МИНИСТЕРСТВО ТРАНСПОРТА И КОММУНИКАЦИЙ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«Белорусский государственный университет транспорта»

Кафедра «Информационно-управляющие   
системы и технологии»

Отчет  
по лабораторным работам

по дисциплине «Средства и технологии анализа и разработки информационных систем»

Выполнил Проверил

студент группы ГИ-31 м.т.н., ст.преп.

Болдачева М.Д. Козлов В. Г.

Гомель, 2025

**Лабораторная работа №11  
Тема: РАЗРАБОТКА СИСТЕМЫ ОБРАБОТКИ СООБЩЕНИЙ С ИСПОЛЬЗОВАНИЕМ APACHE KAFKA.**

Цель: Разработка системы асинхронной обработки сообщений с использованием Apache Kafka, включая создание продюсеров, консьюмеров и настройку их взаимодействия.

**Задание**

1. Установка и настройка Kafka

* Запустите ZooKeeper и Kafka-брокер (локально или через Docker).
* Создайте топик user\_actions с 3 партициями.

2. Разработка продюсера

* Напишите приложение (Java/Python), отправляющее JSON-сообщения в топик. Пример сообщения: {"user\_id": 101, "action": "purchase", "timestamp": "2023-10-01T12:00:00"}
* Реализуйте ввод сообщений через консоль или чтение из файла.

3. Разработка консьюмера

Напишите консьюмер, обрабатывающий сообщения из топика:

* Фильтрация по типу действия (например, "purchase").
* Вывод статистики в консоль (количество сообщений, частые действия).

4. Обработка ошибок и масштабирование

* Настройте Dead Letter Topic (DLT) для некорректных сообщений.
* Запустите 2 консьюмера в одной группе. Продемонстрируйте распределение сообщений между ними.

5. Интеграция с внешней системой

* Сохраняйте данные в PostgreSQL/MySQL.

**Код KafkaProducer.java**

package org.example;

import org.apache.kafka.clients.producer.\*;

import org.apache.kafka.common.serialization.StringSerializer;

import org.apache.kafka.clients.producer.ProducerConfig;

import org.apache.kafka.clients.producer.ProducerRecord;

import java.util.Properties;

import java.util.Scanner;

public class KafkaProducer {

public static void main(String[] args) {

// Конфигурация продюсера

Properties props = new Properties();

props.put(ProducerConfig.BOOTSTRAP\_SERVERS\_CONFIG, "localhost:9092"); // Указываем адрес Kafka

props.put(ProducerConfig.KEY\_SERIALIZER\_CLASS\_CONFIG, StringSerializer.class.getName());

props.put(ProducerConfig.VALUE\_SERIALIZER\_CLASS\_CONFIG, StringSerializer.class.getName());

// Создание продюсера

org.apache.kafka.clients.producer.KafkaProducer<String, String> producer = new org.apache.kafka.clients.producer.KafkaProducer<>(props);

// Чтение данных из консоли

Scanner scanner = new Scanner(System.in);

System.out.println("Введите сообщение (JSON формат):");

while (true) {

String message = scanner.nextLine();

if ("exit".equalsIgnoreCase(message)) {

break;

}

// Если сообщение не является корректным JSON, отправляем его в DLT

if (!message.contains("\"user\_id\"") || !message.contains("\"action\"") || !message.contains("\"timestamp\"")) {

System.out.println("Некорректное сообщение, отправляем в DLT");

ProducerRecord<String, String> record = new ProducerRecord<>("user\_actions\_dlt", null, message);

producer.send(record);

} else {

// Создание сообщения и отправка его в Kafka

ProducerRecord<String, String> record = new ProducerRecord<>("user\_actions", null, message);

producer.send(record, new Callback() {

@Override

public void onCompletion(RecordMetadata metadata, Exception exception) {

if (exception != null) {

System.out.println("Ошибка при отправке сообщения: " + exception.getMessage());

} else {

System.out.println("Сообщение отправлено в топик " + metadata.topic() + " на партицию " + metadata.partition());

}

}

});

}

}

producer.close(); // Закрытие продюсера

}

}

**Код KafkaConsumer.java**

package org.example;

import com.fasterxml.jackson.databind.JsonNode;

import com.fasterxml.jackson.databind.ObjectMapper;

import org.apache.kafka.clients.consumer.\*;

import org.apache.kafka.common.serialization.StringDeserializer;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.SQLException;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import java.util.Properties;

public class KafkaConsumer {

private static final String DB\_URL = "jdbc:sqlite:user\_actions.db";

private static final ObjectMapper objectMapper = new ObjectMapper();

public static void main(String[] args) {

// Конфигурация консьюмера

Properties props = new Properties();

props.put("bootstrap.servers", "localhost:9092"); // Адрес Kafka

props.put("group.id", "user-action-group"); // Идентификатор группы

props.put("key.deserializer", StringDeserializer.class.getName());

props.put("value.deserializer", StringDeserializer.class.getName());

props.put("auto.offset.reset", "earliest"); // Чтение с самого начала

// Создание консьюмера

org.apache.kafka.clients.consumer.KafkaConsumer<String, String> consumer = new org.apache.kafka.clients.consumer.KafkaConsumer<>(props);

consumer.subscribe(List.of("user\_actions", "user\_actions\_dlt")); // Подписка на оба топика

Map<String, Integer> actionCounts = new HashMap<>();

int messageCount = 0;

int invalidMessageCount = 0; // Счетчик для неправильных сообщений

// Подключение к базе данных и подготовка SQL-запросов

try (Connection conn = DriverManager.getConnection(DB\_URL)) {

String insertValidSQL = "INSERT INTO user\_actions (user\_id, action, timestamp) VALUES (?, ?, ?)";

String insertInvalidSQL = "INSERT INTO invalid\_messages (message) VALUES (?)";

PreparedStatement insertValidStmt = conn.prepareStatement(insertValidSQL);

PreparedStatement insertInvalidStmt = conn.prepareStatement(insertInvalidSQL);

while (true) {

// Получение сообщений

ConsumerRecords<String, String> records = consumer.poll(1000); // Пауза 1 секунда

for (ConsumerRecord<String, String> record : records) {

String message = record.value();

messageCount++;

// Проверка, из какого топика пришло сообщение

if ("user\_actions\_dlt".equals(record.topic())) {

// Сообщение из топика Dead Letter (некорректное)

invalidMessageCount++;

try {

insertInvalidStmt.setString(1, message);

insertInvalidStmt.executeUpdate();

} catch (SQLException e) {

System.err.println("Ошибка при сохранении в invalid\_messages: " + e.getMessage());

}

continue; // Пропускаем дальнейшую обработку для сообщений из DLT

}

// Обработка валидного сообщения

try {

JsonNode jsonNode = objectMapper.readTree(message);

String action = jsonNode.get("action").asText();

if ("purchase".equals(action)) {

// Увеличиваем счетчик для действия "purchase"

actionCounts.put("purchase", actionCounts.getOrDefault("purchase", 0) + 1);

// Сохранение в базу

int userId = jsonNode.get("user\_id").asInt();

String timestamp = jsonNode.get("timestamp").asText();

insertValidStmt.setInt(1, userId);

insertValidStmt.setString(2, action);

insertValidStmt.setString(3, timestamp);

insertValidStmt.executeUpdate();

}

} catch (Exception e) {

System.err.println("Ошибка при обработке сообщения: " + e.getMessage());

invalidMessageCount++;

try {

insertInvalidStmt.setString(1, message);

insertInvalidStmt.executeUpdate();

} catch (SQLException ex) {

System.err.println("Ошибка при сохранении в invalid\_messages: " + ex.getMessage());

}

}

}

// Вывод статистики

System.out.println("Обработано сообщений: " + messageCount);

System.out.println("Статистика по неправильным сообщениям: " + invalidMessageCount);

System.out.println("Статистика по действиям: " + actionCounts);

// Пауза 5 секунд

try {

Thread.sleep(5000); // Пауза в 5 секунд

} catch (InterruptedException e) {

e.printStackTrace();

}

}

} catch (SQLException e) {

System.err.println("Ошибка подключения к базе данных: " + e.getMessage());

} finally {

consumer.close();

}

}

}

**Результаты**

bin\windows\zookeeper-server-start.bat config\zookeeper.properties

bin\windows\kafka-server-start.bat config\server.properties

.







