Sistemas Distribuidos

TÉCNICO LISBOA

Relatório da 2ª Entrega sobre Tolerância a Faltas

Grupo T06

Francisco Aguiar
84718

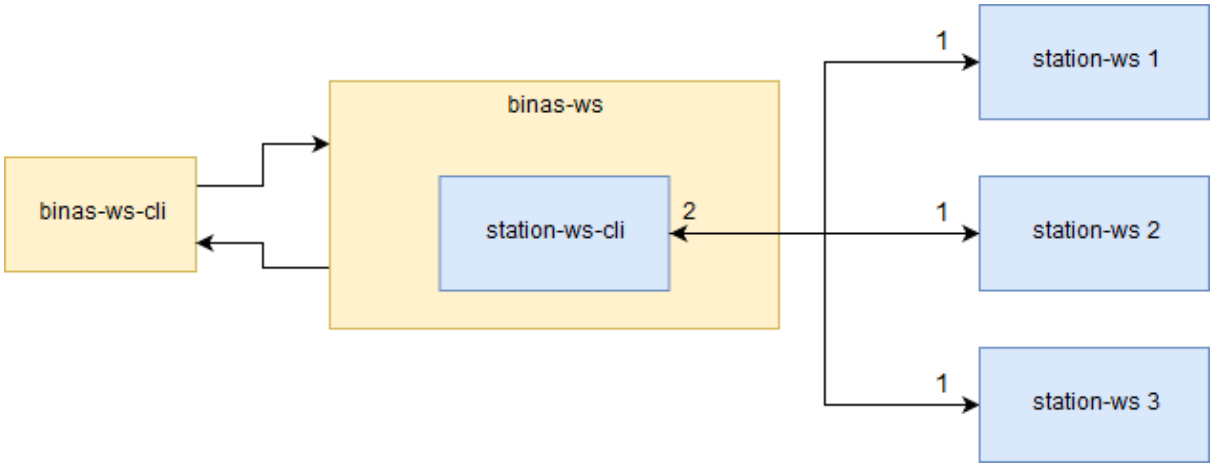
Gonçalo Marques
84719

Manuel Sousa
84740

https://github.com/tecnico-distsys/T06-SD18Proj

Descrição da solução

Pedido de lerSaldo()



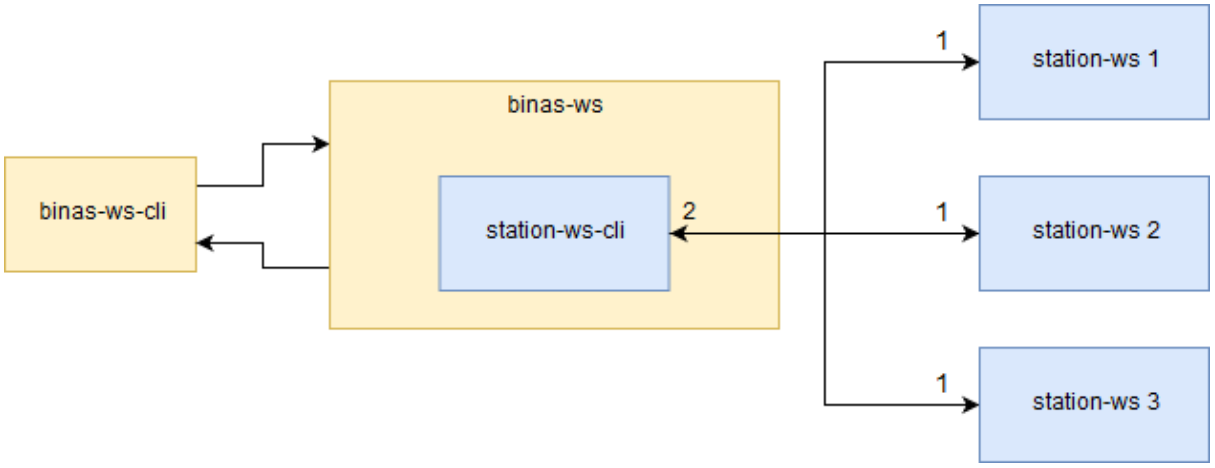
1 - O pedido para ler saldo é feito a todas as estações de modo assíncrono pelo binas-ws.
2 - O binas-ws só devolve uma resposta até ter recebido respostas suficientes das estações para fazer o Quorum Consensus.

2.1 - Se o binas-ws receber 2 ou mais respostas com informação, seleciona aquela que tem o timestamp mais recente, e guarda em Cache o par: (saldo + timestamp), sendo esse par o valor de retorno da função **lerSaldo()** sobre a forma de uma estrutura chamada de BalanceView.

2.2 - Se o binas-ws receber 2 ou mais respostas por parte das estações que digam que o utilizador não existe, retorna uma mensagem de erro ao binas-ws-cli.

2.3 - Se houver o caso extremo em que 2 ou mais estações não retornam nada ou existe um problema de comunicação com as mesmas, o binas-ws também comunica esse erro ao utilizador de modo a que este possa tentar a operação mais tarde.

Pedido de escreverSaldo()



1 - O binas-ws, não tendo nenhum registo na cache sobre o saldo mais recente do cliente e sobre a sua timestamp correspondente, faz o pedido **lerSaldo()** da mesma maneira que foi

referido no ponto anterior e guarda-o na cache, caso contrário, acede à informação disponível em cache.

1.1 - Efectua operações sobre o saldo obtido no último ponto.

1.2 - Envia o novo saldo e a timestamp com a hora local para todas a estações

2 - O binas-ws só espera por respostas até ter informação suficiente para fazer o Quorum Consensus.

2.1 - Se o binas-ws receber 2 ou mais respostas com confirmação da transação, conclui a operação.

2.2 - Se o binas-ws receber 2 ou mais respostas por parte das estações que digam que houve um erro com os argumentos, retorna uma mensagem de erro ao binas-ws-cli.

2.3 - Se houver o caso extremo em que 2 ou mais estações não retornam nada ou existe um problema de comunicação com as mesmas, o binas-ws também comunica esse erro ao utilizador de modo a que este possa tentar a operação mais tarde.

Optimizações:

Decidimos implementar duas caches no binas-ws, uma para os Timestamps e outra para os saldos dos utilizadores, cada uma mapeada através do email do utilizador correspondente. No final da operação **escreverSaldo()**, o saldo final do utilizador correspondente é introduzido nas caches, mantendo-as assim atualizadas com o valor mais recente.
Na operação **lerSaldo()**, a cache dos saldos vai ser pesquisada com a chave do utilizador correspondente. Se aquele utilizador tiver o seu saldo guardado na cache, assume-se que é o mais atualizado, e é esse valor que vai ser utilizado para as operações consequentes em vez de ser feito um pedido a cada uma das estações.

Observações:

Escolha de funções assíncronas:

Escolhemos usar funções com **callback**, em vez de funções com polling, devido à sua facilidade de lidar com várias respostas seguidas, visto que é mais fácil incrementar uma variável na função de callback dos pedidos, do que guardar todas as respostas aos pedidos e percorrê-las todas para verificar se já se atingiu o Quorum Consensus.

Uso de timestamps:

Em alternativa à solução apresentada pelo corpo docente relativo ao Quorum Consensus, decidimos abdicar do uso de tags por incremento, e recorrer ao uso de TimeStamps. Desta forma interagimos com algo mais *human-readable* que nos permite manter registos reais das acções realizadas entre o binas-ws e as station-ws's e podem também ser usadas como *log* em caso de problemas no sistema. O uso de TimeStamps também não é influenciada pelos diferentes clocks ondes os módulos possam vir a correr, visto que apenas um módulo é responsável pela geração destas marcas (binas-ws).