CROSSJOIN

Dev Backend Challenge

Are you up to the challenge?

Perform to perfection

Objective:

The challenge is to develop a parallelized account processor.

This processor reads CSV files from a directory, processes and aggregates the information from those files, and returns a single CSV file with the result.

Each line of the input files contains an ID, followed by a comma, followed by a mathematical expression.

The goal of the processor is to produce a result file with one line per unique ID found, containing the result of evaluating all the expressions associated with that ID.

Example of input:

Input file - contas.csv:

- AAA,10+12-8
- CCC,(1*12+3)/3
- BBB,90+7
- AAA,10/2

Input file - contas2.csv:

- CCC,2
- DDD,2-1
- AAA,3
- BBB,7-8

Expected result file - resultado.csv:

- AAA,22
- BBB,96
- CCC,7
- DDD,1

Solution Requirements:

You can choose the language you are most comfortable with and that is suitable for this challenge

The application must read each file in parallel

The application must also process each line of each file in parallel, producing in the end the output file with the calculations completed and sorted alphabetically by identifier

It is assumed that the expressions only contain the symbols +, -, *, /, (and)

External libraries may be used

Nice-to-have requirement:

While reading each file in parallel, the application should send each line's content to an Apache Pulsar message topic

Suggestion: Launch a Docker container with Apache Pulsar in standalone mode (no security mechanism is required – authentication, https, etc.)

The application should then read the messages from that topic and process them in parallel, producing in the end the output file under the same conditions as in the previous requirements

Valued aspects

The application should then read the messages from that topic and process them in parallel, producing in the end the output file under the same conditions as in the previous requirements

- Correct and coherent parallel processing of the information
- Code modularity
- Ability to receive relevant input variables that modify the application's behavior (for example: path to the directory containing the input files)