MBA em Desenvolvimento de Aplicações Java – SOA e Internet das Coisas
Turma 31SCJ - Desenvolvimento em Arquitetura de Microsserviços

Trabalho para final da disciplina.

Nós gostaríamos de ter uma API restful de estatística. O principal caso de uso para nossa api é calcular uma estatística realtime dos últimos 60 segundos dentro de um timezone. Essa API deverá possuir dois endpoints, one deles é chamado toda vez que uma transação é feita. A outra retorna a estatística baseada nas transações dos últimos 60 segundos.

INSTRUÇÕES:

1. Complete o desafio utilizando java em 7 dias;
2. O código não precisa ser “deployável”, mas precisa estar rodando, ou seja o comando mvn clean install deve completar com sucesso;
3. Disponibilize seu código no Github ou Bitbucket para que possamos revisá-lo;
4. Garanta que não usará nenhum banco de dados, utilize uma solução em memória;
5. Testes Unitários e Testes Integrados;
6. Dockerize seu microsserviço e coloque no README como subir duas instâncias de seu microsserviço;
7. A API tem que ser threadsafe para requisições concorrentes;
8. O Endpoint tem que executar em tempo constant e memória (O(1));
9. Utilize Java para completar o teste;
10. Gere uma documentação da API utilizando Swagger;

Avaliaremos a qualidade e capricho da codificação, e o uso adequado das tecnologias escolhidas para a implementação. Não esqueça de considerar a concorrência, no caso de termos duas requisições no mesmo momento para inserção de uma conta contábil ou um registro.

Especificação da API

A API deve suportar os endpoints e métodos listados mais abaixo. Observe bem os tipos de dados e os formatos utilizados, e use return codes adequados para cada requisição. Não esqueça também de seguir as melhores práticas de mercado no tratamento de erros, como buscar um registro inexistente ou fazer uma inserção de registro sem preencher todos os campos obrigatórios (entre outros…).

1) Toda vez que uma nova transação acontecer, este endpoint será chamado.

```
POST /transactions/

{
  "timestamp": 1478221904000,
  "amount": 25000.15
}

--

201 Created
```

Onde:

- amount - valor da transação
- timestamp - tempo da transação em millisegundos no timezone UTC .

Retorno: Body vazio com 201 ou 204.

- 201 - no caso de sucesso
- 204 - se a transação ocorreu há mais de 60 segundos

Onde:

- amount - é um valor double
- timestamp é um long com unix format in millisegundos.

2) Este é o principal endpoint, este endpoint tem que executar em memória e tempo constante (O(1)). Ele retorna a estatística baseada nas transações no qual aconteceu nos últimos 60 segundos;

```
GET /statistics

--

200 OK

{
  "sum": 25170.91,
  "min": 20.00,
  "max": 25000.17,
  "avg": 8390.30,
  "count": 3
}
```

Onde:

- sum é um double do total da soma das transações dos últimos 60 segundos;
- avg é um double com a média dos valores das transações dos últimos 60 segundos;
- max é um double com o valor máximo da transações dos últimos 60 segundos;
- min é um double com o valor mínimo das transações dos últimos 60 segundos;

- count é um long com o número total das transações que aconteceram nos últimos 60 segundos;

Conforme observações em sala de aula segue informações complementares.

No exercício 1, existem dois argumentos, o primeiro argumento um **timestamp** e o segundo um argumento com valor **amount**.
É necessário realizar uma validação, considerando se o valor enviado no **timestamp** é **maior** que **60** segundos (em relação ao horário da máquina). Se for maior do que 60 segundos, então um ExceptionHandler com o código HTTP Status **204** tem que ser gerado. Se o **timestamp** for **menor ou igual** do que **60** segundos, grave a informação em memória e retorne o HTTP Status **201**.