ИНФРАСТРУКТУРА ЭЛЕКТРОННОГО ПРАВИТЕЛЬСТВА

ВЫПОЛНЕНИЕ РАБОТ ПО РАЗВИТИЮ ТИПОВОГО ТИРАЖИРУЕМОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ВИТРИН ДАННЫХ

Витрина данных НСУД

Техническое описание программы «Витрина данных Лайт»

Версия 1.13.0

Листов 49

Москва, 2024

СОДЕРЖАНИЕ

Аннотация

В данном документе приведено «Техническое описание программы» для ПО «Витрина данных Лайт» (далее – Программа), предназначенного для загрузки публикуемых данных в отдельную базу данных на стороне поставщика данных, а также для формирования отдельной базы данных в соответствии с результатами выполнения запросов на предоставление или репликации данных со стороны получателя данных. «Витрина данных Лайт», в отличии от ПО «Витрина данных НСУД», имеет ограниченный функционал, невысокие требования к аппаратному обеспечению и предназначена для небольших организаций.

В разделе «Общие сведения» указаны назначение и возможности Программы, основные характеристики Программы, технические и программные средства, режим работы, требования к персоналу.

В разделе «Структура программы» приведены сведения о составных частях Программы, ее модулях, составе компонентов в дистрибутиве, связях между составными частями и другими программами, а также карта портов.

В разделе «Архитектура программы» приведены архитектурная и компонентная схемы и алгоритм работы Программы.

В разделе «Описание технических решений» приведены описания задач реализованных технических решений, а также их описание.

В разделе «Вызов и загрузка» приведены описания действий по ручному запуску и остановке программы.

В разделе «Входные и выходные данные» приведено описание организации используемой входной и выходной информации.

Оформление программного документа «Техническое описание программы «Витрина данных Лайт»» произведено по требованиям ЕСПД (ГОСТ 19.101-77, ГОСТ 19.103-77, ГОСТ 19.104-78, ГОСТ 19.105-78, ГОСТ 19.106-78, ГОСТ 19.503-79, ГОСТ 19.604-78).

1 ОБЩИЕ СВЕДЕНИЯ

1.1 Обозначение и наименование программы

Полное наименование: Типовое тиражируемое программное обеспечение витрин данных конфигурации «Лайт».

Условное обозначение: ПО «Витрина данных Лайт».

1.2 Назначение программы

Национальная система управления данными (далее – НСУД) представляет собой систему, состоящую из взаимосвязанных элементов информационно-технологического, организационного, методологического, кадрового и нормативно-правового характера и обеспечивающую достижение целей и выполнение задач, обозначенных в Концепции Национальной системы управления данными, утвержденной распоряжением Правительства Российской Федерации от 3 июня 2019 года № 1189-р.

НСУД предназначена для управления информацией, содержащейся в информационных системах органов и организаций государственного сектора, а также в информационных ресурсах, созданных в целях реализации полномочий органов и организаций государственного сектора (далее – государственные данные) и для осуществления информационного обмена между Поставщиками и Получателями данных, присоединившимися к НСУД (далее – Участники НСУД).

Управление процессами информационного обмена между Участниками НСУД осуществляется средствами федеральной государственной информационной системы «Единая информационная платформа Национальной системы управления данными» (далее – ФГИС «ЕИП НСУД»).

Для передачи данных между Участниками НСУД используется среда взаимодействия НСУД, состоящая из Системы межведомственного электронного взаимодействия 3.0 (далее – СМЭВ) и (или) подсистемы обеспечения доступа к данным СМЭВ (далее – ПОДД СМЭВ) (СМЭВ 4.0), обеспечивающих транспорт и процессинг данных, а также агентов ПОДД СМЭВ, устанавливаемых на стороне Участников НСУД.

Программа является частью НСУД и предназначена для загрузки публикуемых данных в отдельную БД на стороне Поставщика данных. Программа представляет собой типовое программное обеспечение, устанавливаемое на стороне поставщиков данных.

1.3 Возможности программы

Программа обеспечивает выполнение следующих функций:

- автоматическая настройка взаимосвязей между компонентами программы;
- автоматический запуск всех необходимых компонентов программы после установки;
- автоматическая настройка витрины и структуры ее таблиц на основании содержимого XML-файла, загружаемого через пользовательский web-интерфейс;
- выгрузка шаблона через графический интерфейс (для упрощения процесса

подготовки загружаемых данных);

- загрузка данных в витрину:
  - через графический интерфейс;
  - REST API;
  - файловый обмен.
- настройка параметров работы витрины через графический интерфейс;
- выполнение запросов на предоставление данных в соответствии с протоколом ПОДД через механизмы СМЭВ ПОДД.

1.4 Рекомендуемые технические и программные средства

В разделе приведены рекомендации по аппаратному и программному обеспечению, а также необходимая конфигурация сети для оптимального баланса между производительностью и стабильностью работы всех компонентов программы. Рекомендация основана на использовании программы в режиме стандартной рабочей нагрузки на тестовом стенде.

> Примечание:
>
> Следует учитывать, что невозможно дать универсальной рекомендации для развертывания программы т.к. вариантов конфигурации оборудования, характера нагрузки и других факторов может быть очень много. Предварительный расчет параметров оборудования на этапе внедрения для каждой организации должен быть рассчитан индивидуально. Советуем установить программу с приведенными ниже рекомендациями на тестовом стенде для того чтобы определить оптимальную конфигурацию для ваших сценариев работы.

Рекомендуем выполнить следующее:

- продумайте сценарии работы с программой необходимые для достижения ваших целей;
- установите программу (см. Руководство по установке) на тестовом стенде с рекомендуемыми техническими характеристиками (Таблица 1.1);
- создайте структуру Витрины;
- подготовьте тестовые данные для загрузки и определите количество загружаемых данных;
- в процессе загрузки данных проведите измерение ключевых параметров нагрузки серверного оборудования;
- линейно экстраполируйте эти данные на целевую систему, получив таким образом загруженность целевого оборудования;
- выберете оборудование, которое будет соответствовать нагрузке для ваших задач.

Ниже приведены параметры тестового стенда, на котором проверялась работоспособность программы.

1.4.1 Серверное оборудование

Рекомендованные требования к серверному оборудования приведены в таблице (Таблица 1.1).

Таблица 1.1 Требования к серверному оборудованию

Сервер	Требования	CPU	RAM,ГБ	HDD, ГБ
Витрина данных	Минимальные требования	4	16	100
Витрина данных	Рекомендованные требования	10	128	500

1.4.2 Программное обеспечение

Таблица 1.2 Минимальный состав программных средств

Название	Описание	Версия
Операционная система (выбор опционален)	CentOS РЕД ОС АЛЬТ Сервер 8 СП Astra Linux 1.7 (уровень защищенности «Воронеж»)	7.9 7.2 8 1.7
Docker	Программное обеспечение для автоматизации развёртывания и управления приложениями	20.10.2

1.4.3 Требования к каналам связи

К каналам связи сервера, на котором будет установлена программа, предъявляются следующие требования:

- пропускная способность 100 Гбит/сек с протоколами TCP/IP и UDP;
- отсутствие ПО, блокирующего или замедляющего трафик.

К каналам связи, используемым для взаимодействия с клиентскими сетями Ведомства, предъявляются следующие требования:

- пропускная способность 10 Гбит/сек с протоколами TCP/IP и UDP.

Необходимо учитывать процент свободных ресурсов оборудования и пиковые нагрузки.

1.5 Режим работы программы

Программа предназначена для круглосуточного функционирования, 24 часа день 7 дней в неделю, с перерывами на плановое техническое обслуживание, восстановление работоспособности.

Программа поддерживает функционирование в следующих режимах:

- штатный;
- сервисный;
- аварийный;
- остановка.

1.5.1 Штатный режим

Штатный режим является основным режимом функционирования Программы. В этом режиме функционирования обеспечивается круглосуточная работа Программы без потери функциональности и работоспособности ее программно-технических средств.

1.5.2 Сервисный режим

Режим функционирования, при котором Программа выполняет часть основных функций, но параллельно с этим проводятся работы по техническому обслуживанию, накладывающие ограничения на функциональность Программы. В сервисном режиме функционирования обеспечивается возможность круглосуточного функционирования Программы.

1.5.3 Аварийный режим

Режим функционирования, при котором в Программе доступны только базовые функциональные возможности, необходимые для восстановления работоспособности.

1.5.4 Остановка

Режим работы, когда Программа не выполняет ни одну из своих функций.

1.6 Используемые языки программирования

Таблица 1.3 Языки программирования

Название	**Версия**	**Описание**
Kotlin	1.6.20	Язык программирования
Java	1.8.0_352	Язык программирования
SQL	SQL:2016	Язык программирования, применяемый для создания модификации и управления данными в реляционной базе данных
Html	5	Язык разметки
CSS	3	Язык разметки. Описание внешнего вида документов
Python	3.0	Язык программирования
JavaScript	ES2017 (ES8)	Язык программирования

1.7 Требования к персоналу

1.7.1 Требования к системному администратору

Системный администратор обеспечивает штатную работу программы. Выполняет установку, настройку и мониторинг работоспособности программного и аппаратного обеспечения.

В основные обязанности системного программиста входит:

- установка и настройка программы;
- установка обновлений;
- резервное копирование;

– настройка логирования;
– настройка системы мониторинга программы, аппаратного обеспечения и др.

Необходимые навыки для работы:

– Знание TCP/IP.
– Docker (развертывание, установка библиотек, администрирование).
– Ansible.
– Умение работать с веб-серверами (Apache).
– SSH.
– Резервное копирование.
– Логирование.
– Обновление программы.
– Понимание модели баз данных.
– Знание операционных систем CentOS на уровне администрирования.
– Опыт работы по управлению и администрированию баз данных.
– Навык проводить диагностику и анализ проблемных мест.
– Умение распознать следствие/причины некорректной работы ПО или техники.
– Умение анализировать сетевой трафик.

1.7.2 Требования к программисту

Программист решает задачи связанные непосредственно с сопровождением и доработкой программы под требования пользователей. Определяет схемы и алгоритмы обработки данных программой.

Программист должен обладать знаниями следующих языков программирования:

– Java;
– Kotlin;
– Javascript.

Уметь использовать в своей работе следующее программное обеспечение и технологии:

– Apache Zookeeper;
– Apache Kafka;
– Docker;
– Vertx;
– Spring (Spring-boot);
– Web (http, настройка SSH).

В перечень задач, выполняемых программистом, входит:

– поддержание работоспособности программы;

– доработка программы под требованиям пользователей системы.

2 СТРУКТУРА ВИТРИНЫ ДАННЫХ

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

2.1 Составные части Витрины данных

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

Общую схему взаимосвязей компонентов можно просмотреть в разделе Архитектура Витрины данных.

Функционально, программа состоит из следующих частей:

2.1.1 Основные компоненты

- **ProStore** — основной компонент программы с открытым исходным кодом, обеспечивает единый интерфейс к хранилищу разнородных данных. Определяет структуры данных, запись и чтение данных Витрины. Позволяет работать со входящими в состав хранилища СУБД одинаковым образом, используя единый синтаксис запросов SQL и единую логическую схему данных. *ProStore* включает следующие компоненты:
  - *Сервис исполнения запросов* — анализирует и исполняет SQL-запросы; предоставляет REST API для JDBC-драйвера и взаимодействует с сервисом мониторинга статусов Kafka по REST API. В свою очередь состоит из следующих компонентов:
    - Коннектор *Kafka-Postgres reader* - считывает данные из PostrgeSQL и передает их в брокер сообщений Kafka.
    - Коннектор *Kafka-Postgres writer* - записывает данные из брокера сообщений Kafka в PostrgeSQL.
  - *Сервис мониторинга статусов Kafka* — отслеживает состояние топиков брокера сообщений Kafka; предоставляет REST API для сервиса исполнения запросов.
- **PostrgeSQL** — база данных ProStore.
- **Apache ZooKeeper** — необходим для поддержки информации о конфигурации и распределенной координации между компонентами Витрины, также используется как сервисная база данных ProStore, для хранения технической информации (метаданных) от поступающих в Витрину данных запросах.
- **Брокер сообщений Kafka** — используется для непрерывной передачи сообщений между:

- CSV-uploader и ProStore;
- ПОДД-адаптер-Модуль исполнения запросов и Агент ПОДД.

- **CSV-uploader** — модуль программы, который предназначен для загрузки и выгрузки csv/xml-файлов.
- **ПОДД-адаптер** — общее название логических модулей программы, которые обеспечивают подключение к ПОДД СМЭВ, как информационной системы участника взаимодействия. В зависимости от предназначения логические модули обеспечивают загрузку запросов из очереди ИС УВ в ПОДД СМЭВ, формирование и отправку ответов в ПОДД СМЭВ, инициативное формирование уведомлений об изменении данных в экземпляре ПО «Витрина данных НСУД», отправку уведомлений в ПОДД СМЭВ, регистрацию реплики данных ИС УВ, подписки на репликацию и поддержку реплики в актуальном состоянии.

ПОДД-адаптер состоит из следующих логических модулей:

- **ПОДД-адаптер-Модуль исполнения запросов** — предназначен для подключения программы к ПОДД СМЭВ (через протокол коммуникации Агент ПОДД), как информационной системы участника взаимодействия;
- **ПОДД-адаптер – Модуль MPPR** — предназначен для загрузки данных табличных параметров в многопоточном режиме (massively parallel processing, MPP);
- **Модуль подписок** — предназначен для управления подписками между Получателем данных (consumer) и Поставщиком данных (producer).

2.1.2 Дополнительные компоненты

Дополнительное программное обеспечение для администрирования и мониторинга:

- **Prometheus** — используется как система мониторинга системных ресурсов «Витрина данных Лайт». 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 «Витрина данных Лайт» 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 «Витрина данных Лайт», без прерывания работы сервисов в продуктовой среде. Официальный сайт разработчика приложения: https://www.docker.com/.
- **Portainer** — web-приложение для управления docker-контейнерами. Официальный сайт разработчика приложения: https://www.portainer.io/.
- **Graylog** — программное обеспечение для управления лог-файлами. Официальный сайт разработчика приложения: https://www.graylog.org/.
- **MongoDB** — база данных *Graylog*. Официальный сайт разработчика приложения: https://www.mongodb.com/.
- **Elasticsearch** — утилита полнотекстового поиска и аналитики, которая позволяет быстро в режиме реального времени хранить, искать и анализировать большие объемы данных и сохраняет их для *Graylog*. Для передачи сообщений в *Graylog* использует *Filebeat*. Официальный сайт разработчика приложения: https://www.elastic.co/elasticsearch/.
- **Filebeat** — агент на сервере для отправки различных типов оперативных данных в *Elasticsearch*. Официальный сайт разработчика приложения: https://www.elastic.co/elasticsearch/.
- **Node_exporter** — процессы, обеспечивающие сбор и передачу системных метрик серверу *Prometheus*. Также, используется для сбора метрик *ПОДД-адаптера* и *CSV-uploader* см. https://github.com/prometheus/node_exporter.

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);
- РЕД ОС версии 7.2 (далее - РЕД ОС);
- АЛЬТ Сервер 8 СП (далее - АЛЬТ ОС);

– Astra Linux 1.7 (уровень защищенности «Воронеж»).

2.2 Модули Витрины данных

2.2.1 ПОДД-адаптер — Модуль исполнения запросов

2.2.1.1 Общее описание

ПОДД-адаптера - Модуль исполнения запросов Логический модуль ПОДД-адаптера, предназначен для исполнения запросов ПОДД СМЭВ (через протокол коммуникации Агент ПОДД).

Установка опциональна

Обмен сообщениями между *ПОДД-адаптера - Модуль исполнения запросов* и Агент ПОДД происходит через заранее согласованные топики брокера сообщений Kafka.

2.2.1.1.1 Общая схема взаимодействия через ПОДД-адаптер- Модуль исполнения запросов

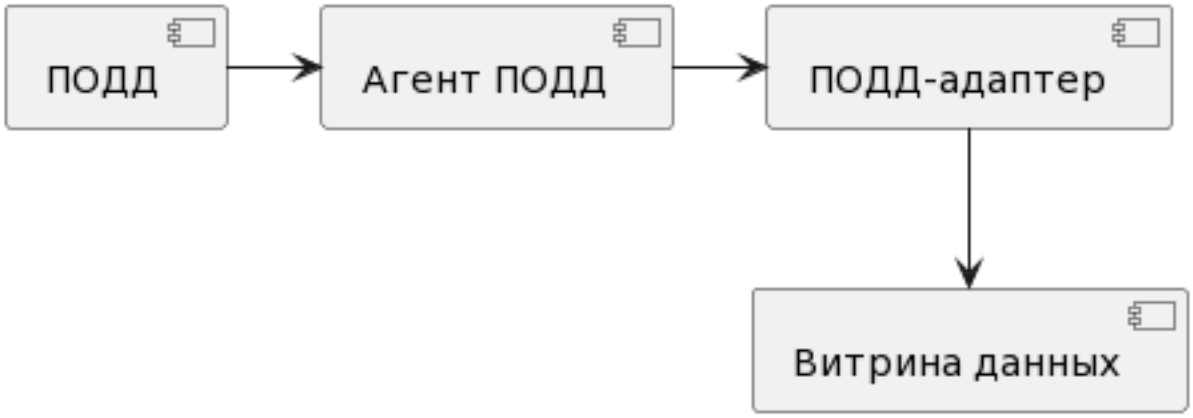

Рисунок - 2.1 Взаимодействие программы с ПОДД

2.2.1.1.2 Процесс обработки запроса через ПОДД-адаптер- Модуль исполнения запросов

1. Получатель данных отправляет через ПОДД запрос к Витрине данных.
2. Запрос поступает в Агент ПОДД.
3. *ПОДД-адаптер- Модуль исполнения запросов* (через заранее согласованные топики брокера сообщений Kafka) получает запрос от Агента ПОДД на предоставление данных.
4. *ПОДД-адаптер- Модуль исполнения запросов* обрабатывает запрос и отправляет его в Витрину данных.
5. Витрина данных обрабатывает запрос и формирует на него ответ в ПОДД-адаптер.
6. *ПОДД-адаптер- Модуль исполнения запросов* обрабатывает ответ, записывает результат в заранее согласованные топик обмена сообщениями и предоставляет ответ Агенту ПОДД.
7. Агент ПОДД отправляет полученный ответ через ПОДД Получателю данных.

Процесс получения BLOB-объектов через *ПОДД-адаптер- Модуль исполнения запросов* описан в разделе Взаимодействие через ПОДД-адаптер.

2.2.2 ПОДД-адаптер — Модуль MPPR

2.2.2.1 Общее описание

Логический модуль ПОДД-адаптер - Модуль MPPR является частью ПОДД-адаптер и предназначен для чтения данных в многопоточном режиме (*massively parallel processing*, MPP).

ПОДД-адаптер - Модуль MPPR предназначен для следующих задач:

1. Многопоточное параллельное чтение данных.
2. Отправка ответа с результатом запроса в Агент ПОДД.
3. Удаление временных таблиц, созданных на основе табличных параметров.

Обмен сообщениями между **ПОДД-адаптером** и **ПОДД-адаптером - Модуль MPPR** происходит через топик `mppr.query`.

2.2.2.1.1 Общая схема взаимодействия

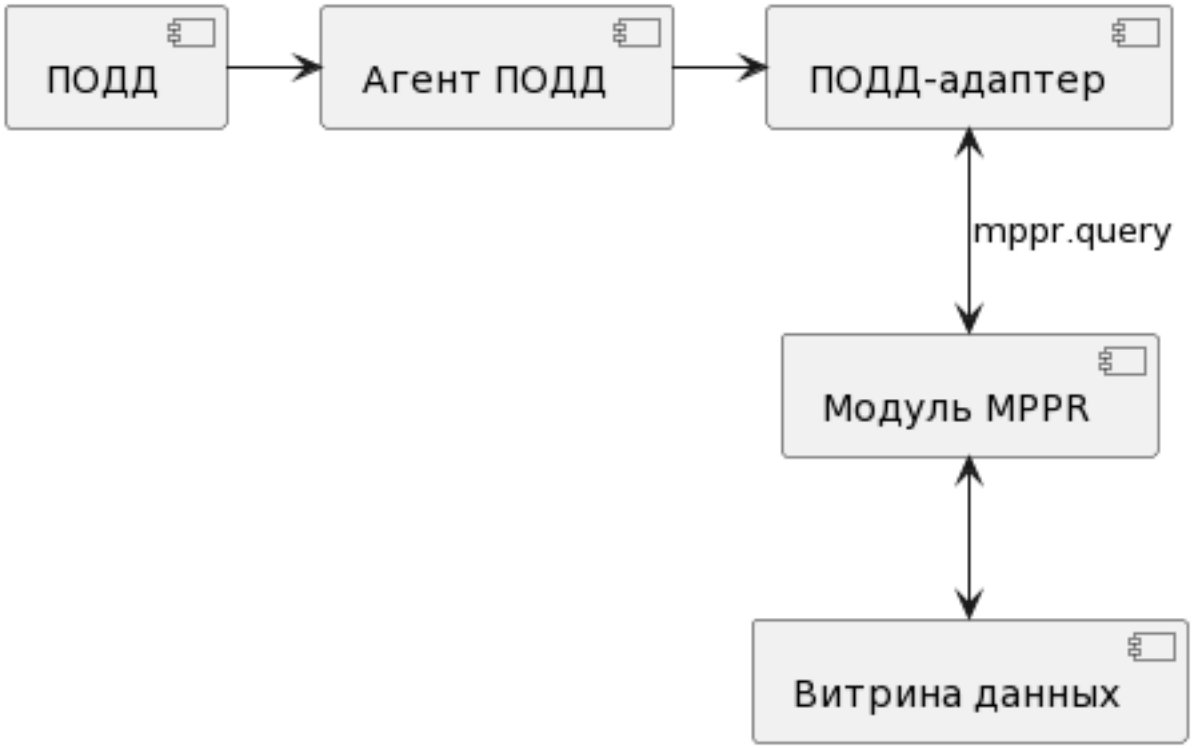

Рисунок - 2.2 Взаимодействие через Модуль MPPR

2.2.2.1.2 Процесс обработки запроса через Модуль MPPR

1. Получатель данных отправляет через ПОДД запрос к Витрине данных.
2. Запрос поступает через **Агента ПОДД** в **ПОДД-адаптер**.
3. Если формат обработки данных предполагает MPP, то **ПОДД-адаптер** отправляет запрос через топик `mppr.query` в **ПОДД-адаптер - Модуль MPPR**.
4. **ПОДД-адаптер - Модуль MPPR** создает временную таблицу (по результатам запроса) и временный топик с запросом для Витрины.
5. Витрина считывает топик, обрабатывает запрос, формирует на него ответ.

6. **ПОДД-адаптер - Модуль MPPR** получает ответ и выкладывает полученные данные во временную таблицу.
7. **ПОДД-адаптер** считывает ответ из временной таблицы и отправляет данные в **Агент ПОДД**.
8. **Агент ПОДД** отправляет полученный ответ через ПОДД Получателю данных.
9. **ПОДД-адаптер - Модуль MPPR** удаляет временный топик и таблицу.

2.2.3 Модуль подписок

2.2.3.1 Общее описание

Модуль **ПОДД-адаптер-Модуль подписок** предназначен для управления подписками между Получателем данных (consumer) и Поставщиком данных (producer).

Модуль используется для получения результатов комплексных запросов из нескольких Витрин источников. Подписка позволяет автоматически загрузить и поддерживать в актуальном состоянии данные из Витрины Поставщика в специальном хранилище на стороне Потребителя - Хранилище данных по подписке. Потребитель посылает запросы напрямую в своё *Хранилище данных по подписке*, в результате чего сокращается продолжительность сеансов обмена и необходимость «склейки» запросов на стороне ПОДД.

Обмен между Витринами осуществляется по предварительно созданной *подписке на уведомления об изменениях или репликацию*.

Модуль решает следующие задачи:

- Запрос создания подписки (Поставщик данных);
- Запрос отмены подписки (Поставщик данных);
- Запрос дельты (Поставщик данных);
- Запрос создания структуры по подписке (Получатель данных);
- Запрос применения дельты (Получатель данных).

Потребители данных могут получать сведения из Витрин Поставщиков данных путем:

1. отправки регламентированных запросов;
2. подписки на изменения сведений.

Подписка позволяет автоматически загрузить и поддерживать в актуальном состоянии данные из Витрины Поставщика в специальном хранилище на стороне Потребителя (Хранилище данных по подписке) и Потребитель посылает запросы напрямую в своё Хранилище, в результате чего сокращается продолжительность сеансов обмена и необходимость «склейки» запросов на стороне ПОДД.

Информационный обмен по подписке состоит из следующих этапов:

1. Регистрация подписки в Витрине Поставщика данных и создание структуры данных в Хранилище Потребителя данных.
2. Передача снапшота из Витрины Поставщика данных в Хранилище Потребителя данных (только для подписки на репликацию). В текущей реализации снапшот не содержит

историчность.

3. Актуализация данных посредством передачи пакета дельт от Витрины Поставщика данных в Хранилище Потребителя данных в одном из режимов:

   – по расписанию (если оно указано в подписке);
   – по событию об изменении данных (если расписание не указано в подписке).

   Подписка определяется следующими параметрами:

   – уникальный идентификатор подписки;
   – источник данных по подписке – мнемоника Витрины Поставщика данных;
   – адресат данных по подписке – мнемоника Витрины Потребителя данных;
   – набор SQL-выражений, каждое из которых описывает подмножество данных Витрины;
   – расписание синхронизации (может отсутствовать).

   Виды подписок:

   – Подписка на репликацию - снэпшот текущего состояния витрины;
   – Подписка на уведомление - выгружаем данные только по дельте.

   Реализованы два варианта подписки:

   – одиночная;
   – распределенная.

   Ключевые особенности одиночных подписок:

   – подписка только на один датамарт;
   – в одиночных подписках можно создать подписку с множественными SQL-запросами к разным таблицам одной витрины.

   Ключевые особенности распределенных подписок:

   – количество витрин-источников больше 1;
   – одной подписке соответствует один SQL-запрос;
   – один датамарт может фигурировать в нескольких подписках витрины потребителя.

В случае необходимости отключить подписку, осуществляется отмена подписки через ВС «Отмена подписки на репликацию или уведомлений в изменении данных».

2.2.4 CSV-uploader

2.2.4.1 Общее описание

CSV-uploader - программный модуль Витрины данных, который предназначен для загрузки CSV-файлов в Витрину данных.

CSV-uploader предназначен для следующих задач:

- загрузка CSV-файлов;
- загрузка CSV-файлов со структурой Витрины;
- выгрузка CSV-шаблонов с демо-шаблонами структуры Витрины;
- автоматический запуск загрузки CSV-файлов по расписанию из выбранного каталога;
- просмотр *Журнала операций*.

Внимание:
Загружаемые файлы обязательно должны быть в кодировке UTF-8

2.2.4.1.1 Общая схема взаимодействия через CSV-uploader

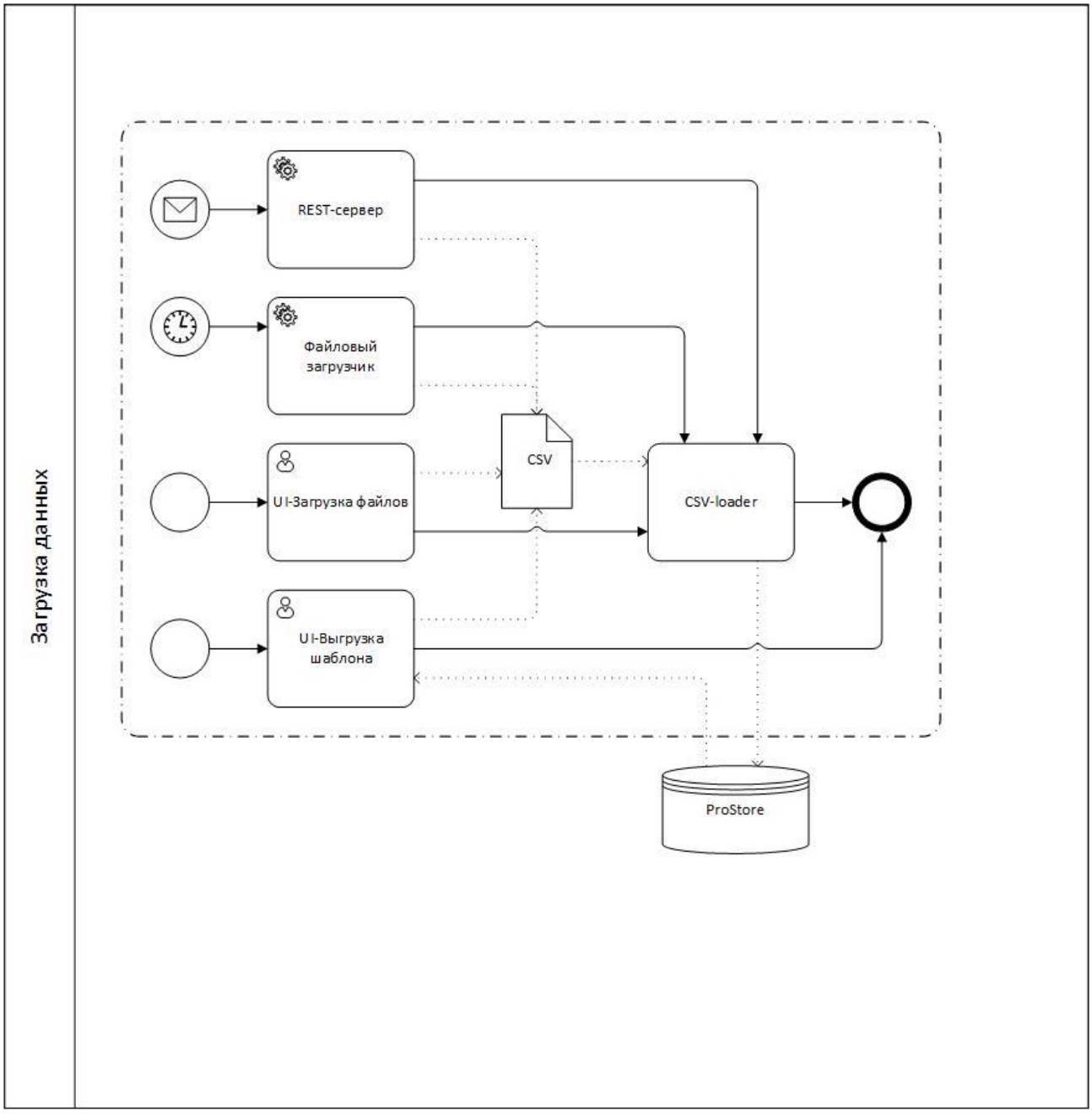

Рисунок - 2.3 Общая схема взаимодействия через CSV-uploader

2.3 Состав компонентов в дистрибутиве

Перечень состава компонентов программы версии 1.13.0 приведен в таблице ниже (см. Таблица 2.1)

Таблица 2.1 Состав компонентов в дистрибутиве программы

Наименование компонента	Версия	Техническое наименование
graylog	4.3.15	graylog:4.3.15
prometheus	2.34.0	Prometheus:2.34.0
podd-adapter-query	1.13.0	podd-adapter:1.13.0
podd-adapter-mppr	1.13.0	podd-adapter-mppr:1.13.0
podd-adapter-mppw	1.13.0	podd-adapter-mppw:1.13.0
podd-adapter-replicator	1.13.0	podd-adapter-replicator:1.13.0
podd-adapter-group-repl	1.13.0	podd-adapter-group-repl:1.13.0
csv-uploader	1.13.0	csv-uploader:1.13.0
kafka-jet-writer	1.2.0	kafka-jet-writer:1.2.0
query-execution	6.8.1	query-execution:6.8.1
grafana	9.2.15	Grafana:9.2.15
node_exporter	1.3.1	node_exporter:1.3.1
filebeat	7.10.2	Filebeat:7.10.2
mongo	4.4	Mongo:4.4
elasticsearch	1.3.14	Elasticsearch:1.3.14
PostgreSQL	13.4	PostgreSQL:13.4
kafka	2.13	Kafka:2.13-2.6.0-alt-p10-r3
zookeeper	3.5.7	Zookeeper:3.5.7-alt-p10-r3
portainer	2.14.0	Portainer:2.14.0
kafka-postgres-writer	0.6.1	kafka-postgres-writer:0.8.0
kafka-postgres-reader	0.6.1	kafka-postgres-reader:0.8.0

2.4 Связи между составными частями

Взаимосвязи между составными частями программы приведены в Таблица 2.2.

Таблица 2.2 Взаимодействие между составными частями

Клиент	Сервер	Способ взаимодействия	Описание
ПОДД-адаптер — Модуль исполнения запросов	ProStore	JDBC Брокер сообщений Kafka	Исполнение запросов.
CSV-Uploader	ProStore	JDBC Брокер сообщений Kafka	Управление логической структурой таблиц. Загрузка публикуемых данных в Витрину.
ProStore	СУБД PostgreSQL	JDBC	Управление логической структурой таблиц. Исполнение запросов.

			Управление загрузкой публикуемых данных в Витрину.
ПОДД- адаптер — Модуль MPPR	Агент ПОДД	Через брокера сообщений Kafka	Предоставляет Результат запрос/подзапрос на получение публикуемых данных (в т.ч. с использованием ТП), делегированного ПОДД-адаптером.

2.5 Связи с другими программами

Взаимодействие с другими программами происходит путем вызова соответствующих модулей программы:

- Внутренняя ИС Ведомства взаимодействует с ProStore через JDBC-driver.
- *ПОДД-адаптер - Модуль исполнения запросов* для взаимодействия с ИС участников взаимодействия через Агента ПОДД.
- CSV-uploader для взаимодействия с ИС участников взаимодействия для передачи файлов в формате XML и CSV.

Связи программы со сторонними программами приведены в см. Таблица 2.3.

Таблица 2.3 Связи с другими программами

Клиент	**Сервер**	**Способ взаимодействия**	**Описание**
Внутренняя ИС Ведомств	CSV-uploader	Файловый обмен (CSV) REST	Загрузка публикуемых данных в Витрину
	ProStore	JDBC Брокер сообщений Kafka	Управление логической структурой таблиц. Исполнение запросов. Загрузка публикуемых данных в Витрину.
ПОДД-адаптер — Модуль исполнения запросов	Агент ПОДД	Брокер сообщений Kafka Kafka	Исполнение запросов.

2.6 Карта портов

Таблица 2.4 Карта портов

Компонент	**Описание**
podd-adapter	Порт: 8083 Протокол: HTTP Описание: Взаимодействие с ПОДД-адаптером
csv-uploader	Порт: 8080 Протокол: HTTP Описание: Взаимодействие с CSV-uploader
query-execution	Порт: 8080 Протокол: HTTP Описание: номер порта сервиса метрик

	Порт: 9090 Протокол: TCP Описание: номер порта сервиса исполнения запросов
status_monitor	Порт: 9095 Протокол: HTTP Описание: сетевой адрес и путь для получения информации о статусе сервиса
prometheus	Порт: 9090 Протокол: HTTP Описание: Подключение к Prometheus WEB UI
grafana	Порт: 3000 Протокол: HTTP Описание: Web-интерфейс для работы с Grafana
node_exporter	Порт: 9100 Протокол: HTTP Описание: Порт для загрузки метрик
filebeat	Порт: нет открытых портов Протокол: - Описание: -
mongodb	Порт: 27017 Протокол:TCP Описание: Подключение к MongoDB. Порт по умолчанию для экземпляров mongod и mongos. Вы можете изменить этот порт с помощью port или –port . Порт: 27018 Протокол: TCP Описание: Подключение к MongoDB. Порт по умолчанию для mongod при запуске с параметром командной строки –shardsvr или значением shardsvr для параметра clusterRole в файле конфигурации
elasticsearch	Порт: 9200 Протокол: HTTP Описание: Подключение к Elasticsearch.
kafka_postgres_writer	Порт: 8096 Протокол: HTTP Описание: Порт используется для записи топиков Kafka в ProStore
kafka_postgres_reader	Порт: 8094 Протокол: HTTP Описание: Порт используется для чтения топиков Kafka из ProStore
postgres	Порт: 5432 Протокол: TCP PostgresSQL Protocol Описание: Источник данных SQL
kafka	Порт: 9092 Протокол: Порт используется для Описание: TCP

zookeeper	Порт: 2181 Протокол: TCP Описание: Порт используется для доступа к Zookeeper
portainer	Порт: 9000 Протокол: HTTP Описание: Web-интерфейс для работы с Portainer

3 АРХИТЕКТУРА ВИТРИНЫ ДАННЫХ

3.1 Общая архитектурная схема

Рассмотрим внутреннюю архитектуру Витрины данных.

Схематичное отображение общей архитектуры Витрины данных приведено на рисунке ниже (см. Рисунок - 3.1).

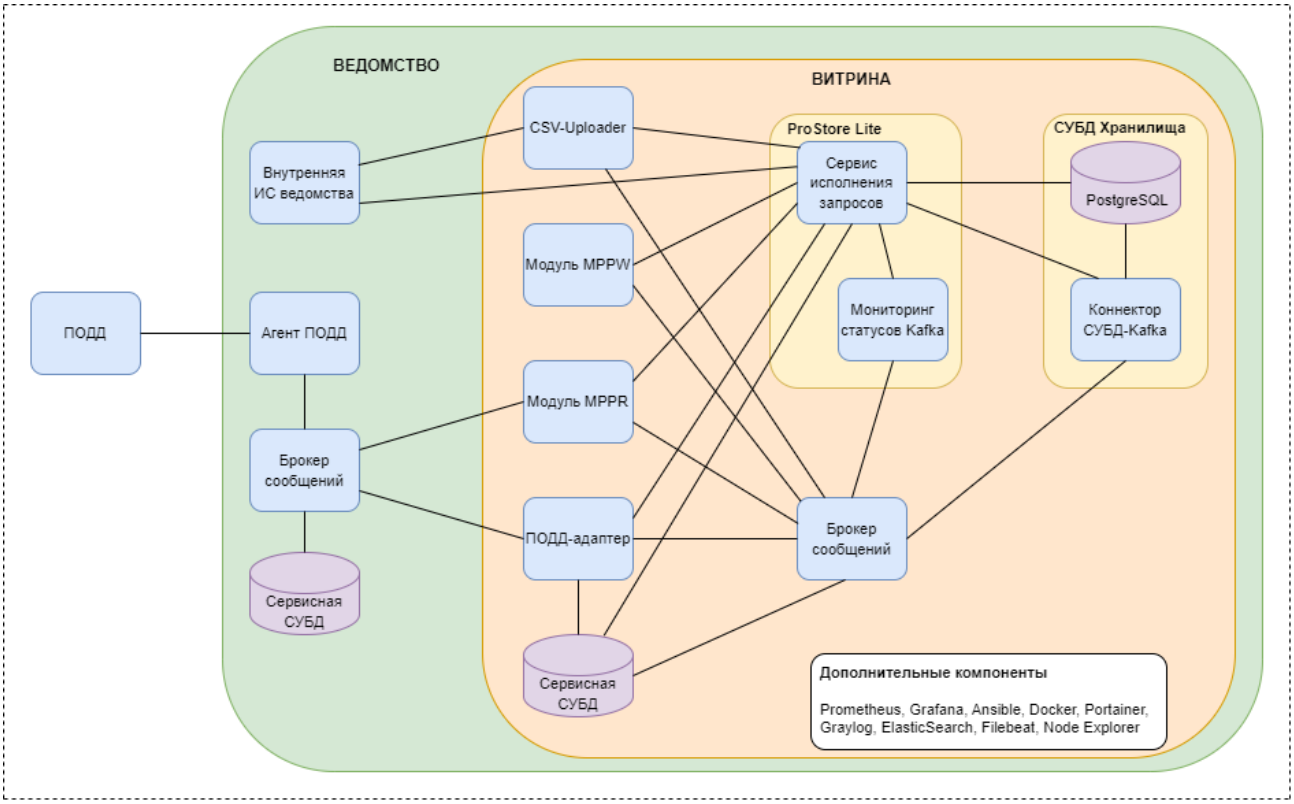

Рисунок - 3.1 Общая архитектура Витрины данных

3.2 Общая компонентная схема

Схема компонентов Витрины данных представлена на рисунке ниже (см. Рисунок - 3.2).

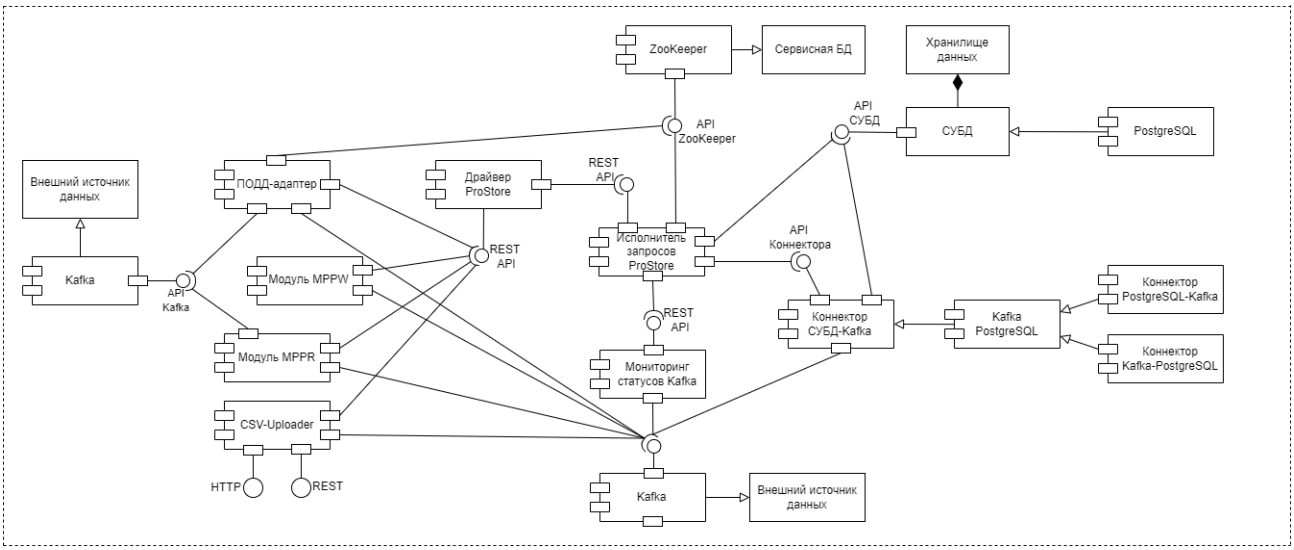

Рисунок - 3.2 Схема компонентов

3.3 Схема развертывания

На рисунке ниже (см. Рисунок - 3.3) приведена схема развертывания программы.

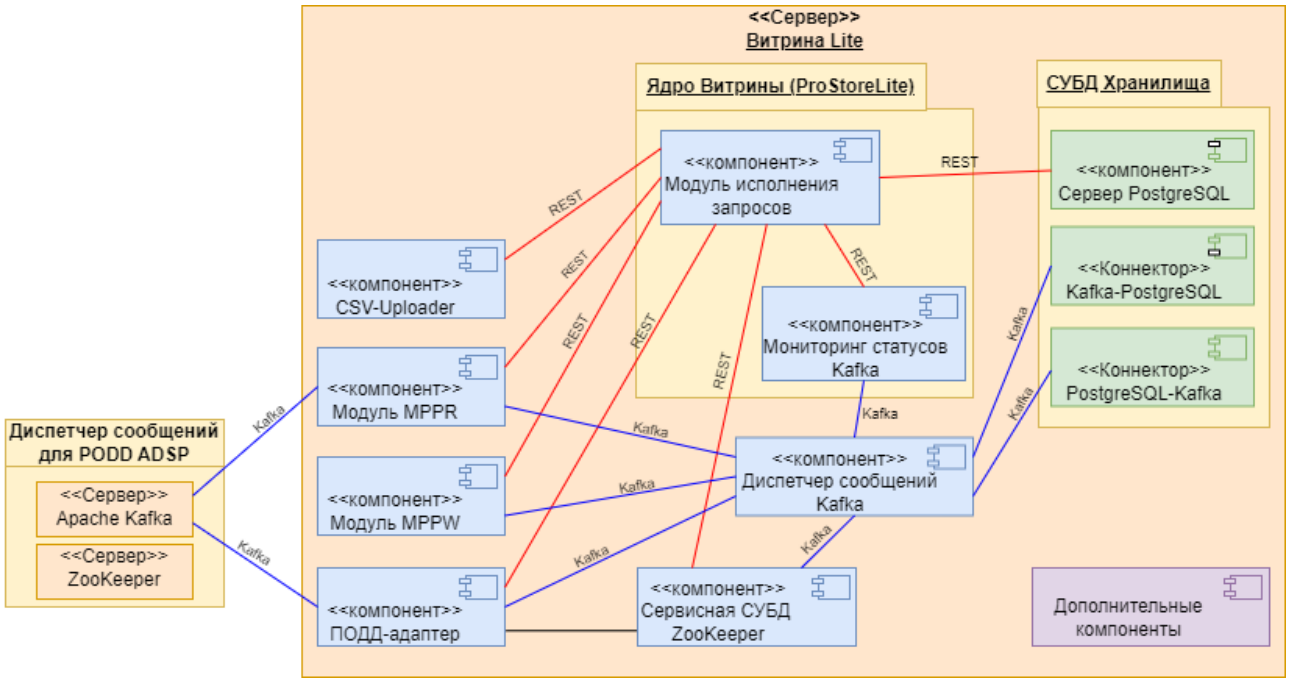

Рисунок - 3.3 Схема развертывания

3.4 Описание логической структуры

1. Установка программы осуществляется с помощью *Ansible* на предварительно сконфигурированный ПК (см. «Руководство по установке»).
2. После установки программа не содержит никакой логической модели данных. Необходимо загрузить структуру витрины через web-интерфейс программы (для хранения данных в качестве Поставщика данных). За загрузку структуры витрины отвечает модуль CSV-uploader. Описание загрузки структуры приведено в документе «Руководство администратора» в разделе «Инструкция по эксплуатации CSV-uploader». При работе с ПОДД структура таблиц настраивается в ЕИП НСУД и передаются в Витрину через ПОДД.
3. После того, как логическая модель данных в Витрине настроена можно:

   – обрабатывать SQL-запросы в качестве Поставщика данных;
   – выгружать шаблон через web-интерфейс;
   – загружать данные в Витрину через:
   – web-интерфейс;
   – файловый обмен;
   – REST.

4 ОПИСАНИЕ ТЕХНИЧЕСКИХ РЕШЕНИЙ

4.1 Задачи реализованных технических решений

Технические решения, реализованные при разработке программы, решают следующие задачи:

- упрощение процесса установки;
- создание структуры таблиц витрины;
- выгрузка шаблона;
- загрузка данных в витрину:
- через графический интерфейс (CSV-файлы);
- через REST API;
- через файловый обмен.
- настройка параметров работы витрины через графический интерфейс;
- журналирование событий функциональных блоков;
- мониторинг информации о работоспособности экземпляра программы;
- совместимость Типового ПО Витрина данных конфигурации Лайт с РЕД ОС версии 7.2;
- совместимость Типового ПО Витрина данных конфигурации Лайт с АЛЬТ Сервер 8 СП;
- совместимость Типового ПО Витрина данных конфигурации Лайт с Astra Linux 1.7;
- выполнение запросов с системными параметрами.

4.1.1 Задача «Упрощение процесса установки»

Для решения этой задачи используется **ansible-плейбук** (`playbook.yml`), который выполняет следующие функции:

- запускает **Docker**;
- конфигурирует сервер (в docker-контейнере);
- устанавливает *Portainer* (web-приложение для управления docker-контейнерами);
- устанавливает программу «Витрина данных Лайт»;
- настраивает взаимосвязи между компонентами витрины;
- запускает программу.

Программа **Portainer** предоставляет графический интерфейс для управления docker-контейнерами с компонентами программы.

4.1.2 Задача «Создание структуры таблиц витрины»

Для решения этой задачи администратор витрины через web-интерфейс **CSV-uploader** загружает заранее сконфигурированный XML-файл со структурой витрины. Витрина на основании содержимого XML-файла создает таблицы с указанными именами и полями в своей БД.

Внимание:
Программа не поддерживает модификацию ранее загруженной структуры данных витрины (изменения возможны только путем удаления и повторной загрузки структуры таблиц и данных).

Процесс загрузки структуры витрины см. в документе «Инструкция по эксплуатации CSV-uploader».

4.1.3 Задача «Выгрузка шаблона»

Для решения этой задачи разработан графический интерфейс, в котором администратор витрины имеет возможность выбрать нужную таблицу и скачать ее на свой ПК (чтобы использовать ее в качестве примера для подготовки загружаемых данных).

В графическом интерфейсе администратор имеет возможность выбрать нужную таблицу, Витрина по метаданным своей БД определяет структуру выбранной таблицы и передает на ПК администратора витрины CSV-файл с шаблоном для выбранной таблицы.

Процесс выгрузки шаблона см. в документе «Инструкция по эксплуатации CSV-uploader».

4.1.4 Задача «Загрузка данных в витрину»

В программе предусмотрено три варианта загрузки данных:

- через графический интерфейс (CSV-файлы);
- REST API;
- файловый обмен.

4.1.4.1 Загрузка данных через графический интерфейс (CSV-файлы)

Для решения этой задачи администратор витрины через графический интерфейс модуля CSV-uploader выбирает таблицу и загружает csv-файл с обновленными/добавленными данными. Витрина записывает данные в таблицу своей БД (см. документ «Инструкция по эксплуатации CSV-uploader»).

4.1.4.2 Загрузка данных через REST API

Для решения этой задачи в программе предусмотрен модуль, который выступает в качестве REST-сервера, он позволяет выполнять стандартные `POST`, `DELETE` запросы для `URL`, содержащие имя таблицы. Полученные данные модуль передает модулю CSV-uploader, который загружает данные в витрину.

4.1.4.3 Загрузка данных через файловый обмен

Для решения этой задачи в программе настроена периодическая проверка папки выбранной для файлового обмена. В случае появления в папке нового CSV-файла, витрина считывает его, содержимое CSV-файла передает модулю CSV-uploader, который загружает данные в витрину. После передачи данных витрина удаляет CSV-файл из папки.

4.1.5 Настройка параметров работы витрины через графический интерфейс

Для решения этой задачи разработан графический интерфейс, в котором администратор Витрины имеет возможность настроить параметры витрины (см. документ «Инструкция по эксплуатации CSV-uploader»).

4.1.6 Задача «Журналирование событий функциональных блоков»

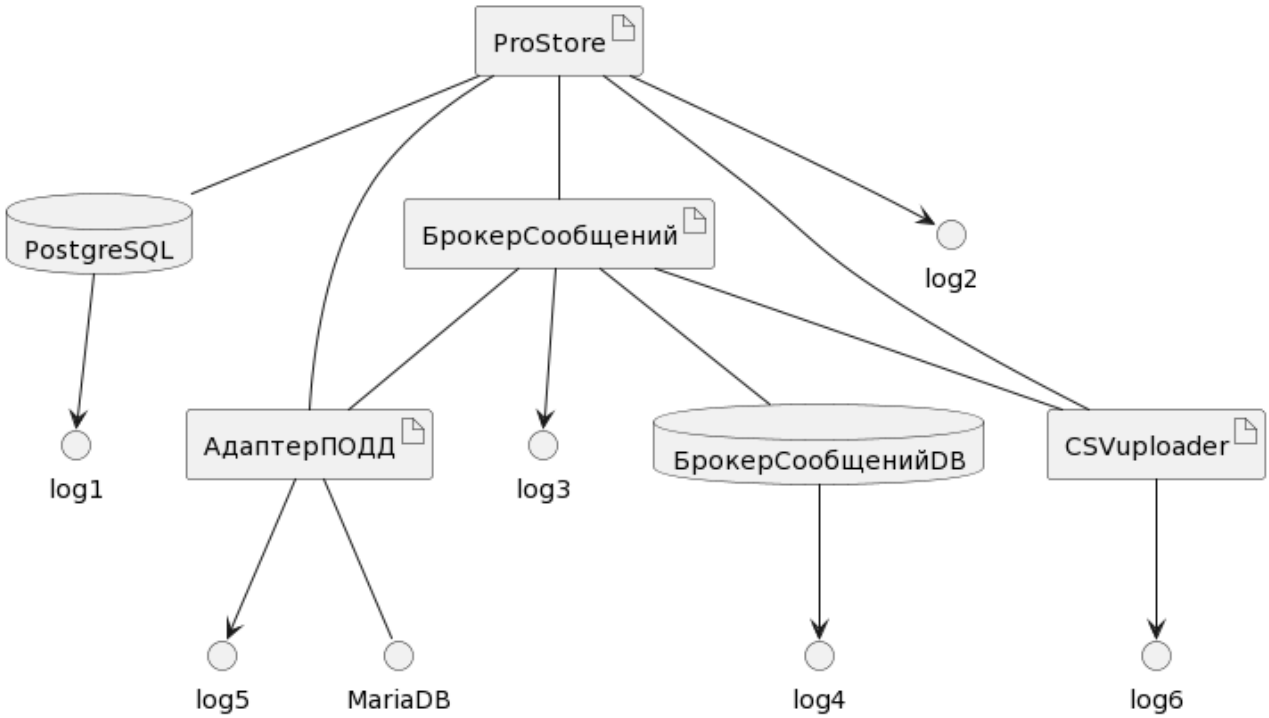

Рисунок - 4.1 Мониторинг и журналирование событий в системе

Задача включает журналирование событий функциональных блоков программы:

- Prostore;
- БД Prostore (PostgreSQL);
- Брокер сообщений;
- БД Брокера сообщений;
- ПОДД-адаптер;
- CSV-uploader.

При реализации указанного метода программа выполняет следующие основные операции по обработке данных:

- запись системных событий (отдельно по каждому функциональному блоку) осуществляется в лог-файлы (см. Рисунок - 4.1).
- предоставление возможности просмотра журнала событий (лог-файла).

Просмотр лог-файлов возможен через графический интерфейс web-приложения *Grafana*.

Настройку детализации лог-файлов осуществляет администратор.

4.1.7 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 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;
- Сервисная СУБД Prostore;
- ПОДД-адаптер – Модуль исполнения запросов;
- ПОДД-адаптер – Модуль MPPR;
- ПОДД-адаптер – Модуль MPPW;
- CSV-uploader.

Сбор лог-файлов программы с записями о событиях производится с помощью **Prometheus**. Процесс просмотра лог-файлов описан в разделе «Дополнительные возможности» документа «Руководство администратора».

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

4.1.8 Задача «Совместимость Типового ПО Витрина данных конфигурации Лайт с РЕД ОС версии 7.2»

Технические решения, реализованные в рамках работ по адаптации типового тиражируемого программного обеспечения «Витрина данных Лайт»» для операционной системы «РЕД ОС», выполнены в соответствии с Государственным контрактом .

ПО «Витрина данных Лайт» – типовое тиражируемое программное обеспечение витрин данных, являющееся частью СМЭВ (1 класс защищенности государственной информационной системы, 3 уровень защищенности персональных данных), применяемое в составе информационных систем обладателей (пользователей) государственных данных для создания

ведомственных витрин.

Защита информации, обрабатываемой с использованием ПО «Витрина данных Лайт» обеспечивается самостоятельно обладателями (пользователями) государственных данных с учетом требований законодательства Российской Федерации.

ПО «Витрина данных Лайт» предназначено для загрузки публикуемых данных в отдельную базу данных на стороне обладателя данных.

В соответствии с положениями нормативных документов, указанных выше, целями создания версии ПО «Витрина данных Лайт» являются:

- Адаптация ПО «Витрина данных Лайт» для расширения перечня операционных систем, под управлением которых работает ПО «Витрина данных Лайт» (операционной системы «РЕД ОС»);
- Реализация и актуализация информационного контента, описывающего функциональные, архитектурные и эксплуатационные возможности ПО «Витрина данных Лайт».
- Проведение тестирования модернизированного ПО «Витрина данных Лайт».

В рамках адаптации программы для работы в операционной системе «РЕД ОС» были выполнены работы:

- Проверка работы CSV Uploader в операционной системе «РЕД ОС»:
  - реализация загрузки схемы из XML ЕИП НСУД;
  - загрузка данных CSV через интерфейс;
  - загрузка данных CSV из каталога;
  - журналирование загрузки со статусами;
  -
    - экран для отображения статусов компонент;
- Проверка работы ПОДД Адаптер в операционной системе «РЕД ОС»:
  - чтение из Агента по алгоритму запроса;
  - ответ в Агент по алгоритму запроса;
  - сериализация в AVRO для запроса данных;
- Проверка работы Health Check в операционной системе «РЕД ОС»:
  - Health-Check - Liveness;
  - Health-Check - Readyness.
- Выполнение нагрузочного и регрессионного тестирования.

4.1.9 Задача «Совместимость Типового ПО Витрина данных конфигурации Лайт с АЛЬТ Сервер 8 СП»

Технические решения, реализованные в рамках работ по адаптации типового тиражируемого программного обеспечения «Витрина данных Лайт»» для операционной

системы «АЛЬТ Сервер 8 СП», выполнены в соответствии с Государственным контрактом.

ПО «Витрина данных Лайт» – типовое тиражируемое программное обеспечение витрин данных, являющееся частью СМЭВ (1 класс защищенности государственной информационной системы, 3 уровень защищенности персональных данных), применяемое в составе информационных систем обладателей (пользователей) государственных данных для создания ведомственных витрин.

Защита информации, обрабатываемой с использованием ПО «Витрина данных Лайт» обеспечивается самостоятельно обладателями (пользователями) государственных данных с учетом требований законодательства Российской Федерации.

ПО «Витрина данных Лайт» предназначено для загрузки публикуемых данных в отдельную базу данных на стороне обладателя данных.

В соответствии с положениями нормативных документов, указанных выше, целями создания версии ПО «Витрина данных Лайт» являются:

- адаптация ПО «Витрина данных Лайт» для расширения перечня операционных систем, под управлением которых работает ПО «Витрина данных Лайт» (операционной системы «АЛЬТ Сервер 8 СП»);
- Реализация и актуализация информационного контента, описывающего функциональные, архитектурные и эксплуатационные возможности ПО «Витрина данных Лайт».
- Проведение тестирования модернизированного ПО «Витрина данных Лайт».

В рамках адаптации программы для работы в операционной системе «АЛЬТ Сервер 8 СП» были выполнены работы:

- Проверка работы CSV Uploader в операционной системе «АЛЬТ Сервер 8 СП»:
  - реализация загрузки схемы из XML ЕИП НСУД;
  - загрузка данных CSV через интерфейс;
  - загрузка данных CSV из каталога;
  - журналирование загрузки со статусами;
  - экран для отображения статусов компонент;
- Проверка работы ПОДД Адаптер в операционной системе «АЛЬТ Сервер 8 СП»:
  - чтение из Агента по алгоритму запроса;
  - ответ в Агент по алгоритму запроса;
  - сериализация в AVRO для запроса данных;
- Проверка работы Health Check в операционной системе «АЛЬТ Сервер 8 СП»:
  - Health-Check - Liveness;
  - Health-Check - Readyness.
- Выполнение нагрузочного и регрессионного тестирования.

4.1.10 Задача «Совместимость Типового ПО Витрина данных конфигурации Лайт с Astra Linux 1.7»

В целях поддержки отечественной операционной системы из реестра Российского ПО, устойчивого предоставления госуслуг и межведомственного обмена обеспечена совместимость Типового ПО Витрина данных на серверах с предустановленной операционной системой Astra Linux.

Для реализации вышеуказанной функциональности выполнены следующие работы:

1. Проведено тестирование процесса развертывания витрины данных на серверах с предустановленными операционными системами Astra Linux.
2. Проведено регрессионное тестирование функционирования витрины данных без подключения к ПОДД на серверах с предустановленными операционными системами Astra Linux.
3. Подготовлен отчет о регрессионном тестировании витрины данных на серверах с предустановленными операционными системами Astra Linux.
4. По итогам регрессионного тестирования выполнены необходимые доработки Типового ПО Витрина данных для обеспечения совместимости с операционными системами Astra Linux.
5. Подготовлены скрипты и проведено нагрузочное тестирование.
6. Подготовлен отчёт о результатах нагрузочного тестирования.

4.1.10.1 Описание тестирования процесса развертывания витрины данных на серверах с предустановленной операционной системой Astra Linux

Целью тестирования процесса развертывания витрины данных на серверах с предустановленными операционной системой Astra Linux:

- проверка работоспособности Типового ПО «Витрина данных» конфигурации Лайт на операционной системе «Astra Linux 1.7 (уровень защищенности «Воронеж»)»;

Для проведения тестирования на операционных системах «Astra Linux 1.7 (уровень защищенности «Воронеж»)» был подготовлен тестовый стенд (схема 1) с программными компонентами, приведенными в Таблица 4.1.

Таблица 4.1 Список компонентов и версий тестового стенда для проверки совместимости Типового ПО Витрина данных

№	Название компонента	Версия
1	ПОДД Агент	einfahrt:3.2.0-pre-2
2	Agent Driver	podd-jdbc-driver-3.2.0-SNAPSHOT-fat.jar
9	Модуль исполнения запросов	podd-adapter-query:5.2.0
10	Модуль подписок	podd-adapter-replicator:1.1.0
11	Модуль группировки чанков репликации	podd-adapter-group-repl:1.1.1

№	Название компонента	Версия
12	Модуль MPPR	podd-adapter-mppr:5.1.6
13	Модуль MPPW	podd-adapter-mppw:5.1.2
19	csv-uploader	csv-uploader:1.0.26
20	Prostore	query-execution:6.0.0 status-monitor:6.0.0
21	Prostore Driver	dtm-jdbc-6.0.0.jar
22	ADP	13.4
23	PG	0.5.0
24	FDW	0.10.2
25	Zookeeper	3.5.8
26	Kafka	2.6.0

В результате установки:

- Типовое ПО «Витрина данных» полностью работоспособно;
- все взаимосвязи между компонентами настроены автоматически;
- все необходимые компоненты запущены.

В процессе установки используется единый конфигурационный файл (настраиваемый перед установкой).

Процесс установки поддерживает детализированную диагностику возникающих ошибок.

Все возникающие в процессе установки ошибки логируются.

В процессе установке автоматически настраиваются взаимосвязи между компонентами программы.

Инструмент установки включает механизм проверки полноты выполненной установки компонентов и того, что все необходимые компоненты запустились.

4.1.10.2 Описание регрессионного тестирования функционирования витрины данных без подключения к ПОДД на серверах с предустановленной операционной системой Astra Linux

Целью проведения регрессионного тестирования является проверка работоспособности Типового ПО «Витрина данных» на операционной системе Astra Linux.

Для проведения регрессионного тестирования подготовлены основные сценарии тестирования модулей Типового ПО Витрина данных конфигурации Лайт.

Каждый сценарий тестирования содержит описание шага тестирования с указанием ожидаемого результата и статусом прохождения шага.

4.1.10.3 Описание подготовки отчетов о регрессионном тестировании витрины данных на серверах с предустановленной операционной системой Astra Linux

По итогам проведения регрессионного тестирования был подготовлен отчет о регрессионном тестировании Типового ПО «Витрина данных» на сервере с предустановленной операционной системой Astra Linux.

Отчет о регрессионном тестировании содержит разделы:

– цели тестирования;
– конфигурация стенда для тестирования;
– методика тестирования и ожидаемые контрольные результаты;
– результаты тестирования;
– замечания к разработке;
– план доработок.

В разделе «Цели тестирования» приведены основные сценарии тестирования.

В разделе «Конфигурация стенда для тестирования» указаны версии программных компонентов ПО «Витрина данных».

В разделе «Методика тестирования и ожидаемые контрольные результаты» прописаны шаги воспроизведения каждого сценария с указанием ожидаемого результата и статуса прохождения шага.

В разделе «Результаты тестирования» приведено описание выявленных дефектов и результатов тестирования.

В разделе «Замечания к разработке» приведена таблица с описанием найденных дефектов.

В приложении «План доработок» приведен план доработок найденных дефектов.

4.1.10.4 Описание выполнения необходимых доработок Типового ПО Витрина данных по итогам регрессионного тестирования для обеспечения совместимости с операционными системами Astra Linux

По результатам регрессионного тестирования Типового ПО «Витрина данных» конфигурации Лайт на операционной системе Astra Linux замечаний к разработке не выявлено.

4.1.10.5 Описание подготовки скриптов и проведения нагрузочного тестирования

На целевых ОС нагрузочное тестирование проводится только на модулях слоя адаптеров, отвечающих за исполнение запросов LLR:

– query-execution;
– status-monitor;
– podd-adapter-query.

Наполнение витрины данными:

– модель данных тестовая с таблицей vehicleregdata (данные регистрации ТС);
– 1,5 млн записей в таблице vehicleregdata;
– ключи генерируемых данных используют диапазон от 1 до 1,5 млн;
– объем одной записи 100 байт;
– избирательность прочих (не ключевых) полей не важна, могут быть одинаковые значения.

Вид нагрузки:

– запросы на получение одной записи по ключу с Limit 1 для работы в режиме LLR,

фильтруемое значение выбирается случайным образом из диапазона генерированных данных;

- запросы направляются в Кафка в топик query.rq, ответы фиксируются в топике query.rs;
- каждый поток создает запрос и ожидает ответа, фиксирует время от запроса до ответа, следующий запрос посылает после получения ответа на предыдущий запрос.

Нагрузка:

- (A) нарастает последовательно до тех пор, пока итоговая производительность не станет снижаться или пока не начнутся массовые ошибки;
- (B) Нарастает до расчётной стабильной и сохраняется на этом уровне.

Перед началом эксперимента необходим перезапуск ADB и стирание файлового кеша на ADB и рабочих узлах.

Накопление кеша в процессе одного эксперимента считаем естественным ускорителем.

4.1.10.6 Описание подготовки отчета о результатах нагрузочного тестирования

По итогам проведения тестирования был подготовлен отчет о нагрузочном тестировании Типового ПО «Витрина данных» на сервере с предустановленной операционной системой Astra Linux.

Отчет о нагрузочном тестировании содержит разделы:

- Общие сведения;
- Цели тестирования;
- Основные сценарии тестирования;
- Конфигурация стенда для тестирования;
- Результаты тестирования;
- Замечания к разработке.

В разделе «Общие сведения» приведены сведения об объекте тестирования.

В разделе «Цели тестирования» приведены основные цели проведения нагрузочных испытаний.

В разделе «Сценарии тестирования» прописаны шаги воспроизведения каждого сценария с указанием ожидаемого результата и статуса прохождения шага.

В разделе «Конфигурация стенда для тестирования» указаны версии программных компонентов ПО «Витрина данных».

В разделе «Результаты тестирования» приведено описание выявленных дефектов и результатов тестирования.

В разделе «Замечания к разработке» приведена таблица с описанием найденных дефектов.

4.1.11 Выполнение запросов с системными параметрами

4.1.11.1 Общее описание

Системный параметр не указывается в исходном sql запросе, передается как именованный в массиве `namedParams` используя служебное именование. Конструкция, соответствующая параметру подставляется во все таблицы запроса, кроме standalone таблиц.

Именованный параметр: указан в sql с `:name`, в зависимости от мнемоники, из массива `namedParams` в момент исполнения запроса в модуле query подставляется значение из элемента массива, соответствующего мнемонике.

Обработка системных и именованных параметров осуществляется Модулем исполнения запросов (podd-adapter-query).

Пример подстановки конструкции, соответствующей системному параметру в sql запрос представлена на рисунке ниже (см. Рисунок - 4.2).

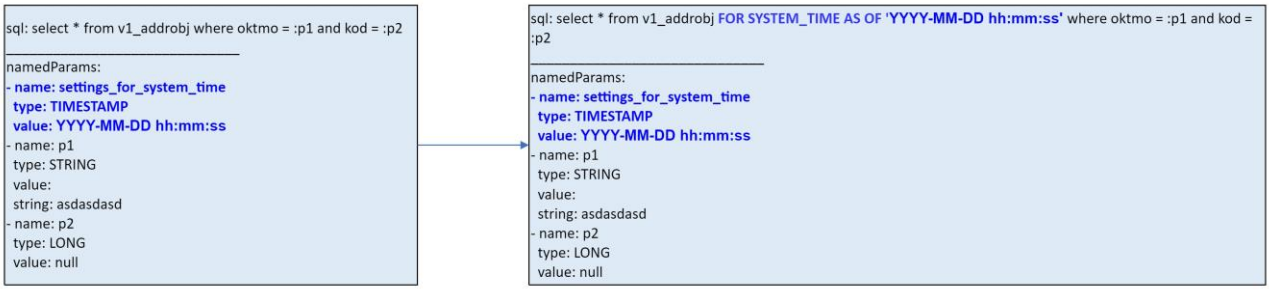

Рисунок - 4.2 Подстановка системного параметра в sql запрос

Массив именованных параметров (для передачи системных и именованных параметров) имеет следующую структуру:

```
- name: namedParams
  description: Именованные параметры запроса
  default: [ ]
  type:
    type: array
    items:
      type: record
      name: NamedParam
      description: Описание именованного параметра
      fields:
        - name: name
          description: Имя (мнемоника) параметра
          type: string
        - name: type
          type: string
          description: Тип параметра
          enum:
            - BIG_DECIMAL
            - BINARY
            - BOOLEAN
            - DATE
            - DOUBLE
            - FLOAT
            - INTEGER
            - LONG
            - SHORT
            - STRING
            - TIME
            - TIMESTAMP
```

```
    - name: value
      description: Значение параметра
      type:
        - string
        - 'null'
```

4.1.11.2 Обработка системных параметров

1. Если блок `namedParams` не пустой, после парсинга запроса **Calcite** модуль podd-adapter-query проверяет наличие мнемоник системных параметров:
   - `settings_for_system_time`;
   - `settings_for_system_time_started`;
   - `settings_for_system_time_finished`.
2. В случае, если системный параметр есть, проверяет:
   - что параметр с префиксом `settings_` один в списке;
   - сам запрос не содержит конструкцию `FOR SYSTEM_TIME` ни в одной из таблиц запроса;
   - в случае, если одно из перечисленных выше условий не выполняется, в топик `query.err` возвращается ошибка: DATAMART-17473 (Неверное sql-выражение: [Двойное указание темпоральности недопустимо]).
3. Модуль проверяет **value** системного параметра по маске:
   - для параметра `settings_for_system_time` **value** запроса должно быть таймштампом в формате YYYY-MM-DD hh:mm:ss
   - для параметров `settings_for_system_time_started` и `settings_for_system_time_finished`:
   - два целочисленных числа разделенных запятой („int1,int2“)
   - в случае, если **value** параметра не соответствует маске, в топик `query.err` возвращается ошибка: DATAMART-17473 (Неверное sql-выражение: [Недопустимое значение параметра [имя параметра] - [значение параметра]])
   - В случае, если значение системного параметра null , возвращается ошибка: DATAMART-17473 (Недопустимое значение параметра [имя параметра] = null)
4. Для параметров `settings_for_system_time_started` и `settings_for_system_time_finished` проверяет, что первое значение int или timestamp меньше последнего:
   - в случае, если первое значение в строке больше второго, в топик `query.err` возвращается ошибка: DATAMART-17473 (Неверное sql-выражение: [Недопустимый (отрицательный) диапазон значений параметра [имя параметра]

- [значение параметра]])

5. После выполнения проверок 1-4 модуль podd-adapter-query подставляет системный параметр во все таблицы запроса, после имени таблицы в блоке from. (если в запросе есть блок where, то перед этим блоком). **Исключение**: временные таблицы загрузки табличных параметров, так как временные таблицы являются standalone (определяем по префиксу `readable` в названии таблицы):
   - если в запросе используется системный параметр `settings_for_system_time` : `FOR SYSTEM_TIME AS OF 'YYYY-MM-DD hh:mm:ss[.microseconds]`,
   - если в запросе используется системный параметр `«settings_for_system_time_started»`:
   - `FOR SYSTEM_TIME STARTED IN (delta_num1, delta_num2)`, если строка состоит из двух целочисленных значений, разделенных запятой
   - если в запросе используется системный параметр `settings_for_system_time_finished`:
   - `FOR SYSTEM_TIME FINISHED IN (delta_num1, delta_num2)`, если строка состоит из двух целочисленных значений, разделенных запятой
6. Подставляет в запрос остальные именованные параметры, в случае, если они присутствуют в запросе.
7. В случае, если в запросе есть **limit** и он меньше порогового значения или есть **FORCE_LLR**, то выполняются следующие действия:
   - в Prostore подается команду на получение данных через Rest-интерфейс Prostore;
   - данные из Prostore преобразуются в формат ПОДД и конвертирует timezone у полей, выгружаемых из Витрины к формату UTC в топик `query.rs`.
   - в случае, если вернулась ошибка , то возвращается ответ в `query.err` с текстом ошибки
8. В случае, если в запросе отсутствует **limit** или настройка **FORCE_LLR** выключена, то передает сообщение в топик `mppr.delegate.rq` и завершает выполнение запроса.

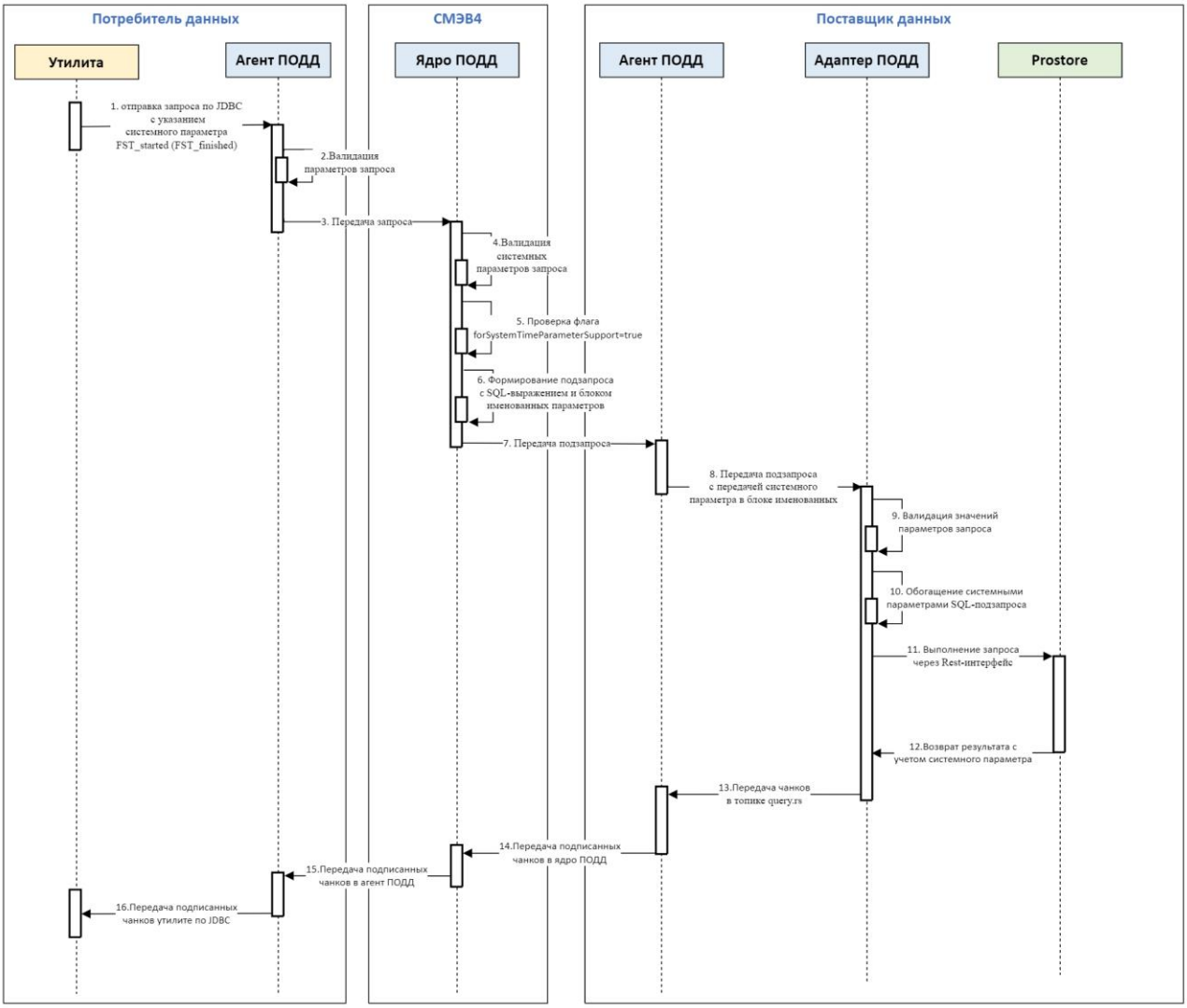

Рисунок - 4.3 Схема взаимодействия компонентов

4.1.11.3 Примеры преобразования запросов

Примеры запроса query.rq с системными параметрами `settings_for_system_time_started` и `settings_for_system_time_finished` с номерами дельт

```
sql: select * FROM demo_view.ticket join demo_view.passenger on demo_view.passenger.id
=demo_view.ticket.passengerid
```

```
"namedParams" : [ {
    "name" : "settings_for_system_time_started",
    "type" : "STRING",
    "value" : {
    "string" : "122, 123"
    }
} ]
```

преобразованный запрос, после подстановки параметра FOR SYSTEM_TIME STARTED IN:

```
select * FROM demo_view.ticket FOR SYSTEM_TIME STARTED IN (122,124) join
demo_view.passenger FOR SYSTEM_TIME STARTED IN (122,124) on demo_view.passenger.id
=demo_view.ticket.passengerid
```

```
___
sql: select * FROM demo_view.ticket join demo_view.passenger on demo_view.passenger.id
=demo_view.ticket.passengerid
_____________________________________________________________________________________
__
"namedParams" : [ {
    "name" : "settings_for_system_time_started",
    "type" : "STRING",
    "value" : {
    "string" : "122,123"
    }
} ]
_____________________________________________________________________________________
____
преобразованный запрос, после подстановки параметра FOR SYSTEM_TIME FINISHED IN:
select * FROM demo_view.ticket FOR SYSTEM_TIME FINISHED IN (122,124) join
demo_view.passenger FOR SYSTEM_TIME FINISHED IN (122,124) on demo_view.passenger.id
=demo_view.ticket.passengerid
```

**Пример запроса query.rq с системным параметром `settings_for_system_time`:

```
sql: select * FROM demo_view.ticket join demo_view.passenger on demo_view.passenger.id
=demo_view.ticket.passengerid
_____________________________________________________________________________________
____
          namedParams:
          - name: settings_for_system_time
              type: TIMESTAMP
              value:
              string: 2022-11-14 16:00:00
_____________________________________________________________________________________
___
преобразованный запрос, после подстановки параметра FOR SYSTEM TIME: select * FROM
demo_view.ticket FOR SYSTEM_TIME AS OF '2022-11-14 16:00:00' join demo_view.passenger
FOR SYSTEM_TIME AS OF :p2 on demo_view.passenger.id =demo_view.ticket.passengerid
```

query.tp, системные параметры (к табличному параметру не подставляем, так как таблица standalone):

```
sql:  select count(*) from @tbl el LEFT JOIN demo_view.passenger on
@tbl.firstname.firstname = demo_view.ticket.firstname
_____________________________________________________________________________________
____
          namedParams:
          - name: settings_for_system_time
              type: TIMESTAMP
              value:
              string: 2022-11-14 16:00:00
_____________________________________________________________________________________
___
преобразованный запрос, после подстановки  параметра  FOR SYSTEM TIME:  select count(*)
from @tbl el LEFT JOIN demo_view.passenger FOR SYSTEM_TIME AS OF '2022-11-14 16:00:00'
on @tbl.firstname.firstname = demo_view.ticket.firstname
```

Примеры запросов с системными и именованными параметрами

Пример недопустимой конструкции запроса с системными и именованными параметрами:

```
sql: select * FROM demo_view.ticket join demo_view.passenger FOR SYSTEM_TIME AS OF :p2
on demo_view.passenger.id =demo_view.ticket.passengerid
_____________________________________________________________________________________
_________
           namedParams:
           - name: settings_for_system_time
               type: TIMESTAMP
               value:
               string: 2022-11-14 16:00:00
           - name: p1
               type: TIMESTAMP
               value:
               string: 2022-11-14 15:00:00
_____________________________________________________________________________________
___
преобразованный запрос, после подстановки  параметра  FOR SYSTEM TIME: select * FROM
demo_view.ticket FOR SYSTEM_TIME AS OF 2022-11-14 16:00:00 join demo_view.passenger FOR
SYSTEM_TIME AS OF :p2 on demo_view.passenger.id =demo_view.ticket.passengerid
```

Пример запроса с системными и именованными параметрами в блоке where:

```
sql: select * FROM demo_view.ticket JOIN demo_view.passenger on demo_view.passenger.id
=demo_view.ticket.passengerid WHERE passengerid=:p1 and price=:p2
_____________________________________________________________________________________
_____________________________________________
           namedParams:
           - name: settings_for_system_time
               type: TIMESTAMP
               value: 2022-11-14 16:00:00
           - name: p1
               type: STRING
               value:
               string: 0fe26963-6cdf-4db6-b632-03825a408d35
           - name: p2
               type: DOUBLE
               value: 719.4501969260996


_____________________________________________________________________________________
___
преобразованный запрос, после подстановки  параметра  FOR SYSTEM TIME: select * FROM
demo_view.ticket FOR SYSTEM_TIME AS OF '2022-11-14 16:00:00' JOIN demo_view.passenger
FOR SYSTEM_TIME AS OF '2022-11-14 16:00:00' on demo_view.passenger.id
=demo_view.ticket.passengerid WHERE passengerid=:p1 and price=:p2
```

5 ВЫЗОВ И ЗАГРУЗКА

Программа не имеет графического интерфейса и запускается с жесткого диска сервера как `systemd` сервис.

При необходимости любой из сервисов/модулей можно остановить и запустить заново.

Для ручной остановки и запуска необходимо подключиться по *ssh* на сервер и с правами `sudo` использовать штатный функционал команд `systemctl`, которая является инструментом центрального управления для контроля системы инициализации.

Например:

```
sudo systemctl stop query-execution
sudo systemctl start query-execution
```

Описание первоначального запуска системы описано в документе «Руководство по установке».

6 ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

6.1 Входные данные

Входные данные программы приведены в таблице ниже (см. Раздел 6.2).

6.2 Выходные данные

Выходные данные ПО «Витрина данных НСУД» приведены в таблице ниже (см. Таблица 6.1).

Таблица 6.1 Выходные данные

№	Описание
1	Данные Витрины, предоставляемые системам-потребителям в ответ на запрос
2	Реплицированные из других витрин данные, предоставляемые внутренним ИС ведомства
3	CSV и XML-файлы выгруженные через модуль CSV-Uploader
4	Сообщения, передаваемые администратору посредством интерфейса командной

Взаимодействие *ПОДД-адаптера - Модуль исполнения запросов* с Агентом ПОДД производится через список топиков *Брокера сообщений Kafka*. Назначение и формат сообщений, последовательность обмена ими соответствуют документу «Методические рекомендации по работе с подсистемой обеспечения доступа к данным федеральной государственной информационной системы «Единая система межведомственного электронного взаимодействия»» (версия 1.0.0).

Описание формата взаимодействия (название топика, формат сообщений, схема взаимодействия) см. «Руководство администратора».

7 ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

ADCM

Arenadata Cluster Manager (ADCM) — Универсальный оркестратор гибридного ландшафта. Он позволяет быстро устанавливать, настраивать все data-сервисы компании и управлять ими. Наиболее ярко преимущества ADCM раскрываются при работе с гетерогенной инфраструктурой, при которой появляется возможность размещать data-сервисы на различных типах инфраструктур: в облаке, on-premise или в качестве PaaS-сервисов.

ADS

Arenadata Streaming (ADS) - Масштабируемая отказоустойчивая система для потоковой обработки данных в режиме реального времени на базе Apache Kafka и Apache Nifi.

Airflow

открытое программное обеспечение для создания, выполнения, мониторинга и оркестровки потоков операций по обработке данных.

Apache

Организация-фонд, способствующая развитию проектов программного обеспечения Apache.

Apache Airflow

Платформа для программного создания, планирования и мониторинга рабочих процессов.

Apache Hadoop

Свободно распространяемый набор утилит, библиотек и фреймворк для разработки и выполнения распределённых программ, работающих на кластерах из сотен и тысяч узлов.

Apache Spark

Фреймворк с открытым исходным кодом для реализации распределённой обработки неструктурированных и слабоструктурированных данных.

Apache Kafka

Распределённый программный брокер сообщений, проект с открытым исходным кодом, разрабатываемый в рамках фонда Apache.

Apache Avro

Система сериализации данных, разработанная в рамках проекта Hadoop.

API

Application programming interface (англ.) – описание сервисов взаимодействия компьютерной программы с другими программами.

Avro

(Object Container File) Линейно-ориентированный формат хранения файлов Big Data

BLOB-адаптер

Информационно-технологический компонент Витрины, обеспечивающий чтение бинарных файлов из Хранилище BLOB-объектов ведомства.

ClickHouse

Колоночная аналитическая СУБД с открытым кодом, позволяющая выполнять аналитические запросы в режиме реального времени на структурированных больших данных, разрабатываемая компанией Яндекс.

Docker

Программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации, контейнеризатор приложений.

Docker Compose

Платформа контейнеризации, предназначена для конфигурирования многоконтейнерных приложений. В Docker Compose можно управлять несколькими контейнерами Docker.

DATA-uploader

Модуль исполнения асинхронных заданий.

Counter-Provider

Сервис генерации уникального номера.

CSV

Comma-Separated Values (англ.) – значения, разделённые запятыми) — текстовый формат, предназначенный для представления табличных данных.

CSV-extractor

Специализированное программное обеспечение, которое извлекает данные из csv-файлов в собственную БД-хранилища сервиса (Tarantool)

CSV-Uploader

Программный модуль Витрины данных, который предназначен для загрузки csv-файлов в Витрину данных.

Grafana

Веб-приложение для аналитики и интерактивной визуализации показателей мониторинга с открытым исходным кодом.

Greenplum

Массово-параллельная СУБД для хранилищ данных на основе PostgreSQL.

DAG

Файл, содержащий блок данных.

DBeaver

Клиентское приложение для управления базами данных (БД), которое использует программный интерфейс JDBC для взаимодействия с реляционными БД через драйвер JDBC-драйвер.

DDL

Data definition language (англ.) – семейство компьютерных языков, используемых в компьютерных программах для описания структуры баз данных.

DNS

Domain Name System «система доменных имён» — компьютерная распределённая система для получения информации о доменах. Чаще всего используется для получения IP-адреса по имени хоста (компьютера или устройства), получения информации о маршрутизации почты и/или обслуживающих узлах для протоколов в домене.

Endpoint

Шлюз (в переводе с англ. — конечная точка), который соединяет серверные процессы приложения с внешним интерфейсом. Простыми словами, это адрес, на который отправляются сообщения (работает с API)

ETL

Extract, transform, load (англ.) – решение, используемое при выгрузке данных из различных источников ведомств и дальнейшего хранения их в Витрине ProStore для чтения, использования и взаимодействия с другими ведомствами.

FileZilla

FTP-клиент

Greennplum

Система управления данными из мира big data.

HikariCP

Hikari Connection Pool.

HTTP

HyperText Transfer Protocol (англ.) – протокол прикладного уровня передачи данных, в настоящий момент используется для передачи произвольных данных.

IAM

Сервисы управления идентификацией и контролем доступа (Identity&AccessManagement)

JDBC

Java DataBase connectivity (англ.) – платформенно-независимый промышленный стандарт взаимодействия Java-приложений с различными СУБД.

JDBC-драйвер

Специализированное программное обеспечение, которое размещается на стороне системы, использующей ADTM (клиента ADTM). Драйвер предоставляет JDBC-интерфейс подключения из этой системы к ADTM и взаимодействует с сервисом исполнения запросов по REST API, предоставляемым сервисом исполнения запросов.

JDBC-extractor

Специализированное программное обеспечение, которое извлекает данные из jdbc-источника (ведомства) в собственную БД-хранилища сервиса (Tarantool).

JDBC-CSV-transformer

Специализированное программное обеспечение, которое предназначено для подключения к БД по JDBC, с последующим сохранением в csv-файлы.

Kafka

Распределённый программный брокер сообщений, проект с открытым исходным кодом, разрабатываемый в рамках фонда Apache.

Kafka-loader

Специализированное программное обеспечение, которое загружает данные, извлеченные и приведенные в соответствие логической структуре данных Витрины, собственно в Витрину.

Loki

Приложение для агрегирования log-файлов, используется совместно с Prometheus.

MD5

128-битный алгоритм хеширования. Предназначен для создания «отпечатков» или дайджестов сообщения произвольной длины и последующей проверки их подлинности.

MPP

Массово-параллельная архитектура (англ. massive parallel processing, MPP, также «массивно-параллельная архитектура»).

NTP

Network Time Protocol — сетевой протокол для синхронизации внутренних часов компьютера с использованием сетей с переменной латентностью.

OpenAPI

The OpenAPI Specification (англ.) – формализованная спецификация и экосистема множества инструментов, предоставляющая интерфейс между front-end системами, кодом библиотек низкого уровня и коммерческими решениями в виде API.

ProStore

Интеграционная система, обеспечивающая единый интерфейс к хранилищу разнородных данных. Определяет структуры данных, запись и чтение данных Витрины. Позволяет работать со входящими в состав хранилища СУБД одинаковым образом, используя единый синтаксис запросов SQL и единую логическую схему данных.

Prostore

Ядро интеграционной системы ProStore, состоящее из сервиса исполнения запросов и сервиса мониторинга.

Prometheus

Программное приложение, используемое для мониторинга событий и оповещения, которое записывает метрики в реальном времени в базу данных временных рядов, построенную с использованием модели HTTP-запроса, с гибкими запросами и оповещениями в режиме реального времени.

Proxy API

Проксирование запросов через Datamart Studio к инсталляциям приложений Витрин данных

PSQL

Терминальный клиент для работы с PostgreSQL

PuTTY

Свободно распространяемый клиент для различных протоколов удалённого доступа, включая SSH, Telnet, rlogin.

PXF

Фреймворк, позволяющий ADB (Greenplum) параллельно обмениваться данными со сторонними системами.

REST

Representational state transfer (англ.) – архитектурный стиль взаимодействия компонентов распределенного приложения в сети.

REST-адаптер

Сервис, реализующий публикацию конечных точек API для обработки запросов с использованием спецификации OpenAPI версии 3. Используется для сохранения обратной совместимости получения данных из ведомства по REST.

REST API

Набор правил, по которым различные программы могут взаимодействовать между собой и обмениваться данными с помощью протокола HTTP

REST-Uploader

Модуль асинхронной загрузки данных из сторонних источников.

SOAP

(от англ. Simple Object Access Protocol — простой протокол доступа к объектам) — протокол обмена структурированными сообщениями в распределённой вычислительной среде.

SQL

Structured query language (англ.) – язык структурированных запросов. Декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

SQL-запрос

Запрос к Базе данных.

SSH

Secure Shell (англ.) – «безопасная оболочка». Сетевой протокол прикладного уровня, позволяющий производить удалённое управление операционной системой и туннелирование TCP-соединений.

Tarantool

Платформа in-memory вычислений с гибкой схемой данных для создания высоконагруженных приложений. Включает в себя базу данных и сервер приложений на Lua.

UDP

Протокол передачи данных. С UDP компьютерные приложения могут посылать сообщения другим хостам по IP-сети без необходимости предварительного сообщения для установки специальных каналов передачи или путей данных.

URI

Унифицированный идентификатор ресурса. URI — последовательность символов, идентифицирующая абстрактный или физический ресурс.

UUID

Стандарт идентификации, используемый в создании программного обеспечения, стандартизированный Open Software Foundation как часть DCE — среды распределённых вычислений. Основное назначение UUID — это позволить распределённым системам уникально идентифицировать информацию без центра координации.

Vert.x

Библиотека для разработки асинхронных приложений, основанная на событиях.

VipNet

программное обеспечение (далее - ПО) для защиты сетевого трафика на рабочих местах пользователей.

XML

eXtensibe Markup Language (англ.) – универсальный текстовый формат для хранения и передачи структурированных данных.

XML-extractor

Специализированное программное обеспечение, для копирования данных из xml-файлов в собственную БД-хранилища сервиса (Tarantool).

ZooKeeper

Сервер с открытым исходным кодом для высоконадежной распределенной координации облачных приложений.

Агент ПОДД

Типовое программное обеспечение, устанавливаемое на стороне УВ и обеспечивающее сопряжение Витрин, хранилищ реплик, ИС Участника взаимодействия с ПОДД. В частности, чтение данных из Витрины, запись данных в реплику, обработка промежуточных/временных массивов данных, порождаемых в процессе выполнения распределённых запросов.

База данных

Совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных.

Брокер сообщений

Архитектурный паттерн в распределённых системах; приложение, которое преобразует сообщение по одному протоколу от приложения-источника в сообщение протокола приложения-приёмника, тем самым выступая между ними посредником.

Витрина данных

Комплекс программных и технических средств в составе информационно-телекоммуникационной инфраструктуры участника НСУД, предназначенный для формирования и (или) получения данных с использованием среды взаимодействия НСУД.

ВС

Вид сведений

ГОСТ

Нормативно-правовой документ, в соответствии требованиями которого производится стандартизация производственных процессов

Дельта

Логически целостная совокупность изменений информации об объектах. Каждой дельте поставлено в соответствие целое число из монотонно возрастающей последовательности целых чисел начиная с 0, отражающее ее место в общей последовательности дельт и дата-время ее исполнения.

ЕИП

Единая информационная платформа

ИС

Информационная система.

КриптоПро

Разработанная одноименной компанией линейка криптографических утилит (вспомогательных программ) — так называемых криптопровайдеров. Они используются в других программах для генерации электронной подписи (ЭП), работы с сертификатами, организации структуры PKI и т.д.

Логическая модель данных

Схема базы данных, выраженная в понятиях бизнес-требований.

набор данных

Совокупность данных (датасетов), систематизированных в определённом формате, представляющих собой базовый элемент для работы с данными во многих отраслях

НСУД

Национальная система управления данными.

ОГРН

Основной государственный регистрационный номер, который налоговая служба присваивает юридическим лицам сразу же после регистрации

ПО

Программное обеспечение.

ПОДД

Подсистемы обеспечения доступа к данным

ПОДД-адаптер

Программно-технический продукт, обеспечивающий взаимодействие витрины и ПОДД СМЭВ.

ПОДД-адаптер - Модуль исполнения запросов

Логический модуль ПОДД-адаптера, предназначен для исполнения запросов ПОДД СМЭВ (через протокол коммуникации Агент ПОДД).

ПОДД-адаптер - Модуль MPPR

Логический модуль ПОДД-адаптера, предназначен для чтения данных в многопоточном режиме (massively parallel processing, MPP).

ПОДД-адаптер - Модуль MPPW

Логический модуль ПОДД-адаптера выполняет загрузку данных в многопоточном режиме.

Поставщик данных

Организация, осуществляющая передачу государственных данных в НСУД.

Потребитель данных

Организация, осуществляющая использование государственных данных, содержащихся в НСУД.

Реплика

СУБД, хранящая реплицируемые наборы данных, полученные от Поставщика данных.

Сервис Формирования документов

Модуль витрины, предназначенный для работы с формируемыми документами.

СМЭВ3

Система межведомственного электронного взаимодействия.

СМЭВ3-адаптер

Информационно-технологический компонент СМЭВ, устанавливаемый на стороне Участника взаимодействия. СМЭВ3-адаптер обеспечивает информационное взаимодействие через единый электронный сервис единой системы межведомственного электронного взаимодействия (СМЭВ).

Сообщение

Сведения в виде законченного блока данных, передаваемые при функционировании информационной системы.

СУБД

Система управления базами данных

Токен

Ключ безопасности (Цифровой сертификат)

Участник НСУД

Федеральный орган исполнительной власти, иной орган государственной власти, государственный внебюджетный фонд, орган местного самоуправления, иное юридическое лицо, являющиеся стороной действующего соглашения о присоединении к Национальной системе управления данными.

ФЛК

Форматно-логический контроль загружаемых в Витрину данных.

Хранилище BLOB-объектов

Место для хранения BLOB-объектов (бинарных данных). Располагается на стороне ведомства и не является частью Витрины данных. Взаимодействие с Хранилищем BLOB-объектов осуществляется через BLOB-адаптер.

Хранилище S3 (объектное хранилище S3)

Хранилище бинарных объектов, позволяющее хранить файлы любого типа и объема. Доступ к хранилищу предоставляется через API.