Содержание

[Введение 3](#_Toc11252892)

[1 Анализ предметной области 4](#_Toc11252893)

[1.1 Описание предметной области 4](#_Toc11252894)

[1.2 Основные понятия 6](#_Toc11252895)

[2 Описание проблемы 7](#_Toc11252896)

[3 Обзор аналогов 8](#_Toc11252897)

[4 Программная документация 10](#_Toc11252898)

[4.1 Техническое задание на программное обеспечение 10](#_Toc11252899)

[4.1.1 Общие сведения 10](#_Toc11252900)

[4.1.2 Назначение и цели создания системы 11](#_Toc11252901)

[4.1.3 Требования к системе 11](#_Toc11252902)

[4.1.4 Требования к пользовательскому интерфейсу 12](#_Toc11252903)

[4.1.5 Черный ящик 17](#_Toc11252904)

[4.1.6 Сервер RabbitMQ 25](#_Toc11252905)

[4.1.7 Серверная часть программы 26](#_Toc11252906)

[4.1.8 Перспективы развития, модернизации системы 27](#_Toc11252907)

[4.1.9 Требования к информационному обеспечению системы 28](#_Toc11252908)

[4.1.10 Требования к программному обеспечению системы 28](#_Toc11252909)

[4.1.11 Требования к техническому обеспечению 29](#_Toc11252910)

[4.1.12 Требования к надежности 31](#_Toc11252911)

[4.1.13 Требования к безопасности 32](#_Toc11252912)

[4.1.14 Требования к метрологическому обеспечению 32](#_Toc11252913)

[4.2 Пояснительная записка к программному обеспечению 33](#_Toc11252914)

[4.2.1 Анализ предметной области 33](#_Toc11252915)

[4.2.2 Формулировка проблемы 33](#_Toc11252916)

[4.2.3 Обзор аналогов 33](#_Toc11252917)

[4.2.4 Назначение и область применения 33](#_Toc11252918)

[4.2.5 Технические характеристики 33](#_Toc11252919)

[4.3 Описание программы 42](#_Toc11252920)

[4.3.1 Общие сведения 42](#_Toc11252921)

[4.3.2 Функционально назначение 43](#_Toc11252922)

[4.3.3 Описание логической структуры 44](#_Toc11252923)

[4.3.4 Используемые технические средства 46](#_Toc11252924)

[4.3.5 Вызов и загрузка 46](#_Toc11252925)

# Введение

Современный мир постоянно развивается. Частью этого процесса является веб­­­‑технологии, отрицание влияния которых невозможно. Все сферы жизни человека, будь то образование, медицина или политика используют современные наработки программистов. Бизнес так же не остался в стороне.

Одно из множества его проявлений – страхование. Страховой рынок содержит большое количество профессионально работающих компаний и перестраховочных обществ, в том числе иностранных и с участием иностранного капитала. Многие из данных организаций для оптимизации своей деятельности и экономии средств, стараются взять новые инструменты на вооружение.

Одной из самых затратных, как с денежной, так и временной точки зрения, является импорт данных в систему для дальнейшего хранения и использования ее. При большом количестве запросов на изменение данных, сервер может не выдержать и перестать функционировать.

Для того, чтобы не произошел во время задачи импорта, необходимо разработать специальный модуль, который будет оптимизировать добавление, обработки, а так же удаление данных из хранилища.

# Анализ предметной области

## Описание предметной области

Страхование – особый вид экономических отношений, призванный обеспечить страховую защиту людей и их интересов от различного рода опасностей. Суть страхования заключается в том, что человек имеет возможность получить перспективы своеобразной защиты от негативных факторов финансового плана. Услуги страхования предоставляют страховые компании.

Страховая компания — это [юридическое лицо](http://discovered.com.ua/glossary/yuridicheskoe-lico/) любой организационно-правовой формы, предусмотренной законодательством, созданное для осуществления страховой деятельности (страховая организация и общество [взаимного страхования](http://discovered.com.ua/glossary/vzaimnoe-straxovanie/)) и получившее в установленном законом порядке лицензию на осуществление страховой деятельности.

Законодательство допускает создание страховых компаний в виде [акционерных обществ](http://discovered.com.ua/glossary/akcionernoe-obshhestvo/), [обществ с ограниченной ответственностью](http://discovered.com.ua/glossary/obshhestvo-s-ogranichennoj-otvetstvennostyu/), государственных предприятий. В качестве специфической формы организации страхового дела предусматривается создание обществ [взаимного страхования](http://discovered.com.ua/glossary/vzaimnoe-straxovanie/).

Процесс сотрудничества со страховой компанией заключается в том, что вы подписываете договор, в рамках которого будете осуществлять определенные выплаты денежных средств за страховку. Если же случится неприятный инцидент, оговоренный в страховом договоре, то компания обязана произвести необходимые финансовые выплаты.

Таким образом, становится понятно, что страхование по своей сути предполагает определенный элемент защиты человека, его интересов и его гражданской ответственности. В настоящий момент появилось существенное количество компаний, которые способны предоставить вашему вниманию услуги страхования.

Проводя анализ предметной области, я выяснил, что множество компаний сталкиваются с проблемой быстрого импорта документов в базы данных. К сожалению, в настоящее время, даже с учетом технического прогресса, серверы не могут обрабатывать большое количество договоров бесперебойно. Серверное оборудование может не выдержать и зависнуть, что, по итогу, может привезти к огромным финансовым потерям. Для устранения данной проблемы было принято решение о создании модуля загрузки и обработки реестров договоров.

## Основные понятия

Страхование — отношения по защите [имущественных интересов](https://ru.wikipedia.org/wiki/%D0%98%D0%BC%D1%83%D1%89%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D0%B5%D1%81_(%D1%81%D1%82%D1%80%D0%B0%D1%85%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) [физических](https://ru.wikipedia.org/wiki/%D0%A4%D0%B8%D0%B7%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%BB%D0%B8%D1%86%D0%BE) и [юридических лиц](https://ru.wikipedia.org/wiki/%D0%AE%D1%80%D0%B8%D0%B4%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%BB%D0%B8%D1%86%D0%BE) при наступлении определённых событий ([страховых случаев](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%B0%D1%85%D0%BE%D0%B2%D0%BE%D0%B9_%D1%81%D0%BB%D1%83%D1%87%D0%B0%D0%B9)) за счет [страховых фондов](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%B0%D1%85%D0%BE%D0%B2%D0%BE%D0%B9_%D1%84%D0%BE%D0%BD%D0%B4), формируемых из уплачиваемых ими страховых взносов.

Страховщик – понятие, которое определяет компанию частного или же государственного формата, которая осуществляет процесс страхования на основании четко проработанных и предусмотренных законодательных норм;

Страхователь – лицо, которое страхует свою ответственность посредством подписания договора. Данное же лицо осуществляет соответствующие выплаты;

Объект страхования - это именно тот элемент, который страхуется. По своей сути, это может быть движимое или же недвижимое имущество, денежные средства, здоровье или же жизнь. В зависимости от вида страхования формируется расчет стоимости взносов;

Договор – документ, который является основным элементом подтверждения наличия страхования;

Случаи страхового характера – возникновение действий, которые провоцируют возможность получения выплат от страховой компании; Возмещение – сумма, которую уплачивает компания в том случае, если возникает страховой случай.

Черный ящик – программа, установленная на клиентском сервере, предназначенная для парсинга данных реестра договоров в промежуточные таблицы базы данных.

# Описание проблемы

Загрузка реестров, обработка, удаление реестров договоров – это сложный процесс, требующий мощного технического обеспечения. К сожалению, даже при наличии дорогой техники, нагрузка может быть настолько велика, что серверы могут не выдержать и повиснуть. Таким образом может остановиться процесс документооборота внутри компании.

Для того, чтобы избавиться от возможности остановить делопроизводство страховой компании необходимо использовать некоторые программные средства, например как брокер сообщений RabbitMQ. Взаимодействие с модулем показано на рис.1.

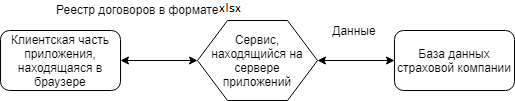


Рис 2.1 - Взаимодействие между элементами модуля импорта реестров договоров страховой компании.

# Обзор аналогов

Для того, чтобы понять целесообразность разработки данного ПО, проведем сравнение с программами, уже существующими на рынке и схожими по функционалу.

1. Система электронного документооборота и автоматизации бизнес-процессов «Е1 ЕВФРАТ»

Данная система хорошо зарекомендовала себя, что показывает ее использование в таких компаниях как: ВТБ Страхование, ВЖД Логистика.

Главными отличиями разрабатываемого модуля от данного аналога являются:

* Неудобный интерфейс
* Отсутствие асинхронной обработки и удаления данных
* Ограничение на количество пользователей
* Цена
* Наличие неиспользуемых функций
* Отсутствие реализации в браузере

Существенным отличием от разрабатываемой системы является то, что отсутствует возможность парсинга импортируемых реестров на отдельные договора, валидация их и приведение к общему виду.

1. Система документооборота «ТЕЗИС»

Используется такими компаниями, как: СПАО «Ингосстрах», Ювелирная сеть «585GOLD».

Главными отличиями разрабатываемого модуля от данной системы являются:

* Отсутствие асинхронной обработки и удаления данных
* Ограничение на количество пользователей
* Цена
* Отсутствие реализации в браузере

В данном случае так же является отсутствие возможности парсинга реестров договоров и валидация их.

По результатам данного сравнения стало известно, что ни один существующий аналоги не может быть использован. Использование уже готовых программных решений приведет к тому, что многие части придется доделывать. Таким образом, вероятность того, что все будет работать успешно, уменьшается.

По этим причинам проще и выгоднее разработать собственное программное обеспечение для реализации потребностей конкретной системы. Таким образом, аналогов, способных реализовать аналогичный функционал, не существует.

# Программная документация

## Техническое задание на программное обеспечение

### Общие сведения

#### Полное название системы и её условные обозначения

Полное наименование системы: Модуль импорта и корректировки реестров договоров страховой компании.

Краткое название системы: Модуль загрузки реестров.

#### Перечень документов, на основании которых создается системы, кем и когда утверждены эти документы

Основанием для разработки интеграционного шлюза являются следующие документы и нормативные акты:

1. Заказ для ООО «БиАйВи» на создание модуля и корректировки реестров договоров от 10 июня 2018 года. Заказчик: ООО СК «Сбербанк страхование»

2. Функциональные требования, присланные заказчиком

#### Порядок оформления и предъявления заказчику результатов работы по созданию системы

Передается в виде функционирующего комплекса на базе средств вычислительной техники заказчика ООО СК «Сбербанк страхование» в сроки, установленные договором.

#### Перечень нормативно-технических документов, методических материалов, использованных при разработке ТЗ

* ГОСТ 19.201-78. Техническое задание. Требования к содержанию и оформлению
* РД 50-34.698-90. Методические указания. Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Требования к содержанию документов

### Назначение и цели создания системы

#### Назначение системы

Модуль импорта и корректировки реестров договоров страховой компании предназначен для автоматизации процесса работы с реестрами договоров, в части исполнения процессов ведения учета, обработки и добавления договоров, а так же нахождения ошибок, содержащихся в договоре

Модуль импорта и корректировки РД страховой компании предполагается использовать для упрощения процесса интеграции договоров.

#### Цели создания системы

Основными целями создания программного продукта являются упрощение ведения учета договоров, автоматизация в процессе обработки и добавления договоров, а так же повышение качества облуживания клиентов.

### Требования к системе

#### Требования к структуре и функционированию системы

В состав модуля импорта и корректировки реестров договоров должны входить следующие компоненты

* Пользовательский интерфейс (UI)
* Черный ящик
* Сервер RabbitMQ
* Серверная часть программы
  + Модуль аутентификации
  + Фасад серверной части (описание http-методов, вызов методов, реализующих логику приложения, используется RestEasy)
  + Модуль импорта и корректировки реестров договоров
  + Модуль для работы с RabbitMQ

Архитектура системы представлена на рисунке 4.1

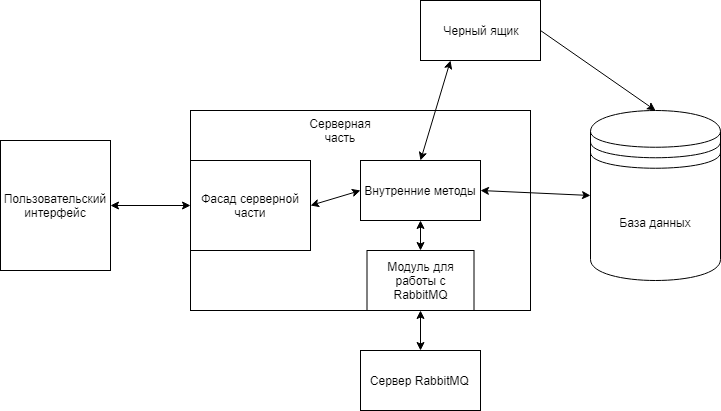


Рисунок 4.1 Архитектура системы

Обмен данными между компонентами системы происходит с помощью REST-сообщений. Для работы с базой данных используется язык запросов SQL.

### Требования к пользовательскому интерфейсу

К пользовательскому интерфейсу предъявляются следующие требования:

* В системе необходим журнал реестров, в котором отображаются все загруженные реестры
* Пользователь должен иметь возможность просмотреть исходные данные из реестра, без возможности изменить
* Пользователь должен иметь возможность просмотреть обработанные данные из реестра, с возможностью редактирования необработанных договоров
* Пользователь должен иметь возможность обрабатывать результаты загрузки реестра с помощью следующих функций:

1. Фильтрация содержимого реестра только по новым и ошибочным записям
2. Просмотр ошибки, возникшей при импорте строки реестра
3. Редактирование данных строки реестра
4. Просмотр ошибки, возникшей при импорте данных

* Пользователь должен имеешь возможность удалять отдельные договоры и реестры целиком. Удаление реестра возможно только в случае, если в нем нет ни одного активного договора.

Макеты интерфейсных форм представлены на рисунках 4.2 , 4.3, 4.4, 4.5



Рисунок 4.2 Макет журнала реестров

На страницу с журналом реестров выводятся все реестры, имеющиеся в БД. При нажатии на кнопку «Добавить» открывается окно, изображенное на рисунке 4.3 «Макет формы добавления реестра». При нажатии на кнопку «Редактировать» открывается содержимое реестра, изображенное которого находится на рисунке 4.4. При нажатии на кнопку «Удалить», выводится диалоговое окно с вопросом «Вы уверены, что хотите удалить реестр?». При согласии запускается процедура удаления реестра.

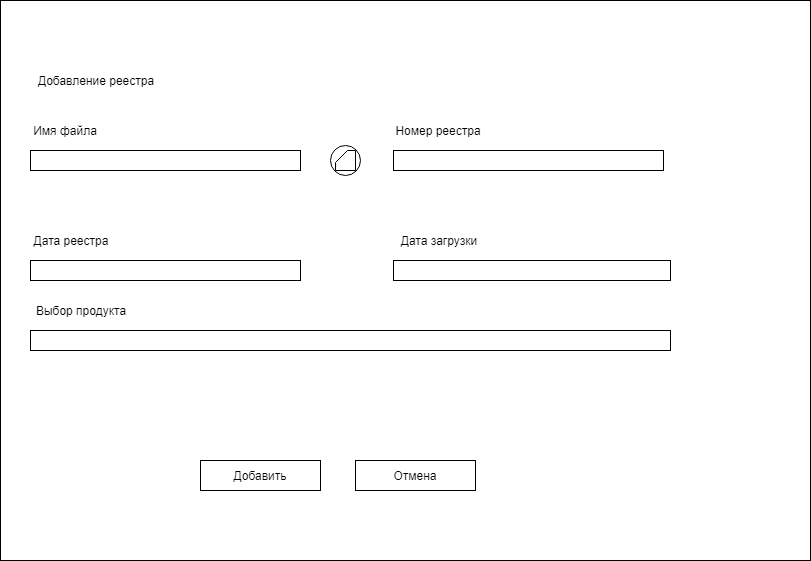


Рисунок 4.3 Макет формы добавления реестра

На форме добавления реестра производится выбор файла с персонального компьютера, а так же заполнение атрибутов реестра, таких как: номер реестра, дата реестра и т.д. При нажатии на кнопку «Добавить» запускается процесс импорта содержимого реестра, появляется окно с прогрессбаром. По итогам появляется диалоговое окно с текстом, меняющимся в зависимости от результата:

1. Если импорт завершился успешно, то «Реестр успешно загружен», При нажатии на кнопку «ОК», возвращаемся в журнал реестров
2. В случае возникновения ошибки, то «В результате загрузки. реестра произошла ошибка *Текст ошибки*». При нажатии на кнопку «Ок» возвращаемся в журнал реестров.

При нажатии на кнопку «Отмена», возвращается в журнал реестров.

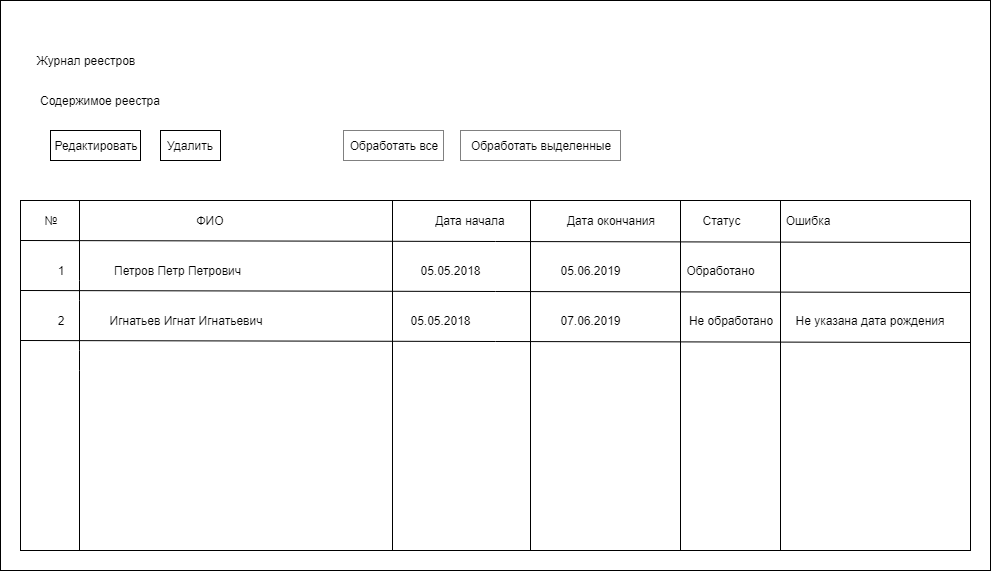


Рисунок 4.4 Макет формы содержимого реестра

На форме отображения содержимого реестра показаны все договоры, содержащиеся в реестре. На ней также отображается статус договора. Если статус «Не обработано», то у пользователя есть возможности редактировать или удалить строку реестра. При нажатии на кнопку «Редактировать», отображается окно редактирования договора. Макет окна представлен на рисунке 4.5. При нажатии на кнопку «Удалить», появляется диалоговое окно с вопросом «Вы уверены, что хотите удалить строку реестра». При согласии запускается процесс удаления строки реестра. При нажатии на кнопку «Обработать все», запускается процедура обработки по всем необработанным строкам реестра. При нажатии на «Обработать выделенные», происходит аналогичная процедура, только по отмеченным строкам реестра. Выделение строк происходит через Ctrl + ЛКМ.

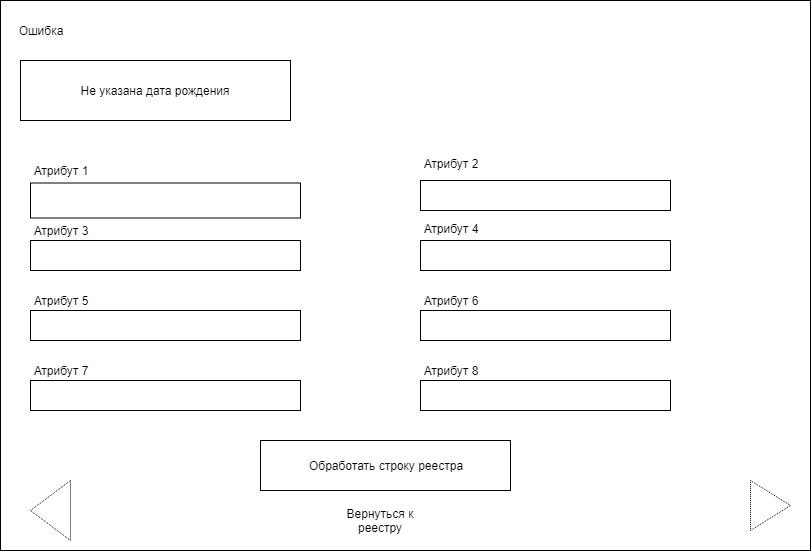


Рисунок 4.5 Макет формы редактирования договора

На форме редактирования договора отображается ошибка, не позволяющая обработать договор, а так же поля атрибутов, которые можно изменять. При нажатии на кнопку «Обработать строку реестра», запускается процедура обработки строки реестра. В случае успеха появляется окно с текстом «Строка реестра успешно обработана», при нажатии на «ОК», осуществляется переход к следующей не обработанной строке реестра. При нажатии на «Вернуться к реестру», происходит возвращение к журналу содержимого обрабатываемого реестра. При нажатии на стрелочки, происходит переход к предыдущей или последующей необработанной строке реестра.

### Черный ящик

Черный ящик представляет собой программу, работающую на клиентском сервере, и предоставляющую возможность парсинга реестров в таблицы базы данных клиента. Изначально модуль импорта и корректировки добавляет загружаемый файл в таблицу БД IS\_BLOB. Далее модуль получает Id загруженного файла. ЧЯ представляет API, который будет принимать этот Id. При получении номера файла, программа обращается к таблице IS\_BLOB и пытается разбить данные из файла реестра на договора. Содержание сообщения, передаваемого в ЧЯ, показана на рисунке 4.6.

После попадания в таблицу IS\_BLOB, данные заносятся в другие таблицы БД, такие как: IS\_REGISTER, IS\_REGISTER\_COV, IS\_REG\_DICT, IS\_REGISTER\_TP. Структура данных представлена в таблицах 4.1, 4.2, 4.3, 4.4, 4.5. В случае успеха, черный ящик в ответ за запрос отправляет текст ошибки или признак успешного выполнения загрузки реестра в БД.

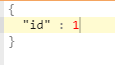


Рисунок 4.6 Содержание сообщения, передаваемого модулем импорта, в черный ящик

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип данных | | Значение |
| SYNK\_ID | NUMBER(17) | | Номер записи |
| BLOB\_FILE | BLOB | | BLOB файла |
| BLOB\_NAME | VARCHAR2(100) | | Наименование файла |
| CREATE\_DATE | DATE default sysdate | | Дата создания |
| SIGN\_BLOCK | NUMBER(1) default 0 | | Блокирована |
| AUTHOR | VARCHAR2(30) | | Автор |
| PRODUCT | NUMBER(17) | | Вид продукта |
| STATUS | NUMBER(1) default 0 | Статус (0 – не сформирован, 1- идет формирование, 2 – сформирован) | |
| JIRA\_ISSUE\_KEY | VARCHAR2(50) |  | |
| MAPPING\_TYPE | NUMBER(1) default 1 | Тип загрузки 1 – обычный 2 - активация | |
| PROCESS\_STATUS | VARCHAR2(200) |  | |
| BLOB\_NUMBER | VARCHAR2(200) |  | |
| BLOB\_DATE | DATE |  | |
| IS\_DELETED | NUMBER(1) |  | |

Таблица 4.1 Структура таблицы БД «IS\_BLOB»

В данной таблице содержится информация, связанная, непосредственно с файлом реестра и данными, вводимыми на форме добавления реестра.

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Значение |
| SYNK\_ID | NUMBER (17) |  |
| REGISTER\_ID | NUMBER(17) | Номер строки реестра |
| CODE | VARCHAR2(50) | Код контрагента |
| OCCUPATION | VARCHAR2(350), | Профессия |
| EMPLOYER\_NAME | VARCHAR2(350), | Место работы |
| FRAME\_REFERENCE | VARCHAR2(350) | Сфера деятельности |
| POSITION | VARCHAR2(350) | Должность |
| LAST\_NAME | VARCHAR2(350) | Фамилия |
| FIRST\_NAME | VARCHAR2(350) | Имя |
| PATRONYMIC | VARCHAR2(350) | Отчество |
| PASSPORT\_SERIES | VARCHAR2(350) | Серия документа |
| PASSPORT\_NUMBER | VARCHAR2(350) | Номер документа |
| PASSPORT\_ISSUEDATE | VARCHAR2(350) | Дата документа |
| PASSPORT\_ISSUEINSIT | VARCHAR2(350) | Кто выдал документ |
| PLACE\_OF\_BIRTH | VARCHAR2(350) | Место рождения |
| STREET\_BUILD2 | VARCHAR2(350) | Номер строения |
| STREET\_FLAT2 | VARCHAR2(350) | Квартира |
| AREA2 | VARCHAR2(350) | Наименование субъекта страны |
| POSTCODE | VARCHAR2(350) | Индекс адреса регистрации |
| TOWN2 | VARCHAR2(350) | Город адреса регистрации |
| STREET2 | VARCHAR2(350) | Улица адреса регистрации |
| DISTRICT2 | VARCHAR2(350) | Район адреса регистрации |
| STREET\_NR2 | VARCHAR2(350) | Номер дома адреса регистрации |
| CONTACT\_TELL2 | VARCHAR2(350) | Телефон рабочий |
| CONTACT\_TELL1 | VARCHAR2(350) | Телефон мобильный |
| CONTACT\_TELL0 | VARCHAR2(350) | Телефон городской |
| CITIZENSHIP\_USA | VARCHAR2(350) | Наличие гражданства США |
| POSTCODE2 | VARCHAR2(350) | Индекс адреса регистрации в США |
| POSTCODE\_ADRESS | VARCHAR2(350) | Адрес регистрации США (без индекса) |
| DOC\_TYPE | VARCHAR2(250) | Наименование документа, удостоверяющего личность |
| RIGHT\_OF\_LIVE | VARCHAR2(350) | Документ, подтверждающий право на пребывание в РФ |
| MIGRATION\_CARD | VARCHAR2(350) | Миграционная карта |
| TAX\_RESIDENT | VARCHAR2(350) | Является ли клиент резидентом другой страны? |
| USA\_TAX\_RESIDENT | VARCHAR2(350) | Является ли клиент резидентом США? |
| DATE\_OF\_BIRTH | VARCHAR2(250) | Дата рождения |
| SEX | VARCHAR2(250) | Пол |
| ADRESS | VARCHAR2(350) | Адрес регистрации |
| PASSPORT\_DATA | VARCHAR2(400) | Паспортные данные |
| CONTACT\_TELL | VARCHAR2(4000) | Контактный телефон |
| TIN | VARCHAR2(250) | ИНН |
| RESIDENT | VARCHAR2(250) | Резидент |
| EMAIL | VARCHAR2(250) | Адрес электронной почты |
| ADRESS2 | VARCHAR2(350) | Фактический адрес проживания |
| CITIZENSHIP | VARCHAR2(350) | Гражданство |
| COUNTRY\_OF\_BIRTH | VARCHAR2(350) | Страна рождения |
| MARITAL\_STATUS | VARCHAR2(250) | Семейное положение |
| TIN\_USA | VARCHAR2(250) | ИНН США |
| TIN2 | VARCHAR2(250) | ИНН2 |
| COUNTRY | VARCHAR2(250) | Страна |
| PASSPORT\_ISSUEINS\_ID | VARCHAR2(250) | Код подразделения документа, удостоверяющего личность |
| MIGRATION\_CARD\_NUM | VARCHAR2(250) | Серия и номер миграционной карты |
| MIGRATION\_CARD\_DSTART | VARCHAR2(250) | Данные миграционной карты: начало срока пребывания |
| MIGRATION\_CARD\_DEND | VARCHAR2(250) | Данные миграционной карты: окончание срока пребывания |
| RIGHT\_OF\_LIVE\_TYPE | VARCHAR2(250) | Наименование документа, подтверждающего право на пребывание в РФ |
| RIGHT\_OF\_LIVE\_SERIES | VARCHAR2(250) | Серия документа, подтверждающего право на пребывание в РФ |
| RIGHT\_OF\_LIVE\_NUMBER | VARCHAR2(250) | Номер документа, подтверждающего право на пребывание в РФ |
| RIGHT\_OF\_LIVE\_DSTART | VARCHAR2(250) | Данные документа, подтверждающего право на пребывание: начало срока пребывания |
| RIGHT\_OF\_LIVE\_DEND | VARCHAR2(250) | Данные документа, подтверждающего право на пребывание: окончание срока пребывания |
| NAME\_SHORT | VARCHAR2(250) | Краткое наименование ЮрЛица |
| NAME | VARCHAR2(250) | Имя |
| ADRESS\_REG | VARCHAR2(350) | Адрес регистрации (застрах) |
| ADRESS\_FACT | VARCHAR2(250) | Фактический адрес (для выгодоприобретателя) |
| ANCESTRY | VARCHAR2(250) | Родственные связи (для выгодоприобретателя) |
| BEN\_SHARE | VARCHAR2(250) | Доля (для выготоприобретателя) |
| RISK | VARCHAR2(250) | Риск (для выгодоприобретателя) |
| CITIZENSHIP2 | VARCHAR2(250) | Другое гражданство |
| PDN | VARCHAR2(250) | Согласие на обработку персональных данных |
| RESIDENCE\_PERMIT | VARCHAR2(250) | Вид на жительство |
| AGE | VARCHAR2(250) | Возраст в годах |
| STREET\_NR | VARCHAR2(350) | Адрес регистрации (Дом) |
| DISTRICT | VARCHAR2(350) | Адрес регистрации (Район) |
| STREET | VARCHAR2(350) | Адрес регистрации (Улица) |
| TOWN | VARCHAR2(350) | Адрес регистрации (Город) |
| AREA | VARCHAR2(350) | Адрес регистрации (Область) |
| STREET\_FLAT | VARCHAR2(350) | Адрес регистрации (Квартира) |
| STREET\_BUILD | VARCHAR2(350) | Адрес регистрации (Номер строения) |
| TYPE\_PERSON | NUMBER(1) default 0 | Тип персоны (Физическое \ Юридическое лицо) |
| OGRN | VARCHAR2(250) | ОРГН |
| EX\_1\_TMP | VARCHAR2(250) | Нагрузка на застрахованного (для случая многих застрахованных) |
| CERT\_DOC\_TYPE | VARCHAR2(250) | Наименование документа, удостоверяющего полномочия уполномоченного представителя |
| CERT\_ISSUEDATE | VARCHAR2(250) | Дата документа, удостоверяющего полномочия уполномоченного представителя |
| CERT\_NUMBER | VARCHAR2(250) | Серия и номер документа, удостоверяющего полномочия уполномоченного представителя |
| POLICY\_PROGRAM\_TMP | VARCHAR2(250) | Поле для временного хранения данных по программе застрахованного |
| E\_AFTER\_SAL\_SERV | VARCHAR2(250) | Согласие на ППО онлайн |
| SNILS | VARCHAR2(250) | СНИЛС |

Таблица 4.2 Структура таблицы БД «IS\_REGISTER\_TP»

Эта таблица служит для хранения данных о контрагентах для каждой строка реестра. Заполняется при парсинге данных из xls-файла реестра.

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Значение |
| SYNC\_ID | NUMBER(17) | Первичный ключ |
| CODE | VARCHAR2(50) | Код |
| VAL | VARCHAR2(350) | Значение |
| REGISTER\_ID | NUMBER(17) | Ссылка на строку реестра |

Таблица 4.3 Структура таблицы БД «IS\_REG\_DICT»

Эта таблица служит для хранения данных о характеристик полиса по каждой строке реестра. Заполняется при парсинге данных из xls-файла реестра.

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Значение |
| SYNK\_ID | NUMBER(17) | Первичный ключ |
| REGISTER\_ID | NUMBER(17) | Номер строки реестра |
| CODE | VARCHAR2(50) | Код покрытия |
| PT | VARCHAR2(350) | Способ страховой выплаты |
| EX\_2 | VARCHAR2(350) | Нагрузка 2 |
| EX\_1 | VARCHAR2(350) | Нагрузка 1 |
| DSP | VARCHAR2(350) | Дата начала выплат |
| DB | VARCHAR2(350) | Дата начала страхования |
| DE | VARCHAR2(250) | Дата окончания страхования |
| PREM | VARCHAR2(250) | Страхования премия |
| SUM | VARCHAR2(250) | Страхования сумма |
| DEP | VARCHAR2(350) | Дата окончания выплат |
| PP | VARCHAR2(350) | Периодичность выплат |
| SUM2 | VARCHAR2(250) | Страхования сумма 2 (альтернативная) |
| DSP2 | VARCHAR2(250) | Дата начала выплаты 2 (альтернативная) |
| DSP1 | VARCHAR2(250) | Дата начала выплаты 1 (альтернативная) |
| EX\_1\_DOP | VARCHAR2(250) | Нагрузка 1 по покрытию (альтернативная) |
| EX\_2\_DOP | VARCHAR2(250) | Нагрузка 2 по покрытию (альтернативная) |
| PAY\_PERIODICITY | NUMBER(2) | Период оплаты премии |
| COV\_PERIOD | VARCHAR2(250) | Период действия покрытия (разница между DB и DE) |
| PREM\_1YEAR | VARCHAR2(250) | Размер первого платежа |
| SEQ | NUMBER(2) | Порядковый номер покрытия |
| SERV\_PACK | VARCHAR2(250) | Пакет услуг |
| PAYM\_SCHED\_START\_DATE | VARCHAR2(250) | Дата начала формирования графика платежей |
| PAYM\_SCHED\_END\_DATE | VARCHAR2(250) | Дата окончания формирования графика платежей |
| SIGN\_CLAIM\_CREATE | NUMBER(1) | Учитывать ли покрытие при создание убытков? 1-да, 0 –нет (определяет фиктивные риски нужные для перестрахования или других целей) |

Таблица 4.4 Структура таблицы БД «IS\_REGISTER\_COV»

Эта таблица служит для хранения данных о покрытиях по каждой строке реестра. Заполняется при парсинге данных из xls-файла реестра.

### Сервер RabbitMQ

Задача импорта и корректировки реестров является сложной ресурсоемкой, которая не может быть умещена в коротком временном интервале HTTP запроса. На помощь приходят очереди. Основная идея очередей – избежать выполнения ресурсоемких задач непосредственно после отправки запроса. Вместо этого задача ставится в очередь для последующего выполнения в асинхронном режиме, то есть, при получении задача инкапсулируется в сообщение и отправляется в очередь, а обработчик очереди достает сообщение в порядке их следования и обрабатывает надлежащим образом. В случае модуля импорта и корректировки реестров договоров очереди используют сервер сообщений RabbitMQ.

Базовыми понятиями являются: продюсер – программа, которая посылает сообщения, брокер – буфер в памяти без каких либо ограничений на количество хранимых сообщений, консьюмер – программа, которая принимает сообщения из очереди

Сервер RabbitMQ по сути является менеджеров очередей, который имеет следующие преимущества:

* При некорректном завершении работы сервера, данные в очереди не теряются. При последующей запуске обработка продолжается с того места, где был обрыв.
* Распределить задачи на несколько очередей, то есть создать распараллеливание на уровне сообщений
* Если результат обработки не удовлетворяет, задачу можно послать в очередь повторно
* Количество хранимых в очереди сообщений неограниченно
* Сервер сообщений может быть расположен удаленно как по отношению к продюсеру, так и по отношению к консьюмеру

Взаимодействие с RabbitMQ происходит во время удаления или обработки реестров договоров, а так же договоров в отдельности.

### Серверная часть программы

Серверная часть программы состоит из нескольких модулей, связанных между собой. Они отвечают за логику программы.

Модуль аутентификации представляет собой набор методов, служащих для подтверждения аккаунта входящего в систему. Ему приходит захешированная строка, передаваемая в каждой JSON части REST-сообщения, содержащая в себе логин, пароль и время. Задача модуля заключается в расхешировании, проверке данных о пользователе, и предоставлении доступа к модулям.

Фасад серверной части представляет собой API, предоставляемый наружу и вызывающий внутренние методы системы, такие как добавление, обработка, корректировка, удаление реестров, а так же их отдельных строк. Он является элементом архитектуры системы и представляет возможности для дальнейшего масштабирования системы.

Модуль импорта и корректировки представляет собой набор методов, который отвечает за функционал системы.

Модуль импорта взаимодействует с черным ящиком и его главная задача заключается в добавлении файла реестра в таблицу БД IS\_BLOB, в которой хранятся все реестры, а так же передача его Id документа ЧЯ и возврат сообщения о корректности выполнения операции.

Модуль корректировки представляет собой метод, который обновляет информацию о необработанных договорах в промежуточных таблицах БД.

Модуль удаления реестров договоров и их строк в отдельности представляет собой набор методов, взаимодействующих с сервером RabbitMQ. Основная его задача – пометка подлежащих удалению договоров, и передача, единого для нескольких строк таблицы, индекса удаления в очередь, для последующего взаимодействия с консьюмером. После получения данных от консьюмера происходит удаление данных о договорах и промежуточных таблиц БД.

Модуль обработки реестров договоров представляет собой набор методов для валидации и донасыщенные данных, находящихся в промежуточных таблицах БД. Задача данного модуля – пометка договоров подлежащих обработке индексом, и передача индекса на сервер RabbitMQ. Так же, после получения данных от консьюмера производится добавление договоров в таблицу БД с активными договорами.

Модуль для работы с RabbitMQ служит для получения из очереди сообщений с данными и, в зависимости от выбранного метода, отправление данных на сервер приложений для дальнейшей обработки или удаления.

### Перспективы развития, модернизации системы

Система должна реализовывать возможность дальнейшей модернизации как программного обеспечения, так комплекса технических средств.

Также необходимо предусмотреть возможность увеличения производительности системы путем её масштабирования.

### Требования к информационному обеспечению системы

Уровень хранения данных в системе должен быть построен на основе современных реляционных или объектно-реляционных СУБД. Для обеспечения целостности данных должны использоваться встроенные механизмы СУБД. Средства СУБД, а также средства используемых операционных систем должны обеспечивать документирование и протоколирование обрабатываемой в системе информации.

Структура базы данных должна поддерживать кодирование хранимой и обрабатываемой информации в соответствии с общероссийскими классификаторами (там, где они применимы).

Доступ к данным должен быть предоставлен только авторизованным пользователям с учетом их служебных полномочий, а также с учетом категории запрашиваемой информации.

Структура базы данных должна быть организована рациональным способом, исключающим единовременную полную выгрузку информации, содержащейся в базе данных системы.

Технические должны средства, использовать обеспечивающие современные технологии, хранение информации, позволяющие обеспечить повышенную надежность хранения данных и оперативную замену оборудования (распределенная избыточная запись/считывание данных; зеркалирование; независимые дисковые массивы; кластеризация).

### Требования к программному обеспечению системы

При проектировании и разработке системы необходимо максимально эффективным образом использовать ранее закупленное программное обеспечение, как серверное, так и для рабочих станций.

Используемое при разработке программное обеспечение и библиотеки программных кодов должны иметь широкое распространение, быть общедоступными и использоваться в промышленных масштабах.

### Требования к техническому обеспечению

В состав технического обеспечения должны входить следующие технические средства:

- Сервер БД

- Сервер приложений

- Веб-сервер

- RabbitMq- сервер

- ПК пользователей

- Сервер черного ящика

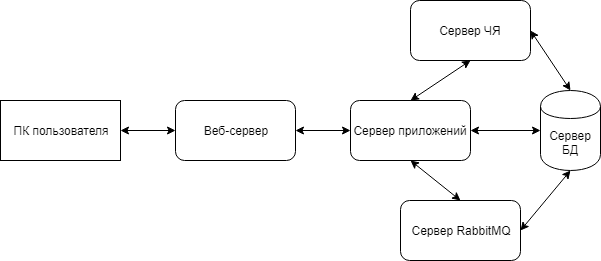


Рисунок 4.6. Схема взаимодействия технических средств

Серверы БД должны быть объединены в отказоустойчивый кластер. Серверы приложений должны образовывать кластер с балансировкой нагрузки.

Серверы БД, серверы приложений и сервер системы формирования отчетности должны быть объединены одной локальной сетью, с пропускной способностью не менее 100 Мбит.

Требования к техническим характеристикам серверов БД:

* Процессор – Intel Xeon Gold 6254 3.2 ГГц
* Объем оперативной памяти – 128 Гб
* Дисковая подсистема – 60 Тб
* Сетевой адаптер – 100 Мбит

Требования к техническим характеристикам системы хранения данных:

- Дисковая подсистема 0,5 Тб Raid Array 5

Требования к техническим характеристикам серверов приложений:

- Процессор – Intel Xeon 5120 2.2 ГГц;

- Объем оперативной памяти – 32 Гб;

- Дисковая подсистема – 1 Тб;

- Устройство чтения компакт-дисков (DVD-ROM);

- Сетевой адаптер – 100 Мбит.

Требования к техническим характеристикам веб - сервера:

- Процессор– [AMD Ryzen **Threadripper** 2950X](https://market.yandex.ru/product--protsessor-amd-ryzen-threadripper-2950x-colfax-str4-l3-32768kb/214768441?show-uid=15576723491050176095016004&nid=55330&context=search)3.5 ГГц;

- Объем оперативной памяти – 32 Гб;

- Дисковая подсистема – 4 Тб;

- Сетевой адаптер – 100 Мбит.

Требования к техническим характеристикам ПК пользователя

- Процессор – Intel Pentium 4560 3.5 ГГц;

- Объем оперативной памяти – 4 Гб;

- Дисковая подсистема – 40 Гб;

- Сетевой адаптер – 100 Мбит.

### Требования к надежности

#### Требования к обеспечению надёжного (устойчивого) функционирования системы

Программа должна работать с надежностью 95 %. 5% резервных накладываются на технические работы и обновления ПО.

Надёжное (устойчивое) функционирование программы должно быть обеспечено выполнением совокупности организационно-технических мероприятий, перечень которых приведён ниже:

* Организация бесперебойного питания сервера, на котором установлена система
* Организация технического обслуживания сервера, обеспечивающего бесперебойную работу всех его составляющих (как программных, так и аппаратных)
* Организация технического обслуживания сети в связи с необходимостью обеспечения бесперебойного доступа к системе через сеть.

#### Время восстановления после отказа

Время восстановления после отказа, вызванного сбоем электропитания технических средств или другими внешними факторами не должно превышать времени запуска всех служб системы-сервера, на котором установлена настоящая система.

### Требования к безопасности

#### Защита от несанкционированного доступа

Для защиты от несанкционированного доступа система должна быть снабжена подсистемой авторизации. Для каждой системы необходимо создать учётную запись. Авторизация должна производиться по логину и паролю.

### Требования к метрологическому обеспечению

Требования к метрологическому обеспечению не предъявляются.

## Пояснительная записка к программному обеспечению

### Анализ предметной области

Анализ предметной области приведен в разделе 1.

### Формулировка проблемы

Формулировка проблемы приведена в разделе 2.

### Обзор аналогов

Обзор аналогов выполнен в разделе 3.

### Назначение и область применения

Цели, назначение и области использования системы приведены в подразделах 4.1.2.1 и 4.1.2.2 технического задания соответственно.

Модуль импорта и корректировки реестров договоров – это программное решение, обеспечивающее оптимизация выполнению работы по внедрению, редактированию и учету договоров страхования, предоставляя разработчикам и администраторам унифицированные и централизованные средства разработки, тестирования и контроля протекания всех интеграционных сценариев.

### Технические характеристики

#### Постановка задачи на разработку программы

Необходимо разработать программный продукт который будет:

1. Выполнять аутентификацию пользователя
2. Загружать файлы реестров в БД
3. Отображать информацию о содержании БД в клиентской части приложения
4. Позволять редактировать данные из реестров
5. Позволять удалять данные из реестров
6. Выполнять обработку договоров
   1. Валидацию строк реестра
   2. Донасыщение строк реестра
7. Выполнять парсинг данных из реестра в промежуточные таблицы

#### Описание алгоритма и функционирования программы

Общий алгоритм программы представлен на рисунке

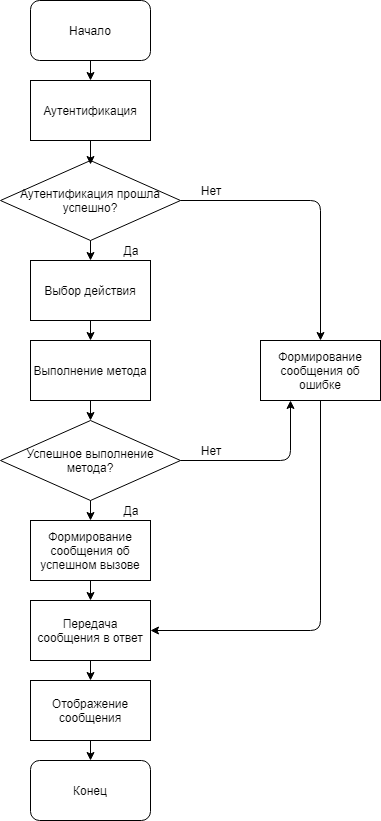


Рисунок 4.7 Алгоритм работы модуля импорта и корректировки реестров договоров

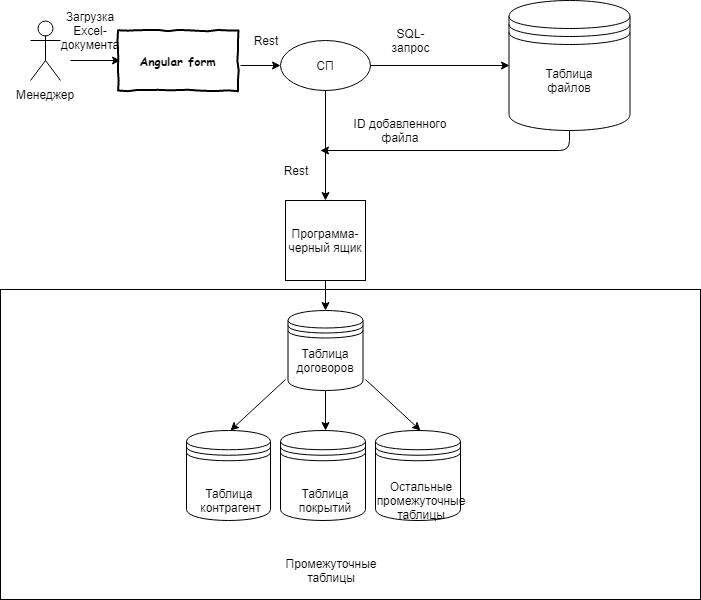
Сценарий работы отдельных частей модуля представлены на рисунка 4.8, 4.9, 4.10 

Рисунок 4.8 Сценарий работы импорта реестра договоров

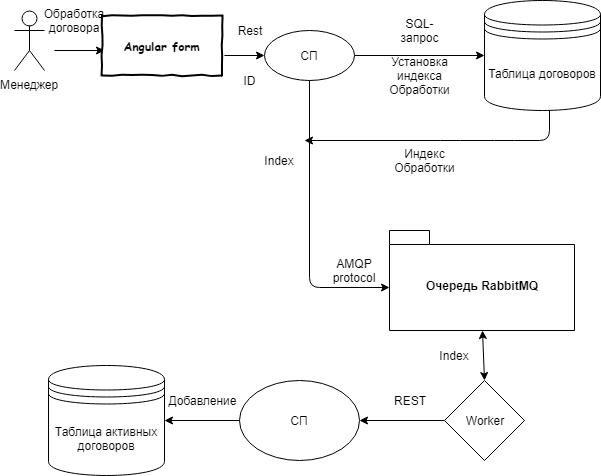


Рисунок 4.9 Сценарий работы обработки реестра договоров

Есть 2 сценария работы данного

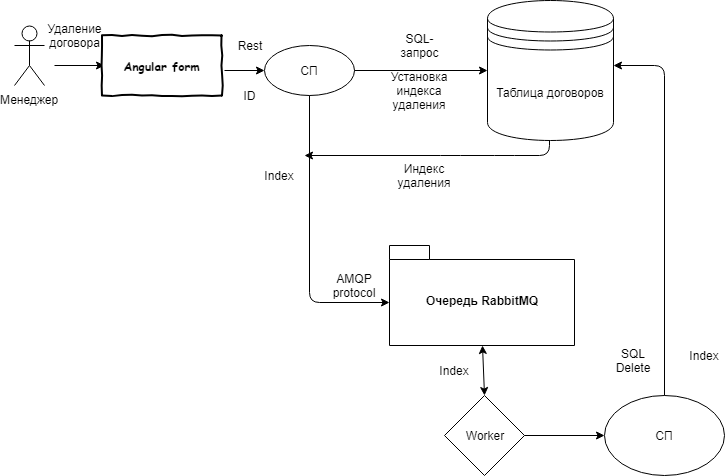


Рисунок 4.10 Сценарий работы удаления реестра договоров

#### Описание и обоснование выбора состава технических и программных средств

**Java** является одним из самых распространенных и популярных языков программирования. Первая версия языка появилась еще в 1996 году в недрах компании Sun Microsystems, впоследствии поглощенной компанией Oracle. Java задумывался как универсальный язык программирования, который можно применять для различного рода задач.

Ключевой особенностью языка Java является то, что его код сначала транслируется в специальный байт-код, независимый от платформы. А затем этот байт-код выполняется виртуальной машиной JVM (Java Virtual Machine). В этом плане Java отличается от стандартных интерпретируемых языков как PHP или Perl, код которых сразу же выполняется интерпретатором. В то же время Java не является и чисто компилируемым языком, как С или С++.

Подобная архитектура обеспечивает кроссплатформенность и аппаратную переносимость программ на Java, благодаря чему подобные программы без перекомпиляции могут выполняться на различных платформах - Windows, Linux, Mac OS и т.д. Для каждой из платформ может быть своя реализация виртуальной машины JVM, но каждая из них может выполнять один и тот же код.

Java является объектно-ориентированным языком. Он поддерживает полиморфизм, наследование, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений.

Язык Java был выбран из за того, что это универсальный язык программировании, который используют сотни тысяч компаний разного масштаба в своем корпоративном серверном ПО, а так же то, что он работает на всех платформах.

**WildFly** (JBoss Application Server) – сервер приложений Java EE, полностью разработанным на [Java](https://ru.bmstu.wiki/Java), и, следовательно, может работать в любой операционной системе, как 32-битной, так и 64-битной. Основными характеристиками, на которых акцентируется внимание, являются возможности подключения, скорость отклика и масштабируемость, а основным фактором, стоящим за улучшениями в этих областях, является новый веб-сервер Undertow, который является более мощным, чем его конкуренты, такие как Jetty Web Server.

Управление памятью очень строгое, чтобы минимизировать максимальное выделение памяти кучи. Все сервисы, используемые WildFly, используют общие индексированные метаданные, которые кэшируются, чтобы избежать дублирования. Они также имеют модульную загрузку, которая предотвращает загрузку дублирующихся классов и должна загружаться поверх системы, необходимой для загрузки.

**Apache Maven** — фреймворк для автоматизации сборки проектов на основе описания их структуры в файлах на языке POM, являющемся подмножеством XML. Проект Maven издаётся сообществом Apache Software Foundation, где формально является частью Jakarta Project.

Maven обеспечивает декларативную сборку проекта. В файлах описания проекта содержится его спецификация, а не отдельные команды выполнения. Все задачи по обработке файлов, описанные в спецификации, Maven выполняет посредством их обработки последовательностью встроенных и внешних плагинов.

**Apache Cayenne** является общедоступным проектом под лицензией Apache, обеспечивающим объектно-реляционной отображение (ORM) и удаленное взаимодействие сервисов. Другими словами, это инструмент для разработчиков Java, которым необходимо обращаться к базе данных (или нескольким базам данных). Сayenne имеет множество уникальных и мощных возможностей, может решать широкий круг потребностей разработчиков. Он легко связывает одну или несколько схем баз данных непосредственно через Java объекты, управляет автоматическими фиксациями и откатами, генерирует SQL , объединения, последовательности, и многое другое.

**RabbitMQ** — программный брокер сообщений на основе стандарта AMQP — тиражируемое связующее программное обеспечение, ориентированное на обработку сообщений.

Состоит из сервера, библиотек поддержки протоколов HTTP, XMPP и STOMP, клиентских библиотек AMQP для Java и .NET Framework и различных плагинов (таких как плагины для мониторинга и управления через HTTP или веб-интерфейс или плагин «Shovel» для передачи сообщений между брокерами).

**Angular** представляет собой фреймворк от компании Google для создания клиентских приложений. Прежде всего он нацелен на разработку SPA-решений (Single Page Application), то есть одностраничных приложений. В этом плане Angular является наследником другого фреймворка AngularJS. В то же время Angular это не новая версия AngularJS, а принципиально новый фреймворк.

Angular предоставляет такую функциональность, как двустороннее связывание, позволяющее динамически изменять данные в одном месте интерфейса при изменении данных модели в другом, шаблоны, маршрутизация и так далее. Одной из ключевых особенностей Angular является то, что он использует в качестве языка программирования TypeScript. Одной из ключевых особенностей Angular является то, что он использует в качестве языка программирования TypeScript.

#### Ожидаемые технико-экономические показатели

Значительное повышение прибыли страховой компании, вследствие оптимизации документооборота внутри предприятия и уменьшения длительности обслуживания клиентов.

## Описание программы

### Общие сведения

#### Обозначение и наименование программы

Модуль импорта и корректировки реестров договоров страховой компании имеет следующие атрибуты.

* Наименование файла веб-архива: sberimpdog.war
* Исходное имя файла: sberimpdog.war
* Внутреннее имя файла: sberimpdog
* Название продукта: sberimpdog
* Идентификатор группы Maven: com.bivgroup.ws
* Идентификатор артефакта Maven
* Версия 1.01.05

#### Программное обеспечение, необходимое для функционирования программы

Системные программные средства, используемые модулем загрузки реестров должны быть, представлены одной из следующих операционных систем: Windows 7 или выше, Ubuntu 16.04 LTS или выше, UbuntuServer 16.04 LTS или выше.

Так же для функционирования программного продукта необходимо следующее предустановленное программное обеспечение стороннего разработчика:

* Сервер приложений Wildfly версии 10 и выше
* Oracle JDK версии 8 и выше
* Сервер RabbitMQ

#### Языки программирования, на которых написана программа

Исходным языком программирования серверной части приложения является Java. В качестве языка используемого в клиентской части выступает TypeScript

### Функционально назначение

#### Классы решаемых задач

Модуль импорта и корректировки реестров договоров страховой компании - программное решение, обеспечивающее увеличение документооборота, предоставляя работника предприятия удобное средство для ведения учета договоров.

#### Назначение программы

Разрабатываемый программный продукт позволит снизить нагрузку на системы страховой компании, увеличит скорость учета договоров, а так же увеличит число обслуживаемых клиентов в единицу времени.

#### Сведения о функциональных ограничениях на применение

Модуль загрузки реестров предназначен для работы с протоколом HTTP для общения между серверной и клиентской частью, а так же с протоколом AMQP для взаимодействия с брокером сообщений RabbitMQ.

Разрабатываемое веб-приложение может запускаться только внутри сервера приложений.

### Описание логической структуры

#### Алгоритм программы

Алгоритм работы программы приводится в пояснительной записке раздел «4.2.5.2 Описание алгоритма и функционирования программы».

#### Используемые методы

##### Метод «uploadDocument»

Метод «uploadDocument» служит для передачи документа в таблицу базы данных и получить его id. При вызове метода не происходит проверка на тип файла, так как это задача черного ящика. Далее происходит вызов метода «passToBlackBox» для передачи данных с сервисной части программы в черный ящик, для дальнейшего парсинга данных из документа. Результатом выполнения метода «uploadDocument» является отправка сообщения на фронт-часть программы об успешности импорта договоров.

##### Метод «handlingDocument»

Метод «handlingDocument» выполняется, в случае, когда пользователь хочет выполнить обработку данных. Первым делом происходит пометка необходимых строк в таблице «IS\_REGISTER» индексом обработки, затем, в зависимости способа обработки (обработка целого реестра или определенных документов), происходит следующее:

* В случае обработки целого реестра, происходит передача индекса обработки документа методу «sendToQueue», в котором происходит добавление этого индекса в очередь RabbitMQ, где в последующем ее получает программа-консьюмер и передает обратно на сервер приложений в метод «processingDocument». Выходными данными метода являются сообщения о том, что договора находятся в очереди на обработку.
* В случае обработки выбранных вручную договоров, индекс обработки, вне очереди, передается методу «processingDocument». Результатом является сообщение об успешности обработки договоров.

Наличие двух сценариев работы позволяет сделать систему более гибкой, так как программа позволяет обработать нужные договоры все очереди, что требуется заказчику.

##### Метод «processingDocument»

Метод «processingDocument» служит для обработки данных. Под обработкой данных понимается валидация и донасыщение исходных данных реестра. Исходные данные из реестра должны подвергаться проверкам и модификация перечисленным ниже:

* Удаление кратных пробелов в ФИО
* Определение организационно-правового статуса ЮЛ
* Определение страны по полному адресу
* Определение типа документа по тексту паспортных данных
* Проверка на корректность ИНН и т.д.

По окончанию выполнения валидации и донасыщения, данные заносятся в специальную таблицу активных договоров и могут быть использованы в дальнейшем документообороте.

##### Метод «provideToDeleteDocument»

##### Метод «provideToDeleteDocument» выполняется в случае, если пользователю необходимо удалить несколько договоров из реестра или реестр в целом. Его задача заключается в том, чтобы пометить нужные строки таблицы «IS\_REGISTER» индексом удаления, после, в зависимости от того, удаляются ли отдельные договоры или реестр целиком, происходит следующее

* В случае удаления целого реестра, происходит передача индекса удаления документа методу «sendToQueue», в котором происходит добавление этого индекса в очередь RabbitMQ, где в последующем ее получает программа-консьюмер и передает обратно на сервер приложений в метод «deleteDocument». Выходными данными метода являются сообщения о том, что договора находятся в очереди на удаление.
* В случае удаления выбранных вручную договоров, индекс удаления, вне очереди, передается методу «deleteDocument», а сообщение содержит в себе информацию об успешности выполнения.

Наличие двух сценариев работы позволяет сделать систему более гибкой, так как программа позволяет удалять нужные договоры все очереди, что требуется заказчику.

##### Метод «deleteDocument»

Метод «deleteDocument» служит для удаления данных из таблиц БД. При удалении данных из БД, происходит проверка их на связность с другими данными. Если связность отсутствует и документ не активен, то происходит удаление.

#### Структура программы с описание функций составных частей и связи между ними

В состав модуля импорта и корректировки реестров договоров страховой компании должны входить следующие компоненты:

* Подмодуль аутентификации
* Подмодуль загрузки реестров
* Подмодуль обработки
* Подмодуль удаления
* Клиентская часть
* Программы-воркеры

Клиентская часть связывается с серверной посредством REST-сообщений. Сервис общается с воркерами посредством протокола AMQP.

### Используемые технические средства

Техническое обеспечение системы должно максимально и наиболее эффективным образом использовать технические средства заказчика.

Минимальные аппаратные средства:

* Процессор — двухъядерный с тактовой частотой 2 ГГц или лучше
* Оперативная память — не менее 4 Гб
* Свободное место на жестком диске — не менее 2 Гб
* Дополнительные требования — не менее 2 Гб для файла подкачки

Стоит отметить, что чем выше производительность рабочей машины, тем выше скорость работы системы в целом.

### Вызов и загрузка

Загрузка и запуск программы осуществляется способами, детальные сведения о которых изложены в Руководстве системного администратора.