Разработка архитектуры информационно-аналитической системы обработки изображений

Д.И. Сигалов, В.А. Сычев

***Аннотация*—В работе предлагается архитектура информационно-аналитической системы для организации воспроизводимых исследований в области компьютерного зрения. Система обеспечивает унифицированный сбор и хранение визуальных данных из разнородных источников, с возможностью автоматического выполнения аналитических задач. Их обработка осуществляется через интеграцию с произвольными алгоритмами – как классическими методами анализа изображений, так и нейросетевыми моделями в рамках единого интерфейса выполнения. Платформа реализует механизм динамического управления контекстом экспериментов и предоставляет встроенные средства для сравнительного анализа результатов различных алгоритмов. Разработанное решение формализует полный исследовательский цикл, снижая рутинную нагрузку и обеспечивая полную воспроизводимость при работе с визуальными данными.**

***Ключевые слова*—REST API, OAUTH2.0, HashiCorp, RTSP, PostgreSQL, Loki, S3 MINIO, IP, Jaeger.**

I. Введение

Бурный рост объемов визуальных данных из разнообразных источников: от IP-камер и медицинских сканеров до спутниковых снимков, стимулирует развитие методов компьютерного зрения. Однако, если производственные системы видеоаналитики достигли высокой степени зрелости, то инструментальная поддержка исследовательской деятельности в этой области остается фрагментированной. Важной проблемой при сравнительном анализе алгоритмов обработки изображений является обеспечение воспроизводимости экспериментов. Под воспроизводимостью понимается возможность получения идентичных или статистически схожих результатов другими исследователями на том же наборе данных и при одинаковых начальных условиях. Однако проведение таких экспериментов на временных рядах данных с динамически меняющимся контекстом (например, видеопотоки, спутниковые снимки за разные сезоны) сопряжено со значительными ручными усилиями по настройке и слабо формализовано, что снижает надежность и достоверность получаемых выводов. Существующие платформы, будь то системы управления видеопотоками, алгоритмы анализа изображения или видео, решают лишь отдельные аспекты этой проблемы. Они не обеспечивают целостной методологии для сквозного автоматизированного цикла, который объединил бы сбор данных из гетерогенных источников, воспроизводимое выполнение как классических, так и нейросетевых алгоритмов, динамическое управление контекстом и встроенный сравнительный анализ результатов. Для заполнения этой ниши в данной работе предлагается архитектура информационно-аналитической системы, ориентированной на организацию воспроизводимых исследований в компьютерном зрении.

II. Анализ требований к системе

II.1. Сценарии использования формирующие требования к системе

A. Сценарный эксперемент по расписанию

Описание: пользователь ожидает чтобы изображения из источника (например IP- камера) каждые полчаса забирала изображение с камеры, сохраняла изображение в хранилище и собирала метаинформацию с каждого изображения (ширина, высота, размер).

Потребность: сбор информации с источников, выполнение задач по расписанию.

B. Сравнительный анализ алгоритмов на исторических данных

Описание: данный сценарий позволяет провести формализованное сравнение произвольного набора алгоритмов на пользовательской выборке изображений. Система автоматически управляет выполнением всех необходимых запусков, обеспечивает корректность вычислении и агрегирует результаты.

Потребность: управление большими наборами данных, параллельный запуск нескольких обработчиков, агрегация и сравнение разноформатных результатов.

C. Figures Electronic Image Files

Описание: ключевой класс задач предполагает, что условия выполнения эксперимента изменяются со временем или зависят от его предыдущих итераций. Пользователь должен иметь возможность формально описать правило обновления критических параметров. Система автоматически применяет это правило при каждом запуске, обеспечивая актуальность контекста

Статья получена 21 января 2026

Сигалов Давид Игоревич, Национальный исследовательский ядерный университет МИФИ, аспирант, (e-mail: duff1k@inbox.ru).

Сычев Вячеслав Александрович, Национальный исследовательский ядерный университет МИФИ, Старший преподаватель, (e-mail: ssytchov@mai.ru).

обработки.

Потребность системы: механизм динамического управления входными параметрами и данными на основе правил, заданных пользовательским паттерном.

II.2.**Функциональные требования на основе сценариев**

ID	Требование	Описание	Сценарий
Управление данными и источниками			
FR1	Система должна предоставлять адаптеры для приема данных из разнородных источников	Должна быть обеспечена поддержка: REST API, брокерам сообщений, RTSP, HTLS. Требуется единый внутренний интерфейс для работы с данными независимо от источника	1.1.1
FR2	Система должна обеспечивать хранение исходных визуальных данных и метаданных	Для обеспечения воспроизводимости каждый файл или видеокадр должен сохраняться с уникальным идентификатором и метаданным	1.1.1
Планирование задач по расписанию			
FR3	Система должна позволять декларативно описывать план работы с данными источника, включая расписание, входные данные, цепочку обработчиков	FR3	Система должна позволять декларативно описывать план работы с данными источника, включая расписание, входные данные, цепочку обработчиков
Интеграция и выполнение алгоритмов			
FR6	Система должна	Архитектура	1.1.3
	предоставлять единый программный интерфейс (API) для интеграции внешних алгоритмов обработки	системы должна определять чёткую спецификацию для алгоритмов, включая формат входных/выходных данных и метаданных	
FR7	Система должна обеспечивать изолированное и воспроизводимое выполнение алгоритмов с помощью контейнеризации	Исключаются конфликты версий библиотек и пакетов, требуемых разными алгоритмами, так как каждый из них использует собственное, независимое окружение	
Аналитика, сравнение и предоставление результатов			
FR8	Система должна предоставлять встроенные механизмы для автоматического сравнения результатов различных алгоритмов, запущенных на одних и тех же данных	Должна быть возможность настройки правил сравнения и генерации сводных отчётов	Все сценарии
FR9	Система должна предоставлять API и интерфейс (веб-дашборд) для просмотра истории экспериментов, их статуса, а также для визуализации результатов и аналитических отчётов	Исследователь должен иметь возможность отслеживать ход выполнения, изучать результаты и экспортировать данные для дальнейшего анализа без прямого доступа к файловой системе или базам данных	Все сценарии

Таблица 1. Функциональные требования на основе сценариев

II.3.Нефункциональные требования

ID	Требование
НФТ-1	Система должна гарантировать, что любой ранее выполненный эксперимент может быть точно воспроизведён с получением идентичных (бит в бит) или статистически неразличимых результатов при использовании тех же версий данных, алгоритмов и параметров конфигурации
НФТ-2	Все ключевые сущности (исходные данные, версии алгоритмов, конфигурации экспериментов, результаты выполнения) после регистрации в системе должны быть неизменяемыми. Допустимо только создание новых версий сущностей, что обеспечивает целостность и прослеживаемость данных
НФТ-3	Для каждого результата обработки система должна хранить и предоставлять полную цепочку происхождения: идентификаторы исходных данных, версию алгоритма, значения всех параметров запуска, идентификатор среды выполнения, временные метки и идентификатор пользователя.

Таблица 2. Требования к воспроизводимости

ID	Требование
НФТ-4	Время от наступления события (например, по расписанию) до фактического запуска задачи обработки не должно превышать 5 секунд для 95% запусков
НФТ-5	Система должна поддерживать одновременную обработку данных от не менее 10 разнородных источников с интенсивностью до 1 кадра в секунду от каждого
НФТ-6	Архитектура должна позволять горизонтальное масштабирование компонентов оркестрации и слоев выполнения алгоритмов для увеличения

	количества параллельно выполняемых задач в 10 раз (с 20 до 200) за счёт добавления вычислительных узлов без изменения логики приложения

Таблица 3. Требования к производительности и масштабируемости

ID	Требование
НФТ-7	Система должна обеспечивать доступность не ниже 99% для функций, непосредственно обеспечивающих проведение и воспроизведение экспериментов
НФТ-8	Для повышения отказоустойчивости система должна реализовывать политику автоматических повторных попыток. При аварийном завершении задачи обработки должно быть выполнено не менее 3 попыток перезапуска. Задержка между попытками должна экспоненциально увеличиваться, что позволяет системе переждать временные сбои внешних сервисов или кратковременную перегрузку, не усугубляя её моментальными повторными запросами
НФТ-9	Гарантированное сохранение всех исходных данных и результатов должно обеспечиваться за счёт репликации или регулярного (не реже раза в сутки) резервного копирования

Таблица 4. Требования к надежности и доступности

ID	Требование
НФТ-10	Каждый алгоритм должен выполняться в изолированной среде с ограничением доступа к ресурсам и файловой системе узла.
НФТ-11	Система должна разграничивать права пользователей на уровне экспериментов, источников данных и алгоритмов.
НФТ-12	Все передаваемые данные должны быть защищены с использованием криптографических протоколов.

Таблица 5. Требования к безопасности

ID	Требование
НФТ-13	Целевая аудитория и удобство использования. Основным пользователем, работающим с декларативным описанием экспериментов, является исследователь или инженер, обладающий предметными знаниями в области компьютерного зрения. Система должна быть спроектирована так, чтобы такой пользователь, изучив документацию мог самостоятельно создать и запустить эксперимент.
НФТ-14	Система должна предоставлять инструменты для мониторинга ключевых метрик в реальном времени: количество активных задач, загрузка вычислительных ресурсов, статус источников данных, объём хранилища.
НФТ-15	Все значимые действия (изменение конфигурации, удаление данных, запуск эксперимента) должны регистрироваться в журнале аудита с указанием пользователя, времени и сути изменения.

Таблица 6. Требования к удобству использования и сопровождению

III. Архитектура системы

Предлагаемая архитектура информационно-аналитической системы представляет собой многослойную микросервисную платформу [1], спроектированную для обеспечения сквозной воспроизводимости исследовательского цикла при работе с визуальными данным. В основе архитектуры лежит принцип декомпозиции по ответственности, что позволяет независимо масштабировать и развивать ключевые компоненты.

Ядро платформы. Ядро платформы обеспечивает полный сквозной цикл работы с визуальными данными: от регистрации и конфигурации источников, автоматизированного сбора и каталогизации данных до запуска воспроизводимых экспериментов, исполнения алгоритмов анализа и хранения результатов.

Функции ядра платформы:

A. Управление источниками данных

Ядро платформы обеспечивает регистрацию источников визуальных данных, включая камеры, наборы данных, файловые каталоги и HTTP-источники. Для каждого источника поддерживается хранение конфигурации подключения, включающей адреса, параметры доступа и дополнительные свойства, необходимые для корректной эксплуатации.

Предусмотрено управление жизненным циклом источника посредством изменения его состояния, включая перевод в активный режим, постановку на паузу и отключение. Также реализуются операции просмотра перечня источников и их поиска по атрибутам, таким как тип, набор меток и текущее состояние

B. Сбор (захват) визуальных данных

Ядро платформы реализует механизм настройки правил сбора визуальных данных, включающий периодические снимки, получение видеоклипов и выполнение захвата в заданных временных окнах. Поддерживается инициирование сбора в ручном режиме для оперативного получения актуального кадра или клипа.

Для обеспечения контролируемости процесса предусмотрено ведение журнала задач сбора с фиксацией результата выполнения, причин ошибок и временных характеристик. Все данные системы, включая исходные изображения, обученные модели, промежуточные и финальные результаты вычислений, сохраняются в объектном хранилище, соответствующем отраслевому стандарту для работы с большими бинарными данными.

C. Каталог данных (учёт артефактов)

Ядро платформы обеспечивает регистрацию каждого полученного файла в качестве артефакта с фиксацией его идентификатора, связи с источником, временных меток и технических характеристик, включая формат и размер. Реализованы средства поиска и выборки артефактов по источнику, временному интервалу и дополнительным меткам, что необходимо для формирования наборов входных данных экспериментов. Кроме того, предусмотрены механизмы предоставления доступа к файлам, включая получение ссылки на скачивание или

доступ через программный интерфейс.

D.Подключение и учет сервисов анализа

Ядро платформы поддерживает регистрацию сервисов анализа, определяя перечень доступных анализаторов, которые могут быть использованы при построении экспериментальных запусков. Для каждого сервиса хранится информация, необходимая для вызова и воспроизводимости, включая адрес, версию и перечень поддерживаемых операций. Предусмотрено управление доступностью анализаторов, позволяющее включать и отключать их без остановки платформы и без нарушения целостности выполняемых процессов.

E. Управление экспериментами и запусками

В ядре платформы реализуются механизмы создания экспериментов, в рамках которых фиксируются цели исследования, используемые источники и наборы данных, а также выбранные сервисы анализа. Для обеспечения воспроизводимости поддерживается формирование запуска, в котором однозначно фиксируются состав входных данных, параметры выполнения и версии анализаторов. Также предусматривается возможность повторного выполнения запуска при неизменной конфигурации, что необходимо для корректного сравнения результатов и подтверждения стабильности экспериментальных выводов.

F.Исполнение анализа и управление задачами

Ядро платформы выполняет постановку вычислительных задач анализа для выбранных входных данных и назначенных сервисов анализа. Реализованы средства контроля выполнения, включающие отслеживание статусов задач, управление очередями и обработку ошибок, а также применение повторных попыток выполнения при временных сбоях. По завершении вычислений обеспечивается сбор результатов анализа от подключаемых сервисов и их последующая фиксация в системе.

G.Хранение и просмотр результатов

Ядро платформы обеспечивает сохранение результатов анализа в форме численных метрик и структурированных выходных данных, к которым относятся описания детекции и сегментации, а также результаты распознавания текста и иных аналитических процедур. Реализуется связывание результатов с исходными артефактами и конкретными запусками, что обеспечивает прослеживаемость и корректную интерпретацию полученных данных.

Предусмотрены средства просмотра результатов через программный интерфейс и пользовательские представления, обеспечивающие доступ в рамках установленных прав.

H. Базовое администрирование и наблюдаемость

Ядро платформы поддерживает базовые функции администрирования, включая контроль доступа пользователей на основе ролей и прав, управляемых централизованной подсистемой аутентификации и авторизации. Для эксплуатации и диагностики обеспечивается ведение журналов и мониторинг ключевых операций, связанных со сбором данных, запуском анализа и возникновением ошибок, что позволяет контролировать состояние платформы и своевременно выявлять отклонения в работе.

Сервис отчётов. Reporting Service предназначен для формирования результатов экспериментов и запусков, выполняемых ядром платформы. В отличие от ядра, которое отвечает за сбор данных, постановку задач и хранение “сырых” результатов, сервис отчётов реализует слой интерпретации и агрегирования: сводные таблицы, сравнение алгоритмов, построение итоговых отчётов и их публикация пользователю.

Функции сервиса отчетов:

I. Управление отчётами и запросами на отчётность

Сервис отчета обеспечивает формирование и сопровождение отчетных объектов, включая создание и хранение шаблонов отчетов, в которых задаются тип отчета, используемые метрики и критерии группировки данных. Предусматривается запуск построения отчетов по выбранным экспериментам и запускам, что позволяет получать итоговые представления по заранее заданным правилам. Для обеспечения воспроизводимости реализуется версионирование отчетов, при котором фиксируются параметры построения и состав исходных данных, использованных при формировании результата.

J. Сравнительный анализ результатов

Сервис отчета реализует средства сравнительного анализа, ориентированные на сопоставление результатов нескольких запусков, выполненных на едином наборе входных данных. Поддерживается сравнение алгоритмов и анализаторов по выбранным метрикам, а также по различным подмножествам данных, определяемым временными интервалами, источниками или условиями эксперимента. На основе результатов сравнения формируются сводные рейтинги, позволяющие выделять наиболее эффективные решения, в том числе в разрезе отдельных сценариев или сегментов данных.

K.Агрегации и статистика

Сервис отчета выполняет агрегирование метрик по ключевым измерениям, включая источник, временные интервалы, контекст и параметры эксперимента.

Для повышения информативности отчетов формируются статистические сводки, такие как средние значения, медианы и показатели разброса, а при необходимости используются доверительные интервалы. Дополнительно поддерживается группировка результатов по меткам, классам и условиям эксперимента, что обеспечивает интерпретируемость данных при сравнении алгоритмов на неоднородных выборках.

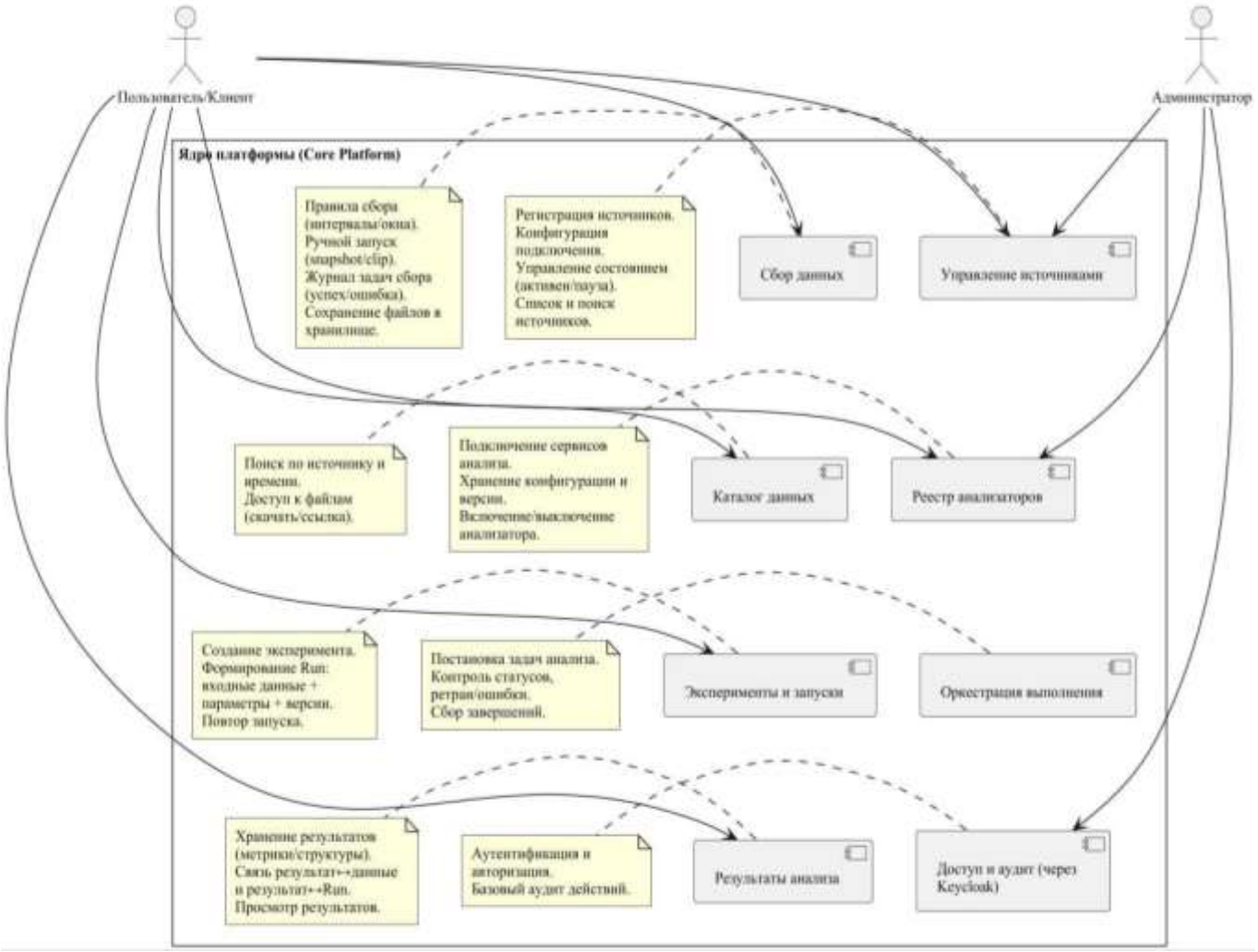

Рис. 1. Диаграмма компонентов ядра платформы

L. Формирование отчётных представлений

Сервис отчета предоставляет механизмы построения представлений, ориентированных на последующую интерпретацию пользователем. К ним относятся табличные формы, включая сводные таблицы и матрицы сравнения, а также графические представления, отражающие динамику метрик во времени и распределения значений. Кроме того, формируются отчетные представления в виде карточек по эксперименту или алгоритму, содержащие сведения о конфигурации запуска, использованных данных и итоговых показателях.

M. Формирование отчётных представлений

Сервис отчета предоставляет механизмы построения представлений, ориентированных на последующую интерпретацию пользователем. К ним относятся табличные формы, включая сводные таблицы и матрицы сравнения, а также графические представления, отражающие динамику метрик во времени и распределения значений. Кроме того, формируются отчетные представления в виде карточек по эксперименту или алгоритму, содержащие сведения о конфигурации запуска, использованных данных и итоговых показателях.

N. Экспорт и публикация

Сервис отчета поддерживает экспорт сформированных результатов в распространенные форматы, включая CSV и JSON, а при необходимости также в HTML или PDF. Предусмотрен программный интерфейс для получения отчета и информации о его состоянии, что позволяет отслеживать этапы построения и фиксировать завершение или ошибку выполнения. Для оптимизации нагрузки и ускорения доступа реализуются механизмы кэширования готовых отчетов и контроля их актуальности, включая возможность повторного построения при изменении исходных данных или условий сравнения.

O. Интеграция с ядром платформы

Сервис отчета интегрируется с ядром платформы для получения исходной информации, необходимой для построения отчетов. В качестве входных данных используются результаты анализа, метаданные запусков и сведения об источниках и контексте экспериментов. Дополнительно может применяться событийный механизм, при котором завершение запуска инициирует перестроение или обновление отчетных представлений, что обеспечивает актуальность аналитических выводов при поступлении новых данных.

Сервис уведомлений. Сервис уведомлений предназначен для доставки сообщений пользователям и внешним информационным системам о значимых событиях, возникающих в ходе функционирования платформы. Его выделение в отдельный компонент позволяет отделить внутреннюю доменную логику ядра, связанную со сбором данных, выполнением экспериментов и запуском алгоритмов анализа, от механизмов внешних коммуникаций, таких как электронная почта или мессенджеры. Такое разделение упрощает развитие интеграций, снижает связанность компонентов и повышает управляемость платформы за счет независимого масштабирования и сопровождения подсистемы доставки уведомлений.

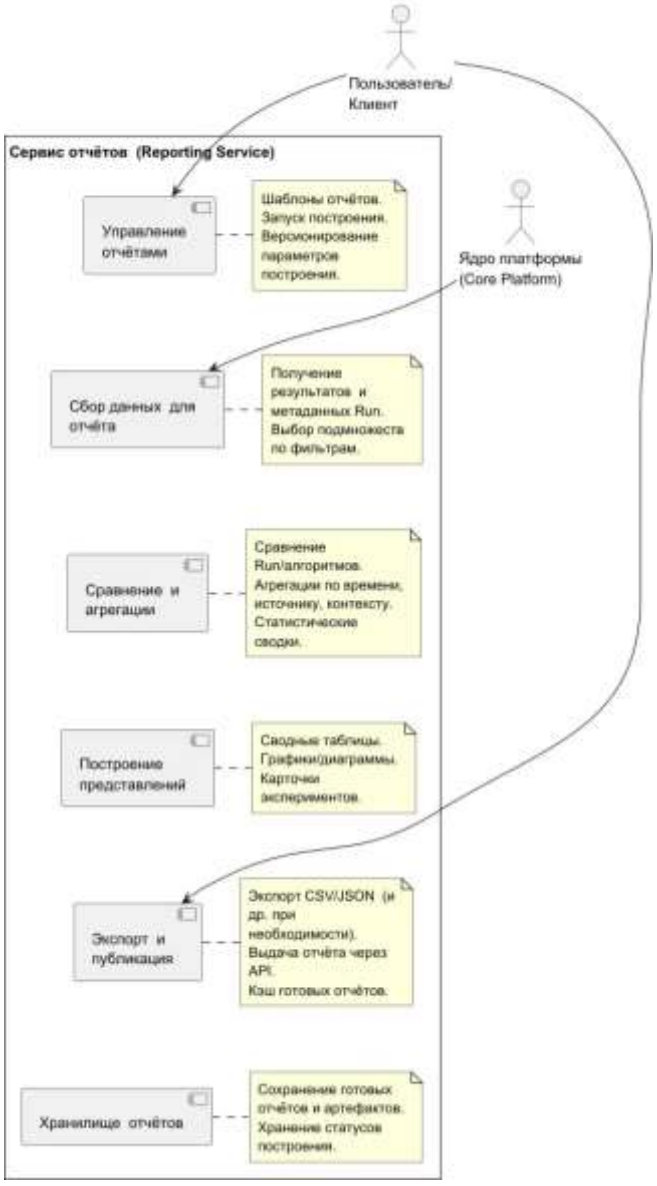

Рис.2. Диаграмма компонентов сервиса отчета

Функции сервиса уведомлений:

P. Приём и обработка событий платформы

Сервис уведомлений обеспечивает получение событий от ядра платформы, включая завершение запусков, ошибки выполнения задач анализа и инциденты, связанные с недоступностью источников. Для обеспечения единообразной обработки выполняется нормализация входящих событий в унифицированный внутренний формат, пригодный для дальнейшей маршрутизации и формирования сообщений. Дополнительно реализуется фильтрация событий по критериям важности, типу, а также принадлежности к проекту или пользователю, что позволяет ограничить отправку уведомлений только значимыми для адресата случаями.

Q.Правила уведомлений и подписки

Сервис уведомлений поддерживает управление подписками, определяющими, какие пользователи или группы получают уведомления по тем или иным типам событий. Реализуются правила маршрутизации, задающие условия доставки, включая ограничения по типам событий, временным интервалам и привязке к конкретным источникам или экспериментам. Для обеспечения единообразия и предсказуемости коммуникаций предусмотрены шаблоны сообщений, которые формируют содержимое уведомления на основе параметров события и контекста выполнения.

R.Каналы доставки

Сервис уведомлений обеспечивает получение событий от ядра платформы, включая завершение запусков, ошибки выполнения задач анализа и инциденты, связанные с недоступностью источников. Для обеспечения единообразной обработки выполняется нормализация входящих событий в унифицированный внутренний формат, пригодный для дальнейшей маршрутизации и формирования сообщений. Дополнительно реализуется фильтрация событий по критериям важности, типу, а также принадлежности к проекту или пользователю, что позволяет ограничить отправку уведомлений только значимыми для адресата случаями. Помимо электронной почты и мессенджеров, доставка уведомлений внешним информационным системам может осуществляться посредством Webhooks [2], обеспечивающих асинхронную интеграцию платформы с внешними сервисами.

S. Надежность доставки

Для исключения блокирования основной логики платформы доставка уведомлений выполняется асинхронно с использованием очереди отправки. В случае временных сбоев внешних каналов поддерживаются повторные попытки доставки, что повышает устойчивость системы к сетевым и инфраструктурным отказам. Дополнительно ведется журнал отправок, фиксирующий статус доставки, причины ошибок и факт повторной обработки, что обеспечивает контролируемость и воспроизводимость эксплуатационных процессов.

T. Аудит и наблюдаемость

Сервис уведомлений обеспечивает ведение истории уведомлений, содержащей сведения о том, какие сообщения были отправлены, кому, в какое время и на основании какого события. Для эксплуатационного контроля формируются метрики, отражающие количество уведомлений, долю ошибок доставки и

задержки обработки. Для сквозной диагностики предусматривается трассировка, связывающая уведомление с исходным событием и контекстом выполнения посредством корреляционного идентификатора, что упрощает анализ причин и последствий возникающих инцидентов.

Сервис аутентификации и авторизации. Разрабатываемая платформа для воспроизводимой обработки визуальных данных ориентирована на работу с источниками, которые потенциально содержат конфиденциальную информацию, включая видеопотоки с объектов, медицинские изображения и данные производственного мониторинга. В связи с этим подсистема безопасности должна обеспечивать однозначную идентификацию субъектов доступа, к которым относятся как пользователи, так и программные компоненты, а также гарантировать, что доступ к функциям и данным платформы осуществляется в строгом соответствии с установленными политиками разграничения прав.

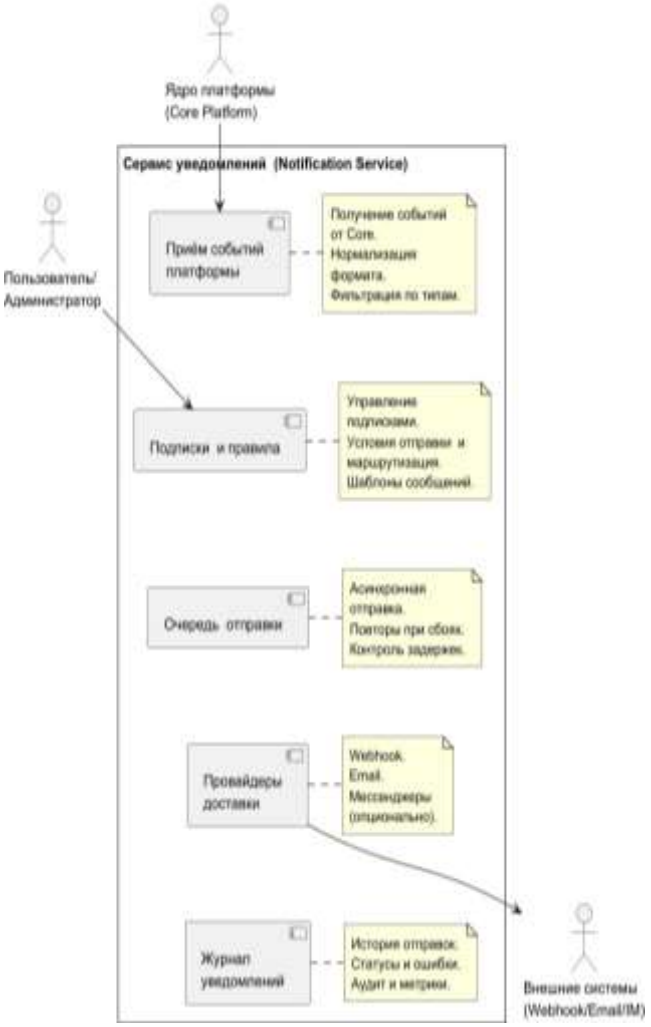

Рис.3. Диаграмма компонентов сервиса нотификации

В проектируемой архитектуре применяется централизованное решение на базе Keycloak, обеспечивающее управление пользователями и клиентами, выпуск и проверку токенов доступа, а также поддержку стандартных протоколов OpenID Connect [3] и OAuth 2.0 [4]. Выбор данного решения обусловлен необходимостью унифицировать механизмы безопасности как в пользовательских сценариях взаимодействия с платформой через веб- интерфейс и REST API, так и в межсервисных взаимодействиях внутри инфраструктуры обращение выполняется с использованием Bearer-токена доступа [5].

Тем самым Keycloak [6] выступает доверенным центром, обеспечивающим управляемость, проверяемость и воспроизводимость процедур аутентификации и авторизации в рамках всей платформы. Для публичного клиента веб-интерфейса поток Authorization Code рекомендуется использовать совместно с механизмом PKCE [7], исключающим возможность перехвата кода авторизации.

Функции ядра платформы:

A. Ролевое разграничение доступа

Для управления доступом к функциям платформы используется ролевая модель, в которой роль определяет класс полномочий субъекта, то есть перечень операций, допустимых на уровне системы в целом. При этом для доступа к конкретным объектам предметной области, включая источники, наборы данных и эксперименты, применяется дополнительная объектная модель контроля доступа. В качестве минимально достаточного набора ролей предлагаются следующие значения. Роль администратора обеспечивает выполнение операций администрирования, включая управление пользователями и правами, регистрацию и удаление источников, подключение и отключение сервисов анализа, а также полный доступ к сущностям платформы в пределах инсталляции. Роль исследователя предназначена для формирования наборов входных данных, создания экспериментов и запусков, анализа и интерпретации результатов, а также генерации отчетов. Роль оператора эксплуатации используется для настройки правил и расписаний сбора, выполнения ручного захвата и диагностики состояния источников. Роль наблюдателя обеспечивает доступ на чтение к разрешенным источникам, данным, результатам и отчетам. Такая модель позволяет сохранить прозрачность полномочий и ограничить выполнение критических операций кругом субъектов с административными правами.

B. Объектные права на источники и данные

Практика эксплуатации систем, работающих с визуальными данными, показывает, что одной ролевой модели недостаточно, поскольку доступ необходимо ограничивать не только по типу действий, но и по множеству объектов, к которым субъект имеет право обращаться. В частности, пользователи с одинаковой ролью могут работать с различными наборами источников и экспериментов, что требует введения объектного контроля доступа. В рамках предлагаемого подхода Keycloak обеспечивает идентификацию пользователя и предоставляет утверждения о его ролях в составе токена доступа, тогда как ядро платформы применяет роль-ориентированные ограничения и дополнительно проверяет объектные разрешения на уровне конкретных идентификаторов источников,

артефактов и экспериментов. Для реализации объектного контроля доступа возможно применение проектной модели, в которой источники и эксперименты закрепляются за проектами, а пользователи получают роль в рамках проекта, либо применение модели списков контроля доступа, в которой для каждого объекта поддерживается перечень разрешений, определяющий права конкретных пользователей или групп.

C. Защита подключаемых сервисов анализа

Существенной особенностью платформы является интеграция с произвольными сервисами анализа, реализующими как классические алгоритмы обработки изображений, так и нейросетевые модели, что делает их вычислительно затратными и потенциально критичными с точки зрения безопасности. В связи с этим требуется исключить прямой публичный доступ к анализаторам, поскольку он может привести к несанкционированному использованию вычислительных ресурсов и компрометации данных.

Для решения данной задачи применяется двухуровневый механизм защиты. На сетевом уровне сервисы анализа размещаются во внутреннем контуре и не экспонируются во внешнюю сеть, вследствие чего пользовательские клиенты не имеют возможности обращаться к ним напрямую. На уровне приложений анализаторы принимают запросы только при наличии действительного токена доступа, выданного Keycloak доверенному сервисному клиенту, а межсервисные вызовы реализуются на основе схемы OAuth 2.0 Client Credentials. В этом случае ядро платформы или выделенный исполнительный узел получает токен по идентификатору и секрету клиента и использует его при обращении к API анализатора. Анализатор валидирует токен и подтверждает, что запрос выполнен доверенным компонентом платформы, что исключает возможность прямого вызова сервиса анализа при наличии только пользовательского токена.

D. Полный поток безопасности

Пользовательский поток (доступ к ядру платформы и отчётам). Практика эксплуатации систем, работающих с визуальными данными, показывает, что одной ролевой модели недостаточно, поскольку доступ необходимо ограничивать не только по типу действий, но и по множеству объектов, к которым субъект имеет право обращаться. В частности, пользователи с одинаковой ролью могут работать с различными наборами источников и экспериментов, что требует введения объектного контроля доступа. В рамках предлагаемого подхода Keycloak обеспечивает идентификацию пользователя и предоставляет утверждения о его ролях в составе токена доступа, тогда как ядро платформы применяет роль-ориентированные ограничения и дополнительно проверяет объектные разрешения на уровне конкретных идентификаторов источников, артефактов и экспериментов. Для реализации объектного контроля доступа возможно применение проектной модели, в которой источники и эксперименты закрепляются за проектами, а пользователи получают роль в рамках проекта, либо применение модели списков контроля доступа, в которой для каждого объекта поддерживается перечень разрешений, определяющий права конкретных пользователей или групп.

Внутренний поток (из ядра к сервисам анализа). Во внутренних сценариях выполнения вычислительных задач платформа обращается к сервисам анализа от имени сервисного аккаунта, используя механизм Client Credentials. Данный поток отделен от пользовательского и исключает возможность инициирования прямого вызова анализатора вне контекста платформы. Перед запуском анализа платформа осуществляет контроль допустимости операции и состава входных данных, после чего инициирует обращение к анализатору от имени доверенного сервисного клиента. Полученные результаты возвращаются в платформу и сохраняются в виде результатов и артефактов, доступных пользователю только при наличии соответствующих прав.

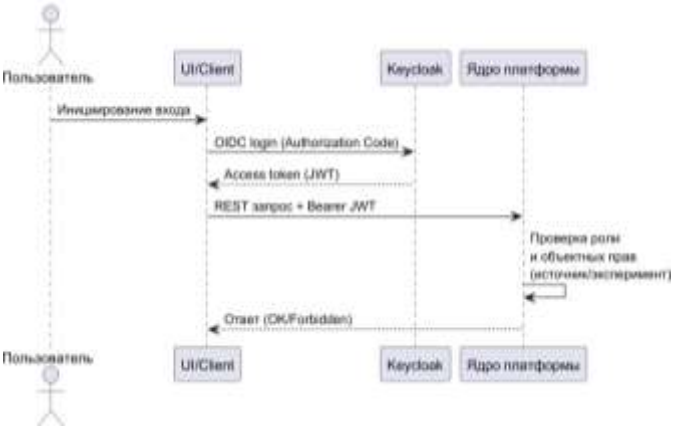

Рис.4. Схема безопасности пользовательского доступа

Рис.5. Схема безопасности доступа к сервисам анализа

E. Предлагаемый набор клиентов KEYCLOAK

Для поддержки пользовательских и межсервисных сценариев в Keycloak формируется набор клиентов. Клиент пользовательского приложения предназначен для веб-интерфейса и реализует типовой поток Authorization Code, включая настройку адресов перенаправления и параметров публичного клиента. Клиент ядра платформы используется для проверки пользовательских токенов как ресурсный сервер, а при необходимости также выступает сервисным клиентом для выполнения внутренних операций. Клиент сервиса отчета обеспечивает проверку токенов при обращениях пользователей к отчетам и может использовать сервисный аккаунт для чтения данных из ядра при

асинхронном построении отчетов. Клиент сервиса уведомлений предназначен для доверенного получения событий и данных от ядра и может использовать механизм Client Credentials при обращении к API платформы для уточнения контекста уведомления. Клиенты сервисов анализа формируются для отдельных анализаторов и используются для проверки доверенных межсервисных вызовов, при этом анализаторы функционируют как ресурсные серверы и принимают токены только от доверенных клиентов платформы.

Сервисы анализа. Сервисы анализа представляют собой подключаемые вычислительные компоненты платформы, реализующие алгоритмы обработки визуальных данных, включая как классические методы компьютерного зрения, так и подходы машинного обучения и нейросетевые модели. Их назначение состоит в том, чтобы принимать входные визуальные артефакты и параметры выполнения, осуществлять вычисления в соответствии с выбранным алгоритмом и возвращать результаты в стандартизированном виде, пригодном для последующего хранения, сопоставления и интерпретации в рамках платформы. Указанные сервисы рассматриваются как вычислительные модули, внешние по отношению к ядру платформы, что позволяет разворачивать их независимо, учитывать различающиеся требования к аппаратным ресурсам, включая использование CPU и GPU, а также обеспечивать независимое масштабирование в зависимости от нагрузки.

Функции сервисов анализа:

A. Реализация операций анализа

Сервисы анализа обеспечивают выполнение прикладных операций обработки изображений и видео, к которым относятся задачи детекции, сегментации, классификации, распознавания текста, оценивания качества, сопровождения объектов и процедур улучшения визуальных данных. Поддерживаются различные режимы вычисления, включая обработку одиночного файла, пакетную обработку набора файлов, а также обработку видеоклипа или серии кадров, что позволяет применять единый подход к запуску анализа в широком спектре исследовательских сценариев.

B. Приём входных данных и параметров

Сервисы анализа получают идентификаторы или ссылки на входные артефакты и осуществляют чтение данных из объектного хранилища посредством механизмов, определенных платформой, включая доступ по разрешенным ссылкам. Наряду с данными принимаются параметры выполнения, включающие пороговые значения, настройки предварительной обработки, выбор модели и ее версии, а также дополнительные метаданные, необходимые для корректной интерпретации входных данных и воспроизводимого запуска алгоритма.

C. Управление версиями и воспроизводимость

Для обеспечения воспроизводимости сервисы анализа фиксируют в результатах сведения о версии используемого алгоритма или модели и параметрах инферейса, примененных в ходе выполнения. Предусматривается возможность параллельной эксплуатации нескольких версий моделей, что позволяет выполнять сопоставительные эксперименты, проводить контролируемые обновления и сохранять корректность ранее полученных результатов при эволюции алгоритмической базы.

D. Формирование результатов

Сервисы анализа возвращают результаты в форме, пригодной для машинной обработки и дальнейшей аналитики, включая численные метрики и структурированные представления, такие как ограничивающие прямоугольники, маски сегментации, ключевые точки и распознанный текст. При необходимости формируются выходные артефакты, включая визуализации и вспомогательные файлы, которые могут быть загружены в объектное хранилище (в качестве технологии объектного хранилища будет использовать S3 Minio [8]) либо переданы в платформу для последующей регистрации и хранения в соответствии с принятой моделью управления артефактами.

Эксплуатационные функции. Для обеспечения корректной эксплуатации сервисы анализа предоставляют средства диагностики, включающие проверку готовности и доступности, а при необходимости контроль очереди и текущей нагрузки. Также обеспечиваются логирование и публикация метрик выполнения, отражающих время обработки, количество запросов и характеристики ошибок, что необходимо для мониторинга производительности и выявления деградации качества обслуживания при росте нагрузки.

Безопасность доступа. Сервисы анализа не предоставляются в публичный доступ и вызываются исключительно ядром платформы либо доверенными исполнительными компонентами, действующими в пределах внутреннего контура. Межсервисная авторизация реализуется посредством централизованной подсистемы безопасности на базе Keycloak с использованием схемы Client Credentials, при которой анализатор принимает запросы только от доверенного сервисного аккаунта и отклоняет обращения, не подтвержденные валидным токеном доступа.

III.1. Регистрация сервисов анализа через HashiCorp Consul

При проектировании платформы, ориентированной на подключение и эксплуатацию множества сервисов анализа визуальных данных, существенное значение приобретает задача актуальной адресации вычислительных компонентов и контроля их доступности. В реальных условиях сервисы анализа часто масштабируются горизонтально, перезапускаются, обновляются по версиям и могут размещаться на узлах с различными аппаратными характеристиками, включая использование GPU. В таких сценариях фиксация сетевых адресов анализаторов в конфигурации ядра приводит к росту операционных затрат и повышает риск обращения к недоступным экземплярам, а также затрудняет параллельное использование нескольких версий одного и того же алгоритма.

Для обеспечения динамического обнаружения и учета доступных экземпляров в платформе применяется

HashiCorp Consul, реализующий механизм service discovery [9]. Consul поддерживает регистрацию экземпляров сервисов, проведение проверок доступности и предоставление централизованного интерфейса поиска по имени и метаданным. Тем самым формируется актуальный реестр работающих анализаторов, обновляемый автоматически без необходимости ручного сопровождения координат.

Использование Consul в подсистеме анализа обеспечивает ряд эксплуатационно значимых эффектов. Во-первых, недоступные экземпляры автоматически исключаются из перечня кандидатов на выполнение запросов, что снижает вероятность повторяющихся отказов.

Во-вторых, упрощается масштабирование, поскольку несколько экземпляров одного анализатора регистрируются под единым логическим именем, а выбор конкретного экземпляра может выполняться динамически. В-третьих, поддержка метаданных позволяет учитывать параметры развертывания, включая версию алгоритма, наличие GPU и режимы обработки, что важно для корректного выполнения экспериментов и фиксации условий воспроизводимости.

В предлагаемой архитектуре доменная регистрация анализатора как доступного функционального компонента хранится в ядре платформы, тогда как Consul используется для хранения сведений о фактически запущенных экземплярах и их текущем состоянии. При запуске каждый сервис анализа публикует в Consul свои сетевые параметры и критерии проверки доступности. Ядро платформы перед вызовом анализа запрашивает у Consul список доступных экземпляров и выбирает подходящий, после чего выполняет защищенный межсервисный вызов с авторизацией через Keycloak на основе схемы Client Credentials. В результате Consul решает задачу актуального обнаружения адресата [10], а Keycloak обеспечивает контроль доверенного доступа.

Таким образом, применение HashiCorp Consul позволяет снизить связанность конфигураций, повысить устойчивость к сбоям, упростить масштабирование сервисов анализа и обеспечить корректную эксплуатацию версий алгоритмов в условиях изменяющейся инфраструктуры, что соответствует требованиям платформы к воспроизводимости и управляемости вычислительного контура.

III.2. Событийная модель

В ранее рассмотренных разделах неоднократно упоминалась публикация событий, что отражает использование в архитектуре платформы событийной модели взаимодействия. Данный подход позволяет декомпозировать систему на относительно независимые подсистемы. Ядро платформы фиксирует факты изменения состояния, например появление нового артефакта данных или завершение запуска, после чего специализированные компоненты, включая сервис отчетов и сервис уведомлений, реагируют на эти факты без необходимости прямых синхронных вызовов.

Такое разделение повышает устойчивость и масштабируемость системы, снижает связанность компонентов и обеспечивает возможность асинхронной обработки длительных операций. В качестве транспорта событий используется брокер сообщений, при этом в рамках настоящей работы применяется Apache Kafka [11], что обусловлено его моделью устойчивого журнала событий и поддержкой масштабируемой доставки сообщений.

Сервис ядра:

1. Событие AssetCreated публикуется при регистрации нового артефакта данных, к которым относятся изображения, видеоклипы и выходные файлы анализа. Данное событие используется для построения реактивных процессов, включая актуализацию витрин данных, запуск формирования отчетов, отправку уведомлений и аудит происхождения артефактов.
2. Событие AssetArchived публикуется при логическом удалении или архивировании артефакта. Его назначение состоит в обеспечении корректного исключения соответствующих объектов из выборок и поддержании согласованности представлений в потребляющих подсистемах.
3. Событие IngestionTaskFailed публикуется при неуспешном выполнении задачи сбора данных, например при таймауте подключения к источнику или ошибке чтения потока. Событие используется для диагностики, формирования оперативных уведомлений и построения статистики отказов.
4. Событие RunCreated публикуется при создании запуска и фиксирует факт формирования воспроизводимой конфигурации выполнения, включающей состав входных данных, параметры и версии анализаторов. Данное событие применяется для ведения журналов экспериментов и аудита.
5. Событие RunStarted публикуется при фактическом начале выполнения запуска и обеспечивает возможность корректного отслеживания активных запусков и отображения актуальных статусов в интерфейсах и отчетных представлениях.
6. Событие RunCompleted публикуется при успешном завершении запуска и отражаетготовность результатов для последующего сравнения и формирования отчетов, а также может служить основанием для уведомления пользователя.
7. Событие RunFailed публикуется при завершении запуска с ошибкой и используется для уведомлений, анализа причин сбоев и организации повторного выполнения.
8. Событие TaskCompleted публикуется при успешном завершении отдельной вычислительной задачи в рамках запуска и применяется в случаях, когда требуется детальное отслеживание прогресса и обработка частичных результатов.
9. Событие TaskFailed публикуется при завершении отдельной задачи анализа с ошибкой и используется для локализации проблем конкретного алгоритма или экземпляра сервиса, а также для повторного выполнения только неуспешных задач.
10. Событие ResultPersisted публикуется после сохранения результата анализа и отражает момент, когда.

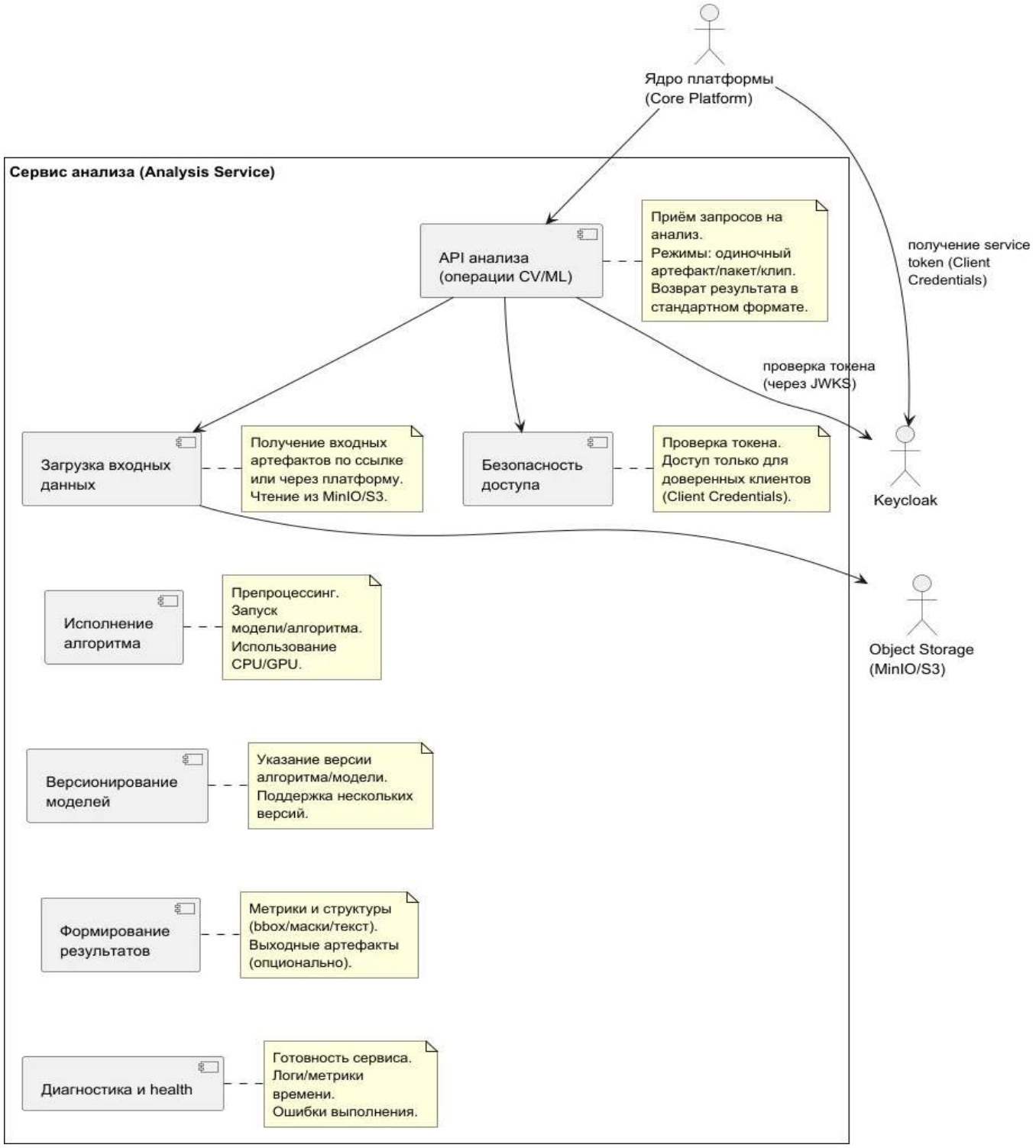

Рис. 6. Диаграмма компонентов сервисов анализа

результат становится доступным через программный интерфейс. Событие целесообразно использовать при инкрементальном построении отчетов по мере поступления результатов, а не только по завершению запуска целиком.

11. Событие ExperimentArchived публикуется при архивировании или закрытии эксперимента и обеспечивает корректное маркирование связанных запусков и отчетов, что необходимо для поддержания согласованности и интерпретации исторических данных.

Сервис отчетов:

1. Событие ReportRequested публикуется при создании запроса на построение отчета, инициированного пользователем либо автоматизированными правилами. Данное событие фиксирует факт обращения к отчетности и используется для запуска асинхронного построения и ведения истории формирования отчетов.

2. Событие ReportBuildStarted публикуется при начале построения отчета и отражает переход запроса в состояние выполнения. Оно применяется для отображения статуса в пользовательских интерфейсах и для предотвращения повторного

параллельного запуска построения.

3. Событие ReportBuildProgress публикуется при

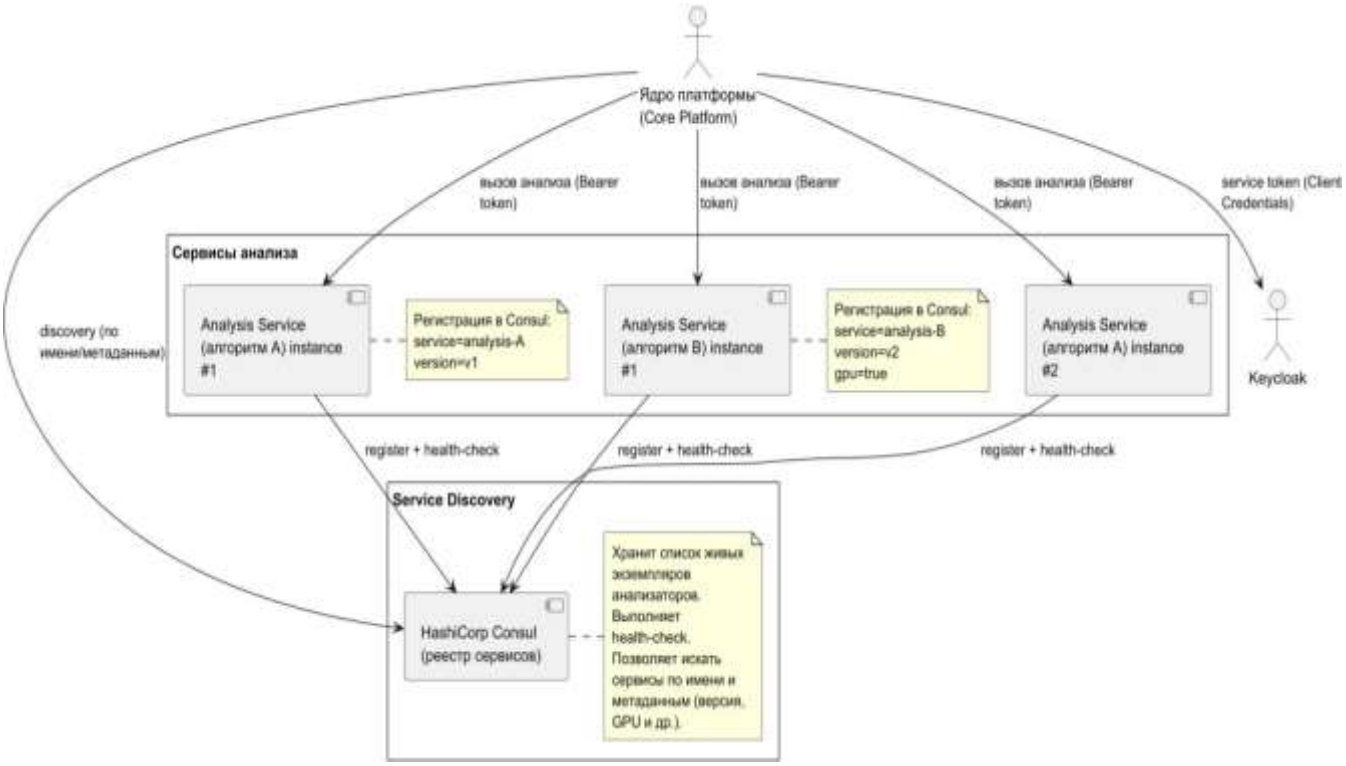

Рис. 7. Схема регистрация сервисов анализа в Consul

обновлении прогресса построения отчета и используется в случаях, когда длительность формирования отчета существенна и требуется промежуточное информирование о ходе выполнения.

4. Событие ReportBuildCompleted публикуется при успешном завершении построения отчета и отражает доступность результата для просмотра или выгрузки. Событие является типовым триггером для уведомлений о готовности отчета.

5. Событие ReportBuildFailed публикуется при ошибке построения отчета и содержит информацию, необходимую для диагностики и информирования пользователя о причине сбоя.

6. Событие ReportInvalidated публикуется при потере актуальности ранее сформированного отчета, например при появлении новых запусков или результатов, влияющих на итоговые показатели. Событие применяется для управления актуальностью отчетов и последующего автоматизированного пересчета.

7. Событие ReportRefreshed публикуется при пересчете и обновлении отчета и фиксирует формирование новой версии, что позволяет обеспечить согласованность отчетных представлений и поддерживать историю изменений. Событие ReportArchived публикуется при логическом удалении или архивировании отчета и используется для исключения отчета из активных списков и поддержания согласованности витрин.

III.3. Событийная модель

В архитектурных схемах платформы выделяется компонент API Gateway [12], который рассматривается как единая входная точка для внешних клиентов, включая пользовательский интерфейс, прикладные интеграции и внешних потребителей программного интерфейса. API Gateway принимает входящие запросы по протоколу HTTP(S) и выполняет их маршрутизацию к внутренним сервисам платформы, включая ядро, сервис отчетов и сервис уведомлений. Централизация входного трафика обеспечивает единообразие политик доступа и эксплуатационных параметров, а также позволяет изолировать внутренние компоненты от прямого внешнего воздействия.

Использование API Gateway является оправданным в силу того, что оно позволяет исключить дублирование типовых инфраструктурных механизмов в каждом микросервисе. Во-первых, обеспечивается единый публичный интерфейс платформы, вследствие чего клиентским приложениям не требуется знание адресов внутренних сервисов, а изменения внутренней структуры не приводят к необходимости модификации внешних интеграций. Во-вторых, API Gateway может выполнять первичную проверку токенов и применять базовые правила авторизации на уровне маршрутов, что снижает риск неконсистентной конфигурации механизмов безопасности и обеспечивает унифицированный контроль входящего трафика. При этом окончательные проверки прав, включая объектные разрешения на уровне источников и экспериментов, сохраняются в доменной логике ядра платформы. В-третьих, API Gateway упрощает поддержку версионирования программного интерфейса и маршрутизацию запросов к соответствующим версиям сервисов, что является существенным фактором при эволюции платформы и обеспечении обратной совместимости.

Дополнительным преимуществом является возможность централизованного управления эксплуатационными политиками. На уровне API Gateway удобно задавать ограничения частоты запросов,

контролировать размеры передаваемых данных, устанавливать таймауты и правила повторов в случаях, где это допустимо, а также конфигурировать политики междоменных запросов для веб-клиента. Кроме того, API Gateway является естественной точкой для унифицированного логирования входящих запросов и внедрения корреляционных идентификаторов, что повышает качество мониторинга и диагностики распределенных сценариев выполнения. Наконец, выделение единой внешней точки входа снижает поверхность атаки, поскольку внешнему окружению экспонируется только API Gateway, тогда как внутренние сервисы остаются в закрытом контуре сети.

В рамках рассматриваемой архитектуры, включающей ядро платформы, сервис отчетов и сервис уведомлений, API Gateway выполняет функцию единого публичного интерфейса. Через него маршрутизируются пользовательские запросы к операциям управления источниками, сбором данных, экспериментами и результатами, а также запросы к операциям формирования и получения отчетов и управлению подписками на уведомления. Подключаемые сервисы анализа при этом не публикуются как публичные конечные точки. Они доступны только во внутреннем контуре и вызываются ядром платформы по защищенному межсервисному взаимодействию. Такой подход обеспечивает, что внешние пользователи и интеграции взаимодействуют с платформой через единый контролируемый канал, тогда как вычислительные компоненты анализа остаются изолированными и защищенными.

III.4. Мониторинг платформы

Эксплуатация платформы, включающей сбор визуальных данных, выполнение вычислительных задач и интеграцию ресурсоемких сервисов анализа, требует обеспечения наблюдаемости как обязательного свойства распределенной системы. Под наблюдаемостью в данном контексте понимается способность получать объективные сведения о текущем состоянии платформы, выявлять отклонения в работе, локализовать причины инцидентов и контролировать использование вычислительных ресурсов.

Наличие мониторинга позволяет обеспечить устойчивость работы, предсказуемость качества обслуживания и воспроизводимость эксплуатационных процессов.

Мониторинг платформы решает задачи оперативного обнаружения сбоев, включая отказ сервисов, недоступность источников и ошибки анализа. Дополнительно он обеспечивает контроль производительности, отражающий задержки обработки, пропускную способность и состояние очередей задач. Существенное значение имеет наблюдение за потреблением ресурсов, включающее нагрузку на процессор, оперативную память, дисковую и сетевую подсистемы, а также использование графических ускорителей для нейросетевых анализаторов. Мониторинг также выполняет диагностическую функцию, позволяя устанавливать причинно-следственные связи между событиями, трассировать распределенные запросы и сопоставлять ошибки с конкретными запусками и задачами. Наконец, он формирует эксплуатационные метрики, отражающие объем собранных данных, интенсивность выполнения запусков и частоту отказов, что необходимо для анализа эффективности и планирования развития платформы.

Логи платформы. Логирование является основным источником детальной информации для расследования инцидентов и анализа ошибок. Для платформы значимыми требованиями являются унификация формата логов, предпочтительно с использованием структурированного представления, а также обеспечение корреляции записей по идентификаторам, связывающим запросы и вычислительные операции в рамках одного сценария выполнения. Централизованный сбор логов со всех сервисов и исполнительных компонентов обеспечивает возможность сквозного поиска и ускоряет диагностику распределенных отказов.

Метрики. Метрики формируют количественное представление о состоянии платформы и позволяют оценивать динамику ее работы. В число технических метрик входят показатели времени ответа программного интерфейса, доли ошибочных запросов, интенсивности обращений и задержек. Для подсистемы сбора данных существенными являются показатели количества успешных и неуспешных операций захвата, длительности захвата и частоты повторных попыток. Для подсистемы анализа важны метрики длительности выполнения задач, состояния очередей и доли ошибок выполнения. Наряду с техническими параметрами применяются прикладные метрики, отражающие интенсивность выполнения запусков, число созданных артефактов и объем сформированных отчетов.

Трассировка. Для распределенной архитектуры существенное значение имеет трассировка запросов в сквозном режиме, позволяющая наблюдать последовательность прохождения запроса через компоненты платформы, начиная от входа через API Gateway и заканчивая обращением к сервисам анализа и возвратом результата. Трассировка дает возможность локализовать источники задержек и ошибок, а также выявлять деградацию производительности на отдельных участках цепочки исполнения.

Мониторинг ресурсов CPU/GPU. Для сервисов анализа, особенно реализующих нейросетевые модели, критичным является наблюдение за ресурсами вычислительной инфраструктуры. Контролю подлежат нагрузка на процессор и оперативную память, а также параметры графических ускорителей, включающие текущую загрузку, объем используемой видеопамяти, температурные характеристики и энергопотребление.

реализации мониторинга используется набор специализированных средств, обеспечивающих сбор, хранение и визуализацию данных наблюдаемости. Сбор и хранение метрик осуществляется посредством Prometheus [13], поддерживающего модель периодического опроса экспортеров. Для построения дашбордов и визуализации применяется Grafana [14].

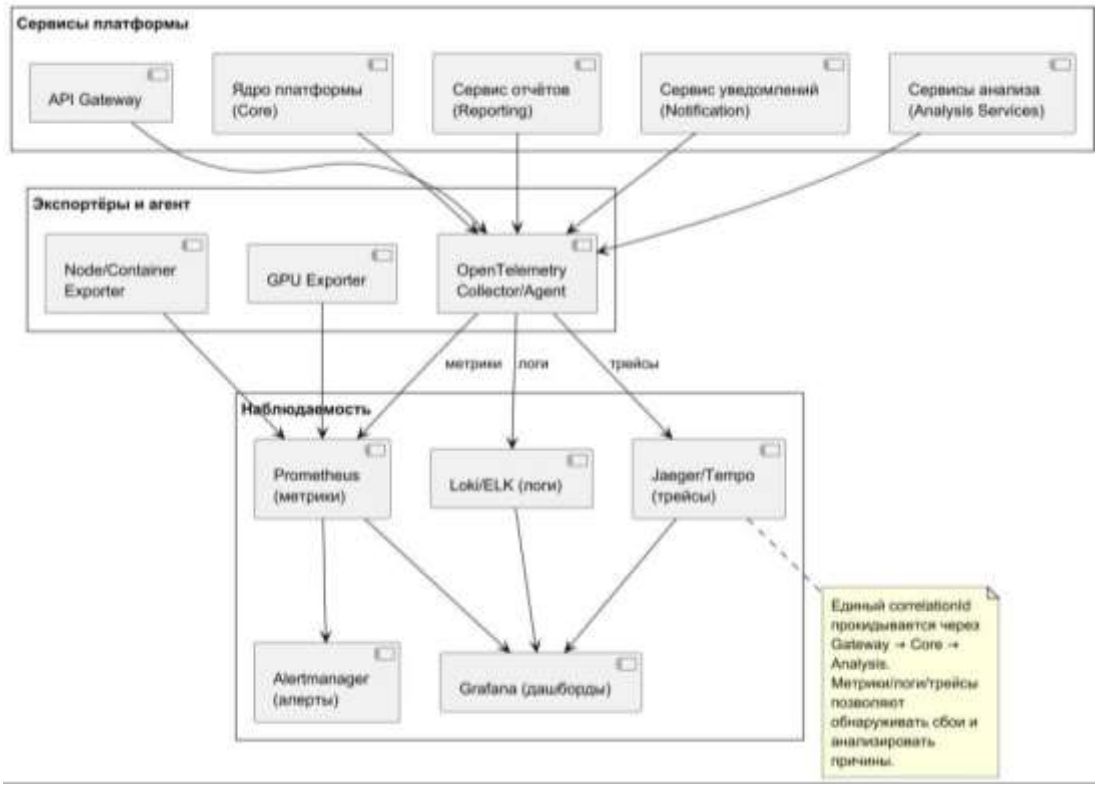

Рис. 8. Схема мониторинга платформы

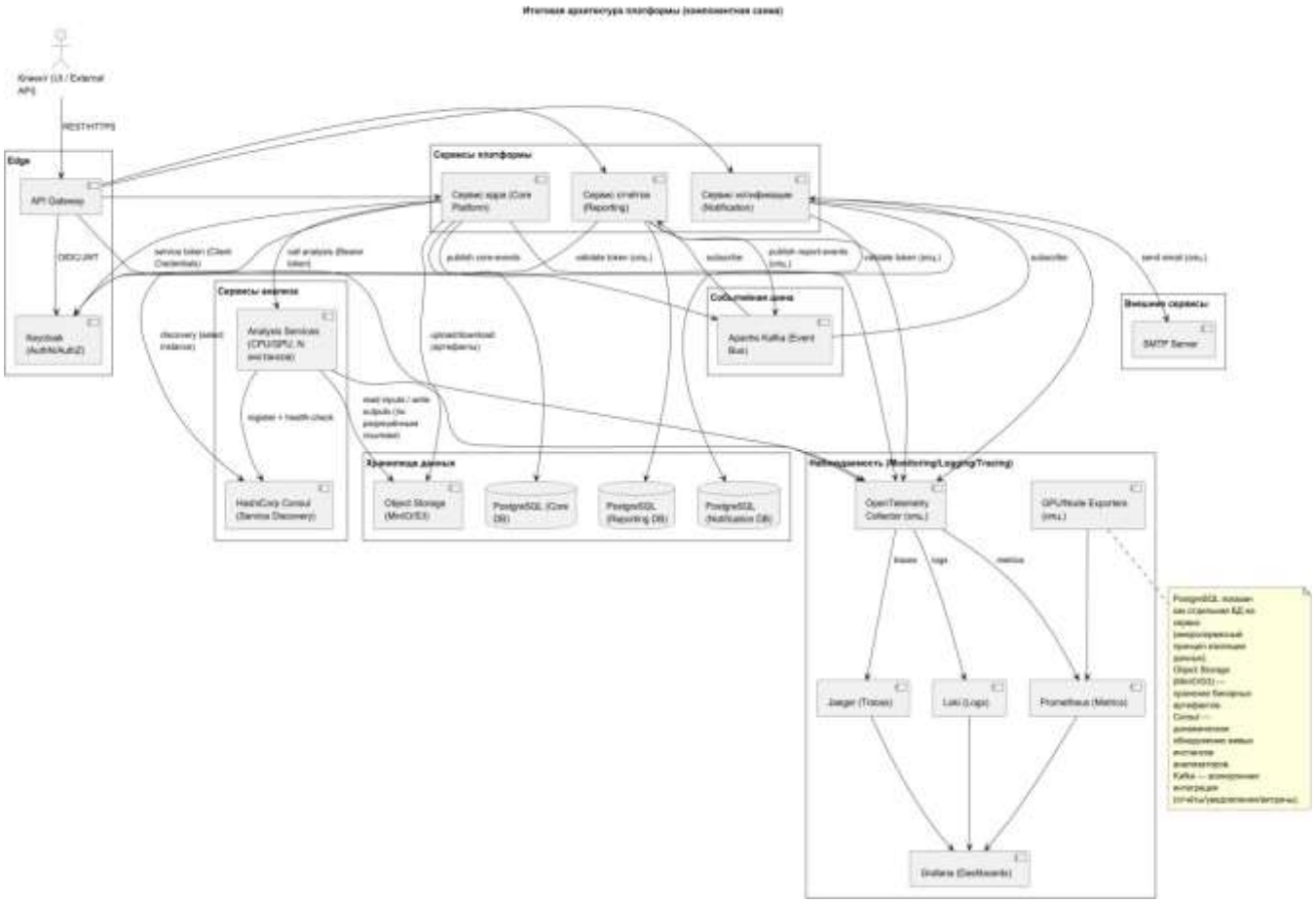

Рис. 9 Итоговая архитектурная схема платформы

Инструменты мониторинга для платформы. Для Настройка правил оповещения по метрикам выполняется с использованием Alertmanager, что позволяет формировать уведомления при превышении пороговых значений и возникновении аномалий. Централизованный сбор и анализ логов реализуется с применением Loki [15] либо альтернативных стеков, ориентированных на индексирование и поиск. Для распределенной трассировки используется Jaeger [16] или Tempo. Инструментализация сервисов и унификация механизмов сбора метрик, трасс и логов обеспечивается стандартом OpenTelemetry [17], что позволяет использовать единый подход к наблюдаемости для всех компонентов платформы, включая вычислительно тяжелые сервисы анализа.

III.5. База данных платформы

Хранение данных в платформе воспроизводимой обработки визуальных данных целесообразно рассматривать как многоуровневую систему, поскольку различные классы данных характеризуются неодинаковыми требованиями к объему, структуре, консистентности и жизненному циклу. Ключевыми типами данных являются метаданные и связи, включающие сведения об источниках, экспериментах, результатах и правилах доступа, которые относятся к транзакционному классу. Отдельную категорию составляют бинарные объекты, представленные изображениями, видеоклипами и производными артефактами, для которых характерны значительные размеры и необходимость масштабируемого хранения. Наконец, существенную роль играют события и эксплуатационные данные, к которым относятся журналы событий, метрики и диагностическая информация, обладающие потоковой природой и требующие специализированных механизмов обработки и агрегирования.

В качестве основной базы данных платформы обоснован выбор реляционной СУБД класса SQL, в частности PostgreSQL [18]. Данное решение соответствует природе доменной модели платформы, где преобладают взаимосвязанные сущности и предъявляются строгие требования к согласованности и транзакционной целостности. Реляционная база данных обеспечивает формализацию и надежное хранение сведений об источниках и их конфигурациях, включая параметры подключения, состояния и правила доступа. Аналогичным образом в ней фиксируются правила сбора данных, задачи захвата, результаты попыток выполнения и диагностические признаки, необходимые для анализа отказов. Значимым элементом является каталог артефактов, в рамках которого сохраняются метаданные объектов, их привязка к источникам и временным меткам, а также атрибуты контроля целостности, включая контрольные суммы и механизмы логической архивации. Кроме того, в базе данных хранится модель экспериментов и запусков, обеспечивающая фиксацию состава входных данных, выбранных анализаторов, параметров выполнения и версий алгоритмов, что является необходимым условием воспроизводимости. В реляционной модели также представлены задачи анализа и результаты, включая статусы, причины ошибок и связи между запуском, задачей, результатом и связанными артефактами. Наконец, в рамках транзакционной базы данных могут храниться сведения, обеспечивающие функционирование отчетности и уведомлений, включая статусы построения отчетов, настройки подписок и, при необходимости, журнал доставки уведомлений для целей аудита. Дополнительно поддерживаются объектные права доступа, позволяющие формализовать привязки пользователей и проектов к источникам и экспериментам и обеспечивать корректное разграничение доступа на уровне данных.

Итоговая архитектурная схема платформы приведена на рис.9.

IV. Заключение

В рамках работы была разработана архитектура информационно-аналитической платформы, ориентированной на организацию воспроизводимых исследований в области компьютерного зрения и обработки визуальных данных. Предложенное решение формализует полный исследовательский цикл - от регистрации и конфигурации источников, автоматизированного сбора и каталогизации артефактов до выполнения вычислительных задач анализа, фиксации результатов и последующего формирования отчётных представлений. Тем самым устраняется характерная для исследовательских проектов фрагментированность инструментов и снижается доля ручных операций при проведении экспериментов.

Библиография

[1] S. Newman, «Building Microservices: Designing Fine-Grained Systems». Sebastopol: O'Reilly Media, 2021. 612 p.
[2] Webhooks documentation, 2025. [Online]. Available: https://docs.github.com/en/webhooks/about-webhooks.
[3] OpenID Foundation. OpenID Connect Core 1.0. [Online]. Available: https://openid.net/specs/openid-connect-core-1_0.html.
[4] Internet Engineering Task Force. The OAuth 2.0 Authorization Framework (RFC 6749). [Online]. Available: https://datatracker.ietf.org/doc/html/rfc6749.
[5] Internet Engineering Task Force. The OAuth 2.0 Authorization Framework: Bearer Token Usage (RFC 6750). [Online]. Available: https://datatracker.ietf.org/doc/html/rfc6750.
[6] Keycloak. Server Administration Guide (latest). [Online]. Available: https://www.keycloak.org/docs/latest/server_admin/index.html.
[7] Internet Engineering Task Force. Proof Key for Code Exchange by OAuth Public Clients (RFC 7636), 2015. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7636.
[8] MinIO, Inc. S3 API Compatibility. AIStor Object Store Documentation. [Online]. Available: https://docs.min.io/enterprise/aistor-object-store/developers/s3-api- compatibility
[9] HashiCorp. Consul. Service Discovery Explained. [Online]. Available: https://developer.hashicorp.com/consul/docs/use-case/service-discovery
[10] HashiCorp. Consul HTTP API. Health. [Online]. Available: https://developer.hashicorp.com/consul/api-docs/health
[11] Apache Software Foundation. Apache Kafka Documentation. [Online]. Available: https://kafka.apache.org/documentation/.
[12] Spring. Spring Cloud Gateway Reference Documentation. [Online]. Available: https://docs.spring.io/spring-cloud-gateway/reference/index.html
[13] The Prometheus Authors. Prometheus. Overview. [Online]. Available: https://prometheus.io/docs/introduction/overview/
[14] Grafana Labs. Grafana Dashboards Overview. [Online]. Available: https://grafana.com/docs/grafana/latest/fundamentals/dashboards-overview/
[15] Grafana Labs. Loki Documentation. [Online]. Available: https://grafana.com/docs/loki/latest/

[16] CNCF. Jaeger Documentation. [Online]. Available: https://www.jaegertracing.io/docs/
[17] OpenTelemetry Authors. OpenTelemetry Documentation. [Online]. Available: https://opentelemetry.io/docs/
[18] PostgreSQL Global Development Group. PostgreSQL Documentation (current). [Online]. Available: https://www.postgresql.org/docs/current/.

Development of the architecture of an information and analytical image processing system

D.I. Sigalov, V.A. Sychev

***Abstract*—The paper proposes the architecture of an information and analytical system for organizing reproducible research in the field of computer vision. The system provides unified collection and storage of visual data from heterogeneous sources, with the ability to automatically perform analytical tasks. Their processing is carried out through integration with arbitrary algorithms - both classical methods of image analysis and neural network models within a single execution interface. The platform implements a mechanism for dynamic management of the context of experiments and provides built-in tools for comparative analysis of the results of various algorithms. The developed solution formalizes the full research cycle, reducing the routine load and ensuring full reproducibility when working with visual data.**

***Keywords*—REST API, OAUTH2.0, HashiCorp, RTSP, PostgreSQL, Loki, S3 MINIO, IP, Jaeger.**

Библиография

[1] S. Newman, «Building Microservices: Designing Fine-Grained Systems». Sebastopol: O'Reilly Media, 2021. 612 p.
[2] Webhooks documentation, 2025. [Online]. Available: https://docs.github.com/en/webhooks/about-webhooks.
[3] OpenID Foundation. OpenID Connect Core 1.0. [Online]. Available: https://openid.net/specs/openid-connect-core-1_0.html.
[4] Internet Engineering Task Force. The OAuth 2.0 Authorization Framework (RFC 6749). [Online]. Available: https://datatracker.ietf.org/doc/html/rfc6749.
[5] Internet Engineering Task Force. The OAuth 2.0 Authorization Framework: Bearer Token Usage (RFC 6750). [Online]. Available: https://datatracker.ietf.org/doc/html/rfc6750.
[6] Keycloak. Server Administration Guide (latest). [Online]. Available: https://www.keycloak.org/docs/latest/server_admin/index.html.
[7] Internet Engineering Task Force. Proof Key for Code Exchange by OAuth Public Clients (RFC 7636), 2015. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7636.
[8] MinIO, Inc. S3 API Compatibility. AIStor Object Store Documentation. [Online]. Available: https://docs.min.io/enterprise/aistor-object-store/developers/s3-api- compatibility
[9] HashiCorp. Consul. Service Discovery Explained. [Online]. Available: https://developer.hashicorp.com/consul/docs/use-case/service-discovery
[10] HashiCorp. Consul HTTP API. Health. [Online]. Available: https://developer.hashicorp.com/consul/api-docs/health
[11] Apache Software Foundation. Apache Kafka Documentation. [Online]. Available: https://kafka.apache.org/documentation/.
[12] Spring. Spring Cloud Gateway Reference Documentation. [Online]. Available: https://docs.spring.io/spring-cloud-gateway/reference/index.html
[13] The Prometheus Authors. Prometheus. Overview. [Online]. Available: https://prometheus.io/docs/introduction/overview/
[14] Grafana Labs. Grafana Dashboards Overview. [Online]. Available: https://grafana.com/docs/grafana/latest/fundamentals/dashboards-overview/
[15] Grafana Labs. Loki Documentation. [Online]. Available: https://grafana.com/docs/loki/latest/
[16] CNCF. Jaeger Documentation. [Online]. Available: https://www.jaegertracing.io/docs/
[17] OpenTelemetry Authors. OpenTelemetry Documentation. [Online]. Available: https://opentelemetry.io/docs/
[18] PostgreSQL Global Development Group. PostgreSQL Documentation (current). [Online]. Available: https://www.postgresql.org/docs/current/.