TÉCNICO LISBOA

Sistemas Distribuídos

PROJETO • Entrega 2

T_09 • LEIC-T • 2017/2018 • 2.º Semestre

Diogo Redin
84711

Gonçalo Matos
81943

Marta Simões
81947

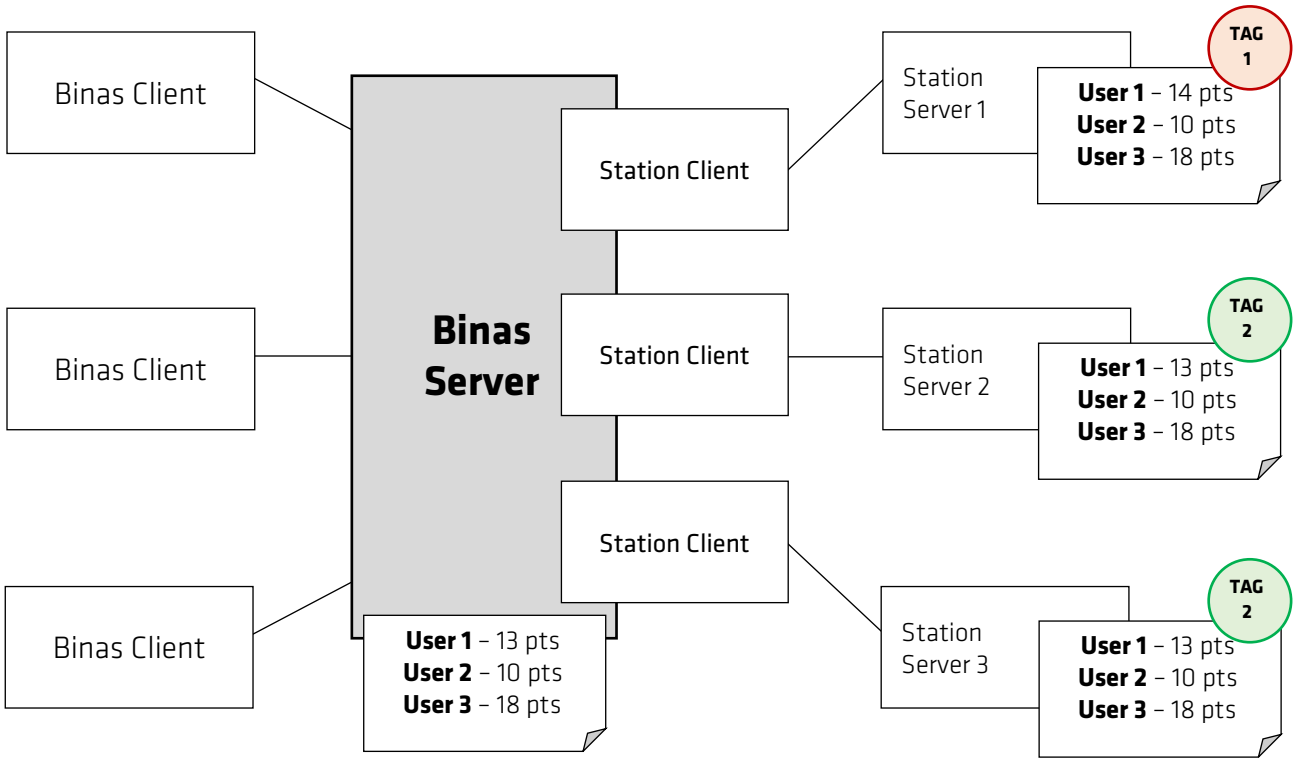

Figura 1 – Modelação da solução implementada.

Introdução

Para a segunda parte do Projeto tínhamos por objetivo permitir que após uma eventual falha do Binas, que o levasse a reiniciar e a perder os seus dados, fosse possível recuperá-los. Para tal, replicámos os dados dos utilizadores pelas Stations para que deixassem de estar disponíveis unicamente no Binas.

Quorum Consensus

O protocolo Quorum Consensus baseia-se num sistema de quórum que consiste em ter um conjunto de subconjuntos de réplicas de dados onde quaisquer dois subconjuntos se intersetam.

Neste Projeto, recorremos ao Quorum Consensus para:

- Leituras: em que é consultada uma maioria de réplicas de dados que permite determinar os dados mais atuais.
- Escritas: em que quando são alterados dados no servidor principal, as alterações são enviadas com uma tag atualizada para uma maioria de réplicas.

O facto de recorrermos a uma maioria de réplicas para leituras e escritas permite que exista sempre pelo menos uma réplica com os dados mais atuais.

Implementação

Com a nossa implementação, garantimos que os dados no Binas estão sempre atualizados e que este só precisa de ir buscar os dados dos seus utilizadores quando reinicia sem quaisquer dados. Para o Binas saber que acabou de acordar, tem um inteiro inicializado a 0, adiante denominado tag, que é incrementado sempre que é feita uma escrita.

Cenários de execução

Cenário I – Binas a operar normalmente

Quando é invocada uma função de escrita do Binas (como *activateUser* ou *setCredit*), a tag é incrementada e os dados que forem escritos são replicados para todas as Stations de forma assíncrona. Juntamente com os dados, é enviada a tag atual do Binas que é utilizada para determinar quão atuais estão os dados de cada utilizador nas estações. Os dados são enviados sob a forma de uma UserView que contém o e-mail, o saldo e uma tag.

Quando estas chamadas são enviadas, o Binas fica bloqueado (utilizando a biblioteca syncronized, vide Figura 2, e o tipo AtomicInteger), impedindo escritas ou leituras que utilizem dados que não são os mais atuais. Assim que o Binas recebe um retorno de uma maioria Stations ($\lfloor n/2 \rfloor + 1$), prossegue a sua execução e desbloqueia-se. Esta funcionalidade de update das stations encontra-se implementada numa única função updateStations, que é chamada nos dois tipos de escrita que podem ser feitas. Assim como o Binas tem uma lista de utilizadores, também cada Station passa a ter uma réplica dessa lista com um tipo User que adicionalmente tem uma tag e que não tem associada a informação de hasBina.

Cenário II – Binas reiniciado sem quaisquer dados

Esta trata-se da única situação em que o Binas estabelece ligação a, pelo menos, uma maioria de Stations para atualizar os dados que possui. Para saber se foi reiniciado, o Binas verifica se a sua tag é 0.

Esta funcionalidade de atualização dos dados do Binas encontra-se implementada numa única função, *updateBinas*, que é invocada em qualquer escrita ou leitura, quando o Binas acabou de acordar. Esta abordagem permite diminuir significativamente o número de ligações às Stations, reduzindo-as ao absolutamente necessário.

Para determinar com que dados atualizar o Binas, é enviado um pedido assíncrono a todas as Stations e aguarda-se uma resposta de uma maioria ($\lfloor n/2 \rfloor + 1$). Este pedido assíncrono encontra-se implementado sob a forma de uma função */listUsers* que retorna uma lista de UserViews de todos os utilizadores guardados numa dada Station.

Quando o Binas obtém resposta de uma maioria, percorre o conjunto de dados enviado por cada Station e vai colocando os dados de cada utilizador numa estrutura. Mediante comparação das tags, sempre que são encontrados dados mais atuais, estes são substituídos na estrutura.

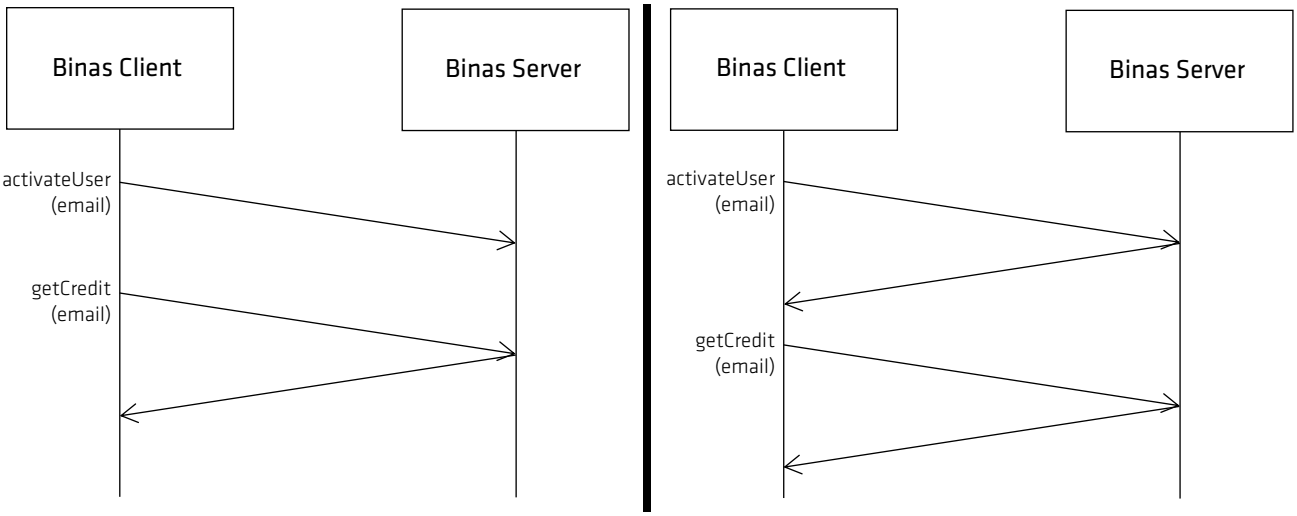

Figura 2 – Caso em que o Binas não fica bloqueado, permitindo leituras e escritas simultâneas (à esquerda) e em que as chamadas são feitas recorrendo à biblioteca syncronized, impedindo leituras e escritas simultâneas (à direita).