

Перечень принятых сокращений

| Сокращение | Расшифровка |
|------------|---|
| APM | Веб-приложение "Автоматизированное рабочее место" |
| БД | База данных |
| Окно | Временной отрезок, полуинтервал - $[t_{start}, t_{end})$, где t_{start} - временная метка начала окна и t_{end} - временная метка окончания окна. |
| ПК | Пользовательский компьютер |
| ПО | Программное обеспечение |
| ХД | Хранилище данных - база данных временных рядов |
| API | Прикладной программный интерфейс, определяющий правила обмена данными между приложениями |
| GPU | Графический процессор (Graphics Processing Unit) |
| GUI | Веб-приложение "Графический интерфейс администратора" |
| LLM | Большая языковая модель (Large Language Model) — тип искусственного интеллекта (ИИ), который обучен на колоссальных объемах текстовых данных и который умеет понимать, обрабатывать и генерировать человекоподобный текст |
| SAP | Комплексное программное решение для автоматизации управления предприятием |

1 Общие сведения

1.1 Назначение документа

Настоящий документ содержит описание функционального назначения программного продукта "Системы сбора, обработки и хранения данных", а также список модулей, из которых состоит система.

1.2 Назначение системы

Системы сбора, обработки и хранения данных предназначена для сбора данных о технологическом оборудовании и процессах на промышленных предприятиях, их обработки, как потоковой, так и постобработки, и хранения в едином объеме, называемом ХД.

Платформа состоит из Коннекторов для сбора данных, Серверного ПО, в состав которого входит брокер сообщений (канал передачи данных), базы данных конфигураций сервисов, хранилище графических объектов и сервисы платформы, ХД, пользовательских веб-приложений (GUI и APM).

Если не указано иное, подразумевается, что в качестве ХД используется база данных ClickHouse, в качестве хранилища конфигурация - PostgreSQL, а в качестве хранилища графических объектов - Minio.

1.3 Функции системы

Система выполняет перечисленные ниже функции:

- Конфигурирование Источников:
 - Конфигурирование коннекторов сбора данных и опрашиваемых тегов;
 - Конфигурирование источников и тегов ручного ввода данных;
- Сбор данных от источников:
 - Опрос источников данных;
 - Буферизация данных;
 - Сериализация данных;
 - Формирование пакетов данных и их передача;
- Передача данных между коннектором и сервисами системы:
 - Получение данных для платформы;
 - Десериализация данных;

- Отправка конфигурации ПО коннектора;
- Запись данных в ХД;
- Импорт данных посредством API;
- Обработка данных:
 - Конфигурирование обработчиков;
 - Поточковая обработка получаемых значений (масштабирование, выделение значащего бита, скрипт);
 - Передача результата в брокер сообщений для последующей записи в ХД;
- Формирование тревог:
 - Конфигурирование обработчиков тревог;
 - Проверка условий формирования тревоги (по уровню, по скорости изменения);
 - Введение тревоги;
 - Передача информации о тревоге в брокер сообщений для последующей записи в ХД;
- Постобработка данных:
 - Формирование расчетных значений на основе данных одного или нескольких источников за окно с помощью скриптов;
 - Конфигурирование технологических окон;
 - Запись расчетных значений в ХД;
- Обращение к данным из ХД:
 - Обращение к текущим значениям;
 - Мониторинг текущих значений;
 - Обращение к историческим значениям за указанный период;
 - Мониторинг исторических значений с помощью графиков;
- Конфигурирование объектной модели:
 - Создание цифровой модели предприятия на базе иерархичной структуры объектов;
 - Создание переменных объектов и назначение им связей с тегами - источниками;
 - Назначение метаинформации объектам и переменным;
 - Создание справочников значений для метаинформации;
- Настройка контентных прав:
 - Настройка контентных прав на Объекты для пользователей системы;
 - Настройка контентных прав на Мнемосхемы для пользователей системы;
- Формирование отчетности:
 - Импорт ресурсов для отчета;
 - Создание скриптов отчетов;
 - Выполнение скриптов отчетов;
 - Экспорт отчетов;
- Конфигурирование уведомлений:
 - Создание уведомлений по расписанию;
 - Создание уведомлений по результатам проверки события;
 - Создание уведомлений по результатам выполнения скрипта;
- Рассылка уведомлений на электронную почту;
- Конфигурирование групп рассылок;
- Мониторинг отправки уведомлений;
- Предиктивная аналитика:
 - Анализа данных из ХД с помощью Jupiter Notebook;
 - Формирование гипотеза на основе анализа данных;
 - Написания скриптов - моделей;
 - Конфигурирование обработчиков моделей;
 - Мониторинг срабатываний обработчиков моделей;
- Интеграция с SAP:
 - Отправка в SAP по расписанию текущих значений выбранных параметров;
 - Получения из SAP при создании/ изменении информации по ТОиР (сообщения, заказы);
 - Получения из SAP при изменении значений наработки и индекса технического состояния (ИТС) для выбранных единиц оборудования и технических мест;
- Визуализация в APM:
 - Созданием мнемосхем;
 - Проигрывание мнемосхем;
 - Мониторинг тревог;
 - Квтирование тревог;
 - Импорт графических элементов, мнемосхем, подложек;
 - Вызов отчетов, ранее сконфигурированных в GUI;
 - Доступ к объектной модели, ранее сконфигурированной в GUI;
 - Добавление значений на мнемосхемы (Значение, Таблица, Датчик, График);
 - Добавление на мнемосхемы графических объектов, примитивов, текстовых подписей;
 - Настройка анимации на мнемосхемах (изменение цвета, прозрачности или мигание) на основе значений из ХД;

- Управление параметрами элементов на мнемосхеме;
- Экспорт конфигурации и печать мнемосхемы;
- Построение базы знаний по объекту:
 - Импорт производственной документации и инструкций в базу знаний объекта;
 - Распознавание текста из загружаемых PDF-документов для пополнения базы знаний;
 - Формирование пар вопрос-ответ для каждого документа;
 - Выявление конфликтов в базе знаний;
 - Дополнение базы знаний с помощью ручного ввода;
- Поиск аномалий и корреляций
 - Автоматическое детектирование аномалий по переменным, для которых выбрана соответствующая опция;
 - Поиск корреляций между аномалиями переменных внутри объекта;
 - Поиск корреляций между аномалиями переменных разных объектов (ИИ-агенты);
 - Генерация текстовых пояснение корреляции LLM моделью на основе базы знаний объекта;
- ИИ-помощник
 - Формирование ответов на вопросы пользователей по использованию веб-интерфейсов на основе руководств пользователя и системного администратора к соответствующим веб-приложениям;
 - Коммуникация с LLM моделью в формате чата;
 - Обсуждение с LLM моделью баз знаний объекта.

1.4 Технические требования к инфраструктуре для разворачивания системы

1.4.1 Рекомендованная конфигурация с учетом резервирования компонент системы

Доступ к веб-приложениям может осуществляться с любого ПК, при наличии сетевой связности с контуром, где развернута система.

Рекомендуемые параметры: 2 vCPU, 16 Gb vRAM и 500 Gb SSD. На ПК должен быть установлен браузер из предложенных, версии от указанной и более поздней:

- **Google Chrome.** Для операционных систем Windows, macOS, Linux - 119.0.6045.199/200 (28 ноября 2023), для Android - 119.0.6045.193 (28 ноября 2023).
- **Mozilla Firefox.** Для операционных систем Linux, Windows, macOS, Android - 120.0.1 (30 ноября 2023).
- **Windows Edge.** Для операционных систем Windows, MacOS, Linux - 119.0.2151.58 (9 ноября 2023), для Android - 118.0.2088.66 (6 ноября 2023).

В Таблице 1 представлен рекомендуемый перечень компонент системы и требования к техническим характеристикам для разворачивания системы с резервированием компонент.

Таблица 1. Рекомендованная конфигурация системы

| Назначение | ОС | vCPU | RAM | SSD, Gb |
|--|-----------|------|-----|---------|
| Kafka node 1 (брокер сообщений, узел 1) | Debian 12 | 8 | 16 | 1000 |
| Kafka node 2 (брокер сообщений, узел 2) | Debian 12 | 8 | 16 | 1000 |
| Kafka node3 (брокер сообщений, узел 3) | Debian 12 | 8 | 16 | 1000 |
| PIMS services node 1 (сервисы, узел 1) | Debian 12 | 8 | 16 | 100 |
| PIMS services node 2 (сервисы, узел 2) | Debian 12 | 8 | 16 | 100 |
| PostgreSQL node 1 (БД конфигураций, узел 1) | Debian 12 | 8 | 16 | 200 |
| PostgreSQL node 2 (БД конфигураций, узел 2) | Debian 12 | 8 | 16 | 200 |
| PostgreSQL node 3 (БД конфигураций, узел 3) | Debian 12 | 8 | 16 | 200 |
| Balancer node 1 (Балансировщик нагрузки ХД) | Debian 12 | 2 | 8 | 40 |
| Balancer node 2 (Балансировщик нагрузки БД конфигураций) | Debian 12 | 2 | 8 | 40 |
| Clickhouse Keeper node 1 (Координатор ХД, узел 1) | Debian 12 | 2 | 8 | 40 |
| Clickhouse Keeper node 2 (Координатор ХД, узел 2) | Debian 12 | 2 | 8 | 40 |
| Clickhouse Keeper node 3 (Координатор ХД, узел 3) | Debian 12 | 2 | 8 | 40 |
| Мониторинг системы | Debian 12 | 2 | 8 | 500 |
| ClickHouse node 1 (ХД, узел 1) | Debian 12 | 16 | 64 | 1000 |
| ClickHouse node 2 (ХД, узел 2) | Debian 12 | 16 | 64 | 1000 |

| | | | | |
|--|-----------|----|-----|------|
| KeyCloak (сервис аутентификации и авторизации) | Debian 12 | 2 | 8 | 40 |
| Minio node 1 (БД графических объектов, узел 1) | Debian 12 | 8 | 16 | 1000 |
| Minio node 2 (БД графических объектов, узел 2) | Debian 12 | 8 | 16 | 1000 |
| Коннектор основной | Debian 12 | 2 | 16 | 500 |
| Коннектор резервный | Debian 12 | 2 | 16 | 500 |
| Opensearch node 1 (Сервис логирования, узел 1) | Debian 12 | 8 | 16 | 200 |
| Opensearch node 2 (Сервис логирования, узел 2) | Debian 12 | 8 | 16 | 200 |
| Сервер с GPU (24 GB vRAM) | Debian 12 | 24 | 128 | 1000 |

1.4.2. Минимальная конфигурация без резервирования

Доступ к веб-приложениям может осуществляться с любого ПК, при наличии сетевой связности с контуром, где развернута система. Рекомендуемые параметры: 2 vCPU, 8 Gb vRAM и 256 Gb SSD. На ПК должен быть установлен браузер из предложенных, версии от указанной и более поздней:

- **Google Chrome.** Для операционных систем Windows, macOS, Linux - 119.0.6045.199/200 (28 ноября 2023), для Android - 119.0.6045.193 (28 ноября 2023).
- **Mozilla Firefox.** Для операционных систем Linux, Windows, macOS, Android - 120.0.1 (30 ноября 2023).
- **Windows Edge.** Для операционных систем Windows, MacOS, Linux - 119.0.2151.58 (9 ноября 2023), для Android - 118.0.2088.66 (6 ноября 2023).

В Таблице 2 представлен минимальный перечень компонент системы и требования к техническим характеристикам для разворачивания системы без резервирования.

Таблица 2. Минимальная конфигурация системы

| Назначение | ОС | vCPU | vRAM | SSD, Gb |
|--|-----------|------|------|---------|
| Kafka (брокер сообщений) | Debian 12 | 8 | 16 | 1000 |
| PIMS services (сервисы) | Debian 12 | 8 | 16 | 100 |
| PostgreSQL (БД конфигураций) | Debian 12 | 8 | 16 | 200 |
| Balancer (Балансировщик) | Debian 12 | 2 | 8 | 40 |
| ClickHouse (ХД) | Debian 12 | 16 | 64 | 1000 |
| KeyCloak (сервис аутентификации и авторизации) | Debian 12 | 2 | 8 | 40 |
| Minio (БД графических объектов) | Debian 12 | 8 | 16 | 1000 |
| Коннектор основной | Debian 12 | 2 | 16 | 500 |
| Сервер с GPU (24 GB vRAM) | Debian 12 | 24 | 128 | 200 |

1.5 Требования к персоналу

Персонал, допущенный к работе с системой в качестве **пользователя** (диспетчеры), должен обладать следующими навыками и опытом:

- Уверенный пользователь ПК;
- Обучен работе с веб-приложением АРМ на базе руководства пользователя АРМ.

Персонал, допущенный к работе с системой в качестве **системного администратора**, должен обладать следующими навыками и опытом:

- Уверенный пользователь ПК;
- Обучен работе с веб-приложением АРМ на базе руководства пользователя АРМ;
- Обучен работе с веб-приложением GUI на базе руководства пользователя GUI;
- Навыки программирования на Python;
- Навыки работы с операционными системами семейства Linux, docker-контейнерами;
- Навыки использования командной строки;
- Навыки работы с БД, умение писать простые SQL-запросы.

Хранилище данных представляет собой базу данных временных рядов, которая хранит значения тегов, принимаемых системой, а также все данные, генерируемые системой. В качестве базы данных используется ClickHouse.

В отдельных таблицах хранятся: значения временных рядов для тегов, история тревог и результаты вычислений оконных функций.

Оптимизация поиска и получения актуальных значений достигается за счет использования материализованных представлений (materialized view) в ClickHouse.

Ограничение операций записи и чтения из БД осуществляется на уровне сервисов.

2.1.3 Обработка данных

За любые манипуляции с данными отвечают сервисы системы. Понятие "обработка данных" в этом контексте достаточно широкое, т.к. включает в себя:

- Получение данных из брокера сообщений и их десериализация;
- Запись данных в хранилище;
- Преобразование полученных значений в соответствии с настроенными правилами;
- Формирование новых элементов и расчетных показателей на основе полученных данных;
- Сопоставление тегов источника и цифровой модели предприятия;
- Доступ к данным для их дальнейшего использования.

Каждый сервис системы выполняют свою конкретную функцию. В следующем параграфе остановимся на каждом из подробнее.

2.2 Сервисы ядра системы

2.2.1 Сервис сбора данных

Основные функции:

1. Вычитывает данные из нужного топика брокера сообщений, в который их складывают источники данных (коннекторы, другие сервисы, внешние интеграции);
2. Сохраняет данные в хранилище данных.

При запуске сервис выполняет последовательно следующие шаги.

1. Открывает соединение к хранилищу данных;
2. Проверяет существование таблицы для значений тегов в хранилище данных. Если таблицы нет, то создаёт её;
3. Запускает задачу, которая считывает сообщения из топика в брокере сообщений и сохраняет полученные теги в буфер в памяти. В буфере может храниться 2 пакета данных в состоянии ожидания записи в хранилище. Размер каждого пакета данных по умолчанию составляет 400 000 тегов.
4. Запускает задачу, которая каждые 5 секунд помечает в брокере сообщений те пакеты данных, которые успешно записаны в хранилище.
5. Запускает задачу, которая периодически записывает данные из буфера в хранилище.

Имя таблицы, имя кластера и период хранения данных задаются в конфигурационном файле сервиса. При использовании в качестве хранилища данных кластера ClickHouse таблица будет создана с движком ReplicatedReplacingMergeTree. Если не используется кластер, то будет выбран движок ReplacingMergeTree.

Дополнительно для быстрого поиска актуальных значений тегов создается отдельная таблица, в которой хранятся значения с максимальной меткой времени по каждому полученному тегу. В случае ClickHouse, таблица наполняется автоматически средствами materialized view.

2.2.2 Сервис потоковой обработки данных

Обработчик — это набор правил, позволяющий преобразовать значение тега, получаемое от источника.

Основные функции сервиса:

1. Настройка обработчиков тегов;
2. Получение из брокера сообщений и обработка пакета значений тегов в соответствии с настроенными правилами;
3. Генерация новых значений и их отправка в брокер сообщений.

Принцип работы сервиса:

1. При запуске сервиса в оперативную память выгружается конфигурация из хранилища конфигураций;
2. Сервис подписывается на топик брокера сообщений, в котором находятся получаемые системой данные;
3. После получения нового пакета сервис осуществляет обработку для тех тегов, которые содержатся в конфигурации обработчиков, и генерирует значения новых тегов;
4. Сервис отправляет новые значения обратно в топик брокера сообщений.

В системе доступны следующие виды обработчиков:

- **Обработчик масштабирования**, который осуществляет линейное преобразование над исходным тэгом с возможностью округления получившегося результата;
- **Обработчик выделения бита**, который возвращает значение бита на заданной позиции в двоичном представлении целого числа;

- **Скриптовый обработчик**, который осуществляет преобразование над исходным тэгом на основе логики, описанной в пользовательском скрипте.

2.2.3 Сервис тревог

Основные функции сервиса:

1. Настройка обработчиков тревог;
2. Получение из брокера сообщений и проверка пакета значений тегов в соответствии с настроенными правилами;
3. Взведение тревог при необходимости и запись истории тревоги в хранилище данных.

Принцип работы сервиса описывается следующими шагами.

1. При запуске проверяется доступ к таблице тревог. Если в хранилище данных таблица отсутствует, то она создается;
2. Запрашивается последнее актуальное состояние из хранилища данных. Если актуального состояния нет, то объект состояния инициализируется пустым;
3. Сервис подписывается на топик брокера сообщений, в котором находятся получаемые системой данные;
4. После получения нового пакета для каждого из тегов сервис проверяет метку времени значения и запрашивает список обработчиков тревог, связанных с данным тегом.
Если метка времени значения меньше последнего пришедшего значения (нарушен порядок во времени) или не существует обработчиков тревог, связанных с тегом, то тег пропускается;
5. Если метка времени значения больше последнего пришедшего значения и существуют связанные с тегом обработчики тревог, проверяются условия взведения тревог. Допустимы следующие сценарии обработки:
 - **Тревога не была взведена, и условие взведения не выполнено.** Сохраняется последнее значение тега и его метка времени в состоянии, при этом в историю тревоги данный тег не попадает.
 - **Тревога не была взведена, и условие взведения выполнено.** Формируется новая запись для истории о взведении тревоги. Сохраняется последнее значение тега и его метка времени, состояние тревоги - взведена.
 - **Тревога была взведена, и условие взведения не выполнено.** Формируется новая запись для истории о снятии тревоги. Сохраняется последнее значение тега и его метка времени, состояние тревоги - снята.
 - **Тревога была взведена, и условие взведения выполнено.** В историю не вносятся никаких изменений. Обновляется последнее значение тега и его метка времени, состояние тревоги не изменяется.
6. В качестве фоновых запускаются процессы записи в хранилище данных, которые обновляют состояния и записывают историю тревог.

Обработчики тревог делятся на разные типы и могут иметь несколько подтипов. При изменении значения тега подтип тревоги может меняться с одного на другой, но тревога при этом продолжает оставаться в состоянии - взведена.

В системе доступны следующие виды обработчиков тревог:

- **Тревога по значению (VALUE)** срабатывает, когда значение тега опускается ниже или поднимается выше заданных уровней

Подтипы:

- **LOLO** — аварийно-низкий уровень;
 - **LO** — низкий уровень;
 - **HI** — высокий уровень;
 - **HIMI** — аварийно-высокий уровень.
- Тревога по скорости изменения (ROC) срабатывает, когда скорость изменения значений за промежуток времени превышает заданные уровни

Подтипы:

- **UP** — увеличение скорости;
- **DOWN** — снижение скорости.

2.2.4 Сервис истории

Основные функции сервиса:

- Получение исторических значений из хранилища данных;
- Получение актуальных значений из хранилища данных.

Принцип работы:

Взаимодействие с сервисом осуществляется при помощи HTTP запросов, которые направляются в хранилище данных с указанием типа элемента и его уникальным идентификатором.

Обращение к значениям может происходить по идентификаторам тегов, а может через идентификаторы объектов и переменных цифровой модели предприятия.

2.2.5 Сервис объектной модели

Объектная модель — это инструмент, который позволяет построить цифровую модель предприятия и связать между собой тэги, значения которых хранятся в хранилище данных, с цифровыми аналогами производственных агрегатов. Основная задача объекта в рамках системы представлять

свойства реального технологического объекта в цифровой модели этого объекта.

Атрибут представляет собой значение, связанное с объектом или переменной и содержащее в себе метainформацию. Это значение может быть фиксированным, либо ссылаться на изменяемое значение в справочнике.

Сервис объектной модели представляет собой иерархическое хранилище объектов, атрибутов и сопоставление имен переменных идентификаторам тэгов. Иерархия модели представляет собой набор объектов и переменных. Объект может как быть родителем для других объектов, так и иметь своего родителя.

Основные функции:

1. Настройка иерархии объектной модели;
2. Хранение конфигураций:
 - объектов, которые могут содержать переменные;
 - связей родитель-потомок между объектами;
 - связей переменных с тегами;
 - атрибутов;
 - справочников.
3. Обеспечение контентных ограничений доступов конкретным пользователям к конкретным узлам иерархии.

Взаимодействие с сервисом осуществляется при помощи HTTP запросов.

2.2.5 Сервис импорта данных

Основные функции:

1. Получает значения тегов посредством API метода;
2. Отправляет полученные значения в брокер сообщений в топик, где находятся получаемые системой данные.

Принцип работы:

1. Сервис получает данные пакетом с фиксированной структурой элементов:
 - Метка времени значения;
 - Идентификатор источника значения;
 - Идентификатор тега значения;
 - Числовое значение (null, если задано строковое);
 - Строковое значение (null, если задано числовое);
 - Качество значения;
 - Метка удаления значения.
2. Направляет данные в брокер сообщений в топик, где находятся получаемые системой данные.
Если процесс переноса данных в брокер сообщений прошел успешно, то в ответе на API запрос вернется код 200.
3. Если в процессе переноса данных в брокер сообщений произошел сбой, то сервис предпримет повторные попытки отправки данных.
Если по истечению всех попыток процесс отправки данных в брокер сообщений не завершился успехом, в ответе на API запрос вернется код группы 400 - импорт завершился с ошибкой.

Количество попыток отправки данных в брокер сообщений и временной интервал между попытками задаются в конфигурационном файле сервиса.

2.2.7 Сервис управления источниками данных

Коннекторы могут быть 2 видов - основной и резервный. Каждый коннектор имеет свой набор параметров конфигурации в зависимости от типа источника, к которому он подключается.

У коннектора есть перечень тегов, которые есть на источнике - доступные теги, и перечень сконфигурированных тегов - выбранные теги.

Основные функции:

1. Мониторинг состояния существующих коннекторов;
2. Настройка конфигураций коннекторов;
3. Получение доступных тегов источника.

Принцип работы:

1. При запуске сервис подписывается на топики топика брокера сообщений, которые отвечают за получение логов и обновлений с коннектора. В сообщении с обновлениями содержится имя коннектора, его роль и одно из 5 заполненных свойств, остальные выставлены в null.
2. Проверяется, что значит пришедшее свойство, и выбирается маршрут обработки. Перечень свойств:
 - **BrowsingResponse** - сообщение с результатом браузинга тегов. Если пришло данное свойство, то необходимо проверить значение в поле `IsSuccessful`.
Если оно `true` - полученные теги добавляются в коллекцию тегов источника;
Если оно `false` - в коллекции `diagnostics` проверяется имя соответствующего коннектора, если имя найдено, то обновляется информация о последнем завершеном браузинге. В противном случае ничего не предпринимается;
 - **ConfigUpdateResponse** - сообщение с результатом обновления конфигурации коннектора. Если пришло данное свойство, то в сообщении либо обновляются поля `IsConfigAccepted`, `LastActivityTime`, `ErrorMessageByHeaders` документа из коллекции `diagnostic`, либо создается новая сущность - коннектор и определяются те же поля;

- **GetDiagnosticsResponse** - сообщение со статусом здоровья коннектора. Если пришло данное свойство, то либо обновляется статус работоспособности коннектора, либо (в случае отсутствия конфигурации на момент получения) создается новая сущность - коннектор, для которой заполняются полученные поля в коллекции `diagnostics`;
- **DiscoveryResponse** - сообщение о появлении нового коннектора с системе. Если пришло данное свойство, то в коллекцию `diagnostics` добавляется новый коннектор;
- **PrimaryConnectorHealthStateChangedEvent** - сообщение о работоспособности основного коннектора. Данное свойство не обрабатывается сервисом.

2.2.8 Сервис API

Основные функции:

1. Точка входа для взаимодействия внешних пользователей с системой;
2. Обеспечение взаимодействия сервисов системы между собой.

Для использования API пользователю необходимо получить пару токенов доступа (access и refresh) в Keycloak. Токен содержит информацию о перечне ролей, назначенных пользователю.

Принцип работы:

Сервис получает HTTP запрос и определяет, в какой сервис нужно его отправить. Сервис проверяет токен авторизации пользователя, если недостаточно прав для выполнения запроса, то возвращается ошибка 403 Forbidden.

2.2.9 Сервис отчетов

Ресурсы - файлы, которые используются в качестве бланков для заполнения отчета в результате работы скрипта.

Шаблон отчета - совокупность ресурсов и программного кода (скрипта), выполнение которого позволяет сформировать отчет.

Отчёт - результат работы сервиса, который получает на вход параметры и шаблон отчета, а возвращает файл выбранного формата.

Каждый Шаблон отчёта может дополнительно содержать список параметров в формате ключ-значение, которые будут подставлены в исходный код скрипта во время генерации отчёта.

В исходном коде скрипта доступны методы библиотеки `ofg_pims_api`, которая позволяет обращаться к API системы.

Основные функции:

1. Настройка шаблонов отчетов;
2. Импорт ресурсов;
3. Генерация отчетов, посредством вызова скрипта на языке Python, в котором реализован модель запроса к данным платформы.

Принцип работы:

1. Взаимодействие с сервисом осуществляется при помощи HTTP запросов;
2. Во время обработки запроса на генерацию Отчета сервис выполняет последовательность команд, указанных в скрипте, и возвращает результат его выполнения отправителю запроса. Скрипт выполняется в отдельном процессе.
3. Если скрипт выполнен без ошибок и файл Отчета был успешно сохранен, то в ответе будет указано имя файла для скачивания. В противном случае в ответе будет указана ошибка, в связи с которой не удалось сохранить отчет.
4. Сервис выполняет фоновую очистку сгенерированных файлов Отчета из хранилища, по умолчанию файл доступен для скачивания в течение 24 часов.

2.2.10 Сервис уведомлений

Уведомление (задача) - часть программы, которая запускается автоматически по расписанию, указанному в конфигурации Уведомления. Задача может проверять ряд условий, которые определяют, нужно ли отправлять уведомление получателю.

Основные функции:

1. Настройка уведомлений;
2. Проверка условий (триггеров) генерации уведомления;
3. Генерация уведомления;
4. Мониторинг отправок уведомлений;
5. Назначение групп рассылок уведомлений.

Принцип работы:

1. Взаимодействие с сервисом осуществляется при помощи HTTP запросов;
2. Сервис осуществляет проверку условий отправки уведомлений;
3. Если условие выполнено, то уведомление отправляется в брокер сообщений, в топика с информацией об отправках ; Уведомление может содержать вложение - отчет, созданный в сервисе отчетов. Для таких уведомлений перед отправкой выполняется формирование отчета с указанными параметрами. Полученный файл является вложением для письма.

Отправка уведомления происходит по электронной почте.

Типы уведомлений:

- **По расписанию.** Уведомления отправляются безусловно по настроенному расписанию;
- **По событию.** Уведомление отправляется только если выбранная в настройках тревога взведена, время её взведения превышает заданную продолжительность и статус тревоги удовлетворяет выбранному в фильтре.
- **По скрипту.** Условие отправки уведомления задаётся скриптовым языком. Отправка уведомления произойдет только, если в результате выполнения скрипта вернулось `true`.

Если задано несколько условий, то для отправки уведомления они все должны быть выполнены.

2.2.11 Сервис отправки

Основные функции:

1. Получение пакетов из брокера сообщений, топика с информацией об отправке, в который их публикуют другие сервисы;
2. Формирование из полученных данных письма;
3. Отправка письма получателям с использованием почтового сервера;
4. Хранение информации по сообщениям. По умолчанию сообщения хранятся в течение 1 недели. Очистка информации происходит автоматически по принципу FIFO.

Принцип работы:

1. Сервис подписывается на топик брокера сообщений, в который поступают запросы на отправку;
2. На основании полученных из пакета данных сервис формирует письмо со структурой:
 - Тип отправки (только email);
 - Email адреса получателей;
 - Тема письма;
 - Текст письма;
 - Вложения.
3. Сервис осуществляет отправку писем с помощью SMTP сервер отправки сообщений;
4. Если почтовый сервер не принял письмо, то сервис отправки сообщений повторно попытается отправить сообщение через некоторое время;
5. Если в конечном итоге не получилось отправить сообщение, то сервис отправляет сообщений в топик брокера сообщений, где содержатся неотправленные сообщения.

Настройки сервера SMTP, учетные данные email адреса отправителя, количество попыток повторной отправки и интервал между ними содержатся в конфигурационном файле сервиса.

2.2.12 Сервис оконных функций

Оконная функция - функция, принимающая в качестве аргументов источники (числовые теги), метки времени значений которых попадают в окно, на котором она рассчитывается. Для описания преобразования значений источников в результат оконной функции используется скриптовый язык.

Результат оконной функции - значение тега, временная метка которого совпадает с временной меткой начала окна.

Сплиттер - функция, которая имеет направление - вперед и возвращает несколько значений в окне. Окно, в котором располагаются метки времени значений сплиттера, зависит от направления сплиттера. Для сплиттера с направлением вперед значение, получаемое от источника на начало окна (метка времени попадает в окно), преобразуется в n значений с метками времени внутри окна.

Граф вычисления оконных функций - однонаправленное дерево.

Основные функции:

1. Настройка функций и сплиттеров;
2. Настройка расписаний;
3. Расчет значений функций и сплиттеров;
4. Запись рассчитанных значений в хранилище данных.

Значения оконных функций и сплиттеров хранятся в отдельной от значений временных рядов таблице в хранилище данных. Таблица содержит столбец, в который каждая оконная функция записывает время последнего вычисления.

Алгоритм работы разового расчета зависимостей

1. Для каждой оконной функции запрашиваются метки времени значений ее источников, которые были вставлены в базу позже метки времени последнего расчета этой оконной функции;
2. Для временных меток, полученных в пункте 1, вычисляется список окон, на которых нужно рассчитать эту оконную функцию;
3. Для расчета в окне по источникам запрашиваются значения, метки времени которых попадают в это окно;
4. Выполняется скрипт расчета значения оконной функции;
5. Если в окне присутствуют значения источников с пометкой удаления, и функция не возвратила значение (вернулся `null`), то для данного окна в таблице значений оконных функций в хранилище данных значение помечается на удаление. Если в БД отсутствовали значения источников для расчета функции в окне, то для данного окна расчет функции не происходит, ничего не записывается в хранилище данных.
6. Если у оконной функции задан параметр "Пересчитать", то после сохранения для этой оконной функции в ХД находятся значения источников, время вставки в ХД которых больше либо равны заданной метке времени. Для выбранных данных определяются границы окон функции и

выполняется расчет.

Алгоритм непрерывного расчета

Расчет происходит по таймеру с учетом графа зависимости источников.

Алгоритм разового расчета запускается не чаще, чем завершился некоторый интервал - таймер. Для каждой оконной функции задан собственный интервал пересчета. Если рассчиталось окно, на основе которого считается другое, но при этом у зависимого окна не завершился таймер, расчет зависимого окна пропускается.

Алгоритм работы сплиттера

При поступлении значения источника сплиттер определит список временных меток - начал окон, заданного в конфигурации типа, и запишет значения в хранилище данных. В конфигурации сплиттера, указан период прихода значений источника и коэффициент, который необходимо использовать для преобразования значения источника в значения сплиттера.

2.2.13 Сервис бэкенда мнемосхем

Основные функции:

1. Хранение библиотеки графических элементов;
2. Хранение превью мнемосхем;
3. Хранение конфигураций мнемосхем.

Для функционирования сервиса необходимо наличие подключения к S3-совместимому хранилищу, где хранятся превью существующих мнемосхем и библиотека графических элементов, а также в базе данных конфигураций, где хранятся структуры мнемосхем.

2.2.14 Сервис обработки моделей предиктивной аналитики

Основные функции:

1. Создание аналитических моделей на языке Python;
2. Настройка обработчиков моделей;
3. Запуск обработчиков моделей;
4. Мониторинг выполнения обработчиков;
5. Отправка расчетных значений в брокер сообщений.

Принцип работы сервиса:

1. При запуске сервис подписывается на топик брокера сообщений, в который в процессе работы помещаются успешные результаты обработки;
2. Каждый обработчик запускается периодически;
3. При запуске отдельного обработчика исторические данные по тегам-источникам и аналитическая модель загружаются в файлы, а также создается пустой файл для записи результатов скрипта;
4. Если исполняемый скрипт аналитической модели завершил свою работу успешно, то результаты вычитываются из файла, проверяются на соответствие заданным выходным параметрам и отправляются в топик брокера сообщений, где находятся получаемые системой данные;
5. По окончании обработки все созданные временные файлы удаляются.

При любом возможном статусе завершения обработки фиксируется история срабатывания обработчика.

Для ошибок и предупреждений в лог записывается текст возникшей ошибки или уведомления. Сервис выполняет фоновую очистку истории, храня данные за последнюю неделю.

Перед загрузкой новой модели существует возможность протестировать ее на данных платформы. Помимо скрипта модели можно симитировать работу для нового или уже существующего обработчика на данных платформы в прошлом.

2.2.15 Сервис интеграции с SAP

Техническое место (ТМ) - объект в иерархии оборудования (корень, ветка) .

Единица оборудования (ЕО)- объект иерархии оборудования (лист).

Точка измерения (ТИ) - показатель единицы оборудования.

Индекс технического состояния (ИТС) - показатель, который относится к ЕО, при этом для него не заведена отдельная ТИ, и характеризует состояние по шкале от 0 до 100, где 100 - наилучшее состояние ЕО.

Наработка - значение ТИ, соответствующее количеству единиц времени работы ЕО.

Соответствие: ТМ и ЕО в SAP - объект в PIMS; ТИ в SAP - переменная в PIMS.

В объектной модели информация о техническом месте и единице оборудования представлена в формате атрибутов объекта. У объекта, соответствующего техническому месту, есть идентификатор ТМ, но нет идентификатора ЕО, а у объекта, соответствующего единице оборудования, есть идентификатор ЕО и идентификатор ТМ родительского объекта.

Основные функции:

1. Конфигурирование списка переменных, значения которых будут отправляться в SAP;
2. Отправка значений переменных в SAP по расписанию;
3. Настройка соответствия идентификаторов переменных в объектной модели и точек измерения в SAP;
4. Перенос данных наработки и ИТС в базу данных временных рядов;
5. Получение и хранение данных по ТОиР.

Принцип работы сервиса:

Поток данных в SAP

Существует набор переменных, по которым необходимо передавать значения. Перечень переменных конфигурируется пользователем в GUI и может изменяться.

Период отправки данных - 1 час, с привязкой к целому астрономическому часу.

1. При срабатывании таймера на очередную отправку сервис запрашивает актуальные значения для переменных для списка;
2. Осуществляет проверку значения по отношению к последнему отправленному
 - o Если ни метка времени, ни значение не менялись, то такое значение в SAP не отправляется;
 - o Если либо значение, либо метка времени, либо и значение, и метка времени изменились, то значение будет отправлено в SAP;
3. Отправляет данные в SAP с помощью API;

Поток данных из SAP

Из SAP в систему отправляются данные по ТОиР:

- Сообщения (дефекты, простои, предписания);
 - Заказы (ремонты плановые и внеплановые);
 - ИТС;
 - Нарботка для ЕО;
1. При создании / изменении информации для сообщений / заказов и при обновлении значений Нарботки / ИТС сервис на стороне SAP с помощью API передает информацию в систему;
 2. Сервис интеграции с SAP распределяет полученные данные между таблицами в PostgreSQL в соответствии с типом полученной информации;
 3. Для Нарботки и ИТС запускается механизм переноса данных в ХД:
 - o Определяются идентификаторы элементов в объектной модели;
 - o Формируется пакет на отправку данных и с периодом - 1 минута;
 - o Пакет данных отправляется через сервис Импорта данных в Kafka;
 - o Сервис Сбора данных вычитывает данные из Kafka и записывает в ХД;
 4. Если для Нарботки из SAP поступает значение с меткой "Сторно" (отмена ранее присланного значения наработки), в этом случае в ХД передается значение наработки с признаком `is_deleted = 1`.

2.2.16 Сервис журнала диспетчера

Основные функции:

1. Хранение внесенных пользователем текстовых заметок;
2. Получение информации о квитированных тревогах;
3. Предоставление доступа к истории заметок и квитированных тревог.

Принцип работы сервиса:

1. При запуске сервис подписывается на топик брокера сообщений, в который поступает информация о квитировании тревог;
2. При появлении в топике брокера сообщений квитированной тревоги сервис делает запись в БД;
3. При создании пользовательской заметки данные записываются в БД;
4. Редактирование записей доступно в течение 1 часа от момента создания записи и только пользователю, создавшему запись. Спустя час редактирование записи недоступно;
5. При удалении записи она помечается как удаленная, но не удаляется из БД. Для удаленных записей доступна возможность восстановления в течение 30 дней с момента удаления;
6. Сервис выполняет фоновую очистку БД журнала диспетчера, данные хранятся за последние 30 дней.

2.2.17 Сервис искусственного интеллекта

Основные функции:

База знаний

1. Загрузка документов (PDF, DOCX, DOC, TXT, XLSX) с автоматическим извлечением пар "вопрос-ответ" из содержимого;
2. Управление парами "вопрос-ответ": просмотр, добавление вручную, редактирование и удаление;
3. Обнаружение конфликтов между парами "вопрос-ответ" из разных документов;
4. Чат по загруженным документам - пользователь задаёт вопрос и получает ответ на основе содержимого базы знаний.

Помощник по инструкциям

1. Загрузка инструкций в формате Markdown с автоматическим разбиением на смысловые фрагменты;
2. Чат-помощник, который отвечает на вопросы пользователей, опираясь на содержимое загруженных инструкций.

Обнаружение аномалий

1. Получение аномалий по переменным объектов за заданный период;
2. Поиск ассоциативных правил: выявление взаимосвязей между аномалиями различных переменных;
3. Автоматический анализ найденных взаимосвязей с помощью LLM для формирования текстового описания причин и следствий.

Управление агентами мониторинга

1. Создание, редактирование и удаление агентов: именованных наборов переменных, по которым выполняется анализ аномалий.

Принцип работы сервиса:

Сервис представляет собой веб-сервер, который использует языковые модели (LLM) и векторные представления (эмбединги) для интеллектуальной обработки данных.

Обработка документов

При загрузке файла в базу знаний сервис выполняет следующие шаги:

1. **Приём файла.** Загруженный файл сохраняется в S3-хранилище. В базе данных создаётся запись с метаданными (имя файла, тип, размер, S3-ключ) и статусом `PENDING`.
2. **Фоновая обработка.** Фоновый поток периодически опрашивает базу данных на наличие файлов в статусе `PENDING`. При обнаружении статус файла переводится в `PROCESSING`.
3. **Извлечение текста.** В зависимости от типа файла текст извлекается разными способами:
 - **PDF** - файл отправляется во внешний OCR-сервис, который выполняет распознавание текста. Сервис периодически опрашивает статус OCR-задачи и обновляет прогресс обработки в базе данных.
 - **DOCX/DOC** - текст извлекается с помощью специализированных библиотек.
 - **XLSX** - таблица анализируется: определяются строки-заголовки, с помощью LLM фильтруются информативные столбцы, после чего строки преобразуются в текстовые записи.
 - **TXT** - текст читается напрямую.
4. **Разбиение на фрагменты.** Извлечённый текст разбивается на фрагменты (чанки) с помощью токенизатора. Границы фрагментов уточняются по абзацам и предложениям. Между фрагментами создаётся перекрытие для сохранения контекста.
5. **Генерация пар "вопрос-ответ".** Фрагменты объединяются в скользящие окна и передаются в LLM, которая формирует из каждого окна набор пар "вопрос-ответ". Прогресс обработки обновляется в базе данных по мере прохождения окон.
6. **Построение эмбедингов.** Для каждой полученной пары "вопрос-ответ" вычисляется векторное представление (эмбединг) и сохраняется в базе данных вместе с текстом.
7. **Проверка на конфликты.** Каждая новая пара сравнивается с уже существующими: находится ближайшая по эмбедингу пара из другого документа, и LLM определяет, содержат ли они противоречивую информацию. При обнаружении конфликта создаётся запись, которую пользователь может просмотреть и разрешить.

При ошибке на любом этапе статус файла переводится в `FAILED` с сохранением сообщения об ошибке. Пользователь может повторно запустить обработку таких файлов.

Чат и поиск по документам

Чат по базе знаний реализован по паттерну Retrieval-Augmented Generation (RAG):

1. **Приём вопроса.** Вопрос пользователя сохраняется в историю чата, привязанную к сессии.
2. **Поиск релевантного контекста.** Вычисляется эмбединг текущего вопроса, а также эмбединги нескольких предыдущих вопросов из истории сессии. По каждому эмбедингу выполняется поиск ближайших пар "вопрос-ответ" по косинусному расстоянию. Результаты объединяются и дедуплицируются.
3. **Генерация ответа.** Найденные пары передаются в LLM в качестве контекста вместе с историей диалога. LLM формирует ответ, опираясь на предоставленный контекст.
4. **Сохранение ответа.** Ответ сохраняется в историю чата для использования в последующих вопросах.

Чат может работать как по всем документам объекта, так и в рамках конкретного документа.

Анализ аномалий

Модуль обнаружения аномалий работает с данными мониторинга переменных:

1. **Хранение аномалий.** Аномалии хранятся в базе данных с привязкой к объекту, переменной и временной метке. При выдаче смежные аномалии (с шагом 15 минут) автоматически объединяются в непрерывные интервалы.
2. **Поиск ассоциативных правил.** Для набора переменных, входящих в состав агента, выбираются все аномалии за указанный период. На основе временных рядов аномалий запускается алгоритм поиска ассоциативных правил, который выявляет устойчивые взаимосвязи между переменными (например, "аномалия в переменной А часто сопровождается аномалией в переменной В").
3. **Анализ взаимосвязей с помощью LLM.** Для каждого найденного правила определяются временные пересечения аномалий. Затем текстовое описание правила передаётся в LLM двумя способами:
 - на основе контекста из базы знаний (RAG) - для формирования объяснения с учётом документации объекта;
 - на основе общих знаний LLM - для независимой интерпретации. При этом учитывается приоритет (вес) документов в базе знаний: более приоритетные документы имеют большее влияние на ранжирование контекста.

Помощник по инструкциям

Помощник по инструкциям работает аналогично чату по базе знаний, но с другим источником данных:

1. **Загрузка инструкций.** Инструкция в формате Markdown загружается и разбивается на фрагменты. Для каждого фрагмента вычисляется эмбеддинг.
2. **Ответы на вопросы.** При поступлении вопроса пользователя выполняется поиск ближайших фрагментов по косинусному расстоянию. Для повышения точности поиска учитываются предыдущие вопросы из истории сессии. Найденные фрагменты передаются в LLM для генерации ответа.
3. **Версионирование.** При загрузке новой версии инструкции предыдущие фрагменты остаются в базе данных, но для ответов используются только фрагменты из последней версии документа.

2.2.18 Сервис OCR

Основные функции:

1. Прием файла и постановка задания в очередь;
2. Пошаговая обработка страниц документа;
3. Обновление статуса обработчик документа;
4. Сохранение распознанного текста;
5. Передача статусов и обработанных данных по API;
6. Автоматическое очищение старых завершенных или ошибочных заданий.

Принцип работы сервиса:

1. При запуске сервиса запускается веб-сервер, создаются таблицы в БД, запускаются два фоновых потока, отвечающих за обработку файлов в очереди и удаление устаревших записей. Поток обработки переводит зависшие после рестарта записи в состояние ожидания обработки;
2. При получении сервисом файла на обработку происходит сохранение файла во временный каталог. В БД создается запись со статусом ожидания обработки и временным путем к файлу. Сервис отправляет клиенту идентификатор задания и подтверждение загрузки;
3. Поток обработки периодически выбирает первый файл, ожидающий обработки, из очереди, фиксирует статус и время начала обработки. При первой задаче инициализирует алгоритм обработки и загружает модель распознавания в GPU. Для каждой страницы PDF происходит конвертация страницы в изображение, выполняется распознавание с контролем за цикливанием текста, результат добавляется в ответ, возвращаемый клиенту, обновляется статус обработки документа;
Если обработка документа успешно завершена, то фиксируется соответствующий статус и время завершения обработки документа в БД.
Если во время обработки пришел запрос остановки, то задание на обработку завершается с соответствующим статусом.
Если во время обработки возникли ошибки, то обработка завершается с ошибкой. Клиенту возвращается информация о возникших ошибках; Временный файл удаляется после обработки (успех/ошибка/остановка).
4. Поток очистки периодически проверяет БД и удаляет записи заданий, которые завершились успешно, с ошибкой или были остановлены, если период между текущей датой и датой обновления статусе больше порога.

3 Импорт данных и конфигураций в систему

3.1 Импорт данных

Данные в систему могут поступать с помощью коннекторов от источников данных, а могут вводиться вручную.

Импорт данных в систему можно осуществлять пакетами, где каждый элемент имеет следующую структуру:

- Временная метка значения;
- Идентификатор источника;
- Идентификатор тега;
- Числовое значение (null, если определено строковое значение);
- Строковое значение (null, если определено числовое значение);
- Качество сигнала;
- Метка удаления.

3.2 Импорт конфигураций

В системе доступен функционал импорта конфигураций элементов из файла, для более быстрого создания перечня однотипных элементов.

Импортировать из файла можно следующие конфигурации:

- Доступные и выбранные теги коннектора;
- Источники ручного ввода и теги источника ручного ввода;
- Единицы измерения;
- Переменные объекта;
- Обработчики тревог;
- Переменные, значения которых отправляются в SAP в рамках интеграции;
- Корневые теги оконных функций.

4 Экспорт данных из системы

Система позволяет обращаться к данным и отображать их в:

- Мониторинге текущих значений или трендах исторических значений в GUI;
- Формате Значения / Таблицы / Датчика для текущих значений или Графика исторических значений в АРМ;
- Ситуативной экранной форме в АРМ;
- Окне "Анализа данных" в АРМ;
- Отчетах.

При экспорте системы поддерживаемые форматы для:

- отчетов `.xlsx` и `.pdf`;
- для мнемосхем `.json` и `.pdf`;
- для ситуативной экранной формы `.xlsx`;
- для данных ХД `.json`, `.xlsx`;
- для журнала диспетчера `.xlsx`.