# Descrição da solução

A abordagem usada tem base no Quorum Consensus, com algumas alterações com o objetivo de melhorar a eficiência do projeto. Nesse sentido, é assumido que existem três réplicas Station, e que não pode haver mais que uma falha em simultâneo. As chamadas remotas esperam apenas por duas respostas. Além disso, o servidor Binas pode falhar ocasionalmente, o que pode causar a perda ou corrupção na informação dos utilizadores.

Uma das alterações realizadas foi na estrutura da tag. Foi decidido que a tag seria um número sequencial, que iria ser incrementado apenas quando é realizado a operação *setBalance,* ou seja, quando é realizada a operação de escrita. Isso é possível porque existe apenas um cliente, pelo que não é necessário guardar o clientId.

Além disso, a tag mais recente é também guardada pelo utilizador, do lado do Binas, tal como o saldo correspondente à escrita com essa tag. Assim, a leitura realizada antes de se fazer uma escrita, com o objetivo de procurar qual a tag mais recente, passa a ser desnecessária, já que pode ser obtida do lado do Binas. Este procedimento funciona apenas quando o servidor Binas não falha, ou seja, quando não são perdidas as informações dos utilizadores. Se assim for, há a garantia que o saldo guardado pelo utilizador é o mais atual.

Numa situação de falha do servidor Binas, em que os registos de utilizadores são completamente perdidos, a associação de email e saldo de cada utilizador persiste nas réplicas station. Assim sendo, na realização de operações returnBina e getBina posteriores, admitindo uma verificação da existência do utilizador (via getBalance), este voltará a ser instanciado no servidor Binas com os valores por omissão.

Quando as leituras na estação devolvem ‘null’ (o utilizador não tem conta instanciada nas stations), isto é passado com valores inteiros ‘-1’ na tag e saldo, e é tratado do lado do Binas. A thread que executa este pedido do lado do Binas espera pelas respostas inertemente – é ‘adormecida’ e verifica se recebeu alguma resposta periodicamente. Isto conserva energia (a do processador e a nossa, que esta implementação é mais fácil).

No caso específico do setBalance, foi implementado de forma a necessitar do getBalance para fazer a verificação (e, caso seja necessário, a re-inicialização) dos utilizadores, antes de executar a operação em si. Isto evita uma chamada de leitura remota desnecessária às estações, como explicado em cima, já que o getBalance devolve o valor guardado no Binas caso este esteja disponível.

Foi optado por uma estratégia de implementação de *polling*, porque, comparativamente ao *callback*, a atualização da informação aproxima-se de ser em tempo real, e apesar de ser mais custosa, o sistema nunca vai estar numa situação de falta de recursos.

# Troca de mensagens

De acordo com o Quorum Consensus, o cliente faz a operação para todos os servidores, e recebendo resposta de uma maioria deles, assume sucesso. Considerando qualquer tipo de exceção erro na resposta, e sabendo que o número de stations simultâneas é 3, se duas chamadas remotas devolverem corretamente, a operação foi bem-sucedida.

No caso do getBalance, a chamada enviará o email do utilizador e devolverá uma estrutura com dois valores – o saldo e a tag. Neste caso, é selecionado o valor cuja tag seja mais recente (maior sequence number). No caso do setBalance, é enviado o email, o novo saldo, e a tag, e o valor de retorno é void.