**Документация, содержащая информацию, необходимую для эксплуатации экземпляра программного обеспечения Модуль «Инфраструктура для сервиса»**

**Содержание**

[1. Предназначение ПО 3](#_s2xpnm9x5zoo)

[2. Описание функциональных характеристик 3](#_7ep7zxhbxr7i)

[3. Архитектура модуля «Сквозная аналитика» 4](#_fuxsqccpbuer)

[4. Схема работы 6](#_spk4gfqhsfnh)

[5. Используемые языки программирования и фреймворки 7](#_pf377y6aljju)

[6. Эксплуатация системы 8](#_s1q4gko3eb1g)

# **Предназначение ПО**

Модуль «Инфраструктура для сервиса» инфраструктуру для загрузки, хранения и обработки больших объемов данных, а также интеграцию с другими модулями Предикто. Включает S3-совместимое объектное хранилище, ПО для оркестрации данных и коннекторы для выгрузки данных из популярных источников.

Основные задачи модуля:

* Организация хранения данных;
* Обработка больших данных;
* Оркестрация данных;
* Интеграция с источниками данных;
* Обеспечение качества данных;
* Обеспечение взаимодействия между модулями.

# **Описание функциональных характеристик**

2.1 Организация хранения данных

Модуль предоставляет объектное хранилище для больших объемов данных, которое разделяется на несколько логических слоев:

raw — необработанные данные, поступившие из внешних источников.

curated — очищенные и подготовленные данные для аналитики.

aggregated — агрегированные данные, готовые для использования в аналитических отчетах и моделях.

Каждому слою данных присваиваются соответствующие права доступа, что позволяет точно контролировать, кто и какие данные может использовать. Также модуль поддерживает интеграцию с ПО для хранения данных в формате Parquet и Delta Lake или аналогичными форматами, что обеспечивает гибкость в работе с различными типами данных.

2.2 Обработка и оркестрация данных

Модуль использует масштабируемую и распределенную систему для обработки данных на основе открытой технологии Apache Spark. Это позволяет эффективно обрабатывать большие объемы данных с использованием параллельных вычислений.

Также включен компонент оркестрации данных, который позволяет:

* Запускать код на языке Python для выполнения различных задач обработки.
* Планировать выполнение задач по расписанию.
* Отслеживать статус выполнения задач, а также ошибки, возникшие при обработке данных.
* Управлять зависимостями между задачами, чтобы гарантировать корректность выполнения всех шагов обработки данных.

2.3 Интеграция с источниками данных

Модуль предоставляет готовые компоненты (коннекторы) для загрузки данных из различных источников, включая реляционные и нереляционные базы данных, такие как:

PostgreSQL

ClickHouse

MongoDB

Кроме того, доступны коннекторы к популярным сервисам аналитики пользовательских событий, таким как Яндекс.Метрика и AppMetrica, для загрузки данных через их API. Для обработки данных в реальном времени также предусмотрена возможность интеграции с потоковыми источниками данных через Kafka.

2.4 Обеспечение качества данных

Модуль включает инструменты для мониторинга качества данных, которые интегрируются в процессы загрузки и обработки. Это обеспечивает:

* Контроль за качеством данных на каждом этапе их обработки.
* Технологические возможности для загрузки данных как в потоковом, так и в пакетном режиме.
* Решения для отслеживания зависимости данных, их происхождения и изменений, что позволяет обеспечить прозрачность и надежность в работе с данными.

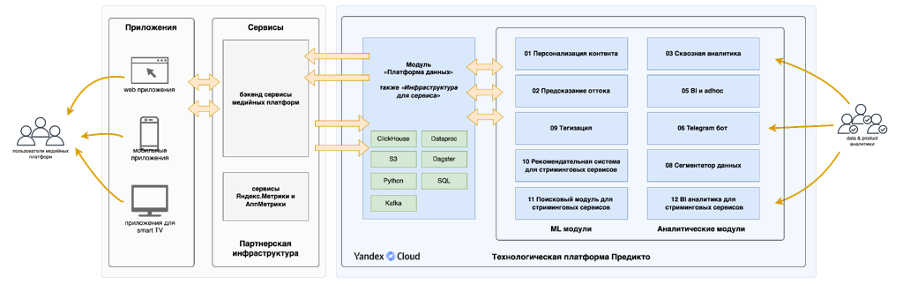
2.5 Обеспечение взаимодействия модулей

Возможность получать данные из модуля «Инфраструктура для сервиса» в необходимом объеме и скорости, как в потоковом, так и в пакетном режиме, используя стандартизированные протоколы и форматы данных.

Загружать данные в модуль «Инфраструктура для сервиса» из других модулей платформы Предикто с использованием тех же стандартов, что обеспечивает безошибочный обмен данными между модулями.

# **Архитектура модуля «Инфраструктура для сервиса»**

Архитектура строится вокруг интеграции приложений, данных и аналитики для создания персонализированного опыта и поддержки аналитических задач, используя мощные инфраструктурные и ML инструменты.



*Рис.1 Концептуальная архитектура модуля «Инфраструктура для сервиса»*

Пользователи медийных платформ взаимодействуют с различными приложениями (web, мобильными, для smart TV) через платформу.

Web-приложения, мобильные приложения и приложения для smart TV направляют данные к сервисам.

Сервисы, предоставляемые для медийных платформ, включают backend-сервисы для обработки данных и сервисы аналитики, такие как Яндекс.Метрика и AppMetрика.

Инфраструктура для сервиса включает модуль «Платформа данных», который используется для обработки и хранения данных. ClickHouse, Dataproc, S3, Dagster, Python, SQL, Kafka — различные технологии и инструменты для обработки данных.

ML модули. Модели машинного обучения (ML), такие как персонализация контента, предсказание оттока, теги, рекомендательные системы, поисковые модули и другие, используют данные из платформы для создания аналитики.

Аналитические модули. Подключение к аналитическим системам для выполнения задач сквозной аналитики, BI, работы с Telegram-ботами, сегментированием данных и создания BI-анализа для стриминговых сервисов.

Data & product аналитики. Данные и аналитики продуктов осуществляют мониторинг и анализ всей работы платформы, обеспечивая связь между сервисами и модулями.

# **Технические и эксплуатационные характеристики**

4.1. Доступность инфраструктуры

Критически важные приложения и данные реплицируются в три дата-центра, расположенных в разных зонах доступности, что обеспечивает высокую степень отказоустойчивости и минимизирует риск потери данных в случае локальных сбоев. Инфраструктура предоставляет возможности для надежного хранения и регулярного бэкапирования данных, что гарантирует сохранность информации в любых условиях. Для обеспечения бизнес-непрерывности предусмотрены механизмы аварийного восстановления, которые активируются при масштабных сбоях или катастрофах, затрагивающих одну из зон доступности. Инфраструктура гарантирует возможность эксплуатации высокодоступных и высоконагруженных приложений, которые способны выдерживать значительные нагрузки.

4.2. Архитектура модулей.

При проектировании системы использована модульная архитектура, что позволяет разбить платформу «Предикто» на отдельные функциональные части. Это обеспечивает независимую разработку, тестирование и внедрение каждого модуля, а также упрощает их интеграцию через модуль «Платформа данных». Разработана система, оптимизирующая потребление аппаратных ресурсов. В частности, используются такие подходы, как кэширование данных и распределенные вычисления, что значительно повышает производительность системы.

4.3. Использование Яндекс.Облака.

Модуль «Платформа данных» был разработан с использованием публичной облачной платформы Яндекс.Облако. Все интеграции с другими модулями платформы проектируются с учетом того, что они также будут развернуты в Яндекс.Облаке. Для снижения операционных издержек, связанных с созданием и эксплуатацией инфраструктуры, активно используются управляемые сервисы, предоставляемые Яндекс.Облаком. Это позволяет существенно уменьшить затраты на обслуживание и повышение доступности.

4.4. Масштабируемость инфраструктуры

Инфраструктура должна обеспечивать горизонтальное масштабирование программного обеспечения до 2000 одновременно используемых CPU и 20 одновременно используемых GPU, что позволяет справляться с большими объемами данных и интенсивными вычислениями. Для долгосрочного хранения данных система поддерживает хранение нескольких петабайт информации, обеспечивая надежную работу с большими объемами данных. Масштабирование инфраструктуры и выделение новых ресурсов может происходить в течение нескольких минут, в отдельных случаях — до нескольких часов, что позволяет оперативно адаптировать систему к изменяющимся требованиям. Для обеспечения высокой доступности и масштабируемости программного обеспечения используются такие методы, как кластеризация серверов и балансировка нагрузки.

4.5. Требования к окружениям.

В рамках модуля предусмотрено разделение контуров для разработки и контуров промышленной эксплуатации, что позволяет изолировать процессы разработки от реальной эксплуатации на уровне доступа к инфраструктуре и данным. Каждый функциональный компонент развернут как минимум в двух окружениях: development (для разработки и тестирования) и production (для промышленной эксплуатации). При необходимости могут быть созданы дополнительные окружения, такие как staging (предварительное тестирование) или QA (контроль качества).

# 

# **5.Используемые языки программирования и фреймворки**

Язык - Python:

Библиотеки и фреймворки

Polars — это высокопроизводительная библиотека для обработки данных, разработанная как альтернатива Pandas. Она использует многопоточность для эффективной работы с большими объемами данных и является одним из ключевых инструментов для анализа данных в модуле.

Pandas — это популярная библиотека Python для работы с табличными данными. Она предоставляет мощные структуры данных (например, DataFrame), которые позволяют эффективно обрабатывать, очищать и анализировать данные в табличном формате.

Delta Lake — это слой хранения данных для Apache Spark, который поддерживает транзакции ACID, улучшая надежность данных. Он используется для управления большими наборами данных с возможностью версионирования и поддержкой гибкой обработки данных.

PyArrow — это библиотека Python для работы с форматом Apache Arrow, который используется для быстрого обмена данными между различными системами и обеспечения эффективной сериализации. PyArrow используется для оптимизации производительности в распределенных вычислениях.

Dagster — это фреймворк для оркестрации данных, который предоставляет возможности для разработки и управления конвейерами данных. Он используется для создания, тестирования и управления сложными задачами и процессами обработки данных.

PySpark — это интерфейс Python для Apache Spark, который используется для распределенной обработки данных. Он позволяет обрабатывать большие объемы данных с использованием кластеров вычислительных ресурсов и поддерживает работу с данными в реальном времени и пакетном режиме.

# **6. Эксплуатация системы**

Развертывание и подготовка окружений.

Прежде чем запустить систему, необходимо развернуть инфраструктуру в соответствующих окружениях. Это включает подготовку окружений development (для разработки и тестирования) и production (для промышленной эксплуатации).

Установка и настройка зависимостей:

Все необходимые зависимости и библиотеки должны быть установлены через PIP.

Настройка системы.

После установки зависимостей требуется настроить конфигурационные файлы системы. Это включает параметры подключения к хранилищу данных, настройку сетевых соединений и обеспечение безопасности системы.

Запуск компонентов системы.

После подготовки и настройки всех компонентов можно запускать основные сервисы, такие как Apache Spark, Kafka, Dagster, а также базы данных, хранилища данных и другие компоненты, необходимые для функционирования системы.

Проверка работоспособности.

После запуска системы необходимо провести проверку работоспособности всех компонентов, чтобы убедиться в корректности их взаимодействия и функциональности. Это включает в себя тестирование обработки и загрузки данных, мониторинг производительности и проверку целостности данных.