# Проектирование архитектуры ПО

# Задания на практическую работу:

## №1

Выполнить развертывание контейнеров или установить локально следующие службы, обеспечивающие различные виды представления хранимых данных:

1. Redis
2. MongoDB
3. Neo4j
4. ElasticSearch
5. PostgreSQL

Выполнить для каждой службы проверку доступности и настройку места хранения данных и информации о состоянии (лог-записи).

## №2

Для каждого из типов хранения, развернутых в рамках практической работы №1, выполнить следующие операции:

- Создать хранилище данных
- Добавить данные в хранилище
- Прочитать данные из хранилища по ключу
- Изменить и сохранить данные в хранилище
- Удалить записи в хранилище
- Удалить хранилище

В качестве основных данных, вносимых в рамках работы, предполагается использовать список студентов группы и изучаемых курсов в рамках семестра.

Для Redis – только список студентов с ключом в виде номера зачетной книжки

Для MongoDB – документ с данными и составом группы

Для Neo4J – Связи между группой-студентом-курсом, рассчитывая, на использование курсов по выбору.

Для ElasticSearch – данные с полнотекстовым описанием курса.

Для PosgreSQL – данные о посещении лекций студентами с партиционированием по неделе посещения. Данные о посещении при заполнении рекомендуется сформировать с помощью случайного выбора.

## №3

На основе информации о группах учащихся, курсах обучения, лекционной программе и составу лекционных курсов и практических занятий, а также структуре связей между курсами, специальностями, студентами кафедры и данными о посещении студентами занятий, сформировать структуру хранения и связей в реляционной базе данных. Разместить информацию по различным видам представления хранения данных (структуры ключ-значение, объекты документ-композиция, наборы типизированных связей, полнотекстовая информация с

метаданными, транзакционные данные с партицированием) для обеспечения оптимальной структуры для выборки информации в целях аналитических запросов. Обосновать свой выбор с точки зрения характеристик типов хранилищ.

## Задания на лабораторные работы:

**Во всех лабораторных работах предполагается для формирования аналитических запросов в качестве источников использовать структуру данных, сформированную в рамках практической работы №3.**

**Реализация предполагает следующую архитектуру, которую необходимо изобразить в нотации C4 и DFD (нотация Йордана-Кода): реализуемая система должна содержать API Gateway с набором API методов для вызова с использованием протокола HTTP(s) одной из лабораторных работ с указанными входными параметрами, а также реализовывать token (JWT) аутентификацию пользователя для доступа к вызываемым методам. Каждая из лабораторных работ выполняется в виде отдельного сервиса, упакованного в контейнер, и предоставляет API для запуска в виде HTTP(s) метода.**

**Выборка должна происходить из оптимального использования характеристик хранения. Способ извлечения данных для каждой работы необходимо обосновать. Результатом каждой лабораторной работы является отчет с описанием структуры хранения, задействованных в выборке хранилищ, обоснованием способа формирования выборки, а также отчет о тестировании выборки в виде последовательности исполнения с выводом результатов на каждом шаге.**

## Задание на лабораторную работу №1:

Выполнить запрос к структуре хранения информации о группах учащихся, курсах обучения, лекционной программе и составу лекционных курсов и практических занятий, а также структуре связей между курсами, специальностями, студентами кафедры и данными о посещении студентами занятий, для извлечения отчета о 10 студентах с минимальным процентом посещения лекция, содержащих заданный термин или фразу, за определенный период обучения. Состав полей должен включать Полную информацию о студенте, процент посещения, период отчета, термин в занятиях курса.

## Задание на лабораторную работу №2:

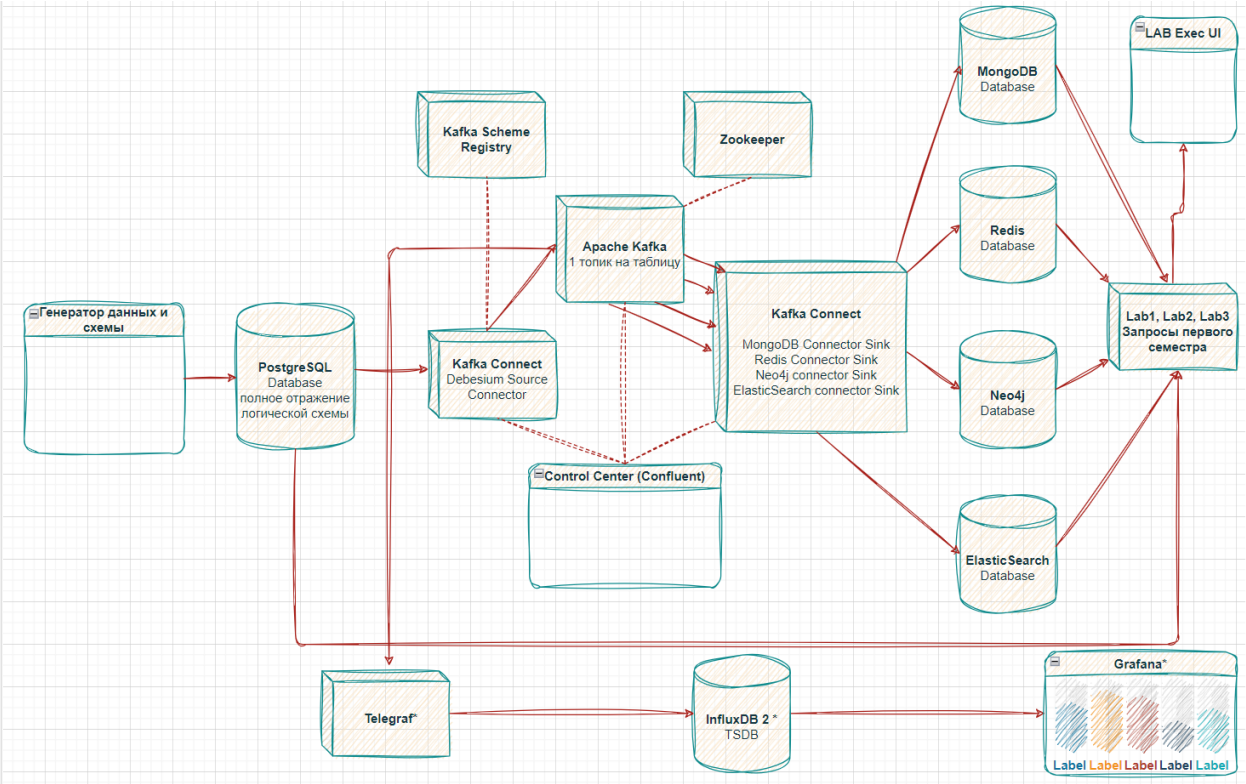
Выполнить запрос к структуре хранения информации о группах учащихся, курсах обучения, лекционной программе и составу лекционных курсов и практических занятий, а также структуре связей между курсами, специальностями, студентами кафедры и данными о посещении студентами занятий, для извлечения отчета о необходимом объеме аудитории для проведения занятий по курсу заданного семестра и года обучения с требованиями в описании к использованию технических средств. В качестве результата необходимо вывести полную информацию о курсе, лекции и количестве слушателей.

## Задание на лабораторную работу №3

Выполнить запрос к структуре хранения информации о группах учащихся, курсах обучения, лекционной программе и составу лекционных курсов и практических занятий, а также структуре связей между курсами, специальностями, студентами кафедры и данными о посещении студентами занятий, для извлечения отчета по заданной группе учащихся с указанием объема прослушанных часов лекций а также необходимого объема запланированных часов, в рамках всех курсов для каждого студента группы.  Предполагается, что одна лекция равна 2-м академическим часам. В отчет должны попасть только лекции, которые содержат тег специальной дисциплины кафедры.  В качестве результата необходимо вывести полную информацию о группе, студенте, курсе, количестве запланированных часов и посещенных часов занятий.

## Задание на лабораторную работу №4

Целевая архитектура потоков данных:




В рамках лабораторной работы необходимо выполнить развертывание СУБД PostgreSQL с включенным механизмом CDC (Change Data Capture).

1. Проверить наличие модуля wal2json в контейнере  СУБД или развернуть контейнер с включением модуля wal2json
2. Сгенерировать схему предыдущего семестра
3. Выполнить публикацию CREATE PUBLICATION pub FOR ALL TABLES;

4. Выполнить настройку конфигурации с postgresql.conf

В рамках лабораторной работы необходимо выполнить развертывание Apache Kafka и дополнительного компонента схемы CDC в виде контейнера Kafka Connect.

1. Провести развертывание APACHE KAFKA на основе примера: Kafka CP docker-compose
2. Выполнить очистку Docker Compose – должны остаться только следующие компоненты:
    - Broker
    - Zookeeper
    - schema-registry
    - kafka-connect
    - control-center – не обязательно, можно заменить на Kafka UI.
3. Добавить секцию для PosgreSQL
4. Выполнить настройку для установки Debezium Postgresql Connector (настройки подключения, Initial Snapshot)
5. Проверить поступление данных в Kafka с помощью Control Center

В рамках лабораторной работы необходимо выполнить конфигурирование Apache Kafka и дополнительного компонента схемы CDC в виде контейнера Kafka Connect, а также развёртывание контейнера ElasticSearch.

1. Добавить секцию в Docker Compose для ElasticSearch
2. Выполнить настройку для установки ElasticSearch Sink Connector
3. Провести настройку соединения и параметры доставки данных CDC из Kafka в ElasticSearch (все топики из PostgreSQL)
4. Проверить поступление данных в ElasticSearch для операций Create, Update, Delete

В рамках лабораторной работы необходимо выполнить конфигурирование Apache Kafka и дополнительного компонента схемы CDC в виде контейнера Kafka Connect, а также развёртывание контейнера Redis.

1. Добавить секцию в Docker Compose для Redis
2. Выполнить настройку для установки Redis sink Connector (использоватьhttps://github.com/lensesio/stream-reactor/tree/master/kafka-connect-redis )
3. Провести настройку соединения и параметры доставки данных CDC из Kafka в Redis (только топики таблиц, кешируемых в Redis)
4. Проверить поступление данных в Redis для операций Create, Update, Delete

## Задание на лабораторную работу №5

В рамках лабораторной работы необходимо выполнить конфигурирование Apache Kafka и дополнительного компонента схемы CDC в виде контейнера Kafka Connect, а также развёртывание контейнера Neo4J.

1. Добавить секцию в Docker Compose для Neo4j

2. Выполнить настройку для установки Neo4j sink Connector
3. Провести настройку соединения и параметры доставки данных CDC из Kafka в Neo4j (только топики таблиц, которые образуют граф связей по схеме данных)
4. Проверить поступление данных в Neo4j для операций Create, Update, Delete

В рамках лабораторной работы необходимо выполнить конфигурирование Apache Kafka и дополнительного компонента схемы CDC в виде контейнера Kafka Connect, а также развёртывание контейнера MongoDB.

1. Добавить секцию в Docker Compose для MongoDB
2. Выполнить настройку для установки MongoDB sink Connector
3. Провести настройку соединения и параметры доставки данных CDC из Kafka в MongoDB
   Использовать конфигурацию для обработчика CDC

   connector.class=com.mongodb.kafka.connect.sink.MongoSinkConnector

   connection.uri=<your connection uri>

   database=<your mongodb database>

   collection=<your mongodb collection>

   topics=<topic containing debezium cdc events>

   change.data.capture.handler=com.mongodb.kafka.connect.sink.cdc.debezium.rdbms.postgres.PostgresHandler
4. Проверить поступление данных в MongoDB для операций Create, Update, Delete
5. Адаптировать структуру объекта для автоматической обработки данных из нескольких таблиц в один иерархический объект в MongoDB: реализовать интерфейс CdcHandler

## Задание на лабораторную работу №6*

В рамках лабораторной работы необходимо выполнить конфигурирование Apache Kafka и дополнительного компонента схемы CDC в виде InfluxDB, средства визуализации Grafana и инструмента доставки данных Telegraf.

1. Добавить секцию в Docker Compose для InfluxDB весии 2
2. Выполнить настройку безопасности подключения к InfluxDB, проверить доступ через web interface
3. Добавить секцию в Docker Compose для Grafana, проверить доступ через web interface
4. Произвести подключение к InfluxDB со стороны Grafana – настроить DataSource для InfluxDB

В рамках лабораторной работы необходимо выполнить конфигурирование Apache Kafka и дополнительного компонента визуализации Grafana и инструмента доставки данных Telegraf.

1. Добавить секцию в Docker Compose для Telegraf
2. Настроить Telegraf для захвата данных из Kafka с использованием https://github.com/influxdata/telegraf/tree/master/plugins/inputs/kafka_consumer
3. Выполнить дополнительное конфигурирование Telegraf для всех топиков из PostgreSQL
4. Выполнить настройку панели мониторинга в Grafana для просмотра количества сообщений, которые были получены в каждый топик.