TECHNICAL CHALLENGE FOR INTEGRATIONS DEVELOPER IN PYTHON

To complete this challenge, you are allowed to use any Python libraries you deem necessary, useful, or required. Please organize your code such that it's readable and easily executed, this entails creating your own functions and/or classes that will be called from within the script.

Please take into consideration that this challenge will assess the functionality of your script, the optimization of imports and resources, the clarity and efficiency of the code, as well as your creative approach to solving the problem in the most straightforward manner.

The challenge at hand requires the creation of two APIs: one for data querying and another for data loading. It is strongly advised that you thoroughly review the entire challenge before commencing the coding process.

Challenge Prep: Download Data

Download the data set that you will use for this challenge from the Kaggle platform. You might need to create an account (free) if you don't already have one.

The dataset is called "pump_sensor_data," and it contains information captured from 52 pressure sensors. Each row of the original dataset contains a measurement date (timestamp), the measurement value for each of the 52 sensors at the indicated moment (sensor_xx), and the state of the machine at that moment (machine_status).

Step 1: Data Request Service

Develop an API capable of sending data from the downloaded dataset through a GET method.

1.1 Read the file and filter data

© Fracttal Tech S.L - Todos los Derechos Reservados.

Construct the necessary code to read the file and filter the data according to the following criteria:

- Include only measurements from April 2018.
- Consider only sensors 07 and 47.
- Include values greater than 20 and less than 30.

1.2 Response format when calling the API

Craft a straightforward API capable of responding with the filtered data through a GET request. No authentication or parameters are required. The response should be a JSON file containing the filtered data as defined in the previous step.

Organize the information in a manner that facilitates its transmission. For this challenge, there is no need to address information security or authentication.

Step 2: Data Reception Service

Develop a second API equipped with a POST method that can accept data in the same format as provided by the previous API.

2.1 Data Reception

Create a secondary API to handle a POST method. The data within the request body should adhere to the same format delivered by the API in stage 1. Like before, you may overlook any concerns regarding information security and authentication.

2.2 Data Organization

Upon receiving the data, arrange it into a Pandas DataFrame with the following columns:

- Date: Contains the date when the measurement was taken (timestamp) in the format YYYY-MM-DD.
- Time: Contains the time of the measurement (timestamp) in the format HH:MM:SS.

© Fracttal Tech S.L - Todos los Derechos Reservados.

- Sensor: Name of the column from which the data was extracted (sensor_xx).
- Measurement: Value from the column from which the data was extracted.
- Status: Value from the machine_status column in the selected row.

NOTE: If you encounter two pieces of data in the same original dataset row that fulfill the conditions (i.e., both sensor 7 and sensor 47 record values between 20 and 30 at the same date and time), ensure they are saved as two distinct rows within the DataFrame.

Once the DataFrame has been constructed, display it to the console.

Deliverables:

- Python Script (.py). If necessary, you can send more than one file. Please include any comments within the script that help us understand how at works and how to execute it.
- Additionally, a .txt file with a list of all libraries required to run the script and need to be installed using PIP.

© Fracttal Tech S.L - Todos los Derechos Reservados.