	3º Projeto / Project #3
UNIVERSIDADE DE COIMBRA FACULDADE DE CIÊNCIAS E TECNOLOGIA *Departamento de Engenharia Informática*	**Integração de Sistemas/ Enterprise Application Integration** **2023/24 – 1ˢᵗ Semester** MEI, MES, MEIG **Deadline: 2023-12-07**

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/

Complete Kafka Producer with Java:
https://www.conduktor.io/kafka/complete-kafka-producer-with-java/

Complete Kafka Consumer with Java:
https://www.conduktor.io/kafka/complete-kafka-consumer-with-java/

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

Make sure you understand the following concepts:
- Producer
- Consumer (and Consumer Groups)

- Topic
- Partition and partition offset
- Broker
- Zookeper (KRaft now)

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? How do you configure the subscriber to read historical data, i.e., all messages from the beginning?

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).
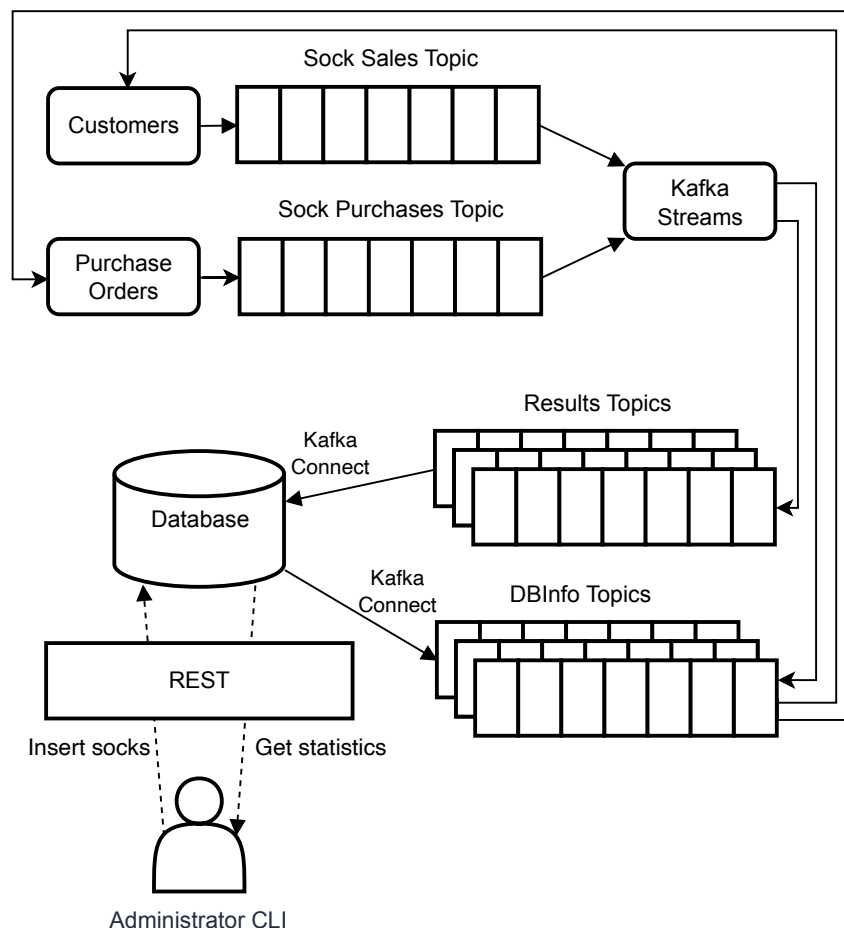
Project Description (for evaluation)

Figure 1 - Overview of the Sock Shop company

Refer to Figure 1 summarizing an E-Commerce Sock Shop company heavily relying on Kafka. The company is built around the following Kafka topics:
- **DBInfo** topics, where one or more source connectors write a list of sock suppliers and a list of socks for sale from the database.
- **Purchases** topic, where a process simulates purchasing socks from suppliers. This process continuously generates orders for socks listed in the DBInfo topic. Each sock includes its price, the number of pairs, type, e.g., invisible, low cut, or over

the calf, and the sock supplier identifier from where the sock pair was purchased. Students are expected to randomly select the socks they want to purchase, along with its price, type and quantity.

- **Sales** topic, where a process simulates customers and continually writes sales data. Each sale has a sock reference, price per sock pair, number of pairs the sock supplier identifier from which the sock pair was purchased, and the buyer identifier. Students need not to be concerned about whether the stock for each sock is positive or if the sock shop company is profitable.

- One or more applications using Kafka streams will listen to these topics and compute a few metrics, as enumerated in the requirements, including revenues, expenses, profit, 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 will 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. No authentication is necessary.

Components to Develop

Students need to develop the following applications of Figure 1: Customers, Purchase Orders, Kafka Streams. 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 Administrator CLI 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 to the database:

1. Add sock suppliers to the database. To simplify sock suppliers cannot be deleted and optionally not changed.
2. List sock suppliers from the database.
3. Add socks to be purchased to the database. Again, these cannot be deleted, but may be changed if students wish.
4. List socks from the database.
5. Get the revenue per sock pair sale.
6. Get the expenses per sock pair sale.
7. Get the profit per sock pair sale.
8. Get the total revenues.
9. Get the total expenses.

10. Get the total profit.
11. Get the average amount spent in each purchase (separated by sock type).
12. Get the average amount spent in each purchase (aggregated for all socks).
13. Get the sock type with the highest profit of all (only one if there is a tie).
14. Get the total revenue in the last hour[1] (use a tumbling time window).
15. Get the total expenses in the last hour (use a tumbling time window).
16. Get the total profits in the last hour (use a tumbling time window).
17. Get the name of the sock supplier generating the highest profit sales. Include the value of such sales.

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 the average amount spent, 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. We 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 based on individual student defenses.
- No report is necessary.

Good Work!

---

[1] Students may change this interval freely for testing and demonstration purposes.