Egen Coding Challenge – Level 2

The goal of this exercise is to build a system that works like an IoT platform – in this case, a personal weight tracker. This system is responsible for,

- Consuming data sent from different sensors (emulators)
- Storing the received data in MongoDB
- Running the data through different rules to make basic predictions

What you are given?

- sensor-emulator - Utilize this java program that acts like a sensor (imagine it is fitted in to your personal weight scale) and sends the recorded weight to your API every 5 seconds. The sensor also takes in another argument that tells the base weight of the person that is using the system. Please follow the README on this repo to start the program.

What you will be building?

- You will be building two modules – the API and the Rules. Create a new repo egen-be-challenge on your github.com account and commit all your code to it. Make sure all IDE and OS related files are in .gitignore. Once completed, send the link of the repo to the recruiter.

API

We would like you to build a Spring Boot microservice that,

- consumes data from the emulator via HTTP API and stores it in a MongoDB collection using Morphia API (please do not use Spring Data)
- uses MongoDB as your datastore with two collections
    - **metrics** – stores the data that comes from sensor
    - **alerts** – stores the alerts that were created by the rules
- exposes the below Metric APIs using Spring MVC,
    - **create** – this is the API that will consume data from the sensor emulator
    - **read** – reads all the metrics stored in your database
    - **readByTimeRange** – reads all the metrics that were created between the given two timestamps
- exposes the below Alert APIs using Spring MVC
    - **read** – reads all alerts that are stored in the database
    - **readByTimeRange** – reads all alerts that are created between the given two timestamps

Rules

You will also create rules, that will be triggered as and when the 'create API' receives new metrics from the sensor. Create these 2 rules using EasyRules,

- Detects under weight – if the weight of the person drops below 10% of his base weight
    - Create a new alert and save it in MongoDB
- Detects over weight – if the weight of the person shoots 10% over his base weight
    - Create a new alert and save it in MongoDB

Write unit tests wherever needed.