CSIS505 Java
Weston Greene

Project: Nubium

A business forms from meeting a perceived need. The parties involved when meeting that need are the supplier–the business–and the receiver–the customer. Having many customers, businesses represent those customers in multiple applications: a marketing platform to advertise the solution to the need, a customer manager to facilitate communication with the customer, a license management software to record the transactions. Each of these applications have an understanding of the customer, but that understanding is narrow. Each application may track that customer with varying identifiers. Nonetheless, regardless of the different understandings, in reality there is always still a single human/buying-group behind those understandings.

Synchronizing these disparate understandings of the single customer would lead to a streamlined customer experience and swifter supply of the customer's needs. Nubium supplies that synchronization. Nubium tracks a central person record of each customer, providing transformation from and to the disparate understandings of that customer. The fields received from one application are mapped to the central person fields, excluding fields that are unique to that application. Once mapped, the values then are applied to the central person record. With the central person updated, the new person definition is then pushed to all other applications, reversing the mapping to the fields of that application that correspond to the central person's fields.

For instance, customer First Last, with email of firstlast@email.com, responds to a marketing campaign by filling out a web form. The submission of that webform is captured in a Marketing Automation Platform (MAP), like Eloqua. The information from that submission is then carried to a Customer Relationship Manager (CRM), like Salesforce, where a company representative is then equipped to contact First Last. First Last purchases from the company, triggering a record in a License Management Software, capturing additional details about their person, including their street address. Those details are then produced to the other applications in this customer's journey, leaving the License Software, the MAP, and the CRM all synchronized.

All the systems being synchronized allows for precision: marketers will now be able to send campaigns to this customer based on geography. It allows for personalization: the sales representatives using the Customer Relationship Manager will be able to see which products the customer has already purchased and suggest supporting products.

Requirements:

- Read from an external source (a .csv file for this example since the custom retrieval is out of scope for this project) and produce to retrieval-kafka-topic
- Consume from retrieval-kafka-topic, transform to person-record, produce to person-canon-topic
- Consume from person-canon-topic, transform back to external source, produce to external source (the aforementioned .csv)

Classes

- Kafka: Communicates with kafka topics
- Transformer extends Kafka: the transformer app that converts either to the canon or to the external source
- Merger extends Kafka: finds previous records of that person and merges the new information with the existing
- Retriever extends Kafka: the custom logic to retrieve the external source events
- Updater extends Kafka: the custom logic to update the external source after having been merged with the person-canon
- Csv Abstract: commonalities among apps that touch CSVs
- Message: the container of the event that occurred in the external source or is being sent to the external source

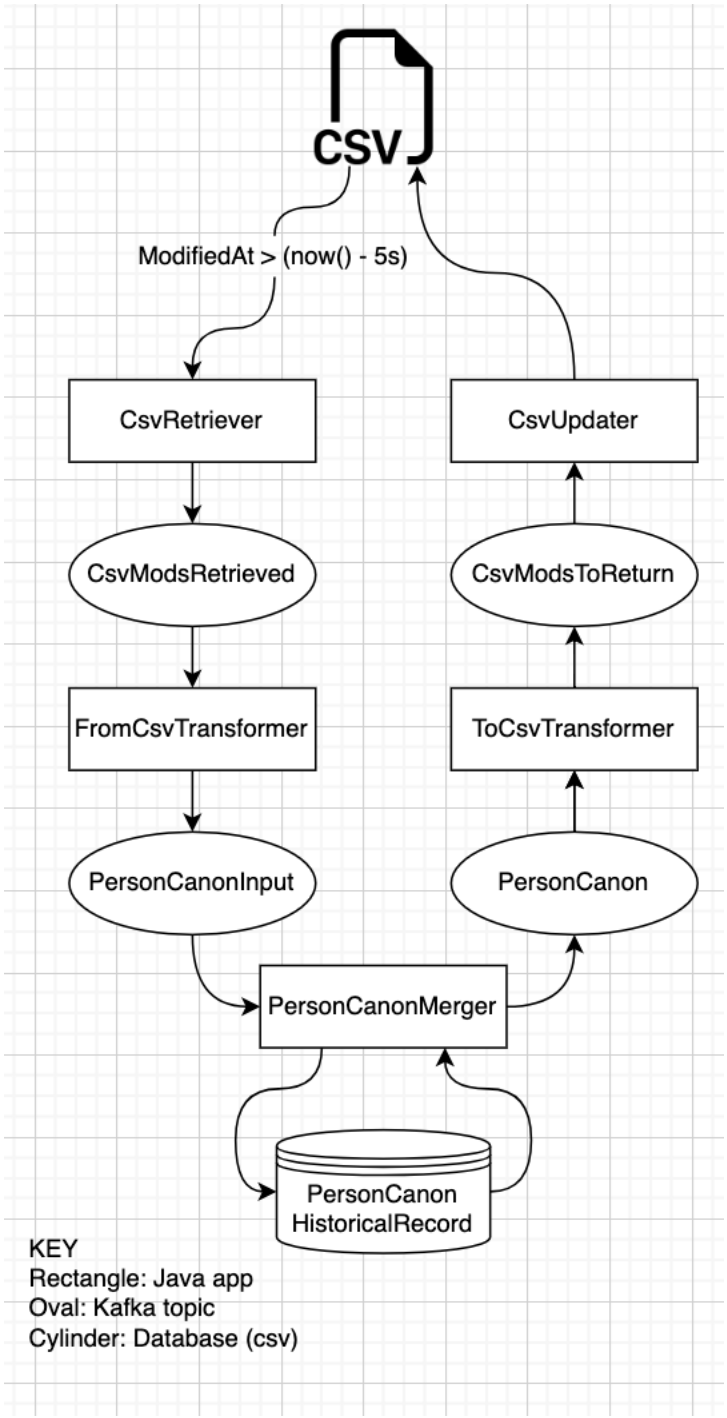
Review of Project

Instead of a Graphical User Interface (GUI), this project will be verifiable via terminal prints and modifications to a CSV spreadsheet.

Steps to verify (will require an internet connection as the applications will communicate with a free Kafka cluster via Confluent):

1. Run applications (may consist of multiple terminals executing multiple Java Applications)
2. Modify and save spreadsheet intregation_a.csv
3. Watch terminal prints
4. Open spreadsheet intregation_b.csv and observe the propagated information
5. Repeat steps 1 through 4 but in the opposite direction: modify intregation_b.csv and observe intregation_a.csv

Data Flow


CSV
ModifiedAt > (now() - 5s)
CsvRetriever
CsvUpdater
CsvModsRetrieved
CsvModsToReturn
FromCsvTransformer
ToCsvTransformer
PersonCanonInput
PersonCanon
PersonCanonMerger
PersonCanon
HistoricalRecord
KEY
Rectangle: Java app
Oval: Kafka topic
Cylinder: Database (csv)

Class Diagram