NOMBRE
Sergio Alonso Benito

EMAIL: ser_al_97@outlook.com
SENT: 14.12.2021
DEADLINE: 21.12.2021

TEST FOR PYTHON BACKEND DEVELOPER

We are pleased you want to do our test. Thank you so much for your interest and time. So let's get started then!

The problem

A financial services company needs to design and implement a system that manages all the processes it needs to conduct its business.

The different components of the system fall into three main categories:

- **Data gathering applications**: these modules collect information from various sources (data providers, third-party websites, files sent by clients) and store it on a database.
- **Data processing applications**: these calculate several financial indicators, based on the prices of financial securities stored on the database. Some of these calculations are straightforward and are calculated "on the fly", when a user requests them; others are more complex and need to be calculated in advance and stored on the database.
- **Data visualization tools**: these allow end users to view processes and analyze financial information, to help them advise clients and produce reports on the evolution of the company's products.

Design of the architecture

The company requires a system that stores and processes the information that is being constantly received.

It is necessary to take into account that all the information that is handled in the system might be of interest to several processes simultaneously.

In this part, we ask you to design the architecture of the system as a whole, explaining how the different types of processes would interact with each other.

We only need you to sketch a general scheme of the system, indicating what technologies you would use in each part. A brief description of the proposed solution will do, it isn't necessary to go into too much detail in this part.

Implementation

To actually see how the system would work in practice, we ask you to implement a small subset of the functionality of the system, coding a module that calculates the price series of a **synthetic index**. The following considerations must be taken into account to develop this part:

- An external data provider sends information on the prices of financial securities, and these are stored on a database by an automatic process. Occasionally, the prices received correspond to a date that is already on the database; in these cases, it is assumed that the latest price is always the correct one. For the purposes of this exercise, you are free to simulate the reception of the data in any way you see fit.

- A synthetic index is an aggregate of some underlying securities; each security is assigned a weight in the index to get the price of the synthetic index for each date, which is calculated according to the formula:

  $P_t = P_{t-1} \times (1 + R_t)$, where

  P_t: price of the index on date t

  P_{t-1}: price of the index on date t-1, with $P_0 = 100$

  R_t: return of the index on date t, calculated as follows:

$R_t = \sum_{i=1}^{n} w_i \times r_t^i$, where:

w_i: weight of each underlying security

r_t^i: return of each underlying security on date t, calculated as follows:

$r_{i_t} = \frac{P_t^i}{P_{t-1}^i} - 1$, where:

P_t^i: price of the underlying security i on date t

P_{t-1}^i: price of the underlying security i on date t-1

(It should be noted that the return series of the underlying securities will have one element less than the original price series)

- The performance of all synthetic indices should be always updated with the latest price; that means that the process should run as soon as a new price is received and perform the calculation.
- The price series of the securities involved typically span several decades.
- The calculation must be implemented in a way that allows both internal processes and third-party applications to have access to the price series.

Requirements and constraints

- All the components of the system must be scalable and capable of dealing with hundreds of concurrent requests.
- Using Python to do the implementation part is mandatory. You are free to choose whatever technology of the Python ecosystem you think is appropriate. (Pandas, Numpy, Django, Flask, SqlAlchemy, Channel, etc.)

- You can rely on any framework, library, third-party component and platform that will meet the requirements of the system.
- The system is going to be extended and improved permanently; it is therefore particularly important that the code is as clear and structured as possible, so that programmers that didn’t develop a particular code can fix bugs or extend the functionality.