Exercício 2 - Parte 1

- Que parte das informações poderíamos ter pre-cacheada, sem ter perdido a precisão no cálculo?

Poderíamos pré-cachear o câmbio no KVS, de forma que checaríamos primeiramente se temos o câmbio salvo em cache no KVS, se não faríamos a requisição para api de de conversão, caso contrário, se tivéssemos o câmbio em cache não precisaríamos mais chamar outra aplicação para realizar a conversão.

- Qual seria a janela de exposição para se ter um erro de cálculo no preço em USD? E como podemos ajustá-la?

Se não definirmos um TTL para o cache salvo no KVS, ficaríamos com o câmbio desatualizado indefinidamente. Com um TTL, teríamos um delay da taxa de câmbio equivalente ao TTL definido, onde somente após o cache expirar que faríamos uma nova requisição para api de conversão para obter a nova taxa de câmbio.

Podemos ajustá-la utilizando algum service assíncrono de mensageria, como por exemplo o Big Queue.

- Como se implementaria esse cache?

Poderíamos implementar o sistema de mensageria utilizando o service de Big Queue, onde seríamos o consumer do topic de taxa câmbio, e toda vez que o câmbio mudasse, e, recebêssemos a mensagem de que houve uma alteração no câmbio, invalidaríamos o nosso cache no KVS para obter a taxa mais recente.

- Que serviços se utilizaria para implementá-lo?

Para realizar o cache do câmbio utilizaríamos o KVS do tipo cache, e para a mensageria do tipo publish/subscribe utilizaríamos o Big Queue.

- Indiquem quais classes se modificaria para realizar estas mudanças.

Adicionaríamos mais uma classe controller para o serviço do Big Queue, e para adicionar o serviço de KVS, seria no service da api de conversão de moedas. Além de também adicionar as classes services necessárias no package de api.

Exercício 2 - Parte 2

- Como vocês fariam para evitar ir "ao vivo" para a API de items?

Poderíamos utilizar um KVS do tipo cache, que armazenaria as informações de cada item na primeira vez que fosse necessário obter as suas informações, desta forma a partir das requisições seguintes não precisaríamos mais fazer a chamada para a api de items, podendo obter as informações diretamente do KVS.

- O que acontece se o preço de um item mudar?

Se não definirmos um TTL para o cache salvo no KVS, ficaríamos com o preço de um item desatualizado indefinidamente. Com um TTL, teríamos um delay do preço de um item seria equivalente ao TTL definido, onde somente após o cache expirar que faríamos uma nova requisição para api de items para obter o seu novo preço.

- Como podemos descobrir imediatamente quando o preço de um item muda? O que vocês fariam a respeito?

Para descobrir imediatamente quando o preço de um item muda seria necessário utilizarmos algum service de mensageria assíncrona como o Big Queue, desta forma, nossa api seria um consumer que ao receber uma mensagem de o preço de um determinado item mudou, nós invalidaríamos o cache, caso exista, no KVS e o atualizaríamos com o preço mais atual.

- Como executamos o código que temos em fury?

Com o docker ativo, podemos ir no terminal no diretório onde a nossa aplicação e rodar o seguinte comando “fury run”. Ou se estivermos utilizando alguma IDE, podemos rodar diretamente por lá.

De modo geral, também devemos configurar o escopo de onde estamos rodando a aplicação, configurando nas variáveis de ambiente em qual ambiente estamos utilizando.