ПРОГРАММНАЯ ЧАСТЬ ПРОГРАММНО-АППАРАТНОГО КОМПЛЕКСА

«ЭВИРИС»

ОПИСАНИЕ ПРОГРАММЫ ЭВМ

Правообладатель: Общество с ограниченной ответственностью «ГК «Иннотех» (ООО «ГК «Иннотех»)

Место нахождения и адрес правообладателя: 125167, г. Москва, вн.тер.г. муниципальный округ Аэропорт, пр-кт Ленинградский, д. 36, стр. 41, помещ. 23

СОДЕРЖАНИЕ

1. Термины и сокращения

Термины и сокращения, используемые в контексте настоящего документа:

Термин, сокращение	Определение
ПАК	Программно-аппаратный комплекс
ИИ	Искусственный интеллект
Проект	Выделенное рабочее пространство в ПАК, предназначенное для взаимодействия участников проекта в задачах: объединения данных от нескольких поставщиков / из нескольких источников, обучения моделей машинного обучения на полученных данных, применения модели для получения прогнозов, имеющих бизнес-ценность для участников проекта
Контур участника (КУ)	Программно-аппаратная часть ПАК, находящаяся в физическом пользовании конкретного участника и реализующая основные интерфейсы и инструментарий использования
Контур владельца (КВ)	Программно-аппаратная часть ПАК, общая для всех участников, физически размещаемая, администрируемая и обслуживаемая Владельцем ПАК
ML-модель	Модель машинного обучения, математический алгоритм, который обучается на наборах данных для выполнения конкретной задачи, такой как классификация или регрессия, без явного программирования
API GW	Сервер-входной шлюз и маршрутизатор запросов ПАК
СУБД	Система управления базами данных
СКЗИ	Средство криптографической защиты информации
API (Application Programming Interface)	Набор правил и спецификаций, позволяющий программным приложениям взаимодействовать друг с другом и обмениваться данными
БД	База данных
S3	Объектное хранилище
TLS (Transport Layer Security)	Криптографический протокол, обеспечивающий безопасную передачу данных
Batch-применение	Применение модели в режиме, при котором прогнозирование или анализ выполняется не для отдельных экземпляров (онлайн-режим), а для группы входных данных (пакета/батча) одновременно

Термин, сокращение	Определение
НД	Набор данных
Пайплайн подготовки	Настраиваемый, сохраняемый, переиспользуемый алгоритм в системе, который предназначен для обработки наборов данных, подаваемых на вход алгоритма с сохранением результирующего набора данных на выходе
Пайплайн AUTO ML	Настраиваемый в части параметров обучения и источников данных, сохраняемый, корректируемый, переиспользуемый алгоритм в системе, который предназначен для обучения моделей машинного обучения в автоматическом режиме, а также применения модели в batch-режиме
Ядро	Вычислительный комплекс, принимающий на вход данные и управляющие воздействия, отрабатывающий исполняемые задания в автоматическом режиме, и выдающий на выходе результат в виде отчетов
CRL (Certificate Revocation List)	Список отозванных сертификатов
REST-запрос	Запрос, отправляемый по протоколу HTTP, который соответствует принципам архитектурного стиля REST (Representational State Transfer)
HTTPS-протокол	Расширение протокола HTTP, которое обеспечивает безопасную передачу данных в интернете путем шифрования.

2. Общие сведения

Правообладателем программной части программно-аппаратного комплекса «Эвирис» является Общество с ограниченной ответственностью «ГК «Иннотех» (ООО «ГК «Иннотех»).

Место нахождения и адрес правообладателя: 125167, г. Москва, вн.тер.г. муниципальный округ Аэропорт, пр-кт Ленинградский, д. 36, стр. 41, помещ. 23.

Наименование программы для ЭВМ: Программная часть программно-аппаратного комплекса «Эвирис».

3. Назначение

Программная часть программно-аппаратного комплекса «Эвирис» (далее – программа) предназначена для защищённого автоматизированного объединения данных из разных источников, в том числе автоматизированной обработки данных в целях разработки и применения моделей ИИ, функционирующих по принципу машинного обучения. Данная программа

исключает возможность внешнего доступа к данным на этапах передачи, хранения и обработки.

4. Архитектура и состав программы

4.1. Архитектура программы

Программа представляет собой программно-аппаратную среду для выполнения защищенных вычислений на основании объединенных массивов данных, предоставленных несколькими участниками. Предоставленные данные проходят обязательный этап шифрования, что исключает последующую возможность их извлечения и несанкционированного стороннего использования на любом этапе работы программы (передача, хранение, обработка данных). Результатом автоматизированных вычислений являются модели, полученные в ходе машинного обучения средствами искусственного интеллекта.

Структурно программа представляет собой Контуры Участников (далее – КУ), через которые Пользователи взаимодействуют с Контуром Владельца (далее – КВ). Обобщенная структурная схема программы представлена на рисунке 1.

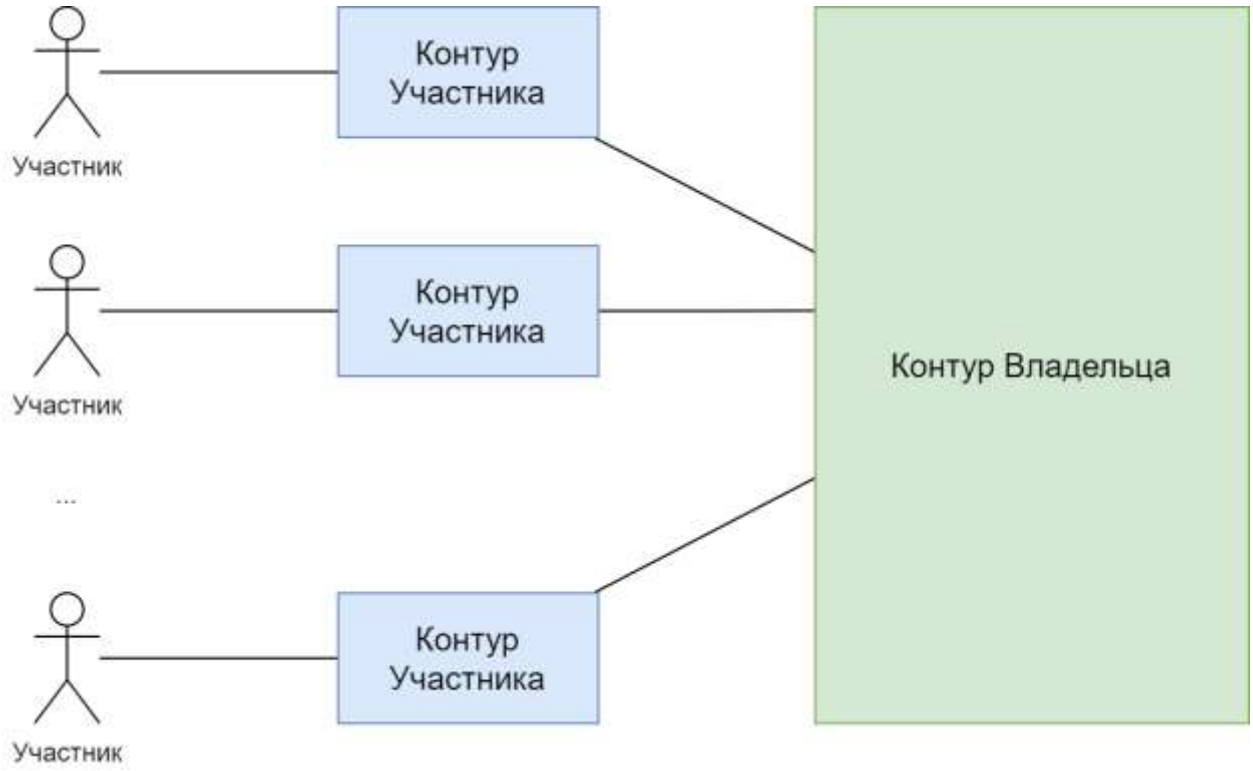

Рисунок 1. Обобщенная структура программы «ЭВИРИС»

Архитектура программы предусматривает неограниченное количество КУ, и один КВ, который, в свою очередь, так же может быть одним из

пользователей и иметь свой КУ. Владелец предоставляет пользователям возможность использования функций по разработке и применению ML-моделей.

Логическая архитектура программы представлена на рисунке 2.

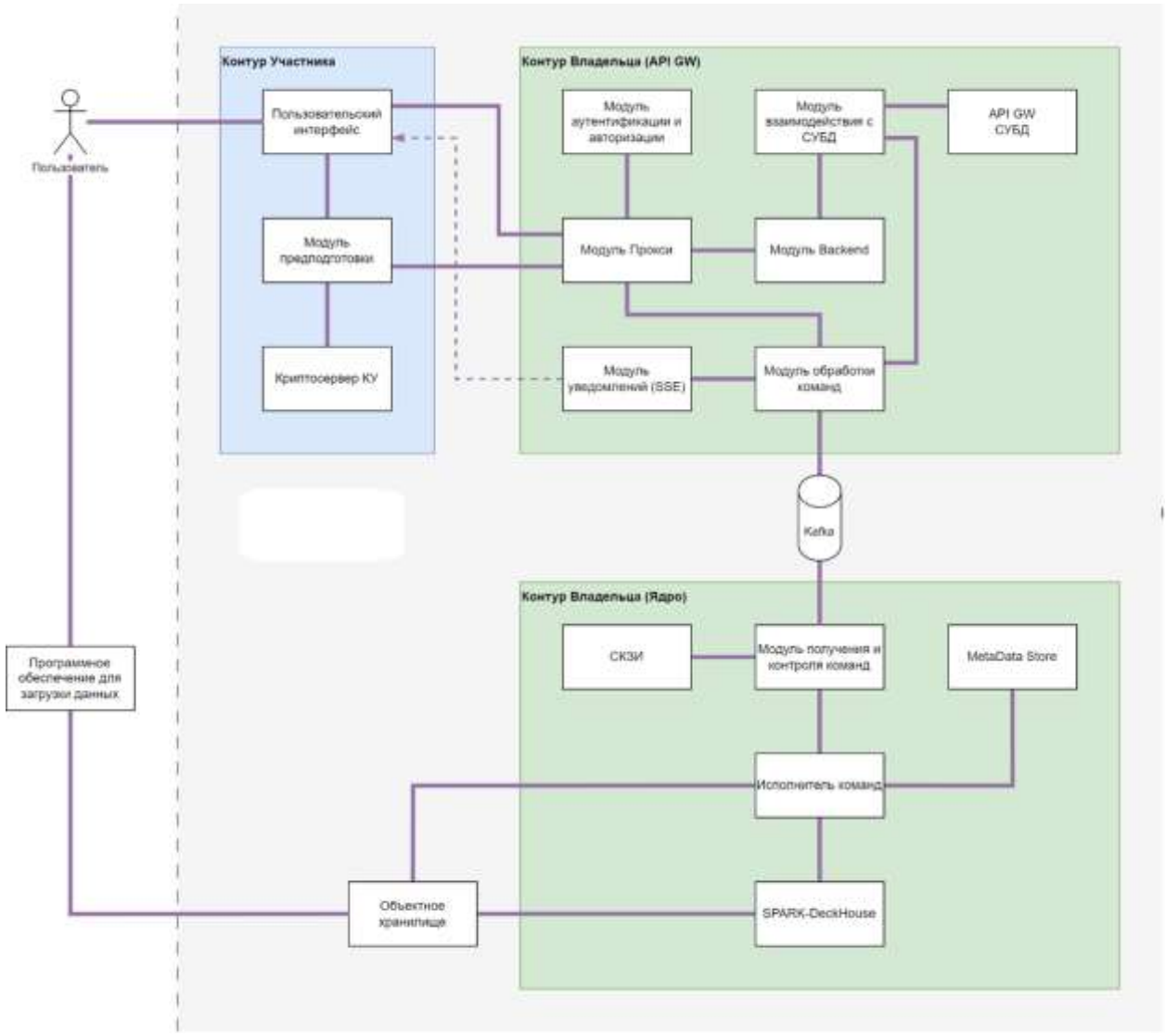

Рисунок 2. Логическая архитектура программы «ЭВИРИС»

4.2. Состав программы

Программа состоит из следующих модулей:

Контур	Модуль	Решаемые задачи
КУ	Пользовательский интерфейс	Предоставляет возможность формирования запросов через пользовательский интерфейс.
КУ	Модуль предподготовки	Модуль отвечает за обработку запросов от пользователя, поступающих через Пользовательского интерфейса. Обеспечивает асинхронное взаимодействие с внешними API, включая отправку команд и

		получение шаблонов. Отвечает за поддержание активности сессии технической учётной записи. Управляет загрузкой шаблонов, генерацией команд и их передачей для подписи в Криптосервер КУ.
КУ	Криптосервер КУ	Отвечает за подписание команд перед передачей в КВ.
КВ API GW	Модуль Прокси	Модуль Прокси предназначен для балансировки нагрузки и маршрутизации внешних запросов между различными модулями системы КВ. Также выполняет проверку прав на выполнение запроса.
КВ API GW	Модуль Backend	Предоставляет API-интерфейс для взаимодействия с пользовательским интерфейсом КУ. Модуль отвечает за обработку запросов и их перенаправление в модуль взаимодействия с базой данных.
КВ API GW	Модуль аутентификации и авторизации	Модуль предназначен для аутентификации и авторизации пользователей ПАК Эвирис.
КВ API GW	Модуль взаимодействия с СУБД	Модуль предназначен для инкапсуляции логики по взаимодействию с базой данных ПАК. Основные функции модуля включают: • Управление данными организаций, проектов, ролей и других сущностей системы. • Поддержку безопасного и быстрого взаимодействия с реляционной базой данных. Модуль обеспечивает стабильное выполнение CRUD-операций (создание, чтение, обновление, удаление) для таблиц базы данных, а также поддерживает расширенные функции, такие как запрос аутентификационных данных и другие.
КВ API GW	Модуль уведомлений (SSE)	Асинхронный сервер уведомлений. Позволяет пользователям и клиентам подписываться на события и получать уведомления в режиме реального времени.
КВ API GW	Модуль обработки команд	Модуль обработки команд предназначен для обработки входящих команд, отправки команд в соответствующие топики Kafka и обработки возвращаемых результатов.
КВ API GW	API GW СУБД	Реляционная БД для хранения информации, необходимой для работы ПАК.
КВ Ядро	Модуль получения и контроля команд (ПиКК)	Модуль получения и контроля команд (ПиКК) предназначен для обработки команд, поступающих из API Gateway (API GW) через топики Kafka, а также для отправления результатов выполнения команд обратную в КВ API GW через Kafka.
КВ Ядро	Исполнитель команд	Модуль является частью системы для оркестрации выполнения команд. Он отвечает за прием команд от других сервисов (в частности, от ПиКК), передачи их для выполнения в SPARK и обработку результатов с последующей передачей обратно отправителю.

КВ Ядро	SPARK + DeckHouse	Является набором модулей, предназначенных - для подготовки данных к использованию для обучения моделей - для автоматизированного обучения моделей машинного обучения, сравнения метрик и выбора лучшей модели. - для автоматизации процесса обработки, трансформации и инференса данных в пакетном режиме.
КВ Ядро	MetaDataStore	Представляет собой хранилище типа «ключ-значение», в котором сохраняется необходимая для работы Ядра часть информации о сущностях предметной области комплекса (проекты и т.п.), аналогичная той, что находится в API GW СУБД. Информация в MetaDataStore и API GW СУБД синхронизирована по ключам сущностей и позволяет получить в Ядре доступ к требуемым данным без выполнения дополнительных запросов в API GW.
КВ Ядро	СКЗИ	Модуль используется для проверки подписи поступающих команд и создания центра сертификации.
КВ	Объектное хранилище	Модуль отвечает за хранение наборов данных (с разными уровнями их обработки) и первичной валидации данных. Обеспечивает разделение прав доступа к данным между Участниками комплекса.

4.3. Используемый стек технологий

4.3.1. Программное и аппаратное обеспечение

Программное и аппаратное обеспечение, которое используется для функционирования программы:

- Сервер «Ядро»;
- Контур участника;
- Сервер API GW;
- Объектное хранилище.

4.3.2. Языки программирования

Языки программирования, на которых написана программа:

- Java:
  - 1.8.0 (Ядро);
  - 17.0.9 (КУ);
- Python:
  - 3.10 (КУ);
  - 3.10 (Объектное хранилище);
  - 3.11 (API Gateway);

- Scala:
  - 2.11.12 (Ядро).

4.3.3. Стандарты, технологии и библиотеки

Стандарты, технологии и библиотеки, которые используются в программе:

- Сервер Ядро:
  - java-1.8.0-openjdk-headless;
  - Scala 2.11.12;
  - Spark 2.4.5;
  - https://github.com/Shamann/griffin (доработка версии 0.7.0-SNAPSHOT);
  - https://github.com/agentlab/TransmogrifAI (доработка версии 0.7.1-SNAPSHOT);
  - Apache Livy (0.8.0-incubating-SNAPSHOT);
  - Eclipse Jetty (9.4.49.v20220914);
  - Deckhouse C.E.;
  - Nginx;
  - Redis;
  - Libgomp1;
- Сервер API Gateway:
  - Python 3.11
    - Flask
    - Flask-cors
    - Flask-restx
    - Flask-sqlalchemy
    - Sqlalchemy
    - Paramiko
    - Pysftp
    - keycloak
  - Deckhouse C.E.
  - Postgres 11.17
  - Keycloak 22.0.5

- Сервер Контура Участника
  - Airflow 2.8.1
  - Docker 20.10.2+dfsq1, build 2291f61
  - Deckhouse C.E.
  - Spark 3.3.4
  - Java-17.0.9-openjdk
  - Python 3.8.18
    - Pyspark
    - Airflow
    - Minio
    - pandas
- Объектное хранилище
  - Minio RELEASE.2023-12-20T01-00-02Z
  - Python 3.10
  - Docker 20.10.2

5. Функциональные возможности

Программа обеспечивает следующие основные функциональные возможности:

- аутентификация и авторизация пользователей;
- управление проектами, организациями и пользователями:
  - создание, удаление и редактирование проектов;
  - разграничение зон хранения и выполнения команд внутри разных проектов;
  - управление пользователями и их доступами внутри проектов;
- защищенная загрузка данных:
  - загрузка данных напрямую в S3 программы посредством защищенного и сертифицированного туннеля TLS-соединения;
  - автоматическое профилирование и валидация загруженных данных;
  - установка ограничений пользователям/организациям для загрузки данных только в разрешенные области S3;
- автоматическое слепое машинное обучение:

  - подбор ключей для объединения наборов данных – автоматический поиск подходящих ключевых полей для объединения двух наборов данных;
  - автоматическое объединение двух наборов данных;
  - автоматическая предобработка, фильтрация и очистка данных перед машинным обучением;
  - автоматическое машинное обучение с выбором модели чемпиона;
- запуск процесса batch-применения на крупном объеме данных;
- администрирование:
  - управление ролями и пользователями (создание, редактирование, удаление);
  - настройка прав доступа;
  - настройка распознавания налоговых регистров.

6. Функциональные требования

Пользовательские истории

Комплекс обеспечивает выполнение следующей функциональности:

6.1.1. Аутентификация, авторизация, смена пароля

Как пользователь, я хочу пройти аутентификацию в Системе, вводя логин и пароль, чтобы подтвердить свою личность, а затем получить доступ к функционалу Системы в соответствии с моими правами.

6.1.2. Создание проекта

Как "Владелец ПАК", я хочу создать новый Проект.

6.1.3. Удаление проекта

Как "Владелец ПАК", я хочу удалить Проект.

6.1.4. Загрузка данных

Как "Поставщик данных" или "Пользователь модели", я хочу загрузить подготовленный Набор данных в Проект.

6.1.5. Команда подбора ключей для объединения наборов данных

Как Разработчик модели, я хочу выбрать два набора данных и запустить автоматический подбор ключевых полей, чтобы получить список пар ключевых полей с оценкой их схожести для объединения данных в Пайплайне подготовки.

6.1.6. Пайплайн подготовки и создание НД

Как Разработчик модели, я хочу создать Пайплайн подготовки на основе загруженных Наборов данных, чтобы получить обогащенный Набор данных для дальнейшего анализа и использования в обучении модели или применения в обученной модели.

6.1.7. Пайплайн AUTO ML и обучение модели

Как Разработчик модели, я хочу запустить автоматический пайплайн обучения на выбранном наборе данных, указав целевую переменную, тип задачи, число фолдов и процент обучающих данных, чтобы обучить модель, сохранить её артефакт в S3 и получить отчёт о качестве для дальнейшего анализа.

6.1.8. Batch применение модели

Как Разработчик модели, я хочу запустить процедуру применения существующей модели на подходящем наборе данных.

6.1.9. Удаление модели

Как пользователь, я хочу удалить Проект.

Принципы работы

6.2. Загрузка данных.

Для того чтобы начать работу Пользователю необходимо загрузить в систему наборы данных, которые ему нужны. Эти данные сохраняются в Объектном хранилище на КВ.

Для каждого Проекта, при его создании, в Объектном хранилище создается свой бакет (директория) с соответствующими правами доступа. Каждый бакет имеет предопределенную структуру внутренних папок, одна из которых предназначена для хранения загружаемых исходных данных. При этом каждый загружаемый набор данных хранится в собственной директории.

Если данные, которые собирается загружать Пользователь, являются общедоступными, т.е. могут использоваться в разных Проектах разными Пользователями, то они сохраняются в отдельном бакете и в отдельной директории в нем.

Непосредственно для загрузки Пользователь может применять специальное программное обеспечения или написать собственный скрипт.

Загрузка данных осуществляется просто путем копирования файлов из окружения, доступного Пользователю, в защищенный КВ. При загрузке данных никаких проверок на их содержимое не происходит, т.е. данные сохраняются «как есть».

Каждый загруженный файл хранится в отдельной директории бакета. Путь к этой директории индивидуален для каждого файла. Для проектных данных он определяется названием файла и датой загрузки. Для общедоступных в пути к данным дополнительно добавляется идентификатор Организации, предоставившей данные.

Помимо непосредственно данных Пользователь можем загрузить дополнительную метаинформацию о данных, представленную в виде json-файла определенной структуры. К метаинформации относится описание данных, информация о ключевых полях и ограничения на них.

6.3. Объединение и обработка данных.

После того, как данные загружены в Объектное хранилище из можно объединить, чтобы впоследствии на этих данных осуществить обучение моделей.

Объединение и подготовка данных на основе набора правил, задаваемых пользователем осуществляются при создании Пайплана подготовки. Результатом выполнения Пайплайна подготовки становится новый (результирующий) набор данных, удовлетворяющий потребностям Пользователя. Этот набор данных, как и все ранее загруженные, является не извлекаемым из КВ и может быть использован как для обучения модели, так и для обогащения других данных при применении модели, а также для создания новых результирующих наборов через Пайплайн подготовки.

При объединении наборов данных Пользователь может воспользоваться дополнительным сервисом, который позволяет выявлять в нескольких наборах данных поля, по которым их можно объединить.

6.4. Обучение модели

Когда данные загружены, объединены и подготовлены на их основе в ПАК можно строить Модели. Построение Моделей производится в "слепом режиме", т.е. Пользователем задаются входные параметры моделирования, а дальнейшие шаги построения Модели происходят в автоматическом режиме (AutoML). На выходе этого этапа в Проекте Пользователь получает готовую Модель и подробный отчет о ее построении и параметрах. В случае, если параметры не удовлетворяют Пользователя, есть возможность удалить получившуюся Модель, изменить параметры обучения, наборы данных для обучения и перезапустить процесс обучения (AutoML).

6.5. Применение Модели

После анализа результатов обучения Модели Пользователь может принять решение об использовании Модели в своих бизнес-процессах и внедрить ее. Делается это без извлечения кода Модели и наборов данных из КВ.

Данные, для которых будет использована (применена) Модель, загружаются в систему по существующей схеме в определенное место Объектного хранилища.

Вместе с данными может быть также сохранен json-файл, содержащий дополнительную информацию.

Поступление данных для применения отслеживается и после загрузки происходит их автоматическая валидация.

После того, как данные для применения загружены, Пользователь через Пользовательский интерфейс может открыть Проект, выбрать Модель и набор данных для применения.

Перед тем как запустить применение Модели на данных, Пользователь может опционально выбрать дополнительный набор данных для их обогащения.

После выполнения этих действий Пользователь запускает операцию применения Модели и в случае успешного получает доступ к ее результатам.

Результатом работы является отчет. Перед скачиванием из КВ отчет проверяется на отсутствие возможности раскрытия исходных данных через его содержание. Отчет представляется в json-формате и может быть использован внутренними системами Участников для дальнейшей автоматической обработки.

7. Техническое обеспечение

Технически программа представляет собой клиент-серверную архитектуру, где клиентская часть – КУ, а серверная часть – КВ. Контуров Участников, в отличие от контура Владельца, может быть больше одного.

Взаимодействие между КУ и КВ производится через зашифрованный канал (VPN-туннель). Объединение контуров защищенным туннелем делает возможным их удаленное расположение. КУ представляют стандартные серверные стойки с преднастроенным программным обеспечением (далее – ПО) и оборудованием. КВ представляет физически защищенную серверную стойку с преднастроенным оборудованием и ПО.

Все ключевые манипуляции с данными (накопление, объединение, последующее моделирование на них, применение моделей) выполняется в КВ, который состоит из блоков, обеспечивающих функционирование комплекса и безопасность данных. Ключевым блоком является Ядро, которое представляет собой вычислительный комплекс, принимающий на вход данные и управляющие воздействия, отрабатывающий исполняемые задания в автоматическом режиме, и выдающий на выходе результат в виде отчетов.

Основное назначение КУ – обеспечение интерфейса взаимодействия Пользователя с КВ. Так же благодаря VPN-туннелю КУ может размещаться удаленно на территории Участника и интегрироваться с информационными системами Участника. КУ позволяет подготавливать шаблоны для данных, получать соответствующие шаблонам наборы данных, передавать и в безопасном режиме в ядро КВ, управлять посредством команд (исполняемых заданий) обработкой переданных данных, создавать на них модели, и применять их на практике.

В рамках программы предусмотрена возможность масштабирования вычислительных ресурсов, что обеспечивается вводом в состав программы дополнительных ядер, имеющих полную или ограниченную функциональность.

8. Принципы защищенности данных

8.1. Основные принципы обеспечения безопасности данных в программе

Безопасность данных в программе обеспечивается следующим:

- доступ к функциональности программы предоставляется только авторизованным пользователям в соответствии с выданными ролями;
- отсутствует возможность просмотра или выгрузки, загруженных наборов данных из программы;
- отправка команд между контурами осуществляется с обязательным подписанием их электронной подписью;
- хранение данных в программе осуществляется в зашифрованном виде;
- транспорт данных в программе осуществляется по защищенным каналам связи;
- использование данных происходит в незашифрованном виде исключительно в энергозависимой памяти, что предотвращает их утечки в случае отключения питания;
- физически защищенная стойка ограничивает неконтролируемый доступ к оборудованию на физическом уровне.

Система физзащиты инициирует, а СКЗИ выполняет удаление ключевых пар в случае несанкционированного физического доступа к Контуру Владельца.

Безопасность в программе обеспечивают следующие ключевые пары:

- Ключевая пара Участника – для работы с машинным обучением и отправки соответствующих команд в Ядро КВ;

- ключевая пара Владельца – для администрирования Ядра без прямого доступа и отправки соответствующих команд в Ядро;
- корневая ключевая пара – для формирования цепочек сертификатов для ключевых пар Участника/Владельца/TLS;
- ключевые пары TLS – формируются в процессе настройки TLS;
- ключи хранилища Ядра – для шифрования данных в хранилище Ядра.

8.2. Процедуры, исключающие компрометацию данных в программе

В случае удаления из Ядра сертификата Владельца и/или Ключевой пары Ядра будет требоваться сброс и повторная генерация ключей в программе.

В случае удаления ключевой пары Владельца из Криптосервера КУ будет требоваться сброс и повторная генерация ключей в программе.

В случае удаления корневой ключевой пары будет требоваться сброс и повторная генерация ключей в программе.

В случае удаления ключевой пары Участника будет требоваться повторное формирование ключевой пары Участника и загрузка сертификата в Ядре.

Для обеспечения защищенности и защиты от угроз реализованы следующие процессы:

- формирование корневой ключевой пары;
- формирование ключевой пары Владельца и загрузка сертификата Владельца в Ядро;
- формирование ключей хранилища Ядра;
- настройка TLS-соединений контура Владельца;
- обновление ключевой пары Владельца и загрузка сертификата Владельца в Ядро;
- формирование ключевой пары Участника и ключей TLS;
- обновление ключей хранилища Ядра;
- обновление ключевой пары Участника;
- обновление ключей TLS;
- отзыв сертификата;
- обновление корневой ключевой пары;
- обновление CRL;
- процесс передачи данных от участника к владельцу;
- процесс передачи отчетов от владельца к участнику;
- процесс передачи команд в Ядро;
- процесс распространения CRL и Корневой ключевой пары.

9. Управление потоками данных

Передача данных между модулями программы выполняется по следующим алгоритмам:

- взаимодействие КУ и API GW через API;
- передача команд (тракт команд);
- загрузка данных (тракт данных).

9.1. Взаимодействие КУ и API GW через API

Данный способ взаимодействия используется для получения на КУ данных из API GW СУБД и последующем отображении их на пользовательском интерфейсе.

Общая схема взаимодействия между модулями представлена на следующем рисунке.

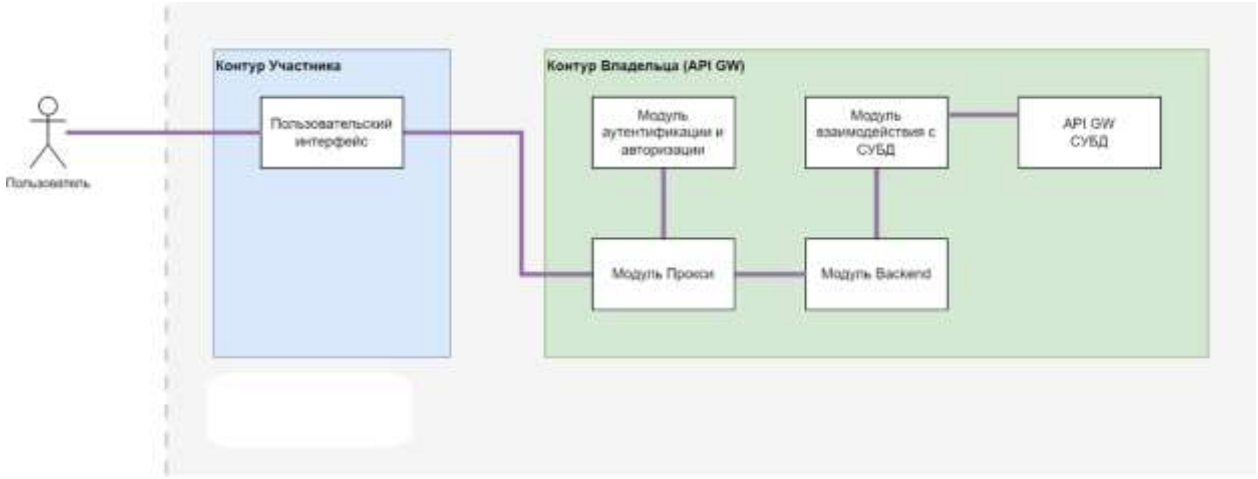

Рисунок 3. Взаимодействие модулей КУ и API GW через API

Алгоритм работы состоит в следующем:

1. Пользователь открывает страницу, на которой должны отображаться определенные данные.
2. Для получения требуемых данных модуль «Пользовательский интерфейс» оправляет на КВ соответствующий REST-запрос по протоколу HTTPS.
3. Этот запрос принимает «Модуль Прокси», который осуществляет проверку прав Пользователя на его выполнение, путем отправки подзапроса в «Модуль аутентификации и авторизации».
4. После успешной проверки прав «Модуль Прокси» перенаправляет оригинальный REST-запрос по HTTPS-протоколу в «Модуль Backend», в котором сосредоточена бизнес-логика обработки запроса.
5. Если при обработке запроса нужны данные, хранящиеся в базе данных (Модуль «API GW СУБД»), то «Модуль Backend» пересылает их в «Модуль

взаимодействия с СУБД», который в свою очередь напрямую связывается с Модулем «API GW СУБД».

6. Результат обработки запроса по обратному маршруту возвращается в КУ в Модуль «Пользовательский интерфейс», где и показывается Пользователю.

9.2. Передача команд (тракт команд)

Данный способ взаимодействия используется для передачи, сформированной Пользователем команды из КУ в КВ и ее последующей обработки.

Общая схема взаимодействия между модулями представлена на рисунке 4.

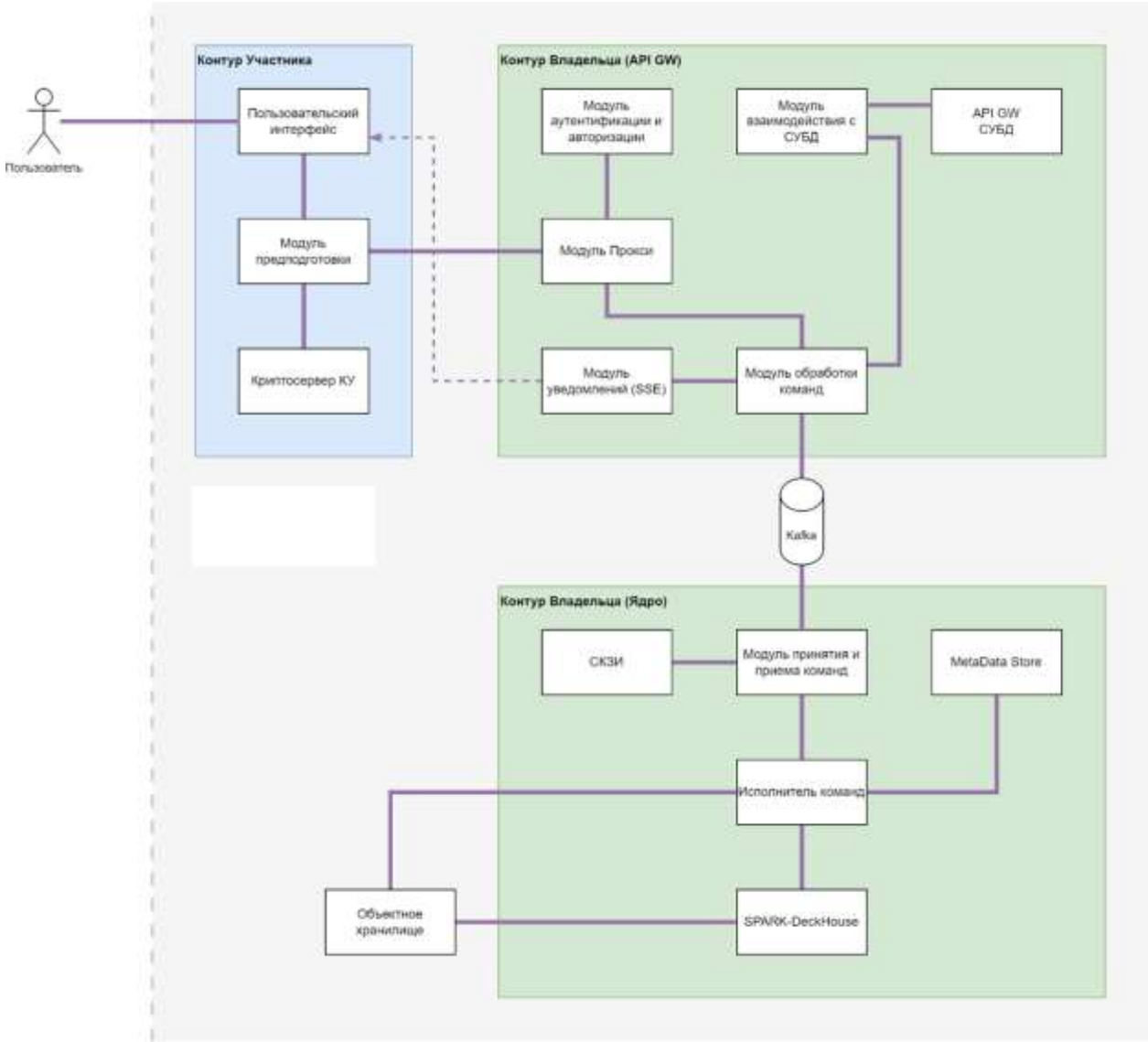

Рисунок 4. Взаимодействие модулей для тракта команд

Алгоритм работы состоит в следующем:

1. Пользователь через «Пользовательский интерфейс» вводит данные, необходимые для выполнения команды, а затем запускает команду на

выполнение. При этом формируется REST-запрос, который по протоколу HTTPS передается в «Модуль предподготовки».

2. Поступившие в «Модуль предподготовки» данные валидируется и в случае успеха по специальному шаблону создается команда – стандартизированный контейнер с данными, которая в виде файла сохраняется в локальной файловой системе КУ.

3. «Криптосервер КУ» скачивает файл с командой и генерирует для него цифровую подпись, которую также выкладывает в локальную файловую систему откуда ее «забирает» себе «Модуль предподготовки».

4. Имея команду и ее цифровую подпись «Модуль предподготовки» отправляет файлы команды и подписи REST-запросом по протоколу HTTPS в «Модуль Прокси» КВ.

5. Этот запрос принимает «Модуль Прокси», который осуществляет проверку прав Пользователя на его выполнение, путем отправки подзапроса в «Модуль аутентификации и авторизации».

6. После успешной проверки прав «Модуль Прокси» перенаправляет оригинальный REST-запрос по HTTPS-протоколу в «Модуль обработки команд», в котором сосредоточена бизнес-логика обработки запроса.

7. Если обработка каких-либо команд требует выполнения запроса к «API GW СУБД», то «Модуль обработки команд» взаимодействует посредством отправки REST-запроса по HTTPS протоколу с «Модулем взаимодействия с СУБД», который в свою очередь отправляет запросы на «API GW СУБД» и полученный ответ переправляет обратно «Модулю обработки команд».

8. «Модуль обработки команд» отправляет команду и ее цифровую подпись в брокер сообщений «Kafka» для передачи управления на сторону КВ (Ядро).

9. Команду и ее цифровую подпись из «Kafka» считывает «Модуль получения и контроля команд».

10. «Модуль получения и контроля команд» формирует REST-запрос и по протоколу HTTPS передает в модуль «СКЗИ» команду и ее цифровую подпись для проверки.

11. Модуль «СКЗИ» осуществляет проверку цифровой подписи и возвращает результат.

12. Если результат успешен, то «Модуль получения и контроля команд» передает команду в модуль «Исполнитель команд».

13. Модуль «Исполнитель команд» в зависимости от типа команды может действовать двумя способами: либо выполнить команду самостоятельно, либо отдать команду на исполнение модулю «SPARK-DeckHouse»:

a. При самостоятельном выполнении команды модуль «Исполнитель команд» по необходимости взаимодействует с «Объектным хранилищем» (через S3 протокол) для чтения записи данных и с «MetaData Store» для чтения, добавления записей типа «key-value».

b. При передаче команды на исполнение в модуль «SPARK-DeckHouse» последний осуществляет обработку данных команды, также при необходимости взаимодействуя с «Объектным хранилищем» и возвращая

результат обработки обратно в «Исполнитель команд». Если с ответом требуется произвести еще какие-то действия, то «Исполнитель команды» их производит имея при этом возможность взаимодействовать как с «Объектным хранилищем», так и с «MetaData Store».

14. Результат обработки из «Исполнителя команд» через REST-запрос по HTTPS протоколу передается в «Модуль получения и контроля команд», который отправляет его в Kafka.

15. Из «Kafka» результат выполнения команды считывается «Модулем обработки команд» и при наличии необходимости полученные данные сохраняются в «API GW СУБД» (через использование «Модуля взаимодействия с СУБД».

16. Если требуется нотифицировать пользователя об успешном завершении обработки команды, то «Модуль обработки команды» связывается с «Модулем уведомлений (SSE)», который и осуществляет доставку сообщения в «Пользовательский интерфейс» КУ.

9.3. Передача данных (тракт данных)

Данный способ взаимодействия используется для передачи наборов данных от Участников в ПАК.

Общая схема взаимодействия между модулями представлена на следующем рисунке.

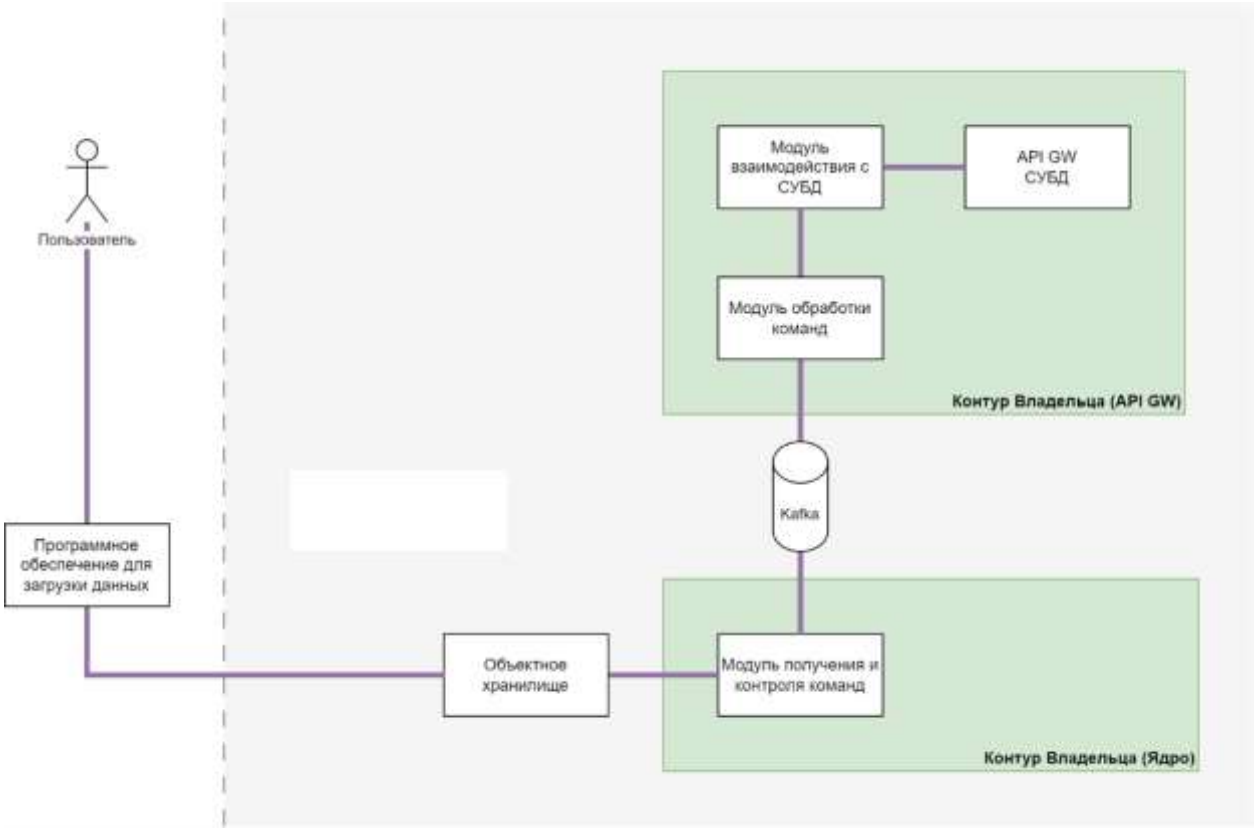

Рисунок 5. Взаимодействие модулей для тракта данных

Алгоритм работы состоит в следующем:

1. Пользователь через «Программное обеспечение для загрузки данных» загружает в «Объектное хранилище» свои наборы данные. Наборы данных загружаются по определенному пути, который определяется в том числе идентификатором проекта, в котором эти данные будут использованы. Данные могут быть использованы как для создания пайплайна подготовки, так и для применения модели. Общая схема использования тракта данных при этом не меняется.

Вместе с набором данных Пользователь может также сохранить json-файл, содержащий дополнительную информацию об нем.

2. Изменения данных в «Объектном хранилище» отслеживаются специальным модулем, который передает информацию об этих изменениях в «Модуль получения и контроля команд».
3. «Модуль получения и контроля команд» составляет соответствующий запрос и отправляет его в Kafka.
4. Из «Kafka» результат выполнения команды считывается «Модулем обработки команд» и полученные данные сохраняются в «API GW СУБД» (через использование «Модуля взаимодействия с СУБД».

10. Запуск программы и инициализация ее элементов

В рамках процесса инициализации программы достигаются следующие цели:

- подготовка программы к эксплуатации;
- обеспечение возможности хранения данных в защищенном виде;
- обеспечение конфиденциальности параметров доступа обслуживающего персонала.

Перед началом инициализации выполняются следующие действия:

- создаются все системные учетные записи;
- создаются учетные записи Владельца для входа в веб-интерфейс программы;
- данные для авторизации передаются Владельцу;
- выполняется развертывание программы у Владельца.

Процесс инициализации состоит из следующих шагов:

1. Формирование корневой ключевой пары.
2. Формирование ключевой пары Владельца и загрузка ее в Ядро.
3. Настройка TLS-соединений.
4. Формирование ключей хранилища.
5. Обновление корневой ключевой пары.
6. Обновление ключей TLS.

7. Обновление ключевой пары Владельца и загрузка сертификата Владельца в Ядро.

11. Организация входных данных

11.1. Данные, необходимых для запуска программы

Входными данными для работы комплекса являются файлы в формате parquet.

Эти файлы используются как контейнеры для наборов данных, которые обогащаются друг другом и используются для обучения моделей.

На файлах этого же формата осуществляется применение моделей и получение финального результата.

12. Организация выходных данных

К отчетам системы относятся следующие данные.

12.1. Отчет о результатах подбора ключевых полей

Отчет представляет собой JSON документ, содержащий информацию о датасетах, чьи поля с указанной вероятностью (поле weight) могут использоваться, как ключевые.

Схематический пример отчета:

```
{
 "match_keys":
 [
  {
   "first_field":
   {
    "dataset_id": <id датасета>,
    "dataset_name": <название датасета>,
    "dataset_field": <название поля датасета>
   },
   "second_field":
   {
    "dataset_id": <id датасета>,
    "dataset_name": <название датасета>,
    "dataset_field": <название поля датасета>
```

```
    },
    "weight": <0-100 - вероятность соответствия>
   },
   ...
 ]
}
```

12.2. Отчет о создании пайплайна подготовки

Отчет представляет собой JSON-документ, содержащий информацию об операциях, которые были выполнены для получения итогового набора данных. Речь прежде всего идет о том, какие признаки были удалены.

Пример отчета:.

```
[
 "Удален признак tele2_calls_1_kvartal_2024.start_call_time",
 "Удален признак tele2_calls_1_kvartal_2024.finish_call_time",
 ...
]
```

12.3. Отчет об обучении модели

Отчет представляет собой JSON-документ, содержащий информацию о моделях, которые были использованы в процессе обучения, их характеристиках и результаты их взаимного оценивания для решения поставленной задачи на обучающем наборе данных.

Пример отчета:

```
{
 "best_model": {
  "metrics": {
   "test": {
    "f1": 0.761,
    "precision": 0.894,
    "recall": 0.692,
    "rocAuc": 0.737
   },
   "train": {
    "f1": 0.768,
    "precision": 0.9,
```

```
      "recall": 0.701,
      "rocAuc": 0.774
    }
  },
  "name": "XGBoost"
},
"trained_models": [
  {
    "model": "RandomForest",
    "test_f1": 0.871,
    "test_precision": 0.879,
    "test_recall": 0.864,
    "test_rocAuc": 0.709,
    "train_f1": 0.932,
    "train_precision": 0.949,
    "train_recall": 0.924,
    "train_rocAuc": 0.966
  },
  {
    "model": "XGBoost",
    "test_f1": 0.761,
    "test_precision": 0.894,
    "test_recall": 0.692,
    "test_rocAuc": 0.737,
    "train_f1": 0.768,
    "train_precision": 0.9,
    "train_recall": 0.701,
    "train_rocAuc": 0.774
  },
  {
    "model": "LogisticRegression",
    "test_f1": 0.756,
    "test_precision": 0.894,
```

```
    "test_recall": 0.685,
    "test_rocAuc": 0.737,
    "train_f1": 0.759,
    "train_precision": 0.896,
    "train_recall": 0.688,
    "train_rocAuc": 0.748
   }
 ]
}
```


12.4. Отчет о применении модели в batch режиме

Отчет представляет собой JSON-документ, содержащий информацию об данных модели и результатах ее применения на наборе данных.

Схематический причем отчета:


```
{
 "model_name":"VTB_1",
 "mergerdata": true,
 "created_at": "2025-03-11T12:45:23+00:00",
 "result": {
   "id1": 0.8235,
   "id2": 0.4567,
   "id3": 0.9083
 }
}
```