Министерство образования и науки РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Рыбинский государственный авиационный технический университет имени П.А. Соловьева»

Факультет радиоэлектроники и информатики Кафедра Математическое и программное обеспечение ЭВС

**ОТЧЕТ**

**по преддипломной практике**

Студент группы …………………………………………Головкин Д. М.

*(Код) (Подпись, дата) (Фамилия И. О.)*

### Руководитель ………………………………………………Волков М. Л.

*(Уч. степень, звание) (Подпись, дата) (Фамилия И. О.)*

### Оценка …………………………………………………………………….

*(Подпись, дата) (Фамилия И. О.)*

### Рыбинск 2019

# Оглавление

Введение………………………………………………………………………………….4

Раздел 1…………………………………………………………………………………..5

1.1 Формулировка темы выпускной квалификационной работы……………………5

1. 2 Анализ предметной области ………………………………………………………6

1.3 Обзор аналогов ……………………………………………………………………...7

1.4 Обоснование актуальности темы…………………………………………………...8

1.5 Цели и задачи………………………………………………………………………..9

1.6 Обзор и анализ литературы по теме задания ……………………….………..…..10

Раздел 2…………………………………………………………………………………11

2.1 Описание функциональных и технологических требований, обоснование и выбор методов, технологий, инструментов разработки ……………………………11

2.1.1 Требования к серверной части …………………………………………………11

2.1.2 Требования к интерфейсу……………………………………………………….12

2.2 Техническое задание на разработку и проектирование программного обеспечения………………………………………………………………………….…14

2.2.1.1 Полное название системы и её условные обозначения……………………...14

2.2.1.2 Плановые сроки начала и окончания работ…………………………….…….14

2.2.1.3 Перечень документов, на основании которых создается система, кем и когда утверждены эти документы……………………………………………………………14

2.2.1.4 Порядок оформления и предъявления заказчику результатов работы по созданию системы……………………………………………………………………...14

2.2.1.5 Перечень нормативно-технических документов, методических материалов, использованных при разработке ТЗ…………………………………………………..14

2.2.2 Назначение и цели создания системы…………………………………………..14

2.2.2.1 Назначение системы…………………………………………………………...15

2.2.2.2 Цели создания системы………………………………………………………..15

2.2.3 Требования к системе……………………………………………………………15

2.2.3.1 Требования к структуре и функционированию системы……………………15

2.2.3.2 Требования к способам и средствам связи для информационного обмена между компонентами системы………………………………………………………..15

2.2.3.3 Общие сведения………………………………………………………………..15

2.2.3.4 Требования к надежности……………………………………………………..16

2.2.3.5 Требования к безопасности……………………………………………………16

2.2.3.6. Требования к интерфейсу…………………………………………………….16

2.2.3.7 Требования к стандартизации и унификации………………………………..17

2.2.3.8 Требования к программному и техническому обеспечению системы……...17

Раздел 3…………………………………………………………………………………20

3.1 Методика тестирования……………………………………………………………20

3.2 Наборы тестов……………………………………………………………………...22

3.3 Алгоритмы работы подмодулей…………………………………………………..23

3.3.1 Алгоритм подмодуля загрузки реестра договоров…………………………….23

3.3.2 Алгоритм подмодуля обработки реестров договоров…………………………24

3.3.3 Алгоритм подмодуля удаления реестров договоров…………………………..24

3.4 Описание методики тестирования………………………………………………...26

3.5 Результаты тестирования………………………………………………………….27

Заключение……………………………………………………………………………..28

Список используемой литературы…………………………………………………29

Приложения…………………………………………………………………………….30

# 

# 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 большой количестве запросов на изменение данных, сервер может не выдержать и зависнуть.

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.

# Раздел 1.

# 1.1 Формулировка темы выпускной квалификационной работы

Тема работы: Разработка модуля импорта и корректировки реестров договоров страховой компании.

# 

# 1.2.Анализ 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 любой организационно-правовой формы, предусмотренной законодательством, созданное для осуществления страховой деятельности (страховая организация и общество взаимного страхования) и получившее в установленном законом порядке лицензию на осуществление страховой деятельности.

Законодательство допускает создание страховых компаний в виде акционерных обществ, обществ с ограниченной ответственностью, государственных предприятий. В качестве специфической формы организации страхового дела предусматривается создание обществ взаимного страхования.

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.

# 

# 1.3 Обзор аналогов

При анализе целесообразности создания данного модуля было проведено исследование, результатом которого стало известно, что ни один существующий аналоги не может быть использован. Это обуславливается тем, что не существует универсального функционала для работы с данными. 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.

# 

# 1.4.Обоснование актуальности темы

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.

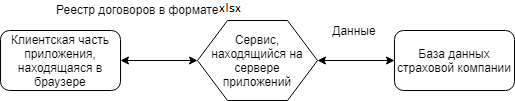


Рис 1 - Взаимодействие между элементами модуля импорта реестров договоров страховой компании.

# 

# 1.5. Цели и задачи

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

В процессе достижения цели необходимо выполнить перечень задач, таких как:

1. Реализовать простой и понятный интерфейс в браузере для загрузки Excel-документов и отображения данных, содержащихся в них.
2. Обеспечить загрузку данных из документа в таблицы БД.
3. Гарантировать асинхронную обработку договоров
4. Предоставить возможность удалять, изменять, сортировать, искать договоры и реестры

# 

# 1.6. Обзор и анализ литературы по теме задания

При выполнении данной работы были использованы книги по программированию и разработке программного обеспечения, а так же электронные ресурсы.

В качестве базовой книги по изучению языка Java, было выбрано 6-ое издание книги Герберта Шилдта «Java 8. Руководство для начинающих». В данной книге заключены базовые понятия языка Java.

# Для получения навыков для разработки веб-сервиса были использованы статьz с сайта JournalDev.com : «RESTEasy Tutorial with Eclipse and Tomcat»

# , а так же с сайта IBM.com: «Web-сервисы RESTful: основы».

# При работе с брокером сообщений RabbitMQ использовалась официальная документация с сайта RabbitMQ.com.

# При взаимодействии с ORM Apache Cayenne была прочитана официальная документация с сайта cayenne.apache.com.

# 

# Раздел 2

# 2.1.Описание функциональных и технологических требований, обоснование и выбор методов, технологий, инструментов разработки

2.1.1 Требования к серверной части

Необходимо реализовать функцию добавления реестров в таблицу БД, предоставить данные из этой таблицу программе - черному ящику для дальнейшего парсинга данных из реестра и помещения в промежуточные таблицы. После выполнения данного процесса необходимо вернуть ответ об успешном импорте на браузерную часть приложения.

Для обработки так же нужно создать метод, при котором сообщение о том, что необходимо обработать договора поступает на сервис. При поступлении такого сообщения сервис должен провести валидацию и стандартизацию данных, содержащихся в таблицах БД, и внести в список активных договоров.

При удалении сервису необходимо проверить, связанны ли удаляемые записи БД с чем то, и, по возможности, изъять из таблиц БД необходимые данные.

При создании должны использоваться следующие технологии:

1. Язык программирования Java

**Java** является одним из самых распространенных и популярных языков программирования. 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).

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.

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.

1. Сервер приложений Wildfly

**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.

1. Брокер сообщений RabbitMQ

**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).

1. ORM Apache Cayenne

**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.

2.1.2 Требования к интерфейсу с пользователем

Процесс загрузки и обработки реестров должен быть прозрачным для пользователя, к пользовательскому интерфейсу предъявляются следующие требования:

1) В системе необходим журнал реестров, в котором отображаются все загруженные реестры

2) Пользователь должен иметь возможность просмотреть исходные данные реестра, без возможности изменить

3) Пользователь должен иметь возможность обрабатывать результаты загрузки реестра с помощью следующих функций:

* Фильтрация содержимого реестра только по новым и ошибочным записям
* Просмотр ошибки, возникшей при импорте строки реестра
* Редактирование данных строки реестра

4) Пользователь может иметь возможность удаления реестра, если по реестру не ни одного связанного с договора

При создании интерфейса необходимо использовать фреймворк Angular.

**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.

# 2.2. Техническое задание на разработку и проектирование программного обеспечения

2.2.1 Общие сведения

2.2.1.1 Полное название системы и её условные обозначения

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

Краткое название системы: Модуль загрузки реестров.

2.2.1.2 Плановые срок начало и окончания работ

Плановый срок начала работ по созданию модуля загрузки реестров 1 февраля 2019 года. Плановый срок окончания работ по разработке 15 июня 2019 года.

2.2.1.3 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

#### 2.2.1.4 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.

#### 2.2.1.5 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

2.2.2 Назначение и цели создания системы

2.2.2.1 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.

2.2.2.2 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.

2.2.3 Требования к системе

2.2.3.1 Требования к структуре и функционированию системы

В состав модуль импорта и корректировки реестров договоров должны входить следующие компоненты:

* Модуль загрузки данных
* Модуль обработки данных
* Модуль удаления данных

Структура частей модуля загрузки реестров показана на рисунках 3.3.3,3.3.3,3.3.4.

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

2.2.3.2 Требования к способам и средствам связи для информационного обмена между компонентами системы

Клиентская часть приложения должна связываться с серверной посредством протокола HTTPS. Для взаимодействия сервера с базой данных используются SQL-запросы. Для взаимодействия с очередью RabbitMQ используется протокол AMQP

2.2.3.3 Общие требования

Ниже представлены требования, присланные заказчиком:

1. Разработка функционала должна соответствовать установленной архитектуре системы
2. Функционал должен использовать готовые микросервисы и окружение системы
3. Функционал должен использовать существующую структуру БД в части хранения договоров, продуктов и справочников

При вызове методов сервисов выполняется проверка:

* Авторизация пользователя
* Прав доступа к сервису для авторизованного пользователя
* Прав доступа к методам сервиса для авторизованного пользователя

2.2.3.4 Требования к надежности

Система должна сохранять работоспособность и обеспечивать восстановление своих функция при возникновении следующих внештатных ситуаций:

* при сбоях в системе электроснабжения аппаратной части, приводящих к перезагрузке ОС, восстановление должно происходить после перезапуска ОС;
* при ошибках в работе аппаратных средств, восстановление функции системы возлагается на ОС, кроме носителей данных и программ;
* при ошибках, связанных с ПО (ОС и драйверы устройств), восстановление работоспособности возлагается на ОС.

2.2.3.5 Требования к безопасности

Соединение с сервером должно происходить в защищенном режиме с использованием протокола HTTPS. Приложение не должно запрашивать излишние разрешения для работы

2.2.3.6 Требования к интерфейсу

Интерфейс должен быть простым и понятным пользователю. Не должно быть различного рода рекламы, которая может закрывать пункты меню, информацию, которую видит пользователь.

2.3.3.7 Требования по стандартизации и унификации

Доступ к личному кабинету осуществляется с помощью ввода адреса в браузере пользователя. Интерфейс должен быть максимально стандартизован:

● приложение должно соответствовать дизайну, выдержанном в одном стиле

● основные навигационные элементы должны быть выделены в горизонтальное меню для обеспечения быстрого доступа к любой страницы приложения;

2.3.3.8 Требования к программному и техническому обеспечению системы

Технические требования:

● На устройстве должен быть установлен один из браузеров: Opera, Google Chrome, Mozilla Firefox, Safari.

В состав технического обеспечения должны входить следующие технические средства:

- Сервер БД

- Сервер приложений

- Веб-сервер

- RabbitMq- сервер

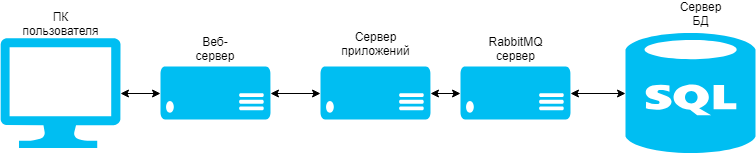
- ПК пользователей

Рисунок 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.

# Раздел 3

# 3.1 Методика тестирования

Тестирование программного обеспечения — процесс исследования, испытания программного продукта, имеющий две различные цели:

· продемонстрировать разработчикам и заказчикам, что программа соответствует требованиям;

· выявить ситуации, в кот 19орых поведение программы является неправильным, нежелательным или не соответствующим спецификации

Стандарт ISO/IEC 25010:2011 (ГОСТ Р ИСО/МЭК 25010-2015) определяет качество программного обеспечения как степень удовлетворения системой заявленных и подразумеваемых потребностей различных заинтересованных сторон. В соответствии со стандартом модель качества продукта включает восемь характеристик:

· функциональная пригодность;

· уровень производительности;

· совместимость;

· удобство пользования;

· надёжность;

· защищённость;

· сопровождаемость;

· переносимость (мобильность).

Состав и содержание документации, сопутствующей процессу тестирования, определяется стандартом IEEE 829-2008 IEEE Standard for Software and System Test Documentation.

В зависимости от доступа разработчика тестов к исходному коду тестируемой программы различают «тестирование (по стратегии) белого ящика» и «тестирование (по стратегии) чёрного ящика».

При тестировании белого ящика (также говорят — прозрачного ящика), разработчик теста имеет доступ к исходному коду программ и может писать код, который связан с библиотеками тестируемого программного обеспечения. Это типично для модульного тестирования, при котором тестируются только отдельные части системы. Оно обеспечивает то, что компоненты конструкции — работоспособны и устойчивы, до определённой степени. При тестировании белого ящика используются метрики покрытия кода или мутационное тестирование.

При тестировании чёрного ящика, тестировщик имеет доступ к программе только через те же интерфейсы, что и заказчик или пользователь, либо через внешние интерфейсы, позволяющие другому компьютеру либо другому процессу подключиться к системе для тестирования. Например, тестирующий модуль может виртуально нажимать клавиши или кнопки мыши в тестируемой программе с помощью механизма взаимодействия процессов, с уверенностью в том, все ли идёт правильно, что эти события вызывают тот же отклик, что и реальные нажатия клавиш и кнопок мыши. Как правило, тестирование чёрного ящика ведётся с использованием спецификаций или иных документов, описывающих требования к системе. Обычно в данном виде тестирования критерий покрытия складывается из покрытия структуры входных данных, покрытия требований и покрытия модели (в тестировании на основе моделей).

Модульное тестирование, или юнит-тестирование — процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы.

Идея состоит в том, чтобы писать тесты для каждой нетривиальной функции или метода. Это позволяет достаточно быстро проверить, не привело ли очередное изменение кода к регрессии, то есть к появлению ошибок в уже оттестированных местах программы, а также облегчает обнаружение и устранение таких ошибок.

# 

# 3.2 Наборы тестов

Необходимо как ведет себя программа при следующих данных:

1. Загружаемый реестр обладает уникальным названием
2. Загружаемый реестр не обладает уникальным названием
3. Количество договоров в реестре меньше 10000 штук
4. Количество договоров в реестре больше 10000 штук
5. Загружается файл не XLS-формата
6. Загружается файл XLS-формата, не содержащий в себе договоры.

# 3.3 Алгоритмы работы подмодулей

3.3.1 Алгоритм подмодуля загрузки реестра договоров

Алгоритм подмодуля загрузки реестра договоров представлен на Рис.3.

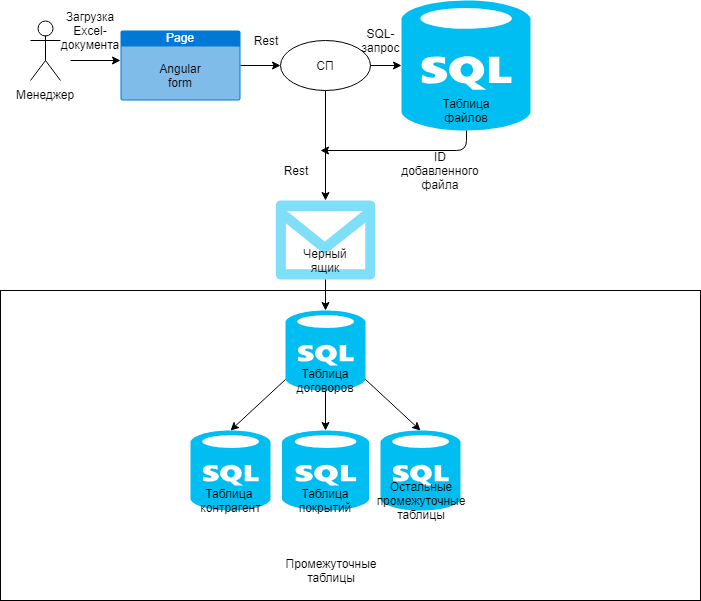
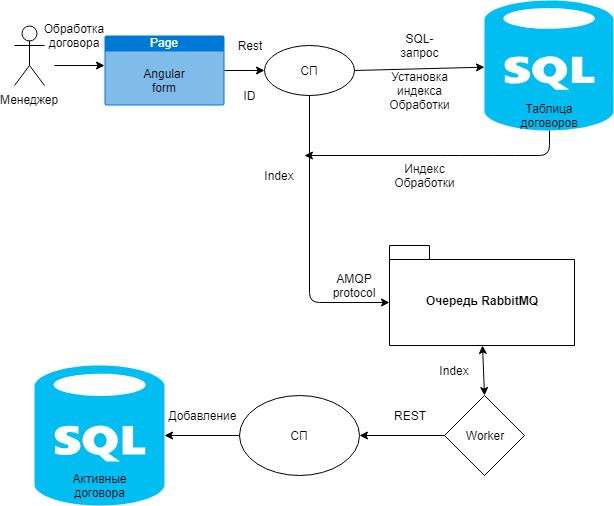
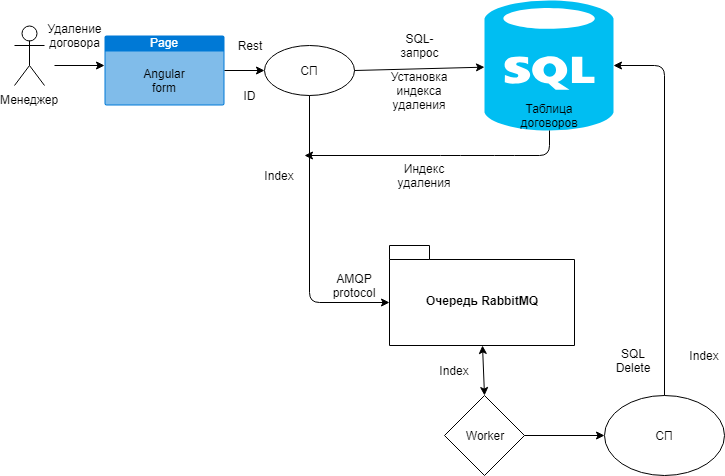


Рис.3. Архитектура подмодуля загрузки реестров договоров

3.3.2 Алгоритм подмодуля обработки реестров договоров



3.3.3 Алгоритм подмодуля удаления реестров договоров



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

# 3.4 Описание методики тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № | Проверяемая функциональность | Входные данные | Ожидаемый результат |
| 1 | Ожидаемый файл должен обладать уникальным именем | Реестр с уникальным именем | Успешный импорт данных в БД |
| 2 | Ожидаемый файл должен обладать уникальным именем | Реестр с именем, которое уже имеется в БД | Импорт данных в БД не производится |
| 3 | Обработка большого количества данных | Менее 10 тысяч договоров | Успешная обработка данных |
| 4 | Обработка большого количества данных | Более 10 тысяч договоров | Успешная обработка данных |
| 5 | Реакция на некорректные файлы | Картинка формата JPEG | Вывод сообщения об ошибке |
| 6 | Реакция на некорректные файлы | Загружается файл XLS-формата, не содержащий в себе договоры. | Вывод сообщения об ошибке |

# 3.5 Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № | Проверяемая функциональность | Входные данные | Ожидаемый результат |
| 1 | Ожидаемый файл должен обладать уникальным именем | Реестр с уникальным именем | Успешный импорт данных в БД |
| 2 | Ожидаемый файл должен обладать уникальным именем | Реестр с именем, которое уже имеется в БД | Импорт данных в БД не производится |
| 3 | Обработка большого количества данных | Менее 10 тысяч договоров | Успешная обработка данных |
| 4 | Обработка большого количества данных | Более 10 тысяч договоров | Успешная обработка данных |
| 5 | Реакция на некорректные файлы | Картинка формата JPEG | Вывод сообщения об ошибке |
| 6 | Реакция на некорректные файлы | Загружается файл XLS-формата, не содержащий в себе договоры. | Вывод сообщения об ошибке |

# Заключение

В результате прохождения практики был разработан веб-сервис, позволяющий производить импорт, корректировку и удаление реестров договоров, а также пользовательский интерфейс, на котором происходит отправка файла в веб-сервис. Была освоена работа с такими инструментами, как: Java, RabbitMQ, Apache Cayenne, Angular, WildFly. Получен опыт работы в IT-компании.

# Список используемой литературы

1. Apache Cayenne guide <https://cayenne.apache.org/docs/4.0/cayenne-guide/>
2. Wildfly <https://wildfly.org>
3. RabbitMQ – начало работы <https://www.rabbitmq.com/#getstarted>
4. Web-сервисы Restful: основы <https://www.ibm.com/developerworks/ru/library/ws-restfu/>
5. Создание Angural приложения. https://shwanoff.ru/angular-5-app/
6. Брюс Эккель. Философия Java ( 4-е издание).

# Приложения

Приложение 1.

Реализация метода добавления реестра в БД

@POST  
@Path(**"/upload"**)  
@Consumes(MediaType.***MULTIPART\_FORM\_DATA***)  
@Produces(MediaType.***TEXT\_PLAIN***)  
**public** Response uploadFile(MultipartFormDataInput input)**throws** IOException {  
 InputPart file = input.getFormDataMap().get(**"foto"**).get(0);  
 **if** (file.getMediaType().toString().equals(**"XLS"**)) {  
  
 Map<String, List<InputPart>> uploadForm = input.getFormDataMap();  
 String fileName = **""**;  
 **try** {  
  
 MultivaluedMap<String, String> header = file.getHeaders();  
 fileName = getFileName(header);  
 **if** (searchInDataBase(fileName)){  
 InputStream inputStream = file.getBody(InputStream.**class**, **null**);  
 **byte**[] bytes = IOUtils.*toByteArray*(inputStream);  
 addToDataBase(bytes, fileName);  
 ObjectId id = getIdOfFile(fileName);  
 ClientRequest request = **new** ClientRequest(  
 **"http://localhost:8080/RESTfulExample/json/product/get"**);  
 request.accept(**"application/json"**);  
 request.body(**"application/json"**, id );  
 ClientResponse<String> response = request.post(String.**class**);  
 **if** (response.getStatus() != 201) {  
 **throw new** RuntimeException(**"Failed : HTTP error code : "** + response.getStatus());  
 }  
  
 System.***out***.println(**"Done"**);  
 } **else** {  
 Response.*status*(Response.Status.***BAD\_REQUEST***).build();  
 }  
  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
  
 **return** Response.*status*(Response.Status.***OK***).build();  
 } **else** {  
 **return** Response.*status*(Response.Status.***BAD\_REQUEST***).build();  
 }

Приложение 2.

Интерфейс модуля импорта реестра договоров

