UNIVERSIDADE DE COIMBRA FACULDADE DE CIÊNCIAS E TECNOLOGIA *Departamento de Engenharia Informática*	**Project #3** **Integração de Sistemas/** **Enterprise Application Integration** **2021/22 – 1st Semester** MEI, MES **Deadline: 2022-12-09**

Nota: A fraude denota uma grave falta de ética e constitui um comportamento não admissível num estudante do ensino superior e futuro profissional. Qualquer tentativa de fraude pode levar à reprovação na disciplina tanto do facilitador como do prevaricador.

MUITO IMPORTANTE: o código entregue pelos alunos vai ser submetido a um sistema de deteção de fraudes.

VERY IMPORTANT: the code delivered by students will be submitted to a fraud detection system.

Message Oriented Middleware (MOM) and Kafka Streams

Objectives

- Learn how to create simple asynchronous and message-oriented applications.
- Learn to use Kafka Streams.

Resources

Apache Kafka Introduction: https://kafka.apache.org

Kafka Streams: https://kafka.apache.org/documentation/streams/

Apache Kafka Tutorial:
https://www.tutorialspoint.com/apache_kafka/apache_kafka_simple_producer_example.htm

Look for the Kafka and Kafka Streams materials available on UC Student.

Make sure you understand the following concepts:

- Producer
- Consumer
- Topic

- Partition and partition offset
- Broker
- Zookeper

Kafka Training (doesn’t count for evaluation)

1. You should start by installing Kafka (or find out where it is installed if you are using a container).

2. You may look at the reading material suggested before and follow that material to have a basic example running with a producer and consumer.

3. What happens to the messages that arrive at the topic before the subscriber makes the subscription?

4. How do you configure different partitions inside a topic?

5. What is the point of Consumer Groups? How do they work?

6. Now, read the Kafka Streams tutorial and run the example that counts the words that go through a stream.

7. Refer to the following blog message for this and the following exercises:
   https://eai-course.blogspot.com/2018/11/playing-with-kafka-streams.html

   Use Kafka Streams to count the events that happened for a given key. Display the result as ‘k->v’ in the final stream.

8. How do you change the type of data of the keys and values that you send?

9. How do you switch the key and value of a stream?

10. Now sum all the values that showed up for each key.

11. Now, control the time considered in the stream to be 1 minute.

12. How could you send complex data instead of primitive types?

13. Imagine that you have two streams, e.g., debit and credit. How do you create a third stream with the current balance?

14. Use the Kafka Connect application to periodically fetch data from a database table (source). You can find help for this operation in the following blog message:
    https://eai-course.blogspot.com/2019/11/how-to-configure-kafka-connectors.html

15. Now do the inverse operation: send from a topic to a database table (sink).

Project Description (for evaluation)

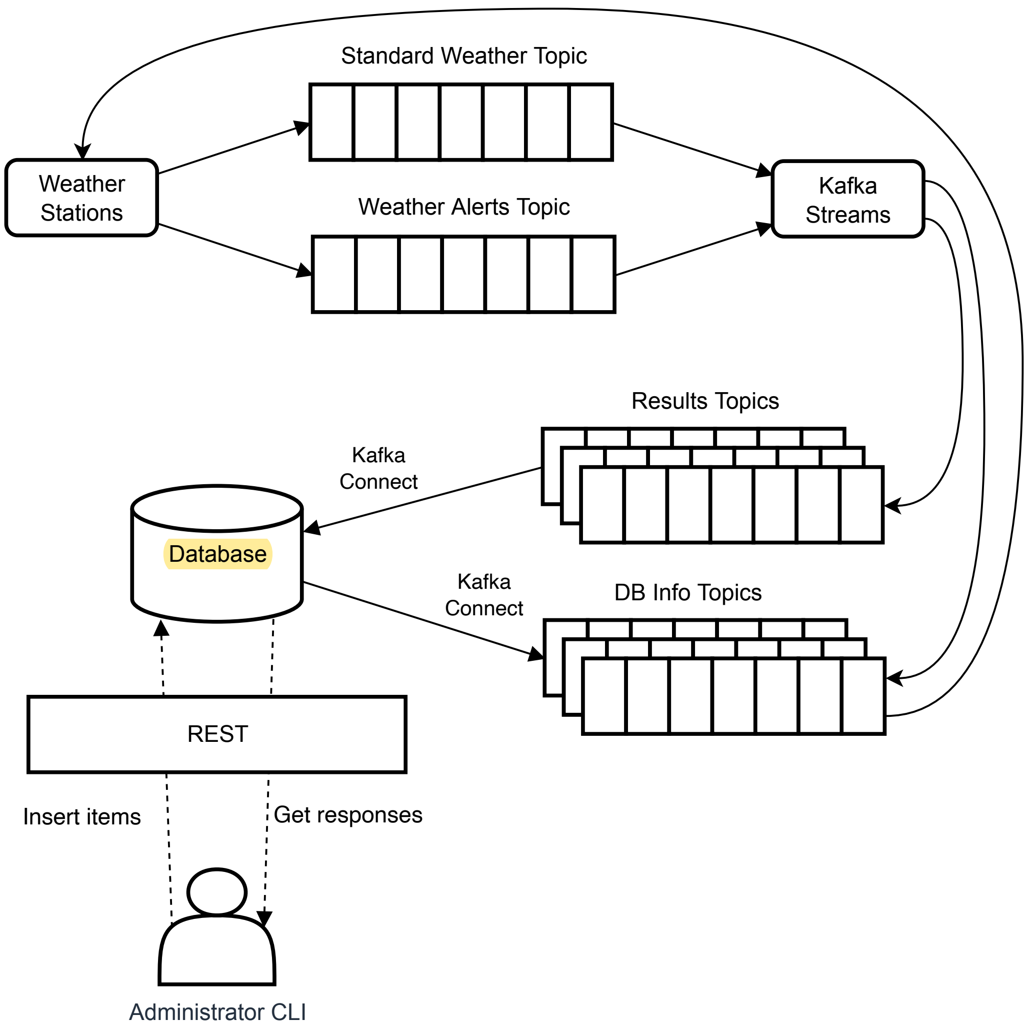

Figure 1 - Overview of the Weather Company

Refer to Figure 1, which summarizes the operation of a Weather Forecasting Company heavily relying on Kafka. The Weather Company has the following Kafka topics:

- **DBInfo** topics, where one or more source connectors write a list of weather stations and locations present in the database.
- **Standard Weather** topic, where a process simulating weather station actions with measurements of atmospheric conditions keeps writing new standard weather events. Each standard weather event includes a location and a temperature. Students

should randomly select the values. Each weather station may produce weather events from different locations, typically in the same city.
- **Weather Alerts** topic, where a process simulating weather station actions (possibly the same) keeps writing weather alert events, e.g., thunderstorms, hurricanes, blizzards, and droughts. Each extreme weather event includes a location and type of alert ("red" for severe alerts and "green" to signal that the weather returned to normal).
- One or more applications using Kafka streams will listen to these topics and compute a few metrics, as enumerated in the requirements, including global average temperatures, average temperature by location, number of extreme weather events by location, etc. This or these applications will then write their computations to another topic, the **Results** topics, from where a Kafka connector will write the results back to the database.

A server-side application would read data from the database and expose the data via REST. This could be accessed, e.g., by a command line application to let the administrator write and read data through the REST services.

Components to Develop

Students need to develop the Weather Stations (publishers) and the Kafka Streams applications form Figure 1. These are stand-alone applications. Students must also configure Kafka connectors to automatically extract and write data from/to the database. They may interact directly with the database, i.e., they do not need to implement the client and the REST server.

The precise definition of the communication format is left for the students to determine. Clean solutions that serialize complex data structures as JSON strings are encouraged.

To simplify the configuration work, the professors will provide a folder with YAML and Docker files with the configuration of multiple services. Students must later configure the `lib` directory mentioned in the YAML file and optionally the `config` directory as well.

Requirements

Students should use Kafka Streams to implement the following requirements and write the results (in Celsius) to the database:

1. Count temperature readings of standard weather events per weather station.
2. Count temperature readings of standard weather events per location.
3. Get minimum and maximum temperature per weather station.
4. Get minimum and maximum temperature per location (Students should compute these values in Fahrenheit).
5. Count the total number of alerts per weather station.
6. Count the total alerts per type.
7. Get minimum temperature of weather stations with red alert events.
8. Get maximum temperature of each location of alert events for the last hour (students are allowed to define a different value for the time window).
9. Get minimum temperature per weather station in red alert zones.

10. Get the average temperature per weather station.
11. Get the average temperature of weather stations with red alert events for the last hour (students are allowed to define a different value for the time window).

Solutions should maximize the computations that students do with Kafka streams and should not depend on complex database queries to extract results (for example, students should use Kafka Stream to compute temperature averages, instead of computing them on the database).

Students should include means in their application to enable a fast verification of the results they are storing.

Students should also leverage Apache Kafka fault-tolerance mechanisms, i.e., configure a multi-broker setup, to prevent data losses due to a potential broker crash. Therefore, students should be able to stop one of the brokers without losing information or halting the entire system.

Final Delivery

- The project should be made in groups of 2 students. I do expect all the members of the group to be fully aware of all the parts of the code that is submitted. Work together!
- To automate the build of the project, students should use Maven.
- Students should submit the project part that is for evaluation in a zip file with the **source** of a complete Maven project, using Inforestudante. Do not forget to associate your work colleague during the submission process.
- Grading is performed according to a grid that will be published later and according to individual student defenses.
- No report is necessary.

Good Work!