Atividades

Assuntos: Estilos Arquiteturais, Comunicação, Concorrência

Pontuação da Atividade Individual: 25 pontos

10/02/2017

Controle de Versão

Versão	Data	Descrição	Responsável
1.0	01/02/2017	Criação do documento (adicionado as questões de 1 a 2)	Ari
2.0	28/02/2017	Adicionado as questões de 3 a 10	Ari

Informações para resolução das questões:

- Todos os códigos devem ser enviados para um repositório no GitHub e dentro dele deverá haver todas as pastas nomeadas com no formato "questao_xx", onde x é o número da questão em dois dígitos, exemplo: *questao_09*;
- Cada pasta deve conter todos os projetos para a solução da questão;
- Deve-se utilizar apenas codificações em Java;
- Todos as classes e métodos devem ser comentados de forma que fique explicado quais os papéis e responsabilidades dos participantes (classes), os estados ou as configurações (atributos) e os comportamentos (métodos);
- Todos os projetos devem ser do tipo Maven e estar preparado para executar em Docker (veja https://ag-ifpb.github.io/blog/post/run-simple-javaapp-in-docker/)

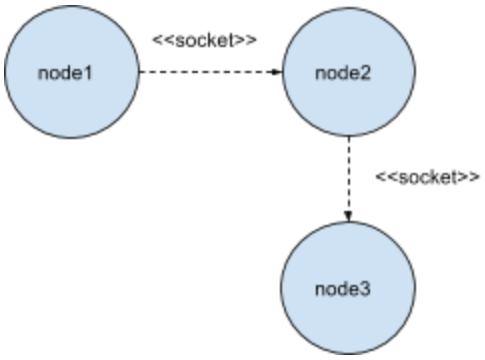
1 - Questão **(estilo arquitetural baseado em camadas)**

Observando a topologia apresentada na figura abaixo, construa as aplicações que possam resolver os seguintes requisitos:

- A aplicação do "node1" deve gerar dois números inteiros entre 0 e 100 e enviar para a aplicação do "node3" através do "node2";
- O "node2" deve resolver receber a mensagem enviada pelo "node1" e enviar para o "node3" somente quando os dois números forem diferentes;
- No caso dos dois números enviados pelo "node1" serem iguais, o "node2" deve responder ao "node1" com o valor "0";

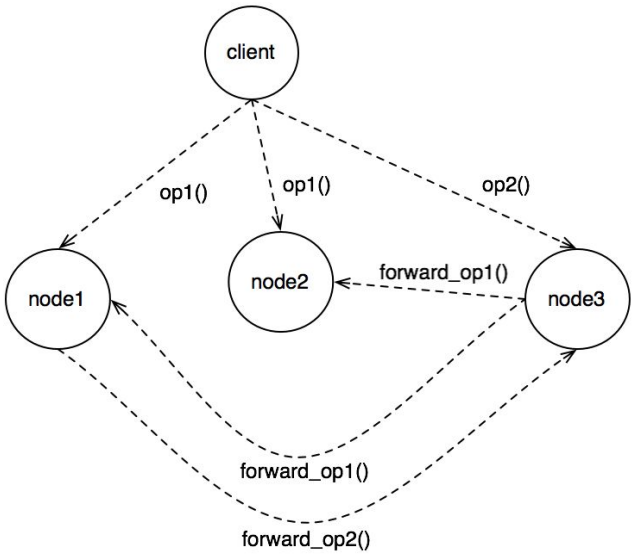
- O "node3" deve usar a equação abaixo para responder as requisições que lhe são feitas:

$$f(x,y) = x^y + y^x$$



2 - Questão **(estilo arquitetural baseado em objetos)**

Considere o cenário onde uma aplicação cliente possui conhecimento de onde encontram-se outros três nós, sendo que dois deles são iguais (réplicas) e que ao necessitar realizar uma consulta para qualquer um dos nós, caso não consiga, tentará em um outro **diferente**. Implemente este cenário o esquema ao lado.

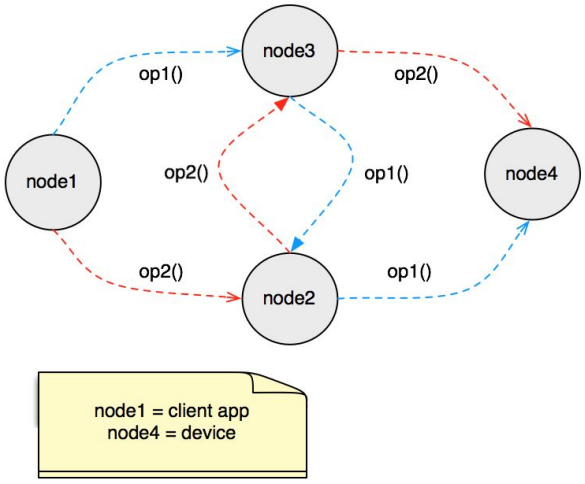


Notas:

- siga a topologia ao lado;
- adote socket
- considere op1 = sum(x,y)
- considere op2 = diff(x,y)

3 - Questão **(estilo arquitetural em camadas)**

Desenvolva uma solução baseada em Socket para um cliente que deseja comunicar-se com um dispositivo móvel, mas sem o conhecimento de onde este dispositivo encontra-se na rede. O fluxo de comunicação deve seguir o esquema da topologia ao lado.

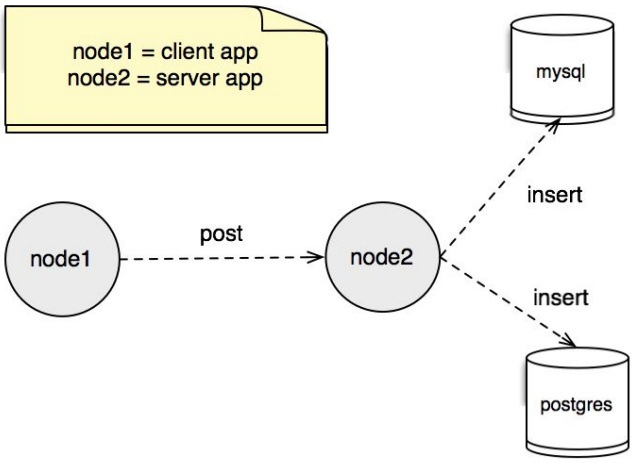


Notas:

- considere op1 = sum(x,y)
- considere op2 = diff(x,y)

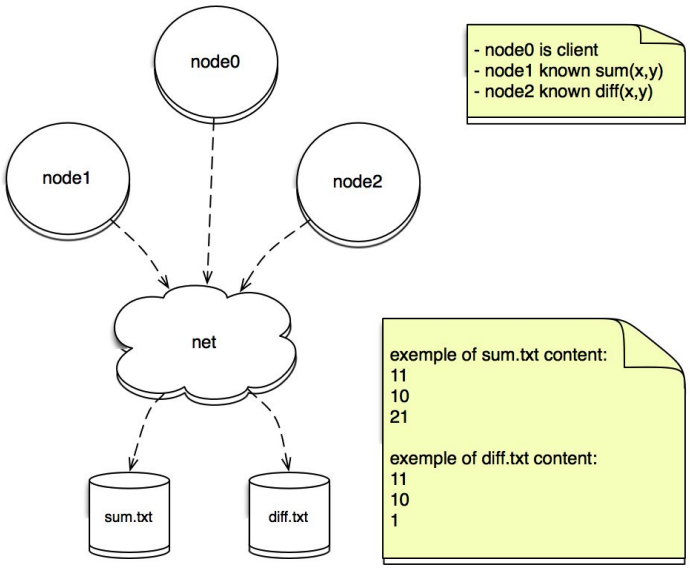
4 - Questão **(estilo arquitetural baseado em camadas)**

Implemente um replicador de dados simples que garanta a consistência dos dados. Adote a topologia do sistema distribuído ao lado para isto.



5 - Questão **(estilo arquitetural baseado em dados compartilhados)**

Adotando uma pasta como repositório de dados e arquivos nomeados de ***sum.txt*** e ***diff.txt*** como mensagens enviadas entre componentes (nós), desenvolva o sistema distribuído que siga a topologia ilustrada ao lado.



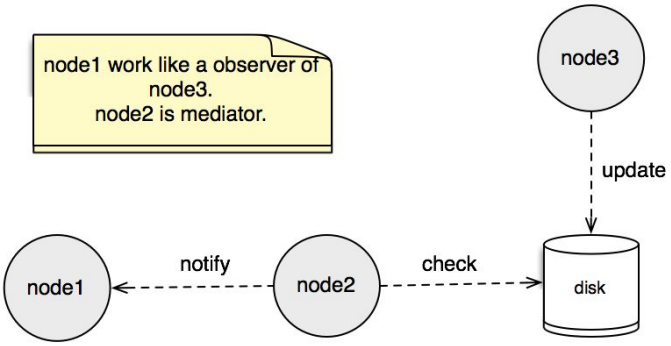
Notas:

- considere op1 = sum(x,y)
- considere op2 = diff(x,y)
- os arquivos *sum.txt* e *diff.txt* não podem ser criados ou destruídos programaticamente
- use o docker para compartilhar arquivos conforme instruído e demonstrado nos links abaixo:

Link para o artigo: https://goo.gl/BO6nlj
Link para o código de exemplo: https://goo.gl/MUOncs

6 - Questão **(estilo arquitetural baseado em dados compartilhados)**

A figura abaixo demonstra a topologia de um sistema distribuído que assemelha-se ao padrão observer. Implemente-o e descreva a diferença entre a sua implementação e o padrão observer descrito em GoF.



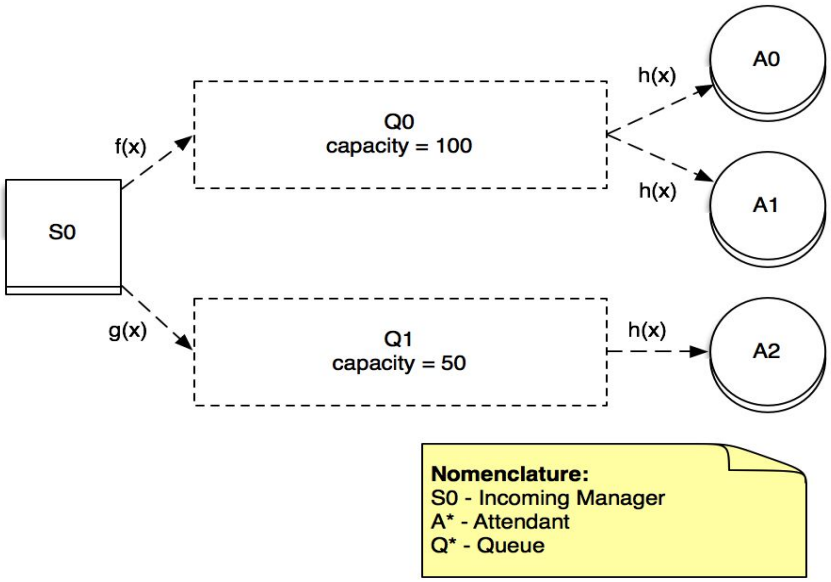
7 - Questão (concorrência)

Considerando o cenário descrito na ***questão 1*** anterior implemente uma solução que permita o "node1" enviar 20 requisições em 1s e cujas respostas também sejam atendidas em 1s.

8 - Questão (concorrência)

Conforme ilustrado abaixo, o sistema de atendimento ocorre em duas filas distintas. A primeira fila (Q1) possui capacidade para 100 pessoas e a segunda (Q2) possui capacidade para 50 pessoas. A primeira fila é atendida por dois postos de trabalho, enquanto que a segunda apenas por um. A taxa de chegada para a primeira fila é de f(x) pessoas por segundo e a taxa de chegada para a segunda fila é de g(x) pessoas por segundo. A taxa de atendimento é de h(x) pessoas por segundo em qualquer um dos postos de trabalho. Considere $x \in \Re$ obtido aleatoriamente a cada 1 segundo, sendo que $0 \leq x < 1.$ Considere ainda que: $f(x) = 0.833e^{-x}$, $g(x) = 3x^2 + 5$ e $h(x) = 15x^x$. Com tais informações construa uma aplicação que simule este sistema de atendimento e responda as seguintes perguntas no fim de 600 segundos:

a) Quantas pessoas entraram na fila?
b) Quantas pessoas foram embora antes de entrar na fila (ocorre quando a fila está cheia)?
c) Quantas pessoas foram atendidas?
d) Quantas pessoas ficaram na fila?



Informações adicionais:

- *deve-se adotar threads e os mecanismos de bloqueio e sincronização para desenvolver o sistema de simulação e*
- *use impressões no console para explicar a ocorrência dos eventos.*

Bom trabalho para todos!

Email para envio do link do github: aristofânio@hotmail.com