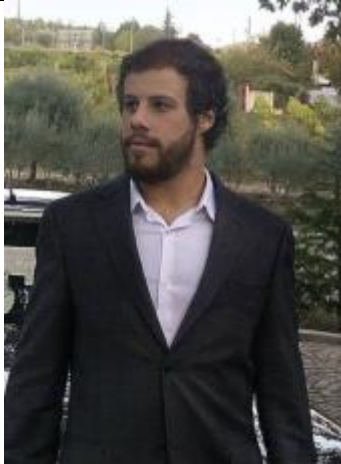






Sistemas Distribuídos

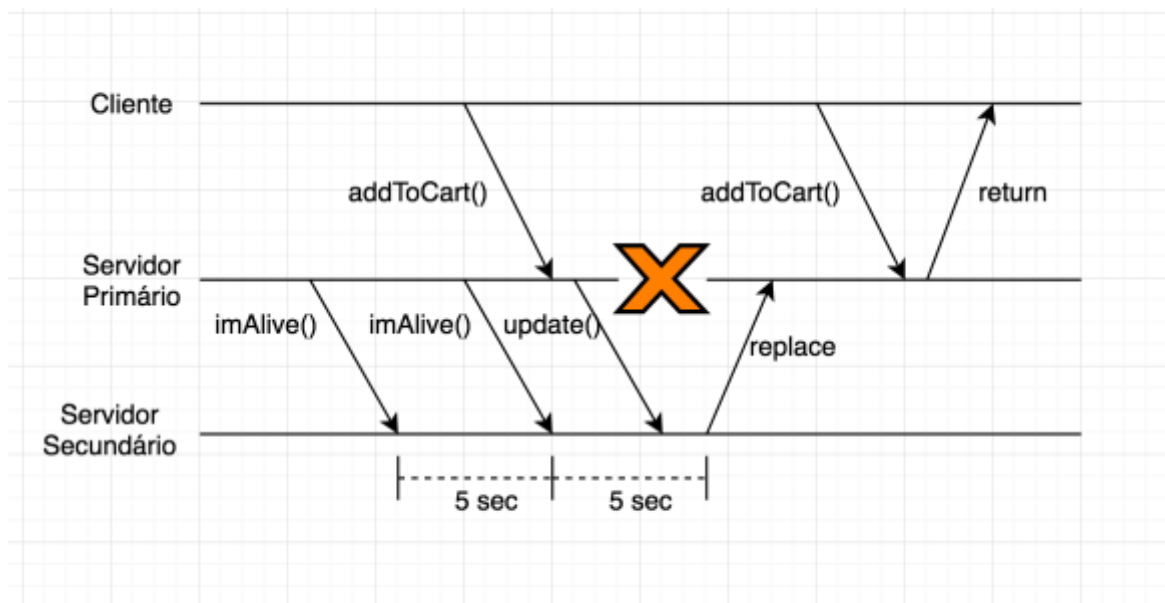
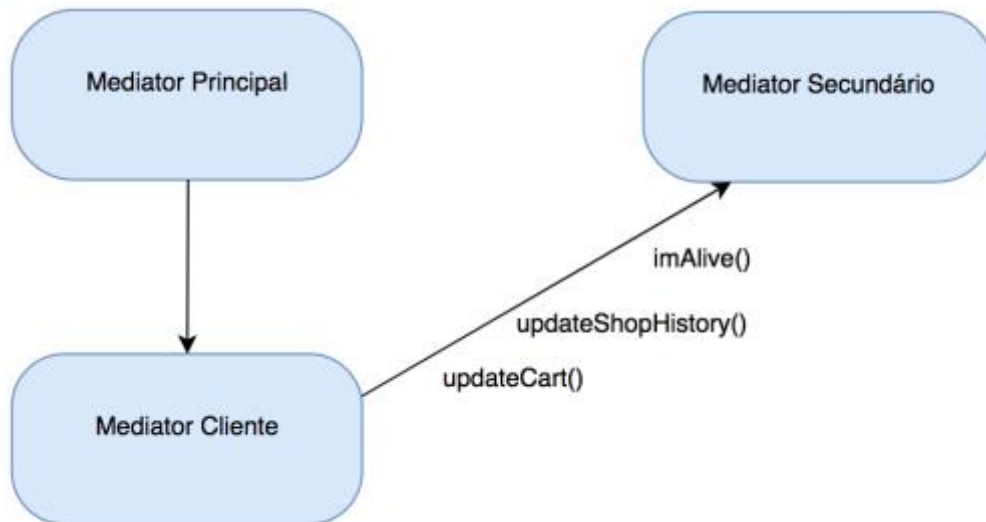
2016/2017

Grupo A65

		
Pedro Sousa	João Ramos	João Silvestre
78024	80915	80996

<https://github.com/tecnico-distsys/A65-Komparator>

Tolerância a Faltas



Replicação

A replicação consiste em ter dois servidores com a mesma informação, sendo o desafio ter a informação consistente entre os dois servidores.

O objetivo desta arquitetura é ter uma solução pronta a ser aplicada, caso o servidor primário falhe. Neste caso, o servidor secundário que é uma réplica do primário assume-se como servidor primário e continua as operações.

O servidor primário envia uma mensagem a cada cinco segundos para dar provas de vida através do método `imAlive()` ao servidor secundário. O servidor secundário só precisa de intervir quando deteta uma falta de provas de vida do primário. Para este projeto consideramos que o servidor primário tem conhecimento do servidor secundário.

As atualizações ao servidor secundário são feitas através dos métodos `update...` Cada operação gera um “timestamp” que serve como identificador único, para mais tarde verificar se existe incoerência entre as mensagens (mensagens repetidas, falha no envio das mensagens). Esta implementação serve essencialmente para evitar a repetição de operações (por exemplo, pagar duas vezes o mesmo carrinho).

Substituição do Servidor

Caso o servidor primário falhe, a consequência imediata é a não invocação do método `imAlive()`. O servidor secundário deteta então que o servidor primário deixou de enviar provas de vida. VERIFICAÇÃO SE O SERVIDOR PRIMARIO AINDA ESTA VIVO. Nesta situação o servidor secundário é registado no UDDI e pode ser encontrado pelo cliente. O servidor secundário toma as funções do servidor primário. Para este projeto consideramos que o servidor secundário nunca falha.

Front-end Cliente

A aplicação desenvolvida está sujeita a falhas de ligação, sendo que a rede nunca é 100% fiável. É então necessário introduzir uma política de “timeouts” para o caso em que o servidor primário falha, o cliente não enviar pedidos consecutivos e desnecessários.

Com a introdução de “timeouts” entre pedidos quando o servidor falha, ganha-se tempo para deixar o sistema recuperar (substituição do servidor primário pelo secundário, tornando-o o novo servidor primário) e é então possível retomar as operações tendo evitado um conjunto de pedidos desnecessários