МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЯДЕРНЫЙ УНИВЕРСИТЕТ «МИФИ» (НИЯУ МИФИ)

ИНСТИТУТ ЯДЕРНОЙ ФИЗИКИ И ТЕХНОЛОГИЙ КАФЕДРА
№40 «ФИЗИКА ЭЛЕМЕНТАРНЫХ ЧАСТИЦ»

УДК

На правах рукописи

ПЛОТНИКОВ АРТЕМ ВАЛЕРЬЕВИЧ

# РАЗРАБОТКА ПРОГРАММНОГО КОМПЛЕКСА ДЛЯ УПРАВЛЕНИЯ ПРОЦЕССАМИ ОБРАБОТКИ ПЕРВИЧНЫХ ДАННЫХ ЭКСПЕРИМЕНТА SPD

Направление подготовки 14.04.02 «Ядерная физика и технологии»
Отчет по преддипломной практике

Научный руководитель,
к.ф-м.н _____ Е. Ю. Солдатов

Научный консультант
к.т.н _____ Д. А. Олейник

Москва 2025

# Содержание

# Введение

Эксперименты в сфере физики высоких энергий всегда характеризовались большими объёмами собираемых и обрабатываемых данных. При столкновениях пучков частиц на коллайдерах регистрируется до нескольких миллионов событий в секунду. Хотя в настоящий момент сверхбыстрые детекторные системы позволяют регистрировать подобные события с достаточно высокой точностью, всё также остаётся необходимость эффективно управлять большим потоком данных с детекторов.

Учитывая ограниченные ресурсы на запись и последующий анализ данных, неизбежно встаёт задача быстрой фильтрации фоновых и малозначимых событий. В этом случае важной оказывается своевременная оценка физической ценности сигналов. Традиционно в таких случаях применяют аппаратные триггерные системы, но в некоторых ситуациях подобные решения не соответствуют особенностям эксперимента. В частности, необходимой может оказаться частичная реконструкция событий, для которой предполагается создание специализированных вычислительных систем, обрабатывающих данные в режиме реального времени.

Именно здесь на первый план выходит концепция программных триггеров, опирающихся на вычислительные мощности многопроцессорных серверов. Плюсом данного подхода является ещё и то, что он даёт возможность динамически менять критерии отбора в зависимости от целей текущего исследования , оперативно адаптируясь к изменяющимся условиям.

*В данной работе будет рассмотрена подобная система для разрабатываемого эксперимента SPD на коллайдере NICA (Дубна, Россия) [1]. А также будет описан этап разработки прототипа одной из подсистем – системы управления процессами обработки.*

# Цель и задачи работы

**Цель работы:** разработка прототипа системы управления процессами обработки (workflow management system - WfMS), позволяющего параллельно управлять большим количеством процессов обработки данных и контролировать статусы выполнения цепочек обработки для физического эксперимента SPD.

**Задачи:**

1. ознакомиться с экспериментом SPD и особенностями его экспериментальной установки;

2. изучить основные принципы работы SPD Online Filter;

3. рассмотреть возможность использования готовых решений в рамках создания системы управления процессами обработки;

4. ознакомиться с проектом системы workflow management system (WfMS): определить основные сервисы WfMS, выявить их основной функционал и принципы взаимодействия с другими системами SPD Online Filter;

5. разработать API для работы с базой данных;

6. разработать сервисы WfMS с использованием предпочтительного стека:

6.1 создать сервис для взаимодействия с оператором обработки:

6.1.1 авторизация пользователей;

6.1.2 обработка CWL-шаблонов;

6.1.3 вывод информации о заданиях и шаблонах;

6.1.4 работа со статусами шаблонов.

6.2 реализовать сервис генерации заданий:

6.2.1 получение датасетов из data management system

(DMS);

6.2.3 создание заданий по шаблонам для последовательностей.

6.3 разработать сервис для опроса DMS:

6.3.1 опрос DMS о готовности входных датасетов для заданий;

6.3.2 создание выходных датасетов и датасетов логов в DMS;

6.3.3 отправление заданий на обработку в workload management system (WMS).

6.4 создать сервис для работы с WMS:

6.4.1 периодический опрос WMS для отслеживания статусов заданий;

6.4.2 закрытие датасетов после завершения заданий;

6.4.3 удаление входного и промежуточных датасетов после выполнения всех заданий в цепочке.

7. протестировать разработанные сервисы и совместную работу систем, исправить баги.

# Глава 1

# SPD — эксперимент по изучению спиновой структуры нуклонов

Несмотря на значительные достижения квантовой хромодинамики при описании взаимодействия кварков и глюонов, остаётся неразрешённым вопрос о том, почему нуклоны такие, какими мы их наблюдаем. Понимание структуры и фундаментальных свойств нуклона на основе динамики его составляющих - кварков и глюонов, остаётся одной из основных нерешенных проблем квантовой хромодинамики.

Модель кварков успешно предсказывает большинство основных характеристик адронов, таких как заряд, чётность, изоспин и свойства симметрии, а также взаимосвязи между ними. Некоторая динамика взаимодействий частиц также может быть качественно понята в рамках этой модели. Однако она не даёт объяснения спиновым свойствам адронов с точки зрения их составляющих. Со времен «спинового кризиса», обозначенного в 1987 году, проблема спиновой структуры нуклонов остается загадкой современной физики высоких энергий. Одной из главных задач, привлекающих огромные теоретические и экспериментальные усилия, является вопрос о том, как спин нуклона формируется из спинов и орбитальных моментов его составных частей - валентных кварков, "морских" кварков и глюонов. Полное объяснение можно получить через так называемые функции распределения партонов, зависящие от их поперечного

импульса.

Эксперименты по поляризованному глубоконеупругому рассеянию (CERN, DESY, JLab, SLAC) и высокоэнергетические поляризованные протон-протонные столкновения (RHIC в BNL) являются основным источником информации о спин-зависимых структурных функциях нуклонов. Тем не менее наши знания о внутренней структуре нуклона все еще ограничены. Особенно это касается глюонного вклада [2].

## 1.1 Spin Physics Detector (SPD)

Экспериментальная установка SPD, которую планируется разместить в одной из двух точек пересечения пучков коллайдера NICA (рис. 1), предназначена для всестороннего изучения спиновой структуры протона и дейтрона в поляризованных p-p, d-d и p-d-столкновениях. Основное внимание будет уделено изучению их поляризованной глюонной компоненты в реакциях инклюзивного рождения чармониев, открытого чарма и прямых фотонов, а также прочих спинзависимых явлений в столкновениях поляризованных пучков протонов и дейтронов с энергией в системе центра масс до 27 ГэВ и светимостью до $10^{32}$ см$^{-2}$ с$^{-1}$.

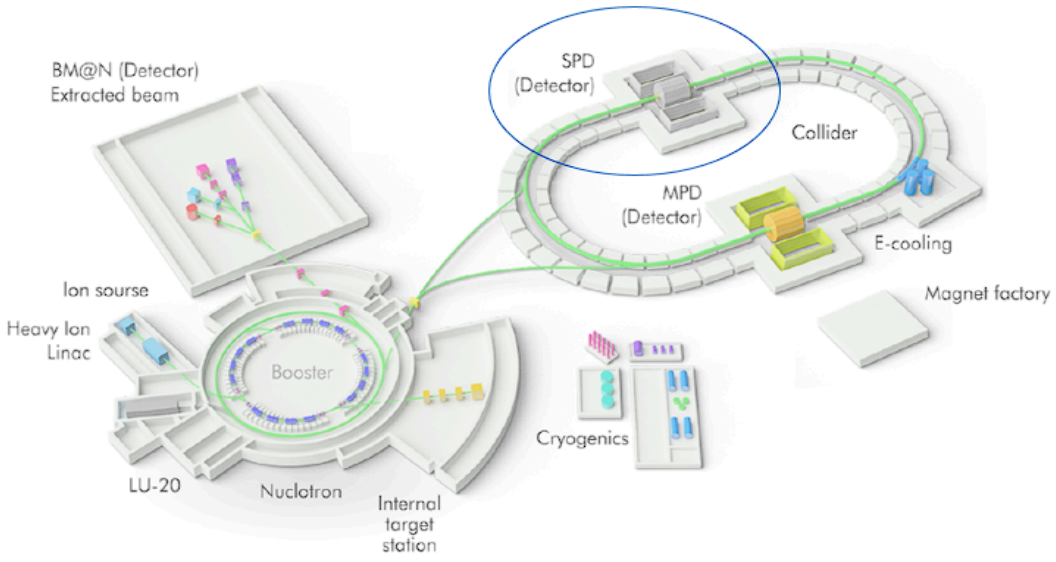



**Рис. 1** Комплекс NICA [3]

Эксперимент SPD планирует получить данные о спиновых асимметриях для анализа корреляций между направлением спина протона (дейтрона), его импульсом и направлением спина, связанным с продольным и поперечным импульсами глюонов внутри протона (дейтрона). В начальной фазе работы установки, до достижения проектных светимости и энергии столкновения, основное внимание планируется уделить изучению спиновых эффектов в упругих p-p и d-d рассеяниях, поиску мультипартонных корреляций и новых связанных состояний, исследованию рождения чарма у порога, изучению поляризаций гиперонов и т.д [4].

Новые данные о спиновой структуре протона и дейтрона, которые будут получены на установке SPD, приблизят нас к пониманию фундаментальных основ квантовой хромодинамики.

Исследования на установке SPD с использованием поляризованных протонных пучков позволят заполнить пробелы между измерениями на низких энергиях на ускорителях ANKE-COSY и SATURNE и измерениями на высоких энергиях на коллайдере RHIC и планируемых на LHC экспериментах с неподвижной мишенью. Важно отметить, что возможность работать с поляризованными пучками дейтронов в этом диапазоне энергий является уникальной особенностью данного эксперимента.

### 1.1.1 Экспериментальная установка SPD

Экспериментальная установка, изображенная на рисунке 2, будет обладать широким геометрическим охватом, приблизительно равным 4π, продвинутой системой для восстановления треков и вершин, а также разнообразными возможностями по идентификации частиц, основанными на передовых технологиях.

Кремниевый вершинный детектор (VD) обеспечит координатное разрешение при восстановлении вершины лучше 100 микрон, что критически

важно для точной реконструкции вторичных вершин распада D-мезонов. Трековая система, использующая строу-трубки (ST) в соленоидальном магнитном поле до 1 Тл, позволит определить поперечные импульсы вторичных частиц с точностью до 2%.

Времяпролетная система (PID) с временным разрешением 60 пс обеспечит разделение пионов, каонов и протонов в широком кинематическом диапазоне. Использование черенковских детекторов на основе аэрогеля позволит расширить этот диапазон. Для регистрации фотонов будет использован электромагнитный калориметр типа «шашлык» (ECal). Для уменьшения эффектов многократного рассеяния и конверсии фотонов минимизировано количество вещества во внутренней части установки.

Мюонная (пробежная) система (RS) предназначена для идентификации мюонов. Она также может служить в качестве грубого адронного калориметра. Пара пучковых счетчиков (BBC) и калориметров нулевого угла (ZDC) будут отвечать за локальную поляриметрию и контроль светимости.

Высокая частота столкновений (до 4 МГц) и большое количество каналов электроники определяют высокие требования к системе сбора данных, онлайн-мониторингу, компьютерной системе последующей обработки данных и программному обеспечению для реконструкции и анализа данных.

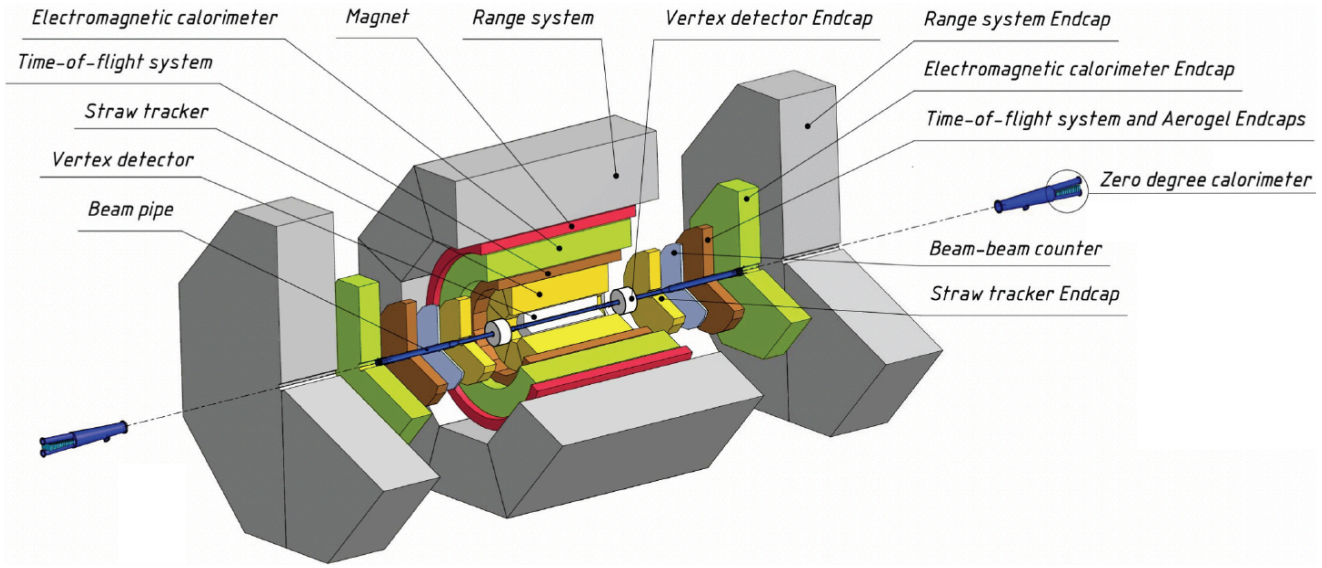



**Рис. 2** Экспериментальная установка SPD [5]

### 1.1.2 Объём данных эксперимента

В физике высоких энергий события, регистрируемые детектором, обрабатываются независимо, поэтому их можно считать наименьшей единицей данных. А значит, зная размер одного события, сложность обработки событий и ожидаемое количество событий для обработки, можно оценить требования к вычислительной системе для эксперимента.

Исходя из приблизительных оценок на SPD будет приходить 20 ГБ/с (или 200 ПБ/год) «необработанных» данных, что соответствует $3 \bullet 10^{13}$ событий/год.

Сбор, обработка и хранение такого объема данных представляет собой серьезную проблему для вычислительной инфраструктуры эксперимента и требует разработки новых методов и подходов для реконструкции событий, моделирования и физического анализа данных с использованием высокопроизводительных и распределенных вычислений.

### 1.1.3 Особенности сбора данных

Системы сбора данных в большинстве подобных экспериментов используют триггеры (рис. 3) для отбора и записи только тех событий, которые являются интересными для дальнейшего анализа. Триггеры – управляющий сигнал, по которому осуществляется запись с детекторных систем.

Триггеры могут быть настроены на различные критерии, например, на обнаружение определенных типов частиц, особенных шаблонов распределения энергии, наличие определенных вершин или корреляций между различными частицами. Такие системы триггеров позволяют экономить ресурсы хранения и обработки данных, фокусируясь только на событиях, которые соответствуют определенным критериям или интересам исследования.

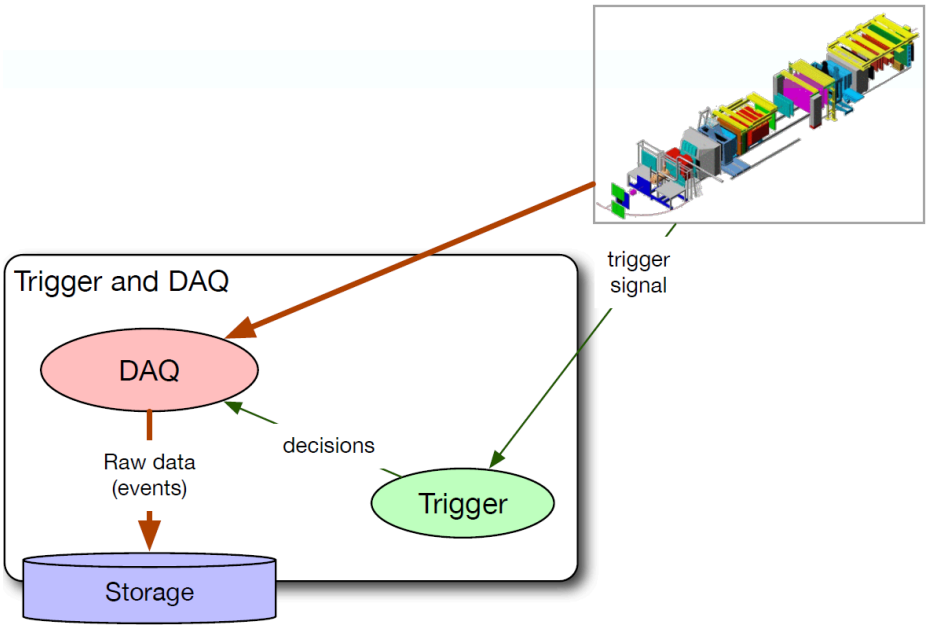



**Рис. 3** Триггерная система сбора данных [6]

Однако ввиду сложности и широты изучаемых процессов (выбор физического сигнала требует реконструкции импульса и вершины) невозможно построить критерии отбора данных на аппаратном уровне, поэтому для минимизации возможных систематических эффектов SPD будет оснащен бестриггерной системой сбора данных (рис. 5).

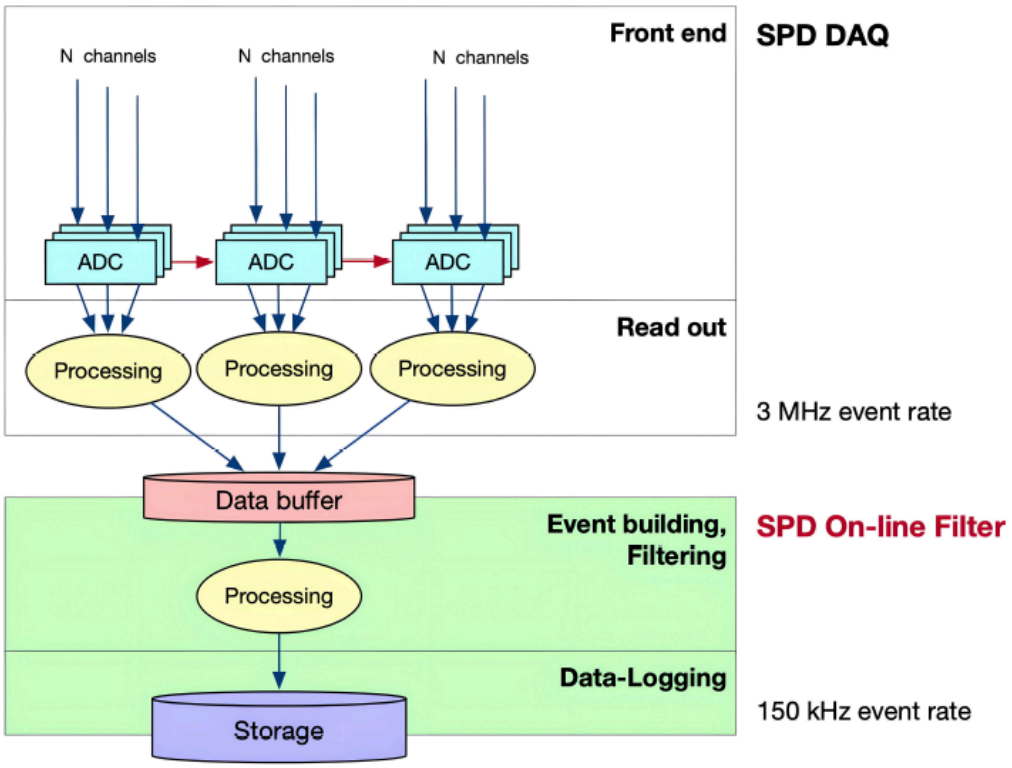



**Рис. 4** Бестриггерная система SPD [6]

Система сбора данных предоставляет данные, организованные по временным интервалам и разбитые на файлы разумного размера (несколько ГБ). При таком подходе отдельные файлы могут обрабатываться параллельно, что повышает эффективность работы системы. Каждый файл может быть обработан независимо, что упрощает процесс.

Такой подход особенно полезен, когда нет необходимости в обмене информацией между файлами на этапе обработки, и результаты обработки одного файла могут быть использованы как входные данные для следующего этапа. Это позволяет эффективно использовать вычислительные ресурсы и ускоряет процесс анализа данных в целом.

# Глава 2

# SPD Online Filter

## 1.1 Принцип работы SPD Online Filter

SPD Online Filter — это высокопроизводительная вычислительная система для высокопропускной первичной обработки данных эксперимента SPD.

Эта вычислительная система должна осуществлять следующее преобразование данных: идентифицировать физические события во временных интервалах; реорганизовать данные в событийно-ориентированном формате; отфильтровать события, оставляя только интересные с точки зрения эксперимента; согласовывать выходные данные, объединять события в файлы и файлы в наборы данных для будущей обработки.

Концепция базовой обработки:

1. Реконструкция треков и связывание их с вершинами;
2. Связывание срабатываний электромагнитного калориметра и пробежной системы с каждой вершиной по времени;
3. Определение несвязанных срабатываний детектора;
4. Объединение с необработанными сигналами от других субдетекторов;
5. Формирование блоков данных и сохранение частично реконструированных событий.

Данные объединяются в наборы файлов в зависимости от стадии их обработки. Каждый файл в наборе может быть обработан независимо, однако наборы обрабатываются в некоторой заданной последовательности. Каждый файл

можно обработать за некоторое разумное время на ограниченном вычислительном ресурсе (вычислительном узле).

Разбиение данных на более мелкие части позволяет управлять объемом обработки и минимизировать последствия при возникновении ошибок при обработке больших объемов данных. Работа с отдельными частями позволяет изолировать ошибки, которые могут возникнуть в процессе обработки, и снизить их воздействие на всю систему.

Такой подход также обеспечивает масштабируемость системы: добавление дополнительных вычислительных ресурсов позволяет обрабатывать еще большие объемы данных без значительного изменения общей архитектуры системы. Это особенно полезно для систем, работающих с высокочастотными данными или при необходимости быстрой обработки информации в режиме реального времени.

SPD Online Filter представляет из себя совокупность прикладного и промежуточного программных обеспечений, а также высокопроизводительного вычислительного кластера.

По результатам анализа первичных требований к системе были выделены основные компоненты промежуточного программного обеспечения:

1. Workflow management system – система управления процессами обработки – создание формального описания цепочек обработки, формирование и контроль исполнения этапов обработки данных;

2. Data management system – система управления данными – регистрация, каталогизация, контроль целостности файлов и датасетов;

3. Workload management system & pilot – система управления нагрузкой с подсистемой pilot – реализация этапов обработки (формирование задач и отправка их pilot).

Pilot – приложение, работающее на вычислительном узле и исполняющее задачи.

Архитектура системы SPD Online Filter приведена на рис. 5.

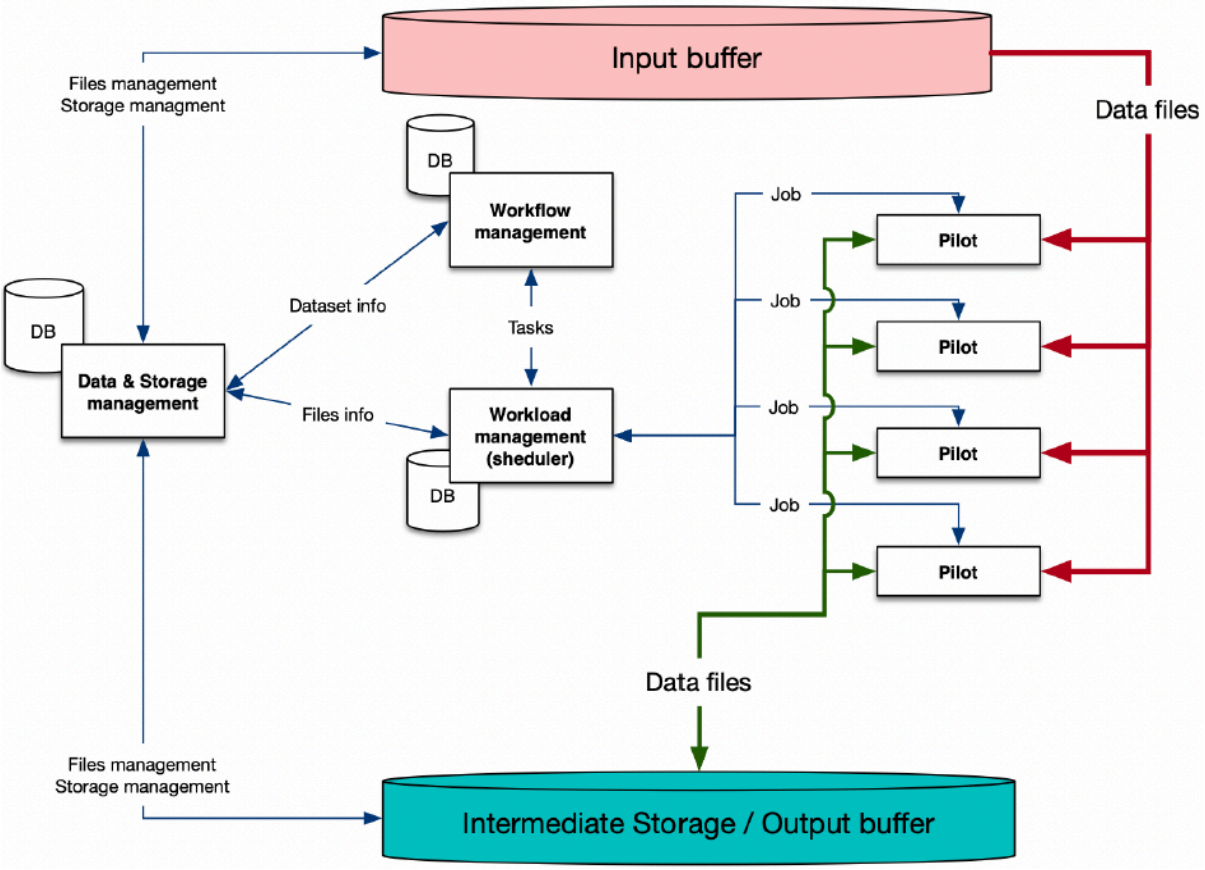



**Рис. 5** Архитектура SPD Online Filter [7]

## 1.2 Входные данные

Входные данные, приходят из системы сбора данных (DAQ) во входной буфер.

- Период набора – ассоциирован с физической задачей. Состоит из набора ранов. Имеет дату начала, дату окончания.
- Ран – интервал, когда условия эксперимента неизменны (калибровки, например). Измеряется часами. Состоит из фреймов.
- Фрейм – нумерованный блок данных с детектора. Может содержать информацию о множестве событий. Продолжительность секунды.
- Датасет – логическое объединение файлов в наборы для обработки. Датасет является единицей обработки для одного задания.
- Файл – количество данных для обработки одной задачей. [8]

# Глава 3

# Система управления процессами обработки (WfMS)

## 1.1 Возможность применения готовых решений

Рассматривая подобные эксперименты в сфере физики высоких энергий, становится ясно, что обойтись без создания собственной системы не получится. Так, например, эксперимент ALICE использует другой принцип обработки данных вкупе с характерным аппаратным решением. Эксперименты ATLAS, CMS и LHCb не афишируют свои решения: статьи поверхностны, а исходного кода нет в открытом доступе.

Использование Dirac [9] и PanDA [10] также не представляется разумной, вследствие использования специализированной вычислительной системы, которая не является географически распределенной, из-за чего большая часть функционала будет излишней.

Интересным решением является продукт с открытым исходным кодом — Apache Airflow [11], который может использоваться в качестве оркестратора для разработки, планирования и мониторинга сложных рабочих процессов.

Работа Apache Airflow основана на создании направленного ациклического графа (DAG), описывающего процессы обработки данных и определяющего связи и правила выполнения задач.

Планировщик обрабатывает DAG, а затем запускает экземпляры задач после выполнения их зависимостей. В фоновом режиме планировщик запускает подпроцесс, который следит и синхронизирует все DAG в указанном каталоге.

WebServer отвечает за отображение веб-интерфейса и аутентификацию пользователей.

Apache Airflow обладает множеством преимуществ и широким функционалом, однако для обработки задач в реальном времени его использование представляется неоптимальным [12].

SPD Online Filter как раз работает с данными по мере их поступления. Помимо всего прочего, нужно учитывать, что WfMS должна взаимодействовать с другими системами, отправлять им запросы, получать ответы и на их основе принимать дальнейшие решения. С помощью Apache Airflow довольно тяжело организовать такую систему.

## 1.2 Принцип работы разрабатываемой WfMS

После попадания в SPD Online Filter, данные регистрируются в системе управления данными (Data Management System — DMS). Затем они подготавливаются к последующей обработке при помощи системы управления процессами обработки (Workflow Management System — WfMS).

WfMS принимает эти зарегистрированные данные и сопоставляет их с определенным шаблоном цепочки обработки, который предоставлен оператором обработки данных. Каждый этап такой цепочки обработки генерирует соответствующие задания для выполнения. Эти задания затем направляются в систему управления нагрузкой (Workload Management System - WMS), где они распределяются на вычислительные ресурсы для выполнения. После этого WfMS контролирует процесс выполнения задания путем периодического опроса WMS об

их статусах и принимает решение об изменении приоритетов заданий или об их отмене.

Такой подход позволяет эффективно организовать и контролировать обработку данных по заданной цепочке этапов, что обеспечивает систему планирования и контроля рабочих процессов, повышающую оперативность обработки данных и предоставляющую возможность мониторинга за выполнением обработки.

Система управления процессами обработки (WfMS) имеет ряд функциональных требований, чтобы эффективно организовать и контролировать последовательности обработки данных:

1. Организация последовательностей обработки данных:
   - Последовательность: состоит из логически связанных шагов обработки данных, где результат одного шага является входными данными для следующего.
   - Шаг обработки: представляет собой действие, которое выполняется над набором данных. Эти шаги могут быть типа "map" (однотипная обработка для каждого элемента данных) или "merge" (объединение гомогенных данных).
   - Описание шаблона: выражается с помощью common workflow language (CWL) [13].
2. Формирование запроса на обработку данных:
   - Для каждого шага обработки определяются необходимые параметры для формирования задания.
3. Выполнение запроса на обработку данных:
   - Создание заданий для каждого шага обработки данных на основе сформированных параметров.
   - Отправка созданных заданий в систему управления нагрузкой (WMS) для выполнения.
   - Осуществление контроля выполнения обработки данных путем опроса системы управления нагрузкой.

Эти функциональные требования позволят системе WfMS эффективно управлять процессами обработки, обеспечивая оптимизацию обработки данных, контроль ошибок и достижение заданных целей обработки.

В основном процессе обработки данных в SPD Online Filter можно выделить следующие шаги:

1. Декодирование данных;
2. Выявление событий и реструктуризация данных;
3. Фильтрация выявленных событий;
4. Верификация данных;
5. Объединение файлов (уменьшение количества файлов, увеличение размера файлов);

Такой набор последовательных шагов обработки будет называться цепочкой обработки. Каждый из шагов в такой цепочке может быть определен шаблоном (рис. 6).

```
cwlVersion: v1.2
label: Decoding of data
class: Workflow

inputs:
  dataset_name: string
  processing_program: string
  processing_program_version: string
  cable_map: File
  input_params: File
steps:
  decoding:
    run:
      class: CommandLineTool
      baseCommand: echo

      inputs:
        dataset_name:
          type: string
        processing_program:
          type: string
        processing_program_version:
          type: string
        cable_map:
          type: File
        input_params:
          type: File
      outputs:
        output_dataset:
          type: File
        log_dataset:
          type: File

    in:
      dataset_name: dataset_name
      processing_program: processing_program
      processing_program_version: processing_program_version
      cable_map: cable_map
      input_params: input_params

    out: [output_dataset, log_dataset]

outputs:
  output_dataset:
    type: File
    outputSource: decoding/output_dataset
  log_dataset:
    type: File
    outputSource: decoding/log_dataset
```

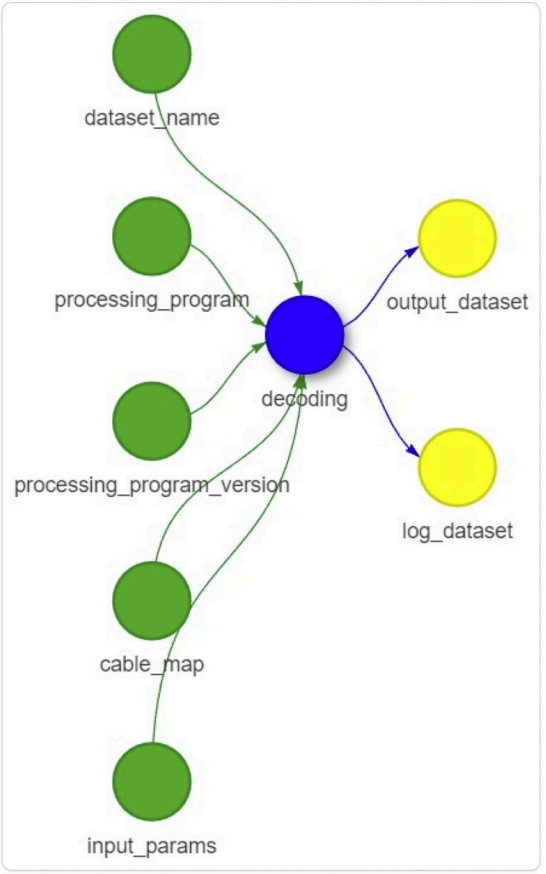



**Рис. 6** Пример CWL-шаблона

Во время работы системы предполагается создание промежуточных данных, которые будут удаляться после получения выходного набора данных.

В результате проектирования были определены основные сервисы системы WfMS:

1. Сервис для взаимодействия с оператором обработки данных:

1) авторизация пользователей;

2) обработка CWL-шаблонов;

3) вывод информации о заданиях и шаблонах;

4) работа со статусами шаблонов;

2. Сервис для опроса DMS:

1) получение информации о готовности входных датасетов;

2) формирование выходных датасетов;

3) отправление заданий на обработку;

3. Сервис генерации заданий:

1) определение последовательностей обработки данных;

2) создание заданий по шаблонам для последовательностей.

4. Сервис для работы с WMS:

1) периодический опрос для отслеживания статуса;

2) завершение заданий с последующим закрытием датасетов;

3) удаление входных и промежуточных датасетов после обработки всех заданий цепочки;

4) изменение приоритетов заданий;

5) отмена заданий.

В данной системе применяется микросервисная архитектура. В рамках этой архитектуры приложение разбивается на отдельные сервисы, которые могут быть развернуты и масштабированы независимо друг от друга. Сервисы взаимодействуют между собой через API-интерфейсы, что позволяет каждому из них функционировать автономно. В отличие от монолитной архитектуры,

микросервисы позволяют быстрее внедрять новые функции и вносить изменения, не требуя переработки большого объема существующего кода.

Архитектура WfMS приведена на рис. 7.

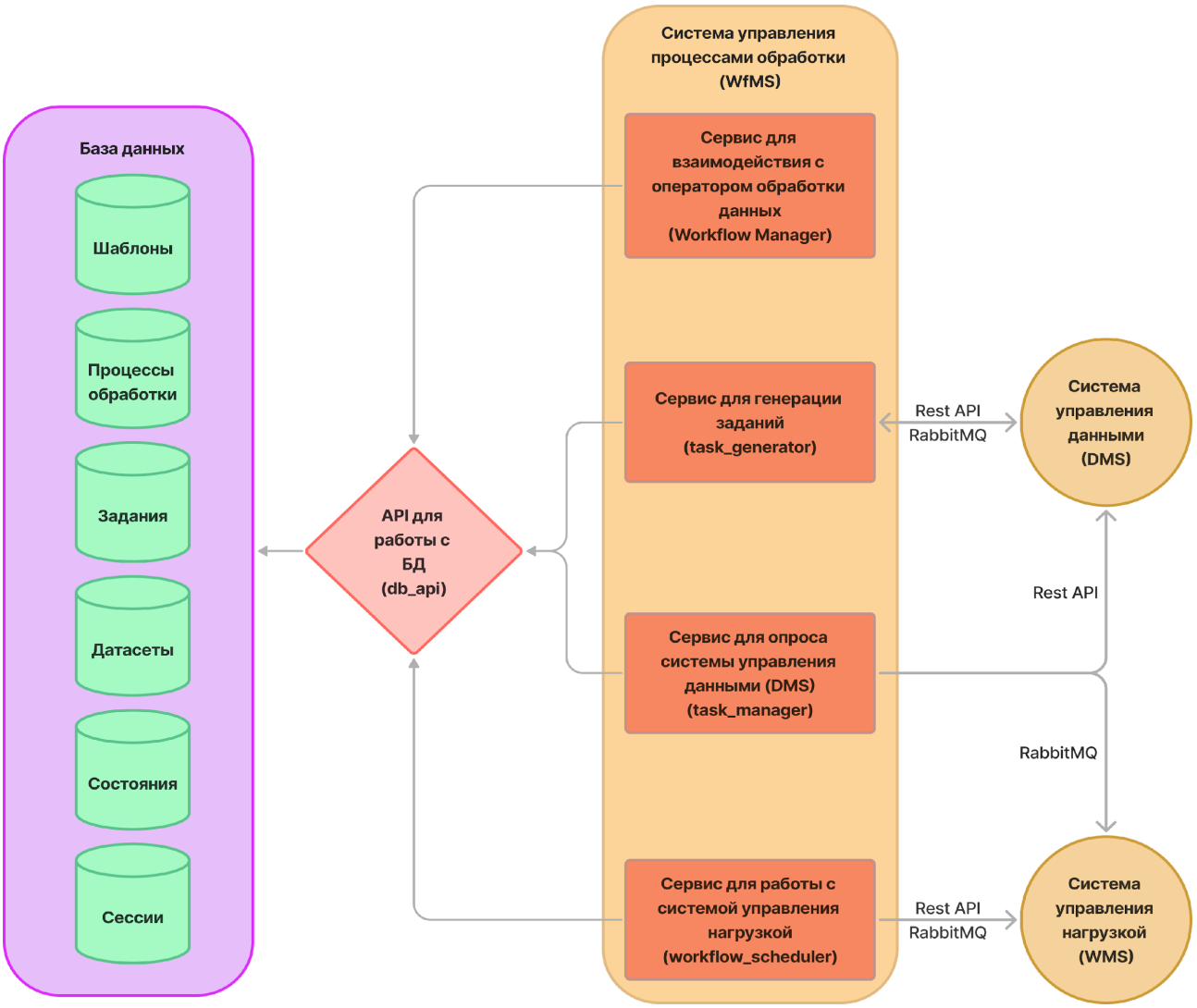



**Рис. 7** Архитектура Workflow Management System

Основные преимущества микросервисной архитектуры включают:

- Повышенную доступность и отказоустойчивость. Сбой одного из сервисов не приводит к отказу всей системы.
- Масштабируемость. При достижении предельной нагрузки на микросервис можно развернуть дополнительные экземпляры этой службы и распределить нагрузку, поддерживая таким образом работоспособность большого количества экземпляров.

## 1.3 База данных и API к ней

Для хранения данных в системе WfMS используется объектно-реляционная база данных. В качестве СУБД была выбрана PostgreSQL [14], развернутая на отдельной виртуальной машине.

Структура базы данных представлена на рис. 8.

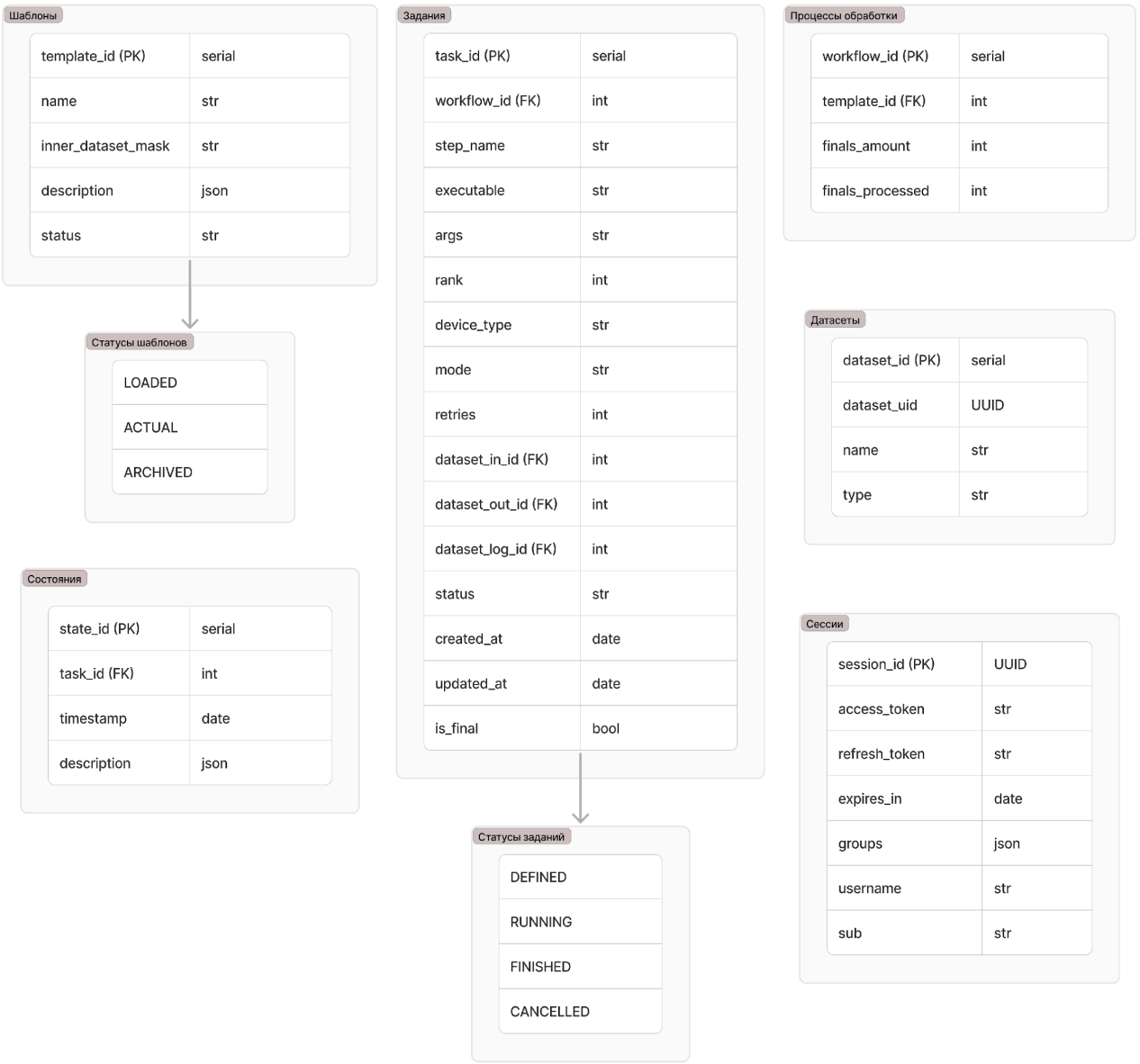



**Рис. 8** Структура базы данных

Для создания и работы с базой данных используется библиотека SQLAlchemy (ORM) [15]. Миграции проводятся с помощью библиотеки Alembic

[16]. Для достижения высокой скорости взаимодействия и асинхронного взаимодействия применяются асинхронные сессии и asyncpg [17].

Взаимодействие каждого из микросервисов с базой данных представлено на рис. 9.

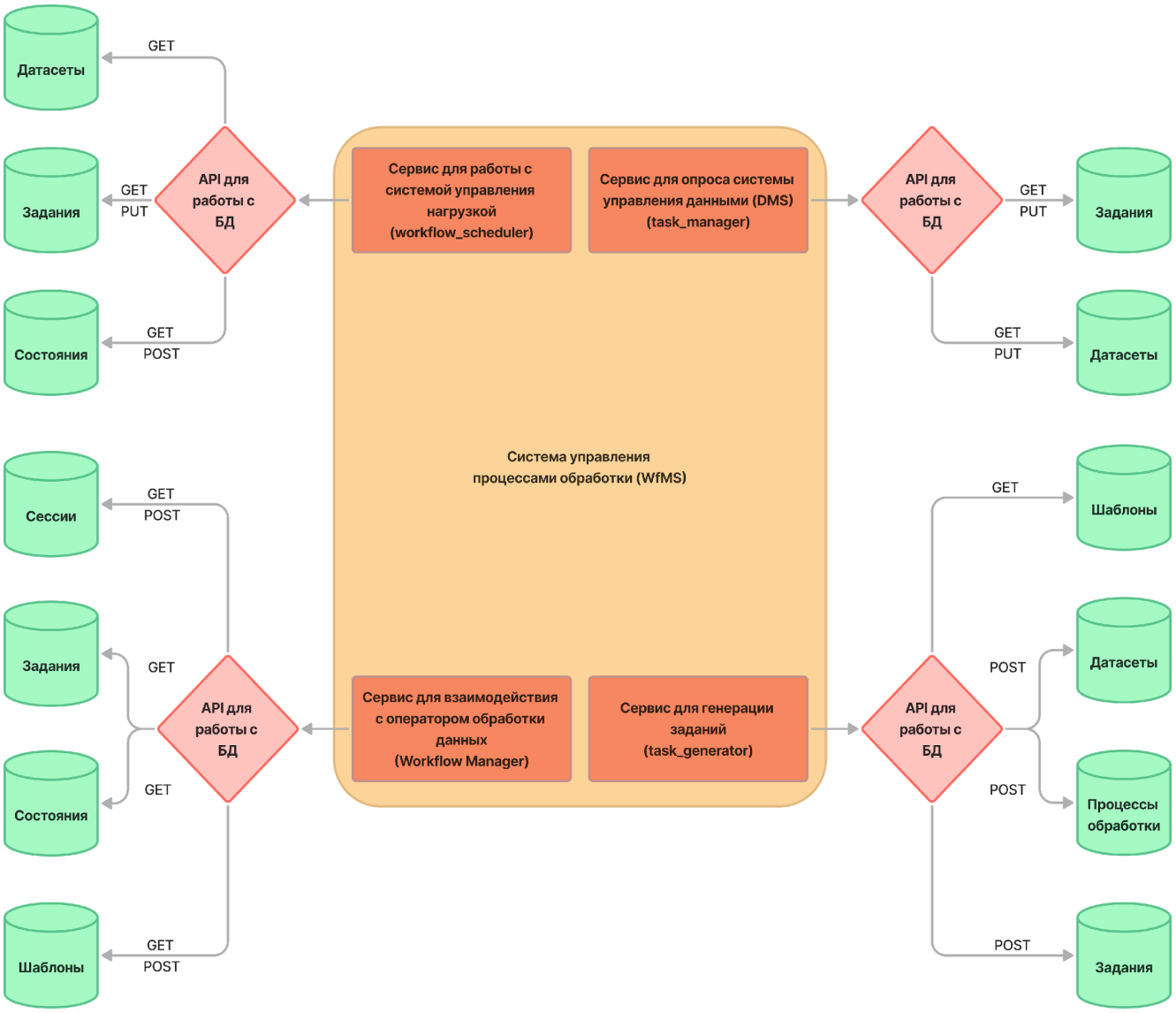



**Рис. 9** Схема взаимодействия микросервисов с БД

Всё взаимодействие с базой данных осуществляется через REST API. Для этого используется фреймворк FastAPI [18]. Вся документация и проверка API производится во встроенном инструменте FastAPI — Swagger (рис. 10).


**Рис. 10** DB API Swagger

## 1.4 Сервис для взаимодействия с оператором обработки данных (Workflow Manager)

Основной функционал сервиса:

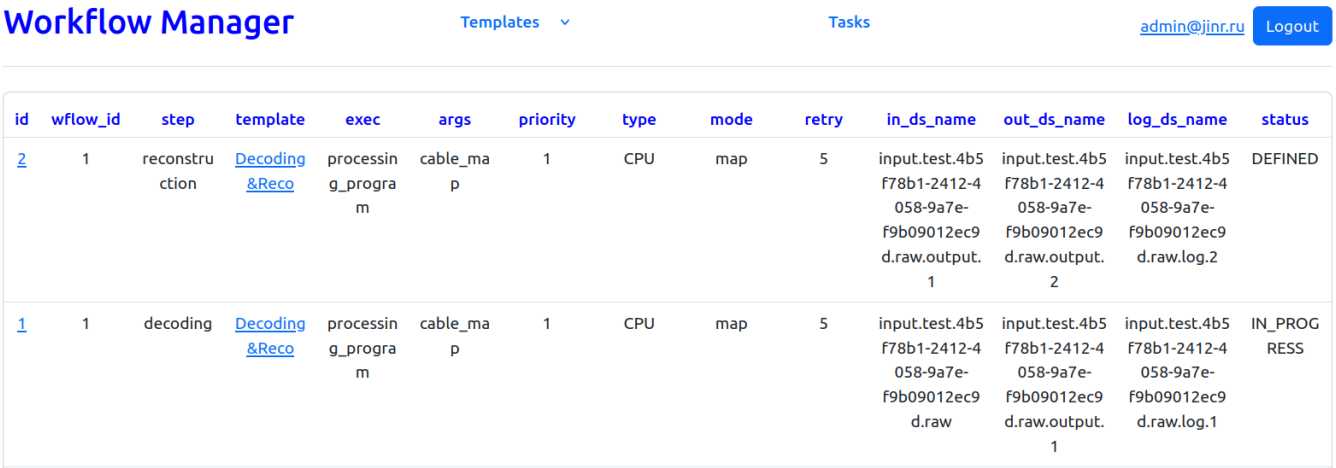
1) Регистрация и авторизация пользователей с разными правами;
2) Вывод CWL-шаблонов;
3) Вывод заданий;
4) Создание CWL-шаблонов суперпользователями;
5) Предварительная валидация и сохранение CWL-шаблонов;
6) Изменение статусов шаблонов суперпользователями на «ACTUAL» и «ARCHIVED»;
7) Удаление суперпользователями шаблонов в статусе «LOADED».

Сервис для взаимодействия с оператором обработки данных обеспечен пользовательским интерфейсом. Backend сервиса написан на FastAPI, для frontend использовался bootstrap [19] (HTML + CSS + JS), а объединяет их воедино шаблонизатор Jinja2 [20].

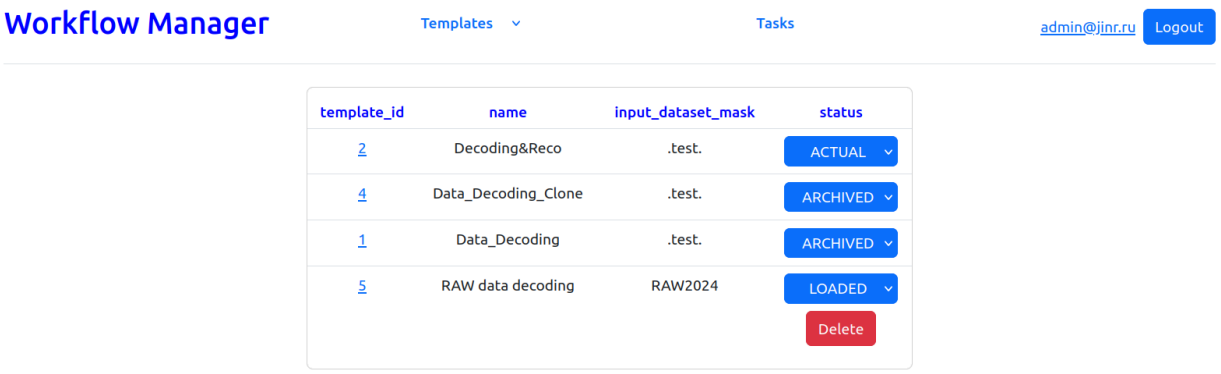
Изначально реализация была следующей: для использования сервиса необходимо было пройти процесс регистрации, а затем аутентификацию и авторизацию. Данная часть сервиса была реализована с помощью библиотеки FastAPI Users на сохранении JWT-токенов в cookie браузера. В дальнейшем было решено отказаться от внутренней регистрации и аутентификации пользователей, в угоду интеграции с новой системой SPD-IAM [21], объединяющей все приложения коллаборации SPD.

Категория пользователя определяется исходя из групп, в которых состоит пользователь в системе SPD-IAM. Суперпользователь (оператор) должен состоять в группе sof/operator.

Суперпользователям, помимо просмотра шаблонов и заданий (рис. 11), так же доступно создание шаблонов, в том числе и с помощью клонирования уже существующих шаблонов или из файла, путем его переноса в область создания, и смена их статусов: LOADED — загруженные шаблоны, которые можно удалить; ACTUAL — шаблоны, используемые при создании заданий; ARCHIVED — неактивные шаблоны. Вернуть шаблон в статус LOADED невозможно.




а)




б)

**Рис. 11** а) Страница с заданиями; б) Страница с шаблонами

Отдельно можно посмотреть и сам шаблон, суперпользователю же доступна возможность его клонирования, для создания на его основе нового экземпляра. Показано на рис. 12.

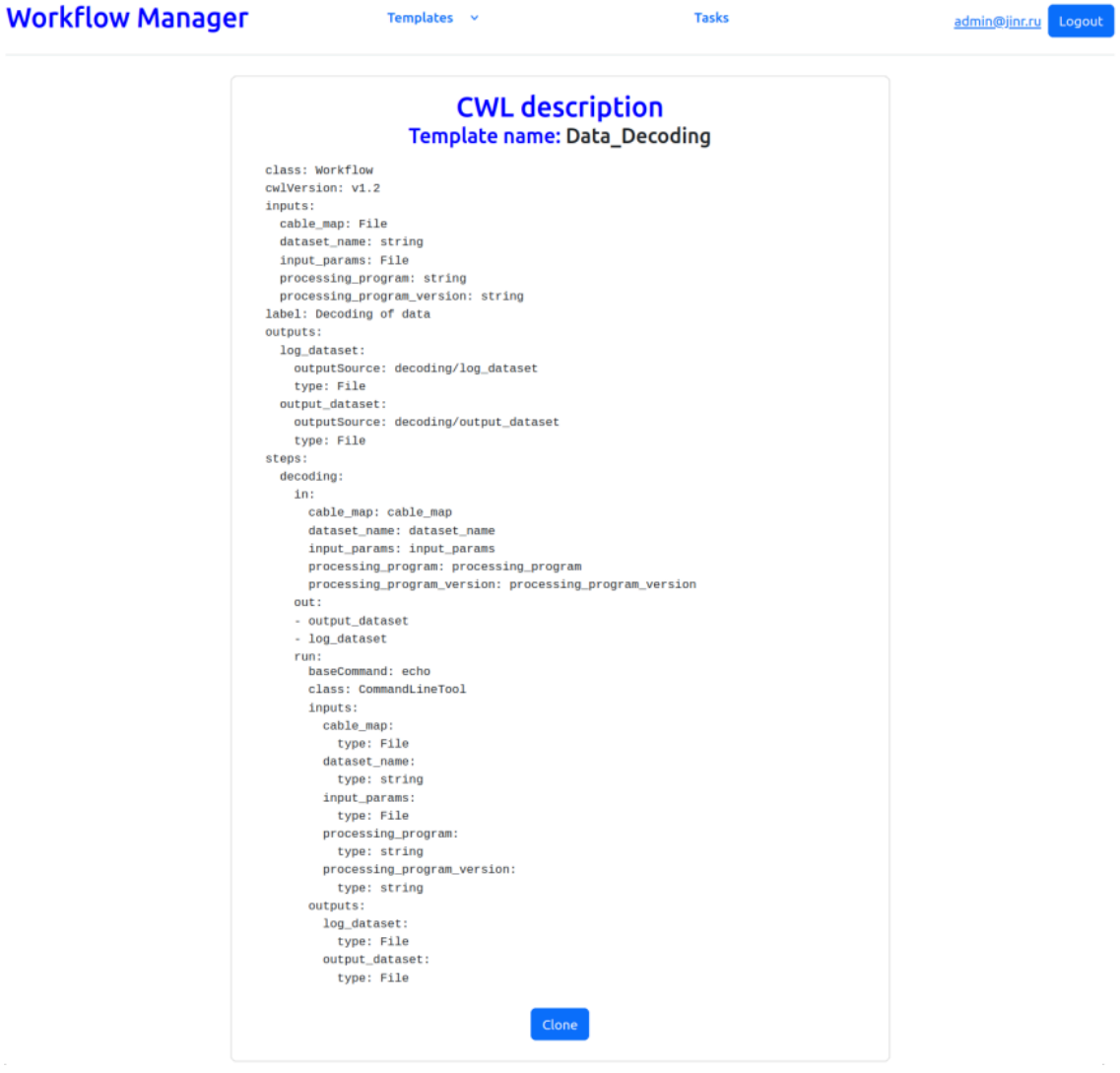



**Рис. 12** Страница с отдельным шаблоном

Шаблон задается в формате Common Workflow Language (CWL). Перед сохранением в базу данных шаблон проходит процесс валидации с помощью инструментов cwltools. Страница создания шаблона представлена на рис. 13.

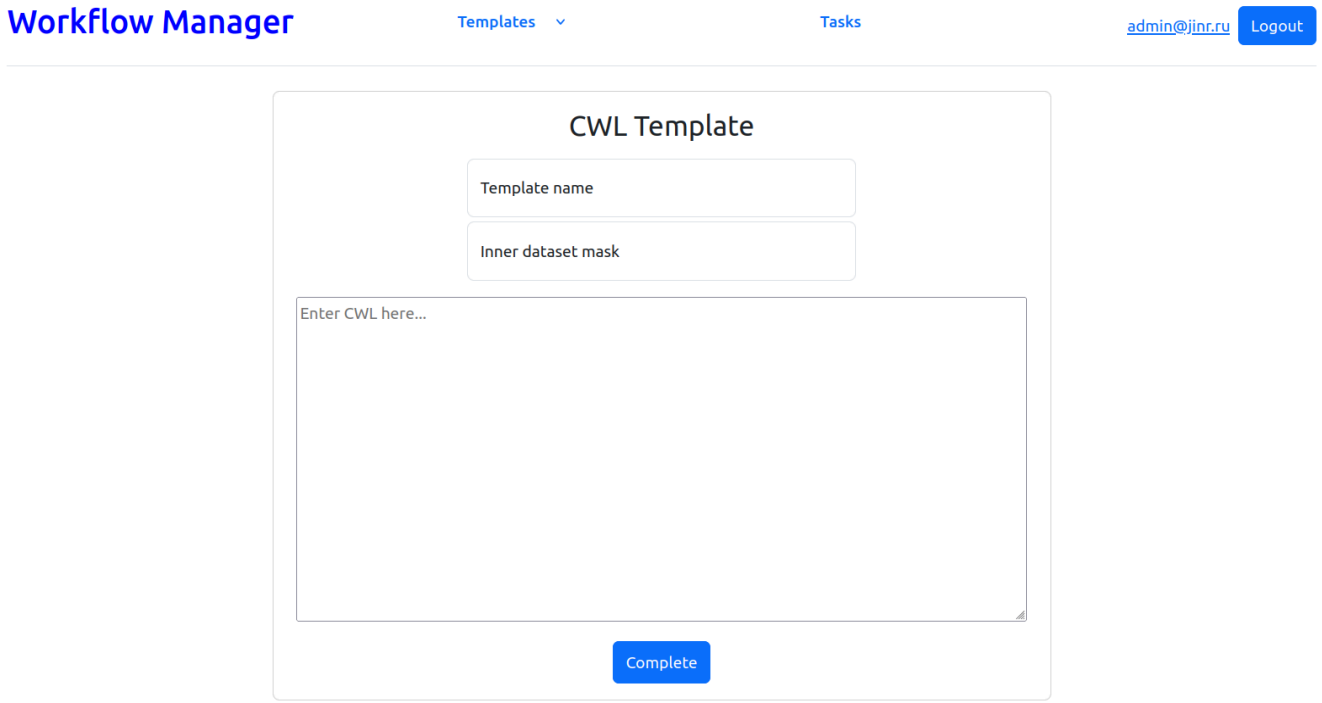



**Рис. 13** Страница создания CWL-шаблона

Для заданий доступна возможность просмотра их состояний, полученных от WMS. В частности, присутствует триггер, сохраняющий состояние при смене статуса задания. Страница с состояниями задания представлена на рис. 14.

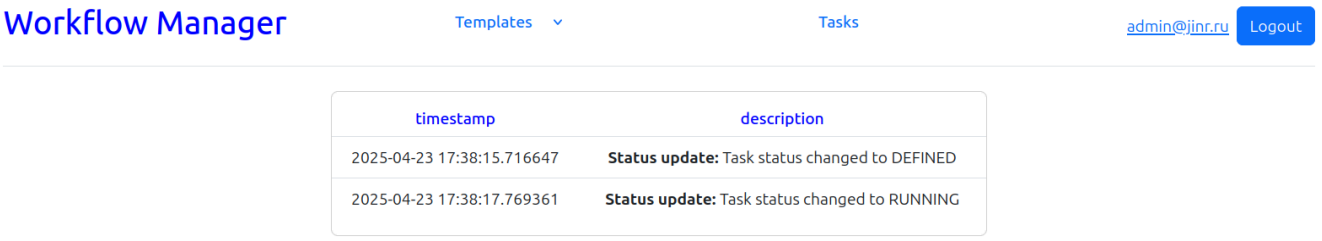



**Рис. 14** Страница с состояниями задания

## 1.5 Сервис для генерации заданий (task_generator)

Сервис для генерации заданий обладает следующим функционалом:

1) Получение зарегистрированных датасетов от DMS из брокера сообщений RabbitMQ [22];

2) Сопоставление датасета по маске имени с нужным шаблоном;

3) Регистрация входного датасета в системе;

4) Создание процесса обработки по шаблону;

5) Создание выходного датасета и датасета логов в системе;

6) Создание заданий.

После того, как датасет был сформирован, DMS отправляет информацию о нем (name, dataset_uid) в брокер сообщений RabbitMQ. Информация о датасетах извлекается по порядку из очереди и сопоставляется по маске имени с шаблонами, написанными оператором обработки в соответствующем сервисе. Используются только те шаблоны, которые находятся в статусе ACTUAL.

Входные датасеты регистрируются в системе, после чего создается процесс обработки по найденному шаблону. Генерируются записи о выходных датасетах и датасетах логов в базе данных для каждого из этапов цепочки обработки с целью дальнейшего их создания в DMS перед отправкой конкретного задания на обработку. Данная реализация позволяет не создавать лишних датасетов в системе в случаях, когда задание отменено.

Далее формируется само задание. Ему присваивается статус DEFINED. Данное задание отличается от того, что будет отправлено на дальнейшую обработку из-за особенностей хранения в базе данных. Таким образом, в этом сервисе формируется «шаблон» задания, который затем будет заполняться перед отправлением на обработку в WMS.

Все этапы генерации заданий осуществляются в асинхронном режиме.

Весь процесс наглядно можно видеть на рис. 15.

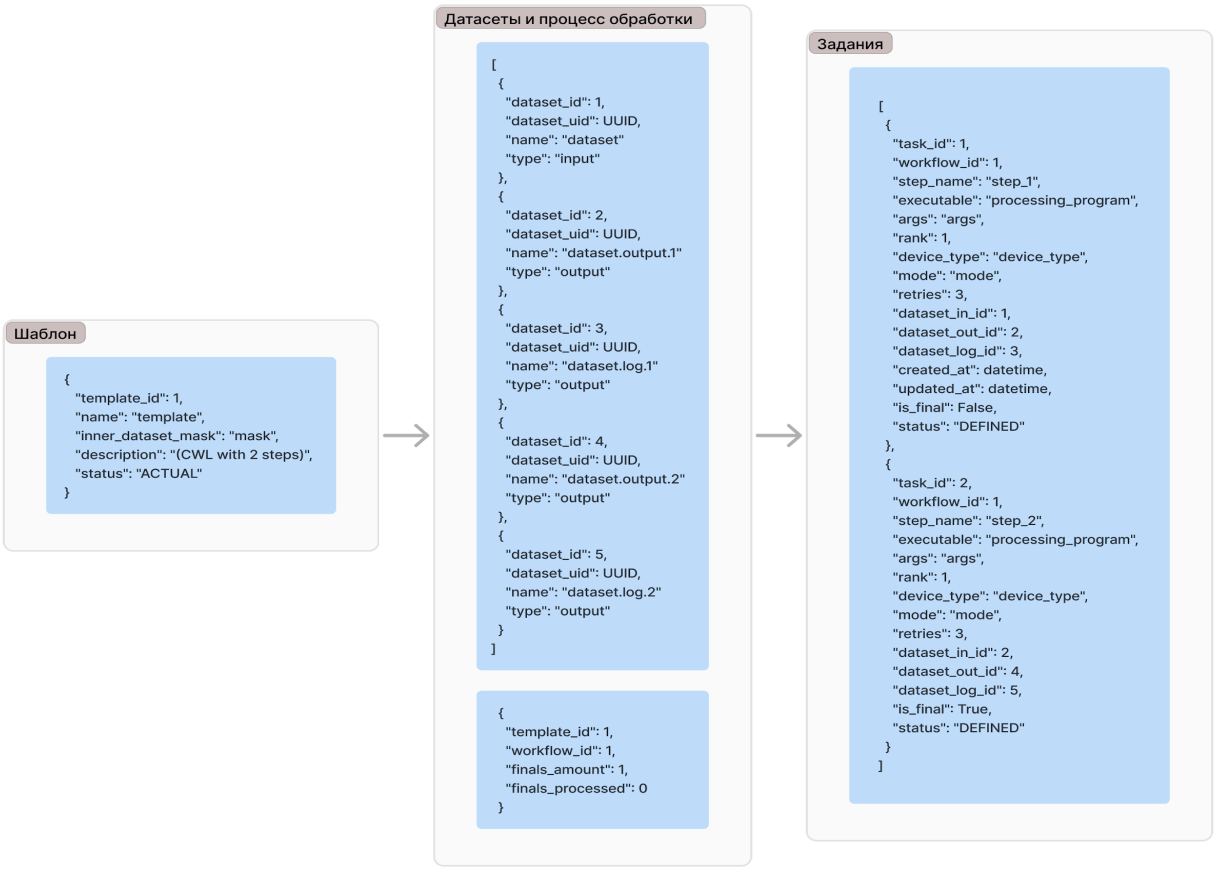



**Рис. 15** Генерация задания по шаблону

Для данного этапа разработки прототипа системы достаточно создания линейной цепочки обработки. Однако в будущем может возникнуть необходимость перехода к обработке общего случая направленного графа. Для решения этой задачи предполагается возможность создания «датасетов-клонов» — разных датасетов, составленных из одних и тех же файлов. Такое усложнение позволяет в дальнейшем легко решать проблему «жадного» удаления датасетов (удаление входного датасета сразу после прохождения этапа обработки), а так же избавляет от необходимости описывать в шаблоне сложную обработку сразу двух выходных датасетов. Подобное представление показано на рис. 16.

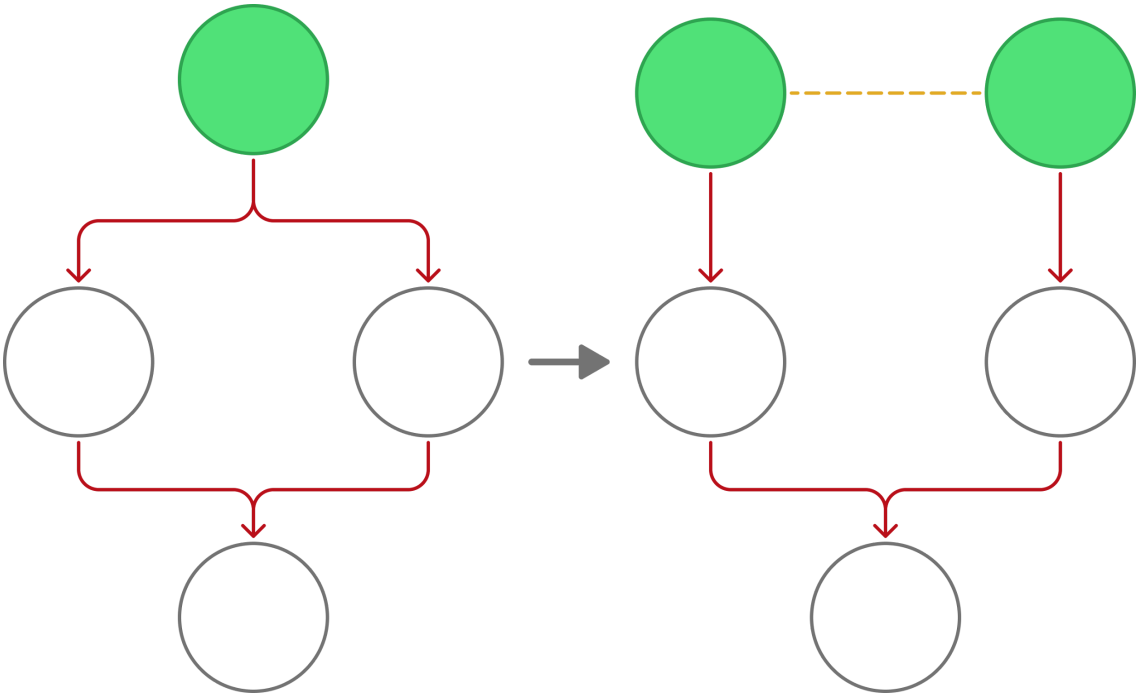


**Рис. 16** Создание «датасетов-клонов»

Это поможет преобразовать ациклический граф общего вида следующим образом (рис. 17).

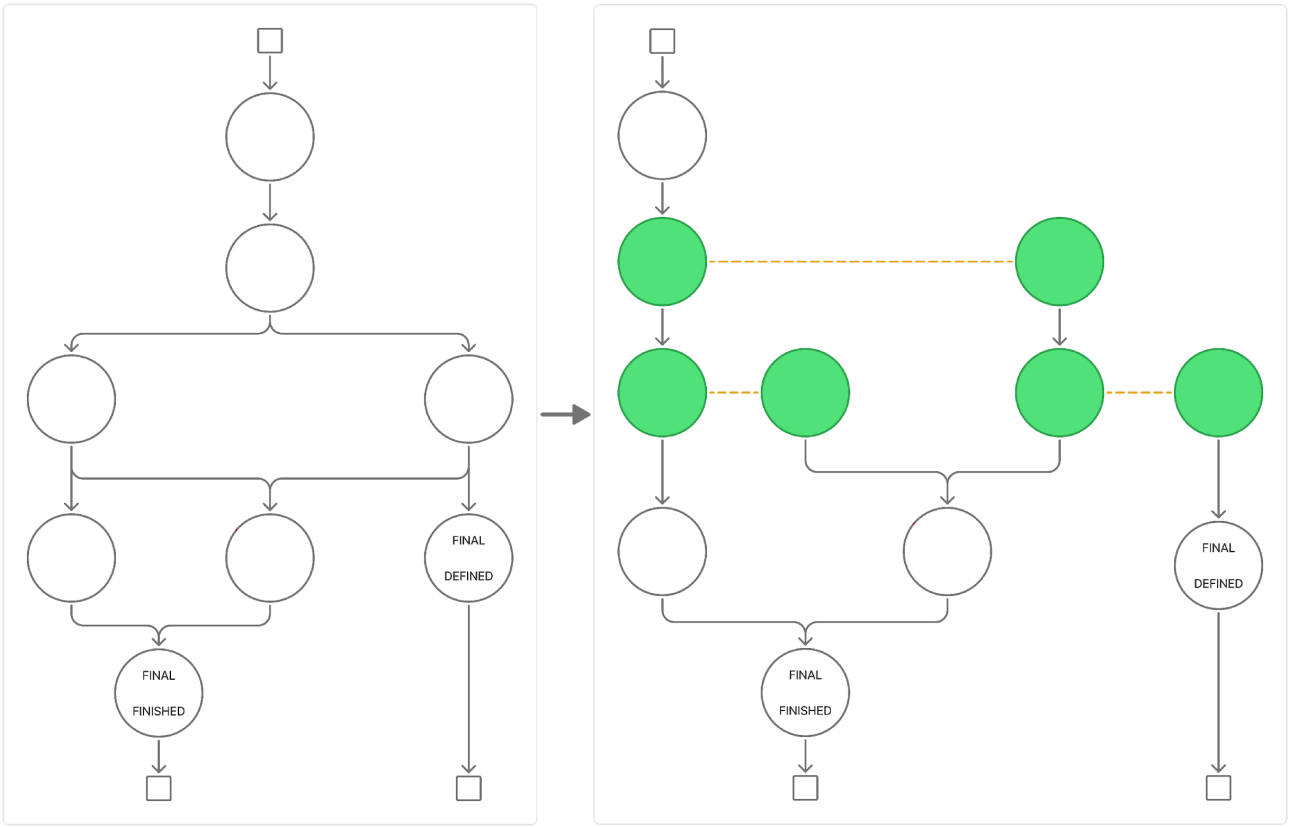



**Рис. 17** Ациклический граф общего вида после создания «датасетов-клонов»

Также в таком случае понадобится еще одна смежная таблица для хранения нескольких датасетов в одном задании (рис. 18).

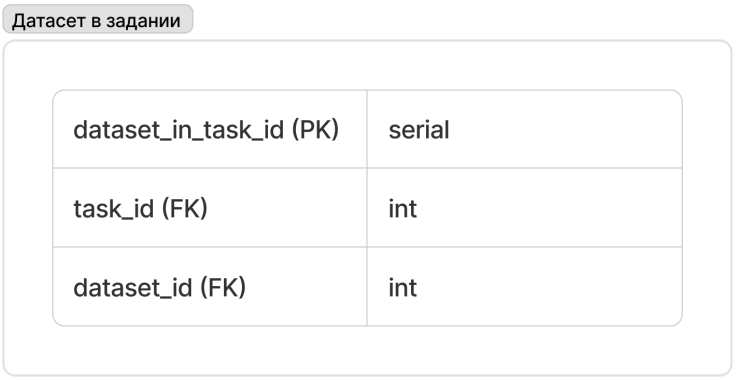



**Рис. 18** Смежная таблица для хранения нескольких датасетов в задании

## 1.6 Взаимодействие с Data Management System (DMS)

Основной задачей данного взаимодействия является получение новых данных, регистрация промежуточных данных и выгрузка выходных данных. Кроме основных функций, взаимодействие должно быть реализовано с учетом обеспечения высокой эффективности и максимальной скорости, исключения узких мест, а также минимизации возможных ошибок.

Взаимодействие с DMS включает следующие этапы:

1. Получение готового входного набора данных. После поступления данных в SPD Online Filter, DMS регистрирует их в своей системе. После этого WfMS может получить информацию об этом датасете из сообщения в брокере RabbitMQ.

2. Регистрация выходного набора данных. Перед любым этапом обработки данных WfMS должен зарегистрировать выходной набор данных, чтобы знать, где и как искать полученные данные. Для этого используется REST API. Промежуточные и выходные данные сопровождаются логами, которые регистрируются аналогичным образом.

3. Обработка промежуточных данных. В ходе обработки данных появляются промежуточные результаты. Если в цепочке обработки имеется n этапов, то при сохранении всех промежуточных данных после каждого этапа объем используемой памяти значительно увеличивается. Поэтому видится возможной реализация двух различных стратегий: «жадной» и «безопасной».

При «жадной» стратегии хранится только один набор входных данных и один набор промежуточных данных. Это значит, что после завершения этапа обработки и появления нового датасета, входные данные для этого этапа удаляются. Входные данные для первого этапа остаются до завершения всей цепочки обработки, чтобы можно было восстановить данные в случае сбоя системы.

При «безопасной» стратегии все данные хранятся до полного прохождения всей цепочки обработки, после чего входные датасеты каждого этапа могут быть безопасно удалены. Плюсом такой стратегии является то, что в любой момент можно вернуться к этапу, где всё пошло не так, как предполагалось.

На сегодняшний момент решено не удалять датасеты логов для удобства отлаживания, однако при переходе от прототипа к финальной версии видится хорошим решением внедрение отдельной переменной окружения, указывающей на то, надо ли выгружать датасеты логов или же стоит их удалять.

В данном взаимодействии используется асинхронный метод, позволяющий не останавливать работу WfMS и не ждать удаления данных, так как оно производится через сообщения в RabbitMQ.

4. Опрос о заполненности выходного хранилища. Периодически DMS необходимо опрашивать о заполненности выходного хранилища, исходя из этого система в дальнейшем будет принимать решения об отмене заданий и изменении их приоритетов. Критический предел заполненности выходного хранилища должен будет задаваться через переменную окружения.

### 1.6.1 Сервис для опроса DMS (task_manager)

После того, как задания были сгенерированы в соответствующем сервисе они получают статус DEFINED. Данный сервис извлекает такие задания из базы данных и итерируется по ним. С помощью REST API в DMS (dsm-manager) отправляется запрос на получение статуса входного датасета для конкретного задания. Данный датасет должен быть готов для обработки (находиться в статусе «CLOSED»).

Для входных датасетов в статусе «CLOSED» создаются выходные датасеты и датасеты логов в DMS через REST API (dsm-manager) при этом вся информация о них передаётся из внутренней базы данных WfMS, где они были созданы на этапе генерации заданий.

Задание формируется в необходимом для дальнейшей обработки виде и отправляется в очередь брокера сообщений RabbitMQ в формате JSON, после чего оно может быть извлечено WMS. Данная очередь создается в момент инициализации сервиса. Сохранность сообщений в очереди и их доставку до получателя обеспечивает RabbitMQ.

После того, как сообщение было отправлено, задание получает статус «RUNNING» (рис. 19), что обеспечит возможность их отслеживания и управления ими в дальнейшем.

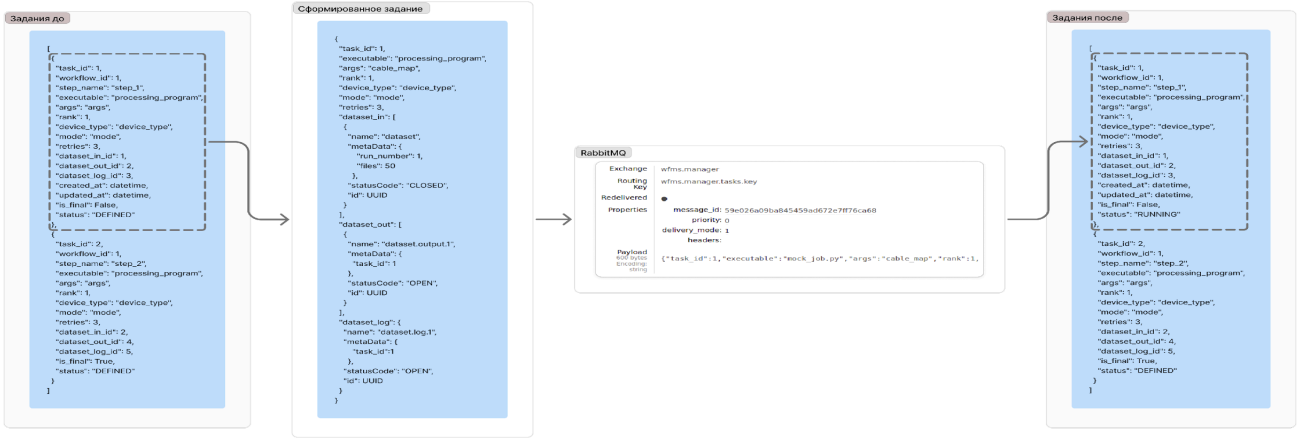



**Рис. 19** Отправление заданий на обработку

Благодаря данной реализации, следующее задание в цепочке будет отправлено на обработку сразу после того, как завершится предыдущее.

## 1.7 Взаимодействие с Workload Management System (WMS)

Основной задачей данного взаимодействия является отправление заданий на обработку и контроль их выполнения. Взаимодействие следует выстроить так, чтобы оно обеспечивало оперативную обработку, устойчивость к сбоям и отсутствие узких мест, препятствующих эффективности системы.

Взаимодействие с WMS включает следующие этапы:

1. Отправление заданий на обработку. Сформированные задания отправляются в очередь RabbitMQ, откуда они забираются одним из сервисов WMS для последующего деления на задачи и их обработки.
2. Опрос о состоянии задания. Периодически WMS опрашивается по REST о состоянии конкретного задания. В ответ приходят метаданные о количестве обработанных файлов относительно всех файлов в датасете и аналогично для задач данного задания, а также текущий статус задания.
3. При достижении критического значения заполненности выходного хранилища будет приниматься решение об отмене заданий или изменении их приоритетов с последующим оповещением WMS. Система принятия решений представляется довольно трудоемкой и громоздкой, поэтому на данном этапе прототипирования было решено её отложить.
4. Приоритеты заданий могут быть изменены и по иным причинам, чем только по заполненности выходного хранилища, предполагается удобным использование следующей функции:

$$rank_{i+1} = \alpha \times x_i + \beta \times y_i + \gamma \times rank_i,$$

$x_i$ – aging, $y_i$ – retries.

### 1.7.1 Сервис для работы с WMS (workflow_scheduler)

Данный сервис извлекает из базы данных задания в статусе «RUNNING». После чего итерируется по ним, опрашивая систему управления нагрузкой WMS об их состоянии. Эти состояния записываются в базу данных для удобного мониторинга оператором обработки и дальнейшего принятия решений.

Если при опросе задание оказывается завершенным (находится в статусе finished в WMS), то выходной датасет данного задания закрывается путем отправления PATCH-запроса в DMS.

После этого задание получает статус «FINISHED» в WfMS. На изменение статуса срабатывает триггер в базе данных, который записывает изменение статуса в состояние.

Далее происходит удаление датасетов по одной из двух стратегий, описанных выше. На данный момент решено реализовать «безопасную» стратегию, так как она удовлетворяет всем необходимым потребностям. В этом случае каждый раз после обработки задания проверяется, является ли оно последним. Если является, то finals_processed (количество обработанных последних заданий) увеличивается на 1. Когда это число становится равным finals_amount (количество последних заданий), происходит удаление всех входных датасетов заданий данной цепочки.

В дальнейшем же переключение между стратегиями будет реализовано через задание переменной окружения.

С отменой задания дела обстоят куда сложнее. Она может быть инициирована как WfMS на основе данных о заполненности выходного хранилища с последующим оповещением системы обработки данных, так и самой WMS. В таких случаях после закрытия датасетов в DMS задания будут получать статус «CANCELLED». В дальнейшем такие задания предполагается обрабатывать по сложной статусной модели.

В будущем данный сервис будет ответственен за изменение приоритетов заданий на основе данных из состояний. Измененный приоритет будет отправляться в WMS с помощью REST.

## 1.8 Логирование, контейнеризация и оркестрация

Для получения необходимой информации о состоянии системы был разработан logger, записывающий необходимую для дальнейшего анализа информацию о работе каждого сервиса. В частности, записываются название сервиса, уровень сообщения (INFO, WARNING, ERROR, CRITICAL), дата и время получения сообщения и описание сообщения.

Одно из сообщений logger можно видеть на рис. 20.

```
task_generator 2024-09-24 20:59:44,380 INFO Logger started
task_generator 2024-09-24 20:59:44,665 INFO Templates successfully initialised.
```

**Рис. 20** Сообщение logger

Для быстрого развертывания приложений на любой системе и налаживания их совместного взаимодействия используются методы контейнеризации и оркестрации с помощью Docker и Docker-Compose [23]. Представление реализованных сервисов в docker compose изображено на рис. 21.

docker-compose

```
services:
  db_api:
    ...
  task_generator:
    ...
  task_manager:
    ...
  workflow_manager
    ...
  workflow_scheduler
    ...

networks:
  wfms:
```

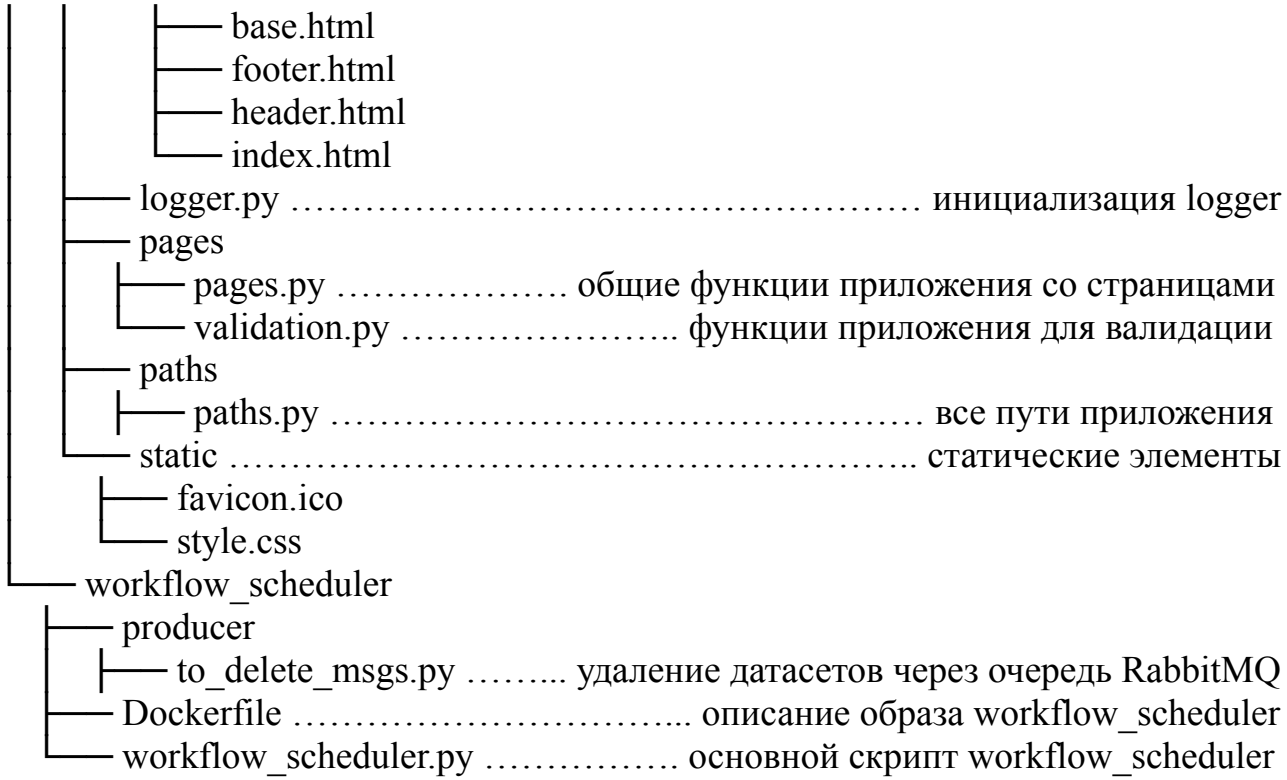
**Рис. 21** Сервисы WfMS в docker-compose

## 1.9 Общая структура системы

Структура кода системы организована согласно общепринятым практикам при разработке на языке Python.

```
wfms-spd-online
├── alembic.ini …… конфигурационный файл для alembic
├── config.py …… файл, подгружающий переменные окружения
├── database …….. модуль по взаимодействию с базой данных
│   ├── api.py …….. инициализация FastAPI
│   ├── db
│   │   ├── engine.py …….. подключение к БД и выдача асинхронных сессий
│   │   └── models.py …… ORM модели SQLAlchemy
│   ├── db_api
│   │   ├── router_functions
│   │   │   └── common_functions.py …….. функции для взаимодействия с БД
│   │   └── routers ……. функции для взаимодействия с отдельными таблицами
│   │       ├── dataset_router.py
│   │       ├── state_router.py
│   │       ├── task_router.py
│   │       ├── template_router.py
│   │       ├── user_session_router.py
│   │       └── workflow_router.py
│   ├── Dockerfile ……... описание образа db_api
│   ├── migrations …… миграции alembic
│   │   ├── env.py
│   │   ├── README
│   │   ├── script.py.mako
│   │   └── versions …….. все миграции
│   └── scripts
│       └── app.sh ……..…… bash-скрипт для запуска web-сервера
├── docker-compose.yml …….. конфигурационный файл docker compose
├── .env ……... переменные окружения
├── .env.example …….. пример файла с переменными окружения
├── .gitignore
├── logger
│   ├── custom_logger.py …….. преднастроенный logger
│   └── logs …… все логи системы
└── paths ……... все внешние пути
    ├── db_paths.py
    ├── dms_paths.py
    └── iam_paths.py
```

```
    └── wms_paths.py
├── README.md ……... описание системы
├── response_models
│   ├── model_enums
│   │   ├── enums.py …… все перечисляемые типы
│   ├── models …….. pydantic [24] модели
│   │   ├── base_models.py
│   │   ├── dataset_models.py
│   │   ├── state_models.py
│   │   ├── task_models.py
│   │   ├── template_models.py
│   │   ├── user_session_model.py
│   │   └── workflow_models.py
│   ├── pk_types
│   │   ├── models_pk_types.py ……... типы первичных ключей
├── task_generator
│   ├── Dockerfile …….. описание образа task_generator
│   ├── rabbit_receiver
│   │   ├── consumer.py …….. получение сообщений из очереди RabbitMQ
│   └── task_generator.py …….. основной скрипт task_generator
├── task_manager
│   ├── app
│   │   ├── producer.py …….. отправление сообщений в очередь RabbitMQ
│   ├── Dockerfile ……... описание образа task_manager
│   └── task_manager.py …… основной скрипт task_ manager
├── workflow_manager
│   ├── app.py ……. инициализация FastAPI
│   ├── auth
│   │   ├── auth_middleware.py ……... промежуточное ПО для авторизации
│   │   ├── current_user.py ……. работа с пользователем
│   │   ├── session_service.py …….. работа с сессиями пользователя
│   │   └── token_manager.py ……. взаимодействие с SPD-IAM
│   ├── Dockerfile …….. описание образа workflow_manager
│   ├── html_templates
│   │   ├── app …… страницы приложения, доступные после авторизации
│   │   │   ├── all_tasks.html
│   │   │   ├── all_templates.html
│   │   │   ├── create_template.html
│   │   │   ├── state.html
│   │   │   └── template_description.html
│   │   └── landing …. базовые элементы страницы и общедоступные страницы
│   │       ├── about.html
│   │       ├── authed_header.html
```




## 1.10 Тестирование системы

Логи task_generator после генерации датасета и отправления его в очередь:

```
task_generator 2025-04-25 16:24:36,562 INFO Task generator initialized
task_generator 2025-04-25 16:24:37,745 INFO Templates successfully initialized.
task_generator 2025-04-25 16:24:37,745 INFO Creating RabbitMQ connection...
task_generator 2025-04-25 16:24:37,767 INFO RabbitMQ connection is open and
active.
task_generator 2025-04-25 16:24:37,767 INFO Creating RabbitMQ channel...
task_generator 2025-04-25 16:24:37,777 INFO RabbitMQ channel created.
task_generator 2025-04-25 16:24:37,780 INFO Queue 'dsm.register.dataset.input'
successfully initialized.
task_generator 2025-04-25 16:40:42,335 INFO Input dataset received.
task_generator 2025-04-25 16:40:43,211 INFO Templates successfully initialized.
task_generator 2025-04-25 16:40:43,211 INFO Reusing existing RabbitMQ connection.
task_generator 2025-04-25 16:40:44,279 INFO Input dataset with
name=input.test.c238ef6e-adac-49ae-b8d8-67cb33da0d42.raw created in database.
task_generator 2025-04-25 16:40:45,118 INFO Workflow with workflow_id=36 created
in database.
task_generator 2025-04-25 16:40:45,579 INFO Output dataset with
name=input.test.c238ef6e-adac-49ae-b8d8-67cb33da0d42.raw.output.1 created in
database.
task_generator 2025-04-25 16:40:45,864 INFO Log dataset with
name=input.test.c238ef6e-adac-49ae-b8d8-67cb33da0d42.raw.log.1 created in
database.
task_generator 2025-04-25 16:40:47,923 INFO Task with task_id=38 created in
```

```
database.
task_generator 2025-04-25 16:40:48,094 INFO Output dataset with
name=input.test.c238ef6e-adac-49ae-b8d8-67cb33da0d42.raw.output.2 created in
database.
task_generator 2025-04-25 16:40:48,264 INFO Log dataset with
name=input.test.c238ef6e-adac-49ae-b8d8-67cb33da0d42.raw.log.2 created in
database.
task_generator 2025-04-25 16:40:48,532 INFO Task with task_id=39 created in
database.
```

Логи task_manager:

```
task_manager 2025-04-25 16:24:36,505 INFO Task manager initialized
task_manager 2025-04-25 16:24:36,505 INFO Creating RabbitMQ connection...
task_manager 2025-04-25 16:24:36,570 INFO RabbitMQ connection created
task_manager 2025-04-25 16:40:48,949 INFO UID of dataset with ID=45 updated.
task_manager 2025-04-25 16:40:49,190 INFO UID of dataset with ID=46 updated.
task_manager 2025-04-25 16:40:49,202 INFO Task with id = 38 sent to RabbitMQ.
task_manager 2025-04-25 16:40:49,352 INFO Status of task with id = 38 updated to
RUNNING.
```

Созданные задания в workflow_manager (рис. 22)

| id | wflow_id | step | template | exec | args | priority | type | mode | retries | in_ds_name | out_ds_name | log_ds_name | status |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 38 | 36 | decoding | d&r | processing_program | cable_map | 1 | CPU | Map | 2 | input.test.c238ef6e-adac-49ae-b8d8-67cb33da0d42.raw | input.test.c238ef6e-adac-49ae-b8d8-67cb33da0d42.raw.output.1 | input.test.c238ef6e-adac-49ae-b8d8-67cb33da0d42.raw.log.1 | RUNNING |
| 39 | 36 | reco | d&r | processing_program | cable_map | 1 | CPU | Map | 2 | input.test.c238ef6e-adac-49ae-b8d8-67cb33da0d42.raw.output.1 | input.test.c238ef6e-adac-49ae-b8d8-67cb33da0d42.raw.output.2 | input.test.c238ef6e-adac-49ae-b8d8-67cb33da0d42.raw.log.2 | DEFINED |

**Рис. 22** Вид созданных заданий по полученному датасету в workflow_manager

Состояния заданий в workflow_manager (рис. 23)

| timestamp | description |
|---|---|
| 2025-04-25 16:40:45.877739 | **Status update:** Task status changed to DEFINED |
| 2025-04-25 16:40:49.207776 | **Status update:** Task status changed to RUNNING |

| timestamp | description |
|---|---|
| 2025-04-25 16:40:48.270340 | **Status update:** Task status changed to DEFINED |

**Рис. 23** Вид состояний заданий  в workflow_manager

Сообщение с заданием в статусе «RUNNING» в очереди RabbitMQ (рис. 24).


**Рис. 24** Сообщение в очереди RabbitMQ

# Результаты работы и дальнейшие планы

**Результаты:**

1) разработан API для работы с базой данных, открывающий доступ для сервисов WfMS к определенному набору функций.

2) создан сервис для работы с оператором обработки данных, позволяющий просматривать задания и шаблоны, а также предоставляющий возможность суперпользователям генерировать шаблоны и управлять их статусами.

3) разработан сервис для генерации заданий, сопоставляющий датасеты с шаблонами по маске имени, создающий цепочки обработки и создающий задания.

4) создан сервис для опроса DMS, опрашивающий его о готовности входного датасета для каждого готового задания, создающий выходные датасеты для таких заданий и отправляющий задания на обработку в WMS.

5) произведена контейнеризация всех приложений и настроена их оркестрация с помощью docker-compose.

6) внедрен logger, сохраняющий информацию об актуальном состоянии системы в каждый момент времени.

7) разработан первый этап сервиса для взаимодействия с WMS, позволяющий завершать задания с закрытием выходных датасетов и удалять датасеты после прохождения всех этапов цепочки обработки.

**Дальнейшие планы:**

1) Последующая доработка сервиса для взаимодействия с WMS;

2) Переход к полной асинхронности;

3) Тестирование системы.

# Заключение

Современные эксперименты в области физики высоких энергий, сталкиваются с необходимостью эффективной обработки огромных объемов данных, генерируемых детекторными системами. Эксперимент SPD на коллайдере NICA предъявляет высокие требования к системе фильтрации данных в реальном времени, учитывая интенсивность событий и необходимость оперативной обработки информации. В этом контексте проектирование системы управления процессами обработки данных становится не просто задачей, а ключевым фактором для достижения успешных результатов в исследованиях физики высоких энергий. Эффективное управление процессами обработки данных в вычислительных комплексах имеет ключевое значение для обеспечения производительности и надежности системы.

В ходе работы были рассмотрены основные аспекты функционирования системы SPD Online Filter эксперимента SPD и разработан прототип её подсистемы — системы управления процессами обработки.

# Список использованных источников


1. International spin physics collaboration at the collider NICA [Электронный ресурс] // https://spd.jinr.ru
2. Abazov V. M. [и др.] Conceptual design of the Spin Physics Detector. 2022. arXiv: 2102.00442 [hep-ex]
3. Комплекс NICA. — URL: https://nica.jinr.ru/ru/complex.php
4. Arbuzov A. [и др.]. On the physics potential to study the gluon content of proton and deuteron at NICA SPD // Prog. Part. Nucl. Phys. 2021. arXiv: 2011.15005 [hep-ex].
5. Экспериментальная установка SPD. — URL: https://nica.jinr.ru/ru/projects/spd.php.
6. Oleynik D. Data processing in HEP experiments. [Электронный ресурс] // https://lit.jinr.ru/sites/lit.jinr.ru/files/pdf/HEPNICA_computing_2022_OleynikD.pdf
7. Abazov V. M. [и др.] Technical Design Report of the Spin Physics Detector at NICA. 2024.arXiv:2404.08317v1 [hep-ex]
8. Пономарев Е. В. Проектирование сервиса управления процессом обработки данных на специализированной распределенной вычислительной системе SPD Online filter. СПб — 2022
9. DIRAC [Электронный ресурс] // https://dirac.readthedocs.io/en/latest/DeveloperGuide/Overview/
10. PanDA [Электронный ресурс] // https://github.com/PanDAWMS/pandawms

11. Apache AirFlow [Электронный ресурс] // https://airflow.apache.org/
12. Основные сущности AirFlow [Электронный ресурс] // https://cloud.vk.com/blog/airflow-what-it-is-how-it-works/
13. Common Workflow Language [Электронный ресурс] // https://www.commonwl.org/
14. PostgreSQL [Электронный ресурс] // https://www.postgresql.org/
15. SQL Alchemy 2.0 [Электронный ресурс] // https://docs.sqlalchemy.org/en/20/
16. Alembic [Электронный ресурс] // https://alembic.sqlalchemy.org/en/latest/
17. Asyncpg [Электронный ресурс] // https://pypi.org/project/asyncpg/
18. FastAPI [Электронный ресурс] // https://fastapi.tiangolo.com/
19. Bootstrap [Электронный ресурс] // https://getbootstrap.com/
20. Jinja [Электронный ресурс] // https://jinja.palletsprojects.com/en/stable/
21. SPD-IAM [Электронный ресурс] // https://spd-iam.jinr.ru
22. RabbitMQ [Электронный ресурс] // https://www.rabbitmq.com/
23. Docker-compose [Электронный ресурс] // https://docs.docker.com/compose/
24. Pydantic [Электронный ресурс] // https://docs.pydantic.dev/latest/