uipath-explorer> (дата обращения: 17.11.2022).
9. Orchestrator [Электронный ресурс]. – 2022. – URL: <https://docs.uipath.com/orchestrator/automation-cloud/latest> (дата обращения: 18.11.2022).
10. UiPath Robots [Электронный ресурс]. — 2022. — URL: <https://docs.uipath.com/orchestrator/docs/about-robots> (дата обращения. 21.10.2022).
11. Alok Mani Tripathi. Learning Robotic Process Automation: Create Software robots and automate business processes with the leading RPA tool – UiPath, 2018, – 362с.
12. Queue Triggers [Электронный ресурс]. — 2022. — URL: <https://docs.uipath.com/orchestrator/standalone/2023.4/user-guide/queue-triggers> (дата обращения. 22.10.2022).
13. UiPath Studio [Электронный ресурс]. — 2022. — URL: <https://docs.uipath.com/studio/standalone/2021.10> (дата обращения. 23.10.2022).
14. UiPath Activity [Электронный ресурс]. — 2022. — URL: <https://docs.uipath.com/activities/other/latest> (дата обращения. 24.10.2022).

15. Assets [Электронный ресурс]. — 2022. — URL: <https://docs.uipath.com/orchestrator/standalone/2023.4/user-guide/about-assets> (дата обращения. 24.10.2022).
16. SecureString [Электронный ресурс]. — 2022. — URL: <https://learn.microsoft.com/ru-ru/dotnet/api/system.security.securestring?view=net-8.0> (дата обращения. 24.10.2022).
17. Authenticator extension [Электронный ресурс]. — 2022. — URL: <https://authenticator.cc/docs/en/overview> (дата обращения – 10.01.2023)
18. HTTP [Электронный ресурс]. — 2021. — URL: <https://developer.mozilla.org/ru/docs/Web/HTTP> (дата обращения – 15.03.2023)
19. Angular [Электронный ресурс]. — URL: <https://angular.io/docs> (дата обращения - 20.03.2023)
20. ASP.NET Core [Электронный ресурс]. — URL: <https://learn.microsoft.com/ru-ru/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-7.0> (дата обращения - 19.02.2023)
21. Веб-сервер IIS [Электронный ресурс]. — URL: <https://learn.microsoft.com/ru-ru/iis/get-started/introduction-to-iis/iis-web-server-overview> (дата обращения - 19.02.2023)
22. Active Directory [Электронный ресурс]. — URL: <https://learn.microsoft.com/ru-ru/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview> (дата обращения - 19.03.2023)

ПРИЛОЖЕНИЕ А

Листинг сервиса RPAAuthService

```
public class RPAAuthService
{
    // <RPASERVICE+"."+TENANCY_Type, obj with auth token>
    private Dictionary<string, RPAAuth> _tokens = new Dictionary<string,
RPAAuth> { };

    private ConfigWrapper _config;
    private APIServicesFactory _servicesFactory;
    private ReaderWriterLockSlim _rwLock;

    public RPAAuthService(APIServicesFactory factory, ConfigWrapper config)
    {
        _config = config;
        _servicesFactory = factory;

        _rwLock = new ReaderWriterLockSlim();
    }

    private RPAAuth FindValidToken(int timeout, string key)
    {
        if (_tokens.ContainsKey(key))
        {
            RPAAuth token = _tokens[key];
            DateTime now = DateTime.Now;

            // if 5 mins left to token expiration (according to config) -> refresh it
            if (token.creationTime.AddMinutes(timeout - 5) > now &&
                token.creationTime < now) // sanity check
                return token;
        }

        return null;
    }

    public async Task<string> GetTokenByProc(string procName)
    {
        return await GetTokenByService(_config.GetServiceRef(procName),
        _config.GetCurrTenantByProccess(procName).tenancyType);
    }

    public async Task<string> GetTokenByService(RpaServiceEnum
RPAProvider, TenancyTypeEnum tenancyType)
```



```

{
    RPAAuth token = null;
    int requestTimeout = _config.GetTimeout(RPAProvider);
    string searchKey = RPAProvider.ToString() + "." +
tenancyType.ToString();
    // search for token in shared dict
    if (_rwLock.TryEnterReadLock(30000))
    {
        try
        {
            token = FindValidToken(requestTimeout, searchKey);
        }
        finally
        {
            _rwLock.ExitReadLock();
        }
    }
    // valid token is not found in dict -> get new token
    if (token == null)
    {
        Uri baseUrl = _config.GetBaseUrl(RPAProvider);
        Tenancy ten = _config.FindServiceTenancy(RPAProvider,
tenancyType);

        var serv = (AuthService)_servicesFactory.Resolve(RPAProvider,
ServiceType.AUTH);
        token = await serv.Authenticate(baseUrl, ten.tenancyName, ten.login,
ten.pswd);
        _rwLock.EnterWriteLock();
        try
        {
            _tokens[searchKey] = token;
        }
        finally
        {
            _rwLock.ExitWriteLock();
        }
    }

    return token.GetToken();
}
}

```