Atividades

Assuntos: Estilos Arquiteturais, Comunicação, Concorrência

Pontuação da Atividade Individual: 20 pontos

09/11/2018

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
3.0	17/03/2017	Adicionado observação na questão 02	Ari, Laerton
4.0	18/03/2017	Adicionado observações nas questões 02 a 06	Ari, Wenstay
5.0	10/06/2017	Atualização para 2017.1	Ari
6.0	09/11/2018	Atualização para 2018.2	Ari

Informações para resolução das questões:

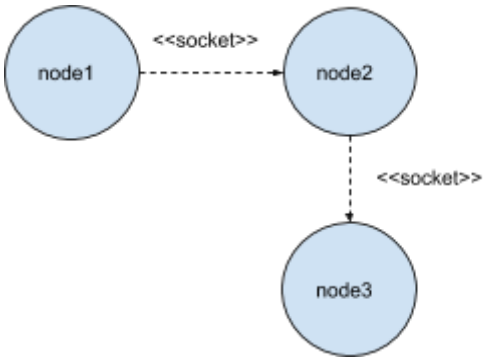
- Todos os códigos devem ser enviados para um repositório no GitHub e dentro dele deverá haver todas as pastas nomeadas com no formato "questao_xx", onde x é o número da questão em dois dígitos, exemplo: *questao_09*;
- Cada pasta deve conter todos os projetos para a resolução da questão correspondente;
- Deve-se utilizar apenas codificações em Java, exceto onde for expressamente solicitado outra linguagem;
- Todos as classes e métodos devem ser comentados de forma que fiquem explicados quais os papéis e responsabilidades dos participantes (classes), os estados ou as configurações (atributos) e os comportamentos (métodos);
- Adote Maven como arquetipo como forma

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

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

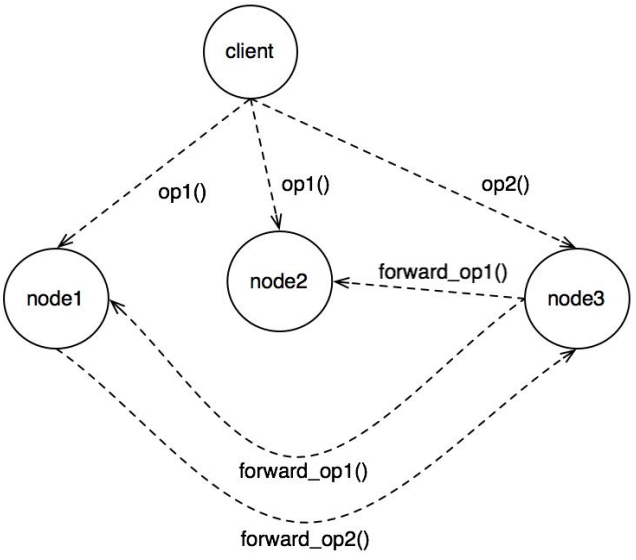
- A aplicação do "node1" deve gerar dois números inteiros entre 0 e 100 e enviar para a aplicação do "node3" através do "node2";
- O "node2" deve resolver receber a mensagem enviada pelo "node1" e enviar para o "node3" somente quando os dois números forem diferentes;
- No caso dos dois números enviados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

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



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 requisição para qualquer um dos nós, caso não consiga, tentará em um outro **diferente (não réplica)**. Implemente este cenário o esquema ao lado.



Notas:

- siga a topologia ao lado;
- adote socket;

Observação

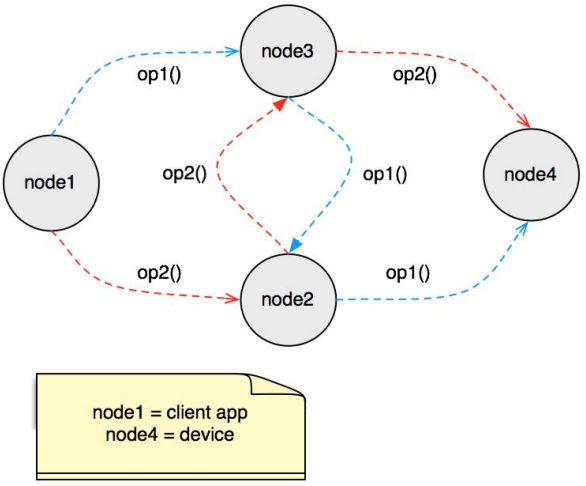
- Node2 é réplica de Node1;
- Node1 e Node2 sabe resolver apenas op1;
- Node3 sabe resolver apenas op2;
- Node1 deve delegar para Node3 a operação de op2;
- Node3 deve delegar para Node1 a operação de op1 e
- Considere a formulação para

$$op1(x,y) = 2yx$$

$$op2(x,y) = 2x/y$$

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. (Utilize socket para comunicação entre os nós e adote a sua segunda linguagem).

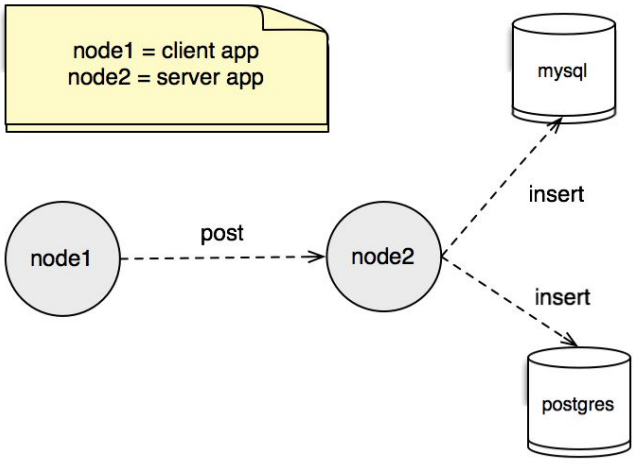


Notas:

- 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 e teste conforme instruções abaixo. (Utilize socket para comunicação entre os nós).



Informações complementares:

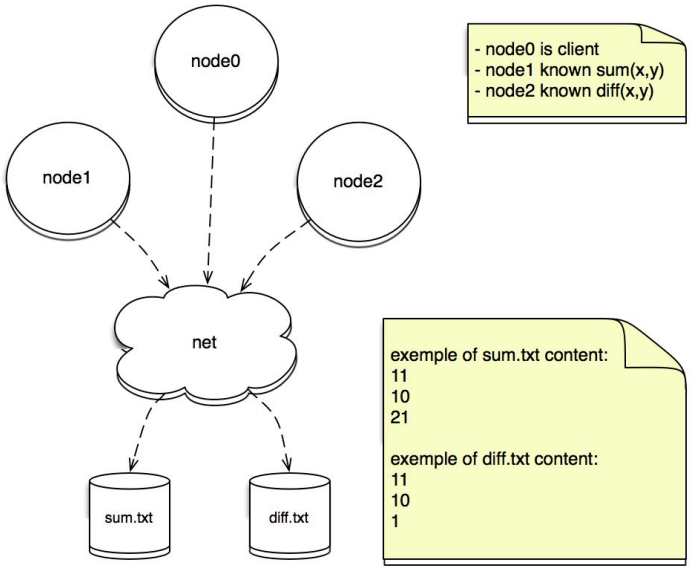
Tabela	Campos	Banco
tb_user	code int name varchar(50)	Mysql / Postgres

Testes:

Carga	Procedimento	Resultado
100 usuários	Envie 100 requisições de inserção de dados de usuários e informe o tempo total (em segundos)	
1000 usuários	Envie 1000 requisições de inserção de dados de usuários e informe o tempo total (em segundos)	
1000 usuários	Adotando threads, implemente uma solução que reduza o tempo de inserção a partir de 1000 requisições e informe o tempo total (em segundos)	

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. (Utilize socket para comunicação entre os nós).



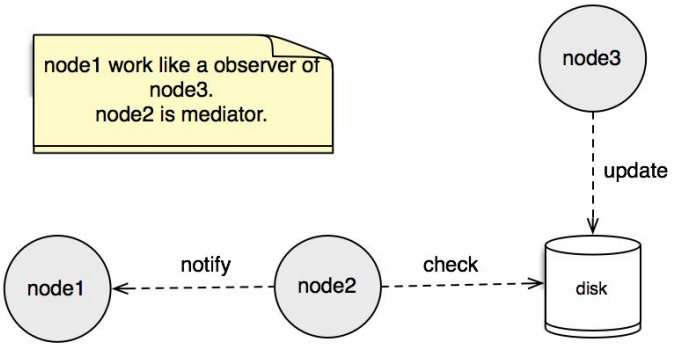
Notas:

- 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. (Utilize socket para comunicação entre os nós e a sua segunda linguagem).



Bom trabalho para todos!

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