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: 2021-12-10

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 Sutdent.

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 change the type of data of the keys and values that you send?

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

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

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

8. 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.

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

10. Use a materialized view to do range queries on the sums on a separate service.

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. 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

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

Project Description (for evaluation)

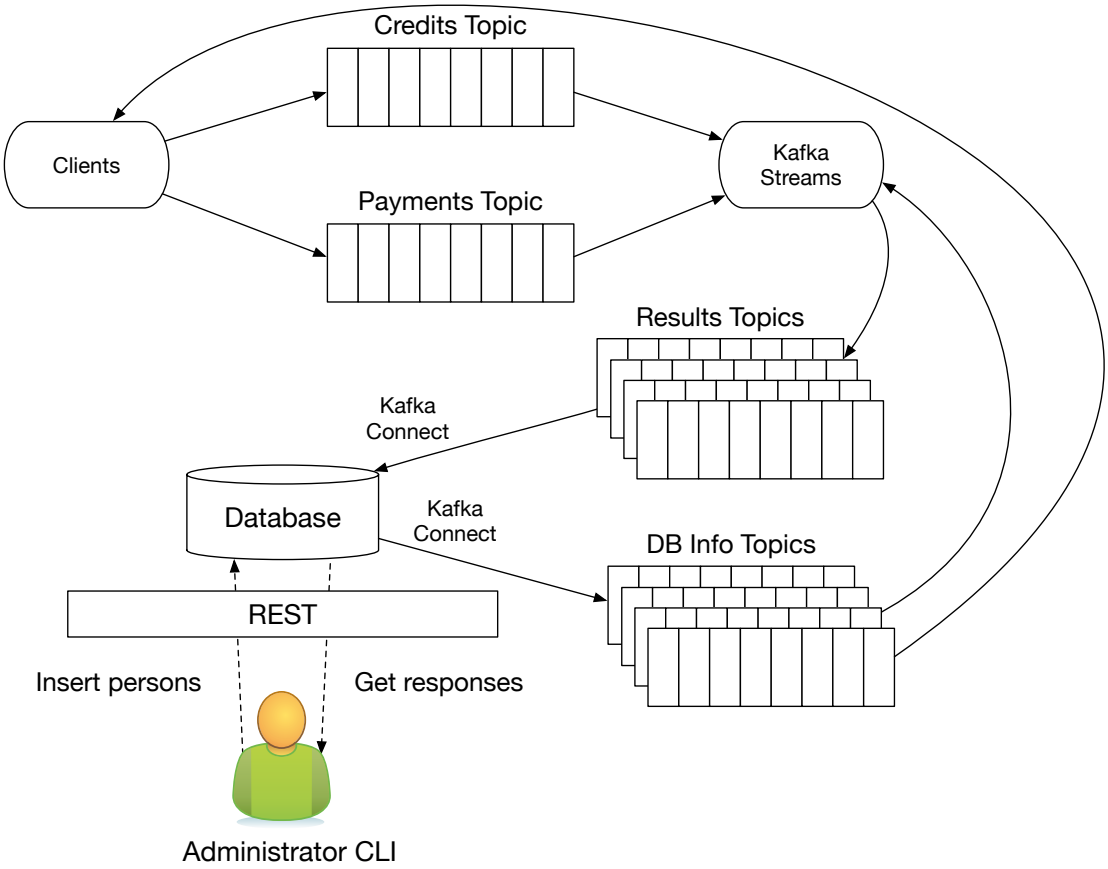

Figure 1 - Overview of the Credit Card Company

Refer to Figure 1 summarizing a credit card company in Kafka. The credit card company has the following Kafka topics:

- **DBInfo**, where one or more source connectors write a list of clients and managers present in the database.
- **Credits** topic, where a process simulating client actions with the credit card keeps writing new outstanding debts from a client. Each purchase includes a price and a currency. Students should randomly select the values and the currency.
- **Payments** topic, where a process simulating client actions (possibly the same) keeps writing payments, just as in the previous case. Students should need not concern if payments exceed the credits.
- One or more applications using Kafka streams will listen to these topics and compute a few metrics, as enumerated in the requirements, including credits,

payments, balances, 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. The final piece of the application is a command line application that allows the administrator to enter items, countries and read statistics from the REST services. No authentication is necessary.

Components to Develop

Students need to develop the following applications of Figure 1: Clients, Kafka Streams, the administrator CLI and the REST implementation. From this list, the former are stand-alone applications, while the latter is a server-side application to deploy on a server. They must also configure Kafka connectors to automatically extract and write data from/to the database.

violates not-null constraint
Detail: Failing row contains (2, 15928798.04099999, null, null).

Except for the data that was mentioned before, 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.

Requirements

As an administrator I can perform the following actions (results should be compute in euros):

1. Add managers to the database. To simplify managers cannot be deleted and optionally not changed. Done
2. Add clients to the database. Again, these cannot be deleted and optionally not changed. Each client has a manager. Done
3. Add a currency and respective exchange rate for the euro to the database. Done
4. List managers from the database. Done
5. List clients from the database. Done
6. List currencies. Done
7. Get the credit per client (students should compute this and the following values in euros). Done
8. Get the payments (i.e., credit reimbursements) per client. Done
9. Get the current balance of a client. Done
10. Get the total (i.e., sum of all persons) credits. Done
11. Get the total payments. Done
12. Get the total balance. Done
13. Compute the bill for each client for the last month[1] (use a tumbling time window). Done
14. Get the list of clients without payments for the last two months.
15. Get the data of the person with the highest outstanding debt (i.e., the most negative current balance). Done
16. Get the data of the manager who has made the highest revenue in payments from his or her clients. Done

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

Students should include means in their application to enable a fast verification of the results they are displaying. Solutions should maximize the computations they do with Kafka streams and reduce the database queries to their simplest possible form (for example, students should use Kafka Stream to compute profits, instead of computing them on the database).

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!