Корпоративный сервис межсистемного электронного взаимодействия (КСМЭВ)

Техническое описание программы «КСМЭВ»

Листов

Москва, 2024

**Содержание**

[**1 Общие сведения 4**](#_1fob9te)

[1.1 Обозначение и наименование программы 4](#_3znysh7)

[1.2 Назначение программы 4](#_2et92p0)

[1.3 Возможности программы 4](#_tyjcwt)

[1.4 Используемые языки программирования 4](#_3dy6vkm)

[**2 Структура программы 6**](#_4d34og8)

[2.1 Составные части программы 6](#_2s8eyo1)

[2.1.1 Основные компоненты 6](#_17dp8vu)

[2.1.2 Дополнительные компоненты 6](#_26in1rg)

[2.1.3 Операционная система 7](#_lnxbz9)

[2.2 Модули программы 8](#_35nkun2)

[2.2.1 Модуль «Адаптер» – adapter-api 8](#_1ksv4uv)

[2.2.2 Модуль «Сервис интеграций» – external-integration-api 8](#_44sinio)

[2.2.2 Модуль «Монитор запросов» – request-monitor-api 12](#_z337ya)

[2.2.3 Модуль Файловый сервис – file-api 14](#_2xcytpi)

[2.2.4 Модуль Шлюз запросов – gateway-api 14](#_3whwml4)

[2.3 Cвязи между составными частями 14](#_3as4poj)

[2.4 Связи с другими программами 14](#_49x2ik5)

[**3 Архитектура программы 16**](#_147n2zr)

[3.1 Общая архитектурная схема 16](#_3o7alnk)

[**4 Описание технических решений 17**](#_ihv636)

[4.1 Задачи реализованных технических решений 17](#_32hioqz)

[4.1.1 Задача «Упрощение процесса установки» 17](#_1hmsyys)

[4.1.2 Задача «Интеграционное межсистемное взаимодействие» 17](#_41mghml)

[4.1.3 Задача «Хранение данных и доступ к ним» 18](#_2grqrue)

[4.1.4 Настройка интеграционного взаимодействия через графический интерфейс 18](#_3fwokq0)

[4.1.5 Задача «Журналирование событий функциональных блоков» 18](#_1v1yuxt)

[4.1.6 Задача «Мониторинг информации о работоспособности экземпляра программы» 18](#_4f1mdlm)

**Аннотация**

В данном документе приведено «Техническое описание программы» для ПО «Корпоративный сервис межсистемного электронного взаимодействия» (далее – КСМЭВ), предназначенного для обеспечения межсистемного взаимодействия и расширения возможности развития функционала ведомственных (корпоративных) информационных систем.

В данном программном документе, в разделе «Общие сведения» указаны назначение и функции программы и сведения о технических и программных средствах, обеспечивающих выполнение данной программы.

В разделе «Функциональное назначение» приведены сведения о функциональном назначении и ограничениях на применение.

В разделе «Описание логической структуры» приведена структура программы с описанием функций составных частей и связи между ними. Общее описание алгоритма работы программы и связи программы с другими программами.

В разделе «Используемые технические средства» приведен минимальный состав технических средств, требования к серверам, телекоммуникационному оборудованию и каналам связи.

В приложениях к документу «Общее описание программы» приведены дополнительные материалы.

# 1 Общие сведения

## 1.1 Обозначение и наименование программы

Полное наименование: Корпоративный сервис межсистемного электронного взаимодействия.

Условное обозначение: КСМЭВ.

## 1.2 Назначение программы

КСМЭВ – это технологичный интеграционный инструмент для обеспечения межсистемного взаимодействия, расширяющий возможности развития функционала ведомственных (корпоративных) информационных систем. Основой программного комплекса является неблокируемый шлюз для обеспечения взаимодействия информационных систем. Он реализован рядом современных технологий, актуализированных в соответствии с действующими требованиями Минцифры России и ГосТех. КСМЭВ зарегистрирован в Российском реестре программного обеспечения.

## 1.3 Возможности программы

Программа обеспечивает выполнение следующих функций:

* сопряжение информационных систем участников взаимодействия при помощи адаптеров или напрямую;
* возможность централизованной маршрутизации потоков запросов средствами настраиваемой сервисной шины;
* автоматический запуск всех необходимых компонентов программы после установки;
* авторизация запросов от информационных систем участников взаимодействия;
* запись основных данных, касающиеся запросов, в базу данных, а также отображение их в удобном виде;
* журналирование и трассировка запросов;
* хранение файлов вложений, конвертов запросов и ответов.
* исключение из внутреннего трафика передачи файлов больших объемов, заменяя его передачей ссылки на файлы вложения;
* настройка параметров работы шины через графический интерфейс.

## 1.4 Используемые языки программирования

Таблица 1.1 Языки программирования

| Название | Версия | Описание |
| --- | --- | --- |
| Python | 2.7 и новее | Язык программирования |
| Java | 17.0.5 | Язык программирования |
| SQL | SQL:2016 | Язык программирования, применяемый для создания модификации и управления данными в реляционной базе данных |
| Html | 4 | Язык разметки |
| CSS | 3 | Язык разметки. Описание внешнего вида документов |
| JavaScript | Стандарт ES2017 (ES8) | Язык программирования |

# 2 Структура программы

В данном разделе приведены сведения о структуре программы, ее составных частях, о связях между составными частями и о связях с другими программами.

## 2.1 Составные части программы

Программа имеет модульную архитектуру и построена на базе различных компонентов (включая разработки сторонних производителей).

Общую схему взаимосвязей компонентов можно просмотреть в разделе Архитектура программы.

Функционально, программа состоит из следующих частей:

### 2.1.1 Основные компоненты

* *PostrgeSQL* – система управления базами данных (СУБД);
* *Apache ZooKeeper* - необходим для поддержки информации о конфигурации и распределенной координации между компонентами КСМЭВ, также используется как сервисная база данных для хранения технической информации (метаданных).
* *Брокер сообщений Kafka* - используется для непрерывной передачи сообщений между модулями КСМЭВ;
* *MinIO Object Storage -* это один из вариантов высокопроизводительного объектного S3-хранилища.   
  API-интерфейс совместим с облачным хранилищем Amazon S3. Он способен работать с неструктурированными данными, такими как фотографии, видео, файлы журналов, резервные копии и образы контейнеров;
* *Keykloak* –провайдераутентификации и авторизации с открытым кодом для реализации [технологии](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F_%D0%B5%D0%B4%D0%B8%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B2%D1%85%D0%BE%D0%B4%D0%B0) единого входа с возможностью управления доступом.

### 2.1.2 Дополнительные компоненты

Дополнительное программное обеспечение для администрирования и мониторинга:

* **Prometheus** - используется как система мониторинга системных ресурсов КСМЭВ. Связь компонентов реализована через HTTP. Данные хранятся локально, в собственной TSBD базе, индексы хранятся в LevelDB. Метрики представляют собой time-series данные. Каждая метрика состоит из имени метрики, временной метки и пары «ключ – значение». Визуализация осуществляется через подключение к Grafana. Официальный сайт разработчика приложения: <https://prometheus.io/>.
* **Grafana** - инструмент реализован в виде панели управления и мониторинга и позволяет визуализировать системные события программы на базе собираемых метрик. Официальный сайт разработчика приложения: <https://grafana.com/docs/>
* **Ansible** - платформа удалённого управления конфигурациями программного обеспечения, предназначенная для упрощения развёртывания КСМЭВ через создание специальных сценариев. Официальный сайт разработчика приложения: <https://www.ansible.com/>
* **Docker** - программное обеспечение для автоматизации развёртывания и управления программы в виртуальных средах с поддержкой контейнеризации. Контейнер позволяет производить изолированный запуск ОС с подключённой файловой системой из образа, изолированно разворачивать приложения и реализовывать микросервисы. Настройки среды хранятся в GitHub, обеспечивая единую точку управления конфигурациями. Может быть использован для развёртывания тестового окружения «КСМЭВ, без прерывания работы сервисов в продуктовой среде. Официальный сайт разработчика приложения: <https://www.docker.com/>
* **Kubernetes** - открытое программное обеспечение для оркестровки контейнеризированных приложений — автоматизации их развёртывания, масштабирования и координации в условиях кластера. Поддерживает основные технологии контейнеризации, включая Docker, также возможна поддержка технологий аппаратной виртуализации.
* **Elasticsearch** - утилита полнотекстового поиска и аналитики, которая позволяет быстро в режиме реального времени хранить, искать и анализировать большие объемы данных и сохраняет
* **Logstash** - бесплатный и открытый конвейер обработки данных на стороне сервера, который принимает данные из множества источников, преобразует их, а затем отправляет их в хранилище.
* **Filebeat** - агент на сервере для отправки различных типов оперативных данных в *Elasticsearch*. Официальный сайт разработчика приложения: <https://www.elastic.co/elasticsearch/>
* **Redis** – (от англ. remote dictionary server) — резидентная система управления базами данных класса NoSQL с открытым исходным кодом, работающая со структурами данных типа «ключ — значение». Используется как для баз данных, так и для реализации кэшей, брокеров сообщений.

### 2.1.3 Операционная система

Операционная система устанавливается на сервер, где будет развернута программа.

Программа может функционировать под одной из следующих операционных систем:

* CentOS 7.9 (далее - CentOS);
* Debian

## 2.2 Модули программы

### 2.2.1 Модуль «Адаптер» – adapter-api

При отправке: по метаданным, полученным от «Сервиса интеграций», Адаптер формирует и отправляет запрос в формат, понятном для системы получателя.

При получении: Адаптер сохраняет запрос в файловом хранилище, формирует метаданные о запросе и отправляет их в «Сервис интеграций».

### 2.2.2 Модуль «Сервис интеграций» – external-integration-api

Основной компонент программы – Сервис интеграций, оперируя метаданными о запросах, обеспечивает отправку запросов и ответов информационным системам – участникам взаимодействия в синхронном или асинхронном режиме взаимодействия, обеспечивает контроль состояния запросов и менеджмент взаимодействия.

Сервис интеграций принимает HTTP-запросы напрямую от систем – участников взаимодействия или от адаптеров.

Получив запрос, отправляет метаданные о запросе в Кафку (очередь «NEW»). Далее Сервис интеграций считывает метаданные о запросе из Кафки («NEW») и регистрирует запрос в системе, сохраняет метаданные о запросе в БД. Затем Сервис интеграций получает присвоенный в БД ID запроса и помещает его в очередь Кафки («registered») зарегистрированных запросов.

Далее Сервис интеграций считывает ID запроса из Кафки («registered») и отправляет метаданные о запросе на соответствующий адаптер для отправки получателю. В случае успешной отправки присваивает статус «*SENT*» в БД.

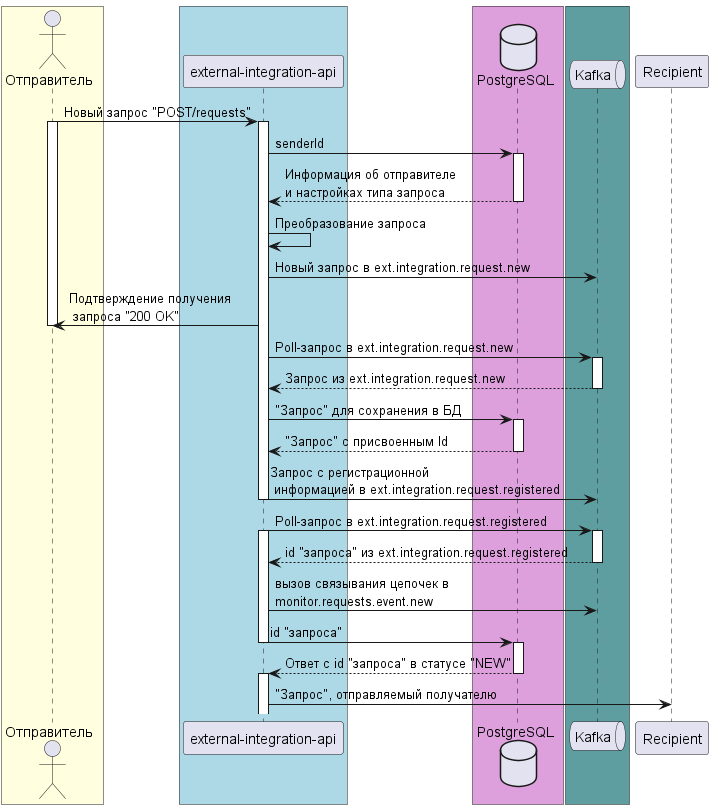


Рисунок - 2.1 *Обобщённая схема получения от отправителя, регистрации и отправки получателю запроса через Сервис интеграций*

В зависимости от режима взаимодействия – синхронное или асинхронное – Сервис интеграций либо сразу получает ответ от получателя, либо ожидает его, пока не получит. Получив метаданные об ответе, Сервис интеграций помещает ID запроса, на который был получен ответ, в очередь Кафки («received») и меняет статус изначального входящего запроса на «*RESPONSE\_RECEIVED*».

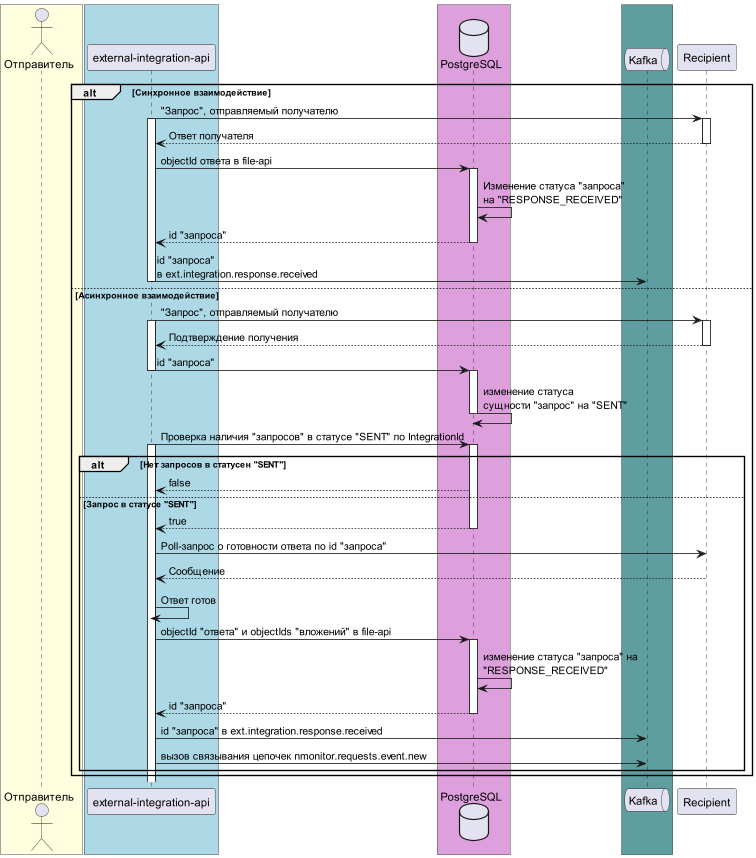
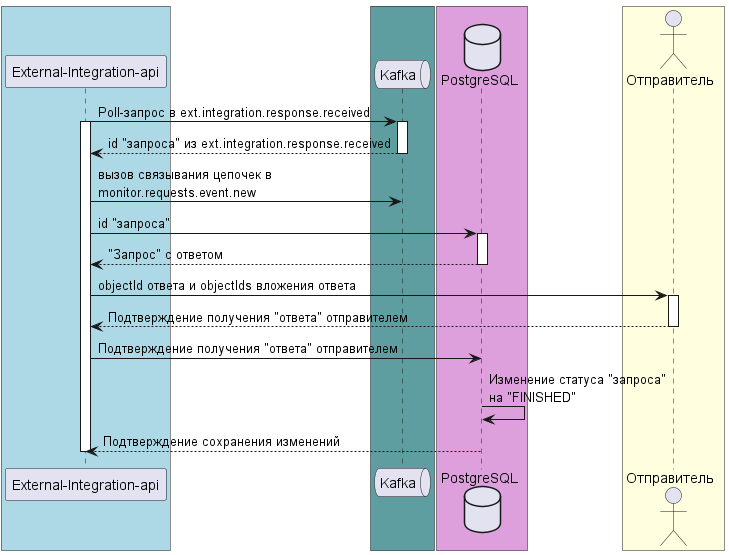


Рисунок - 2.2 *Обобщённая схема получения ответа от получателя входящего запроса*

Далее Сервис интеграций считывает ID запроса из Кафки (очередь «received») и отправляет метаданные о запросе на соответствующий адаптер для отправки ответа отправителю запроса. В случае успешной отправки – то есть, когда пришло подтверждение ответа от отправителя, Сервис интеграций присваивает запросу статус «*FINISHED*» в БД.

Рисунок - 2.3 *Обобщённая схема отправки ответа на запрос отправителю*

Таким образом в БД в каждый момент времени есть актуальное значение статуса запроса, а история изменения статуса пишется в отдельную сущность *status\_history*.

Также Сервис интеграций позволяет настраивать экземпляры интеграций – сущность integration, добавлять и удалять новых участников взаимодействия – сущность participant, создавать и настраивать экземпляры типов запросов – сущность request\_type, хранить настройки интеграций в БД.

Для создания новой интеграции между системами X и Y пользователь при помощи external-integration-ui сначала создаёт новые экземпляры participant X и Y. Затем новый экземпляр integration: выбирает в качестве получателя (integration.recipient\_id) систему X, указывает integration.url – URL-адрес адаптера для системы X, выбирает способ получения ответа (Синхронно/Асинхронно с callback/ Асинхронно с периодическими опросами). Далее пользователь создаёт новый тип запроса (request\_type): выбирает в качестве получателя (request\_type.sender\_id) систему Y, выбирает созданную интеграцию (request\_type.integration \_id), вводит внешний идентификатор (request\_type.external\_integration \_id). Для активации созданной интеграции типу запроса присваивается статус (request\_type.status) «Действует».

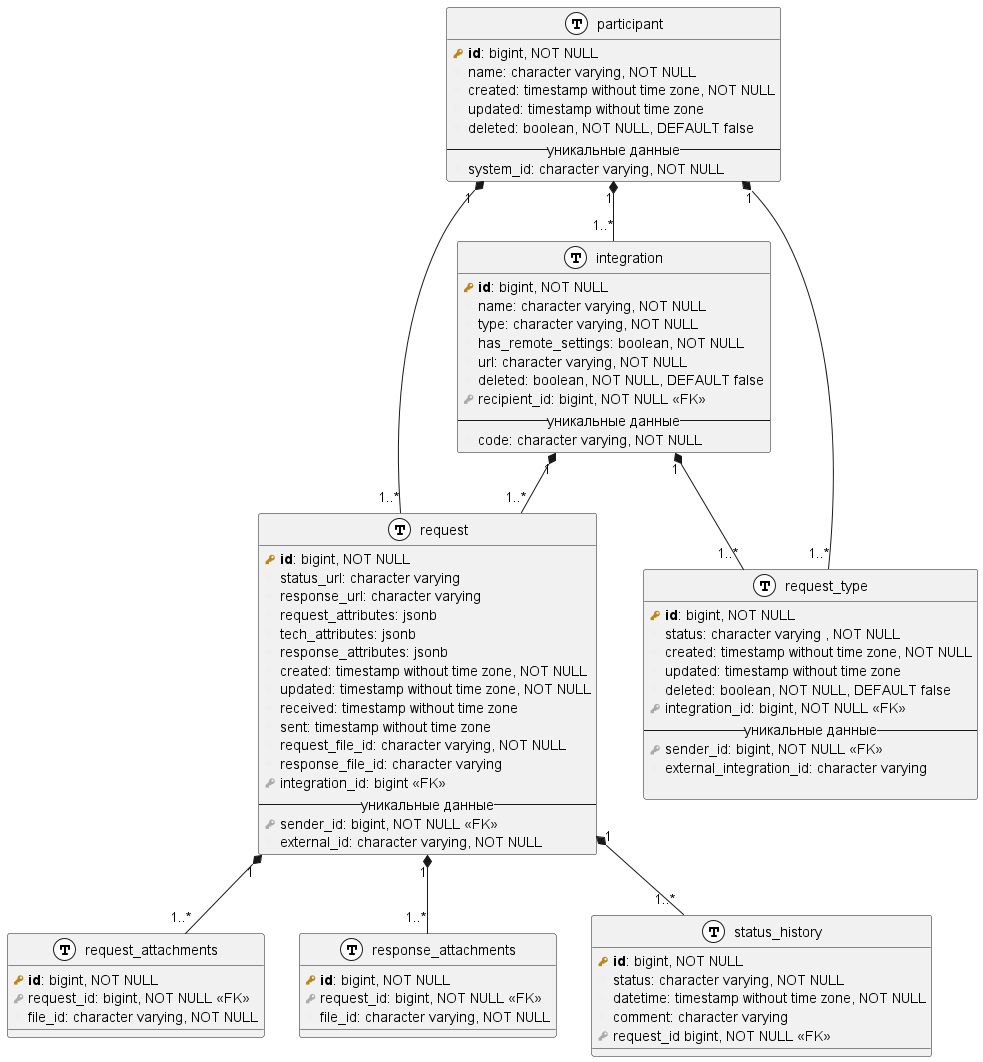


Рисунок - 2.4 *Модель данных БД Сервиса интеграций*

### 2.2.2 Модуль «Монитор запросов» – request-monitor-api

Основные функции Монитора запросов:

* запись и хранение структурированных данных о запросах, проходящих через КСМЭВ;
* связывание цепочек синхронных запросов (сущность correlation) в сущность flow – группу цепочек запросов, порожденных одним бизнес-процессом.

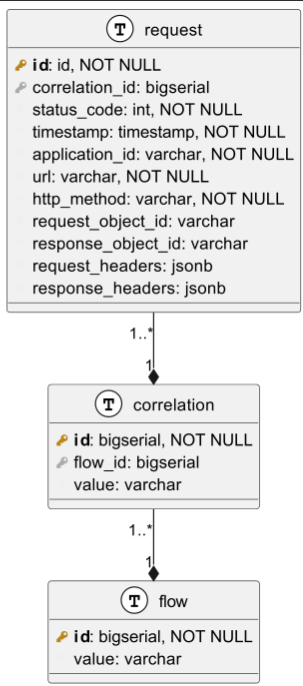


Рисунок - 2.5 *Модель данных БД Монитора запросов*

Если Монитор запросов связывает синхронные технические запросы в цепочки (correlation). Если какой-то бизнес-процесс порождает несколько цепочек запросов, асинхронных по отношению друг к другу – исполняемых в разное время, то такие цепочки Монитор запросов объединяет в группы (сущность flow). Соответственно все запросы цепочки имеют общий correlation\_id, а все цепочки одного бизнес-процесса имеют общий flow\_id (как дома /улицы/города). Это позволяет при возникновении проблем с каким-то одним техническим запросом легко найти все запросы, имеющие отношение к его бизнес-процессу. Это очень полезно при поиске причин инцидентов и оперативном устранении неисправностей системы.

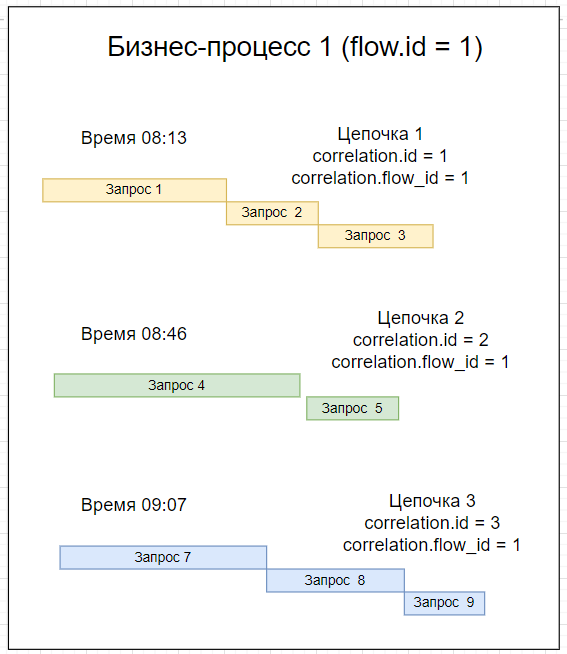


Рисунок - 2.5 *Пример объединения запросов в цепочки и группы цепочек*

### 2.2.3 Модуль Файловый сервис – file-api

Файловый сервис обеспечивает работу с объектным S3-хранилищем. Основные функции: запись, хранение, чтение и удаление файлов. Загрузка файла в корзину происходит либо с использованием Multipart, либо побайтово. Так же Файлы можно загрузить группой с последующей архивацией в ZIP-архив.

### 2.2.4 Модуль Шлюз запросов – gateway-api

Шлюз запросов представляет собой единую точку входа в КСМЭВ для всех взаимодействующих и сторонних систем. Шлюз запросов осуществляет авторизацию запросов путём проверки JWT-токена, прилагаемого к запросу.

Шлюз запросов взаимодействует с компонентом *Keykloak.*

Сервис полнотекстового поиска - поиск по содержимому файлов (конвертов), например поиск по фамилии, по номеру заявки и т.п.

### 2.2.5 Сервис полнотекстового поиска – search-api

Осуществляет поиск по содержимому файлов (конвертов), например поиск по фамилии, по номеру заявки и т.п.

## 2.3 Cвязи между составными частями

Взаимосвязи между составными частями программы приведены на архитектурной схеме (см. рисунок 3.1)

## 2.4 Связи с другими программами

Взаимодействие с другими программами происходит путем вызова соответствующих модулей программы:

* Взаимодействие модулей программы с PostgreSQL происходит через JDBC-driver.
* Взаимодействие модулей друг с другом осуществляется в виде HTTP-запросов напрямую либо через брокер очередей Kafka;
* Взаимодействие модулей с объектным S3-хранилищем осуществляется по RESTfull протоколу Amazon/S3.

# 3 Архитектура программы

## 3.1 Общая архитектурная схема

Рассмотрим внутреннюю архитектуру программы.

Схематичное отображение общей архитектуры решения приведено на рисунке (см. Рисунок - 3.1)

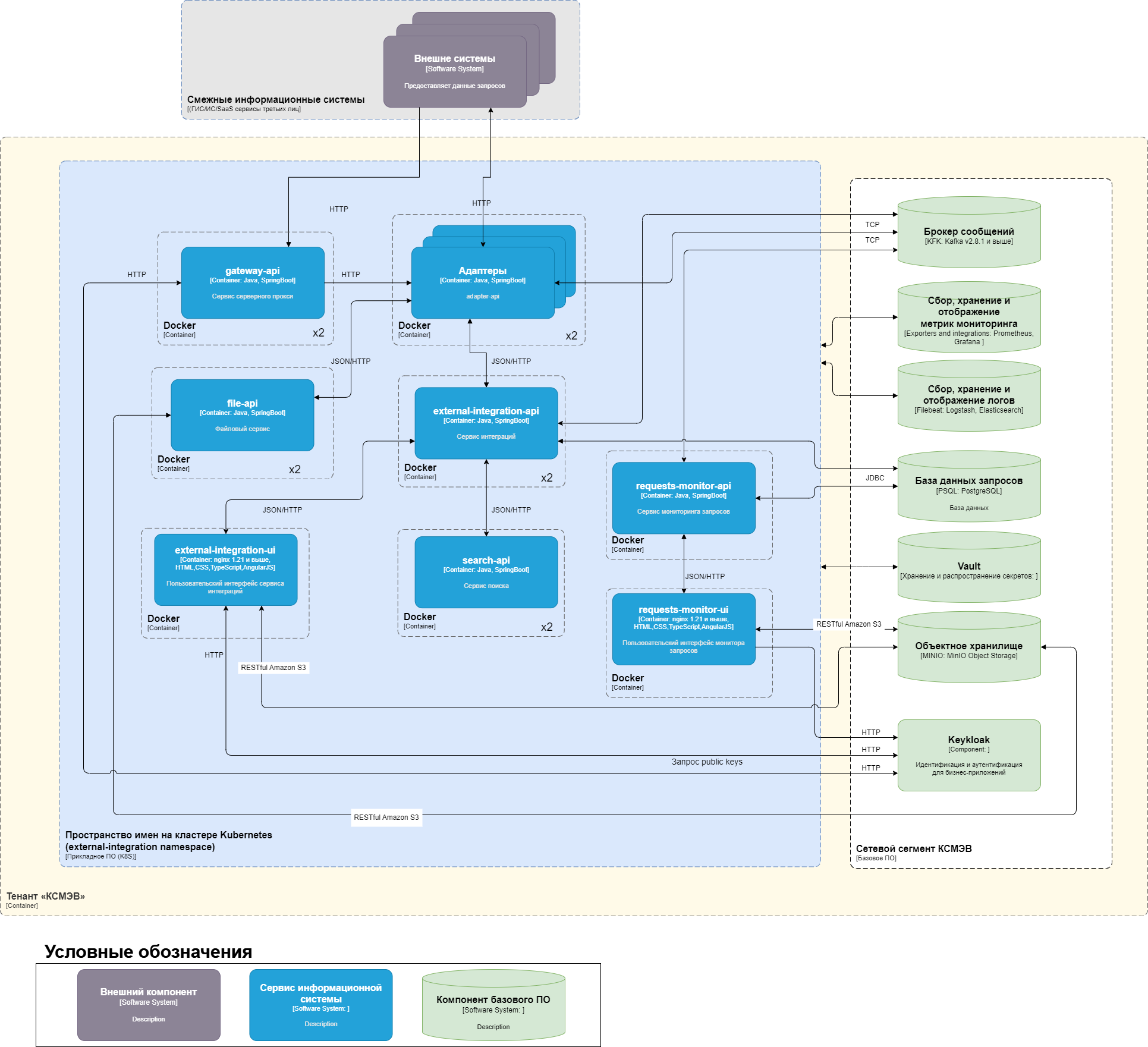


Рисунок - 3.1 Общая архитектура решения

# 4 Описание технических решений

## 4.1 Задачи реализованных технических решений

Технические решения, реализованные в рамках работ по разработке программы, решают следующие задачи:

* + Упрощение процесса установки
  + Интеграционное межсистемное взаимодействие
  + Логирование (протоколирование действий и событий);
  + Хранение данных и доступ к ним;
  + Управление участниками взаимодействия;
  + Управление интеграциями;
  + Управление настройками запросов;
  + Просмотр запросов;
  + Просмотр монитора запросов.

### 4.1.1 Задача «Упрощение процесса установки»

Для решения этой задачи используется *ansible-скрипты*, которые выполняют следующие задачи:

* запускают *Docker*;
* конфигурируют сервер (в docker-контейнере);
* устанавливает КСМЭВ;
* настраивают взаимосвязи между компонентами;
* запускают программу.

### 4.1.2 Задача «Интеграционное межсистемное взаимодействие»

Для решения этой задачи КСМЭВ либо передаёт запрос напрямую, либо задействует адаптеры, которые делают следующее:

* принимают входящие запросы от взаимодействующих систем;
* сохраняют вложения в объектном S3-хранилище при помощи file-api и получают ссылку на файл;
* передают запрос и ссылку на файл в external-integration-api.

Далее external-integration-api либо передаёт запрос системе-корреспонденту напрямую, либо через адаптер, который приводит запрос к нужному виду.

### 4.1.3 Задача «Хранение данных и доступ к ним»

Хранение данных осуществляется СУБД PostgreSQL. Хранение файлов осуществляется объектным S3-хранилищем. Доступ пользователя к данным и файлам осуществляется через графический интерфейс external-integration-ui и request-monitor-ui.

### 4.1.4 Настройка интеграционного взаимодействия через графический интерфейс

Модуль графического интерфейса Сервиса интеграций - external-integration-ui позволяет пользователю создавать новые и настраивать имеющиеся экземпляры интеграций, типов запросов и участников интеграций (при наличии соответствующих адаптеров).

### 4.1.5 Задача «Журналирование событий функциональных блоков»

Задача включает журналирование событий всех функциональных блоков программы.

Запиcь информации о событиях осуществляется конвейером обработки данных *Logstash.* Просмотр и поиск нужной информации по журналу событий осуществляется при помощиутилиты *Elasticsearch*.

### 4.1.6 Задача «Мониторинг информации о работоспособности экземпляра программы»

Задача включает следующие действия:

* Мониторинг информации о работоспособности экземпляра Программы.

Для решения задачи применяются методы:

* метод обработки данных;
* метод журналирования событий.
* метод комплексного сбора данных о занятости вычислительных ресурсов по каждому серверу и их последующему анализу.

При реализации указанного метода программа выполняет следующие основные операции:

* Предоставляет возможность контроля за рекомендуемыми метриками.

Рекомендуемые для отслеживания метрики контроля работоспособности программы приведены ниже:

* Сеть
* CPU

Мониторинг осуществляется методом просмотра лог-файлов следующих компонентов программы:

* Prostore;
* СУБД PostgreSQL;

Сбор лог-файлов программы c записями о событиях производится с помощью *Prometheus*. Процесс просмотра лог-файлов описан в документе «Руководство администратора», раздел «Логирование».

Периодичность обновления значений метрик и их пороговые значения определяются при внедрении и корректируется в процессе последующей эксплуатации программы, в соответствии с пороговыми значениями нагрузки.