**Техническое задание на модернизацию Информационной системы управления образованием (ИСУО) в формате аналитического хранилища данных для Министерства образования и науки Кыргызской Республики**

**1. Общие сведения**

**1.1. Наименование системы**

**Информационная система управления образованием (ИСУО)** Министерства образования и науки Кыргызской Республики (далее - Система)

**1.2. Назначение и цели создания системы**

Система предназначена для консолидации, хранения и анализа данных в сфере образования Кыргызской Республики. Целью создания Системы является поддержка BI-сценариев (дашборды, OLAP-анализ, отчётность) и организация «озера данных» (Data Lake) для всех структурных единиц МОН КР.

**Основные цели:**

* Обеспечить единое централизованное хранилище для сбора и анализа образовательных данных
* Повысить качество и оперативность принятия управленческих решений
* Обеспечить прозрачность образовательных процессов
* Создать основу для внедрения аналитических и прогнозных моделей в сфере образования

**2. Требования к системе**

**2.1. Общие требования**

* Использование только open-source технологий
* Обеспечение работоспособности в среде Linux
* Возможность пакетной обработки данных с перспективой внедрения потоковой обработки
* Минимальные требования к оборудованию и отсутствие дорогостоящих проприетарных решений
* Масштабируемость и отказоустойчивость системы

**2.2. Требования к объёмам данных**

* Система должна обеспечивать хранение и обработку терабайтов данных ежегодно
* Поддержка исторических данных за несколько лет
* Возможность хранения детализированных и агрегированных данных

**2.3. Требования к архитектуре**

* Гибридная архитектура "Data Lake + Data Warehouse" (Lakehouse)
* Использование пакетной архитектуры с возможностью расширения до Lambda-архитектуры (добавление стримингового слоя)
* Модульность и компонентная структура, позволяющая независимое масштабирование каждого слоя

**3. Архитектура системы**

**3.1. Логическая архитектура**

Система должна включать следующие логические слои:

1. **Слой источников данных**: разнородные системы, предоставляющие данные (БД, файлы, API)
2. **Слой интеграции (ингест)**: Apache NiFi для сбора данных из источников
3. **Озеро данных (Data Lake)**: хранилище для сырых данных
4. **Слой обработки и хранения (Data Warehouse)**: ClickHouse в качестве аналитического хранилища
5. **Слой BI и аналитики**: инструменты визуализации и анализа данных
6. **Слой мониторинга и логирования**: контроль состояния системы

**3.2. Компонентная архитектура**

**3.2.1. Системы хранения данных**

**Основное аналитическое хранилище**: ClickHouse

Требования к реализации:

* Колоночное хранение данных с компрессией
* Партиционирование данных по времени (например, по учебным годам)
* Организация таблиц по схеме "звезда" или "снежинка" для аналитических запросов
* Настройка MergeTree-движка с оптимальными параметрами для образовательных данных
* Обеспечение репликации для отказоустойчивости (минимум 2 реплики)
* Настройка шардирования при росте объемов данных

**Озеро данных (Data Lake)**: MinIO

Требования к реализации:

* Объектное хранилище с поддержкой S3-совместимого API
* Организация данных по зонам: landing (сырые данные) и curated (очищенные)
* Структурирование хранилища по источникам данных и датам
* Настройка бакетов и политик хранения данных
* Обеспечение избыточности для отказоустойчивости

**3.2.2. Инструменты оркестрации ETL**

**Сбор данных**: Apache NiFi

Требования к реализации:

* Настройка процессоров для каждого источника данных
* Создание потоков данных для извлечения, первичной обработки и загрузки в Data Lake
* Настройка мониторинга и логирования процессов извлечения
* Обеспечение отказоустойчивости через кластеризацию (минимум 2 узла)
* Разработка шаблонов потоков для типовых источников данных

**Обработка данных**: Apache Airflow

Требования к реализации:

* Разработка DAG (Directed Acyclic Graph) для пакетной обработки данных
* Настройка планирования заданий (ежедневные, еженедельные, ежемесячные процессы)
* Реализация процессов трансформации, очистки и загрузки данных в ClickHouse
* Обеспечение обработки ошибок и механизмов повторного запуска задач
* Настройка оповещений о сбоях в процессах
* Интеграция с системой мониторинга

**3.2.3. Инструменты BI и визуализации**

**Основной инструмент BI**: Metabase

Требования к реализации:

* Настройка подключения к ClickHouse
* Создание базовых дашбордов для анализа ключевых показателей
* Настройка пользовательских ролей и прав доступа
* Создание предопределенных вопросов и аналитических представлений
* Настройка расписания отправки отчетов

**Расширенный инструмент BI**: Apache Superset (опционально)

Требования к реализации:

* Настройка подключения к ClickHouse
* Создание продвинутых аналитических дашбордов
* Настройка детальной системы ролей и прав доступа (RBAC)
* Интеграция с корпоративными системами аутентификации (при наличии)

**3.2.4. Инфраструктура и развертывание**

**Контейнеризация**: Docker/Kubernetes

Требования к реализации:

* Разработка Docker-образов для всех компонентов системы
* Создание Docker Compose файлов для развертывания
* Разработка Kubernetes-манифестов (опционально для продакшн-среды)
* Настройка сетевого взаимодействия между контейнерами
* Настройка хранения данных через постоянные тома

**Автоматизация инфраструктуры**: Terraform, Ansible

Требования к реализации:

* Разработка Terraform-скриптов для провиженинга инфраструктуры
* Создание Ansible-плейбуков для настройки серверов и развертывания компонентов
* Автоматизация процессов установки и обновления системы
* Создание инструментов для резервного копирования и восстановления

**Мониторинг и логирование**:

Требования к реализации:

* Настройка Prometheus и Grafana для мониторинга состояния компонентов
* Настройка сбора и централизации логов
* Создание дашбордов мониторинга для ключевых метрик системы
* Настройка алертинга при критических событиях

**4. Требования к функциональности системы**

**4.1. Сбор и интеграция данных**

**4.1.1. Источники данных**

Система должна поддерживать следующие типы источников:

* Реляционные базы данных (PostgreSQL, MySQL, др.)
* Файлы CSV/Excel
* API внешних систем
* Структурированные и полуструктурированные данные

**4.1.2. Процессы интеграции**

* Извлечение данных по расписанию (инкрементальная загрузка)
* Валидация входящих данных
* Логирование процессов извлечения
* Отслеживание изменений в источниках данных

**4.2. Хранение и обработка данных**

**4.2.1. Структура хранения**

* Организация многослойного хранения (raw, stage, datamart)
* Реализация историчности данных (SCD - Slowly Changing Dimensions)
* Поддержка версионности данных
* Оптимизация хранения для аналитических запросов

**4.2.2. Процессы обработки**

* Очистка и нормализация данных
* Трансформация в соответствии с бизнес-правилами
* Формирование агрегатов и витрин данных
* Расчет ключевых показателей эффективности (KPI)

**4.3. Аналитика и отчетность**

**4.3.1. Дашборды и визуализации**

* Создание интерактивных дашбордов для различных категорий пользователей
* Построение графиков, диаграмм и таблиц по ключевым показателям
* Возможность детализации (drill-down) показателей
* Экспорт отчетов в различные форматы (PDF, Excel)

**4.3.2. Аналитические возможности**

* Ad-hoc запросы к данным
* Многомерный анализ данных (OLAP)
* Фильтрация и группировка данных
* Сравнительный анализ показателей

**5. Требования к масштабированию и отказоустойчивости**

**5.1. Масштабирование**

* Горизонтальное масштабирование ClickHouse через добавление узлов
* Масштабирование NiFi через добавление узлов в кластер
* Увеличение числа воркеров Airflow при росте нагрузки
* Расширение хранилища Data Lake через добавление узлов MinIO

**5.2. Отказоустойчивость**

* Репликация данных в ClickHouse
* Кластеризация NiFi для отказоустойчивости
* Дублирование критических компонентов
* Автоматический перезапуск контейнеров при сбоях
* Периодическое резервное копирование

**6. Требования к безопасности**

**6.1. Аутентификация и авторизация**

* Настройка пользовательских ролей в BI-инструментах
* Разграничение доступа к данным на уровне хранилища
* Интеграция с корпоративными системами аутентификации (опционально)

**6.2. Защита данных**

* Шифрование данных при передаче
* Шифрование чувствительных данных при хранении
* Настройка сетевых правил и брандмауэров
* Логирование доступа к данным

**7. Этапы реализации проекта**

**7.1. Этап 1: Подготовка и проектирование**

* Детальное проектирование архитектуры
* Подготовка инфраструктуры
* Создание тестовой среды
* Разработка скриптов автоматизации

**7.2. Этап 2: Развертывание базовой инфраструктуры**

* Установка и настройка MinIO (Data Lake)
* Установка и настройка ClickHouse
* Настройка мониторинга базовых компонентов

**7.3. Этап 3: Интеграция источников данных**

* Установка и настройка Apache NiFi
* Разработка потоков данных для ключевых источников
* Тестирование процессов извлечения

**7.4. Этап 4: Разработка процессов обработки**

* Установка и настройка Apache Airflow
* Разработка DAG для обработки данных
* Тестирование процессов трансформации и загрузки

**7.5. Этап 5: Настройка аналитики и визуализации**

* Установка и настройка Metabase
* Создание базовых дашбордов
* Настройка прав доступа
* Опционально: установка и настройка Superset

**7.6. Этап 6: Тестирование и оптимизация**

* Комплексное тестирование системы
* Оптимизация производительности
* Нагрузочное тестирование
* Устранение выявленных проблем

**7.7. Этап 7: Ввод в эксплуатацию**

* Миграция в продакшн-среду
* Обучение пользователей
* Подготовка эксплуатационной документации
* Запуск в промышленную эксплуатацию

**8. Требования к документации**

**8.1. Техническая документация**

* Документация по архитектуре системы
* Руководство по установке и настройке
* Инструкции по резервному копированию и восстановлению
* Документация API (при наличии)

**8.2. Пользовательская документация**

* Руководство пользователя BI-инструментов
* Инструкции по созданию дашбордов и отчетов
* Обучающие материалы

**9. Требования к тестированию**

**9.1. Функциональное тестирование**

* Тестирование процессов сбора данных
* Тестирование процессов обработки
* Тестирование аналитических возможностей
* Проверка корректности данных в отчетах

**9.2. Нагрузочное тестирование**

* Тестирование под нагрузкой при пакетной загрузке данных
* Тестирование при одновременной работе множества пользователей
* Тестирование масштабируемости компонентов

**9.3. Тестирование отказоустойчивости**

* Проверка восстановления компонентов после сбоев
* Тестирование процессов резервного копирования и восстановления
* Имитация отказа узлов кластера

**10. Требования к сопровождению**

**10.1. Мониторинг и поддержка**

* Проактивный мониторинг компонентов системы
* Регулярные проверки производительности
* Процедуры обновления компонентов
* Резервное копирование и архивирование данных

**10.2. Развитие системы**

* Добавление новых источников данных
* Внедрение дополнительных аналитических возможностей
* Подготовка к переходу на потоковую обработку данных (Lambda-архитектура)
* Интеграция с системами машинного обучения (в перспективе)

**11. Глоссарий**

* **ETL/ELT** - Extract, Transform, Load / Extract, Load, Transform - процессы извлечения, преобразования и загрузки данных
* **Data Lake** - озеро данных, хранилище сырых данных в различных форматах
* **Data Warehouse** - хранилище данных, оптимизированное для аналитики
* **BI** - Business Intelligence, бизнес-аналитика
* **OLAP** - Online Analytical Processing, аналитическая обработка в реальном времени
* **DAG** - Directed Acyclic Graph, направленный ациклический граф (в контексте Airflow)
* **SCD** - Slowly Changing Dimensions, медленно меняющиеся измерения

**12. Приложения**

**12.1. Пример Docker Compose для базового развертывания**

version: '3'

services:

# ClickHouse

clickhouse-server:

image: yandex/clickhouse-server

volumes:

- clickhouse-data:/var/lib/clickhouse

- ./config/clickhouse:/etc/clickhouse-server/conf.d

ports:

- "8123:8123"

- "9000:9000"

restart: always

# Zookeeper (для ClickHouse кластера)

zookeeper:

image: zookeeper:3.7

volumes:

- zookeeper-data:/data

- zookeeper-logs:/datalog

ports:

- "2181:2181"

restart: always

# MinIO (Data Lake)

minio:

image: minio/minio

volumes:

- minio-data:/data

environment:

MINIO\_ACCESS\_KEY: minioadmin

MINIO\_SECRET\_KEY: minioadmin

command: server /data

ports:

- "9001:9000"

restart: always

# Apache NiFi

nifi:

image: apache/nifi:latest

volumes:

- nifi-conf:/opt/nifi/nifi-current/conf

- nifi-logs:/opt/nifi/nifi-current/logs

- nifi-state:/opt/nifi/nifi-current/state

ports:

- "8080:8080"

environment:

NIFI\_WEB\_HTTP\_PORT: 8080

restart: always

# Apache Airflow

airflow-webserver:

image: apache/airflow:latest

depends\_on:

- postgres

environment:

AIRFLOW\_\_CORE\_\_SQL\_ALCHEMY\_CONN: postgresql+psycopg2://airflow:airflow@postgres/airflow

AIRFLOW\_\_CORE\_\_EXECUTOR: LocalExecutor

volumes:

- ./dags:/opt/airflow/dags

- ./logs:/opt/airflow/logs

- ./plugins:/opt/airflow/plugins

ports:

- "8081:8080"

command: webserver

restart: always

airflow-scheduler:

image: apache/airflow:latest

depends\_on:

- postgres

environment:

AIRFLOW\_\_CORE\_\_SQL\_ALCHEMY\_CONN: postgresql+psycopg2://airflow:airflow@postgres/airflow

AIRFLOW\_\_CORE\_\_EXECUTOR: LocalExecutor

volumes:

- ./dags:/opt/airflow/dags

- ./logs:/opt/airflow/logs

- ./plugins:/opt/airflow/plugins

command: scheduler

restart: always

# PostgreSQL для Airflow

postgres:

image: postgres:13

environment:

POSTGRES\_USER: airflow

POSTGRES\_PASSWORD: airflow

POSTGRES\_DB: airflow

volumes:

- postgres-data:/var/lib/postgresql/data

restart: always

# Metabase

metabase:

image: metabase/metabase

ports:

- "3000:3000"

environment:

MB\_DB\_TYPE: postgres

MB\_DB\_DBNAME: metabase

MB\_DB\_PORT: 5432

MB\_DB\_USER: metabase

MB\_DB\_PASS: metabase

MB\_DB\_HOST: postgres-metabase

depends\_on:

- postgres-metabase

restart: always

# PostgreSQL для Metabase

postgres-metabase:

image: postgres:13

environment:

POSTGRES\_USER: metabase

POSTGRES\_PASSWORD: metabase

POSTGRES\_DB: metabase

volumes:

- postgres-metabase-data:/var/lib/postgresql/data

restart: always

# Prometheus

prometheus:

image: prom/prometheus

volumes:

- ./config/prometheus:/etc/prometheus

- prometheus-data:/prometheus

ports:

- "9090:9090"

restart: always

# Grafana

grafana:

image: grafana/grafana

volumes:

- grafana-data:/var/lib/grafana

ports:

- "3001:3000"

restart: always

volumes:

clickhouse-data:

zookeeper-data:

zookeeper-logs:

minio-data:

nifi-conf:

nifi-logs:

nifi-state:

postgres-data:

postgres-metabase-data:

prometheus-data:

grafana-data:

**12.2. Минимальные системные требования**

* **Сервер для базовых компонентов**:
  + CPU: 8+ ядер
  + RAM: 32+ ГБ
  + Диск: 500+ ГБ SSD (система), 2+ ТБ HDD/SSD (данные)
  + ОС: Linux (Ubuntu Server 20.04 LTS или выше)
* **Дополнительные серверы для кластера (опционально)**:
  + Аналогичные характеристики

**13. Примечания**

* Все компоненты системы должны быть совместимы с последними стабильными версиями используемых технологий.
* При росте объемов данных и нагрузки может потребоваться дополнительное оборудование.
* Рекомендуется начать с одноузлового развертывания для отработки всех процессов, с последующим переходом к кластерному решению.
* Настройки производительности компонентов должны быть адаптированы под конкретное оборудование.