**INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA**

Engenharia Informática e de Computadores

**Infra-Estruturas de Sistemas Distribuídos**

**4º Trabalho Prático**

**Gestor de Transacções**

**Desenvolvido por:**

Ricardo Neto (#26657)

**Análise do Problema**

O trabalho proposto tem o objectivo de permitir consolidar os conhecimentos relativos ao processamento de transacções distribuídas sendo para isso disponibilizado um cenário em que se pretendem fazer transferências inter-bancárias a partir de uma Applet.

Numa primeira fase cada transacção era feita em série não existindo mais nenhum cliente a aceder ao mesmo recurso o que não levantou problemas no que respeita à integridade dos dados.

Após algumas alterações, nomeadamente o aumento do número de transacções a executar em simultâneo, identificaram-se situações em que o resultado final não foi o esperado devido a conflitos no acesso aos recursos pela ausência de mecanismos de sincronização.

No sentido de impedir interferências na execução de uma transacção propõe-se a especificação de um gestor de acessos aos recuros partilhados (contas bancárias) implementado sob a forma de um serviço JINI e que cumpra com os requisitos do protocolo *two-phase locking*.

**Protocolo *two-phase locking***

O protocolo 2PC parte do principio de que um transacção envolve vários gestore

De acordo com os requisitos do enunciado e seguindo as indicações dada pelo docente, no sentido de tentar utilizar uma maior diversidade de tipos de objectos disponíveis, foi implementada a arquitectura em baixo mostrada, sendo a mesma detalhada de seguida.



Diagrama 1 - Diagrama de classes

*ShatServerSession* é uma *Session Bean* responsável por receber as mensagens dos clientes e publica-las em *ShatTopic.* Para a sua implementação optou-se por defini-la como *Statefull* com base na premissa de que não vai existir um número muito elevado de clientes. Assim, possibilita-se que um cliente faça o aproveitamento da sua instância entre chamadas e que os objectos locais da *bean* sejam mantidos penalizando a memória ocupada no contentor mas optimizando a rapidez no acesso. A publicação de mensagens no tópico é interceptada por *ShatInterceptor* que será responsável por alterar a mensagem, traduzindo-a para o idioma utilizado.

A identificação de mensagens com termos impróprios é feita através da *MessageDrivenBean* *ShatParentalControl* que subscreve o tópico. Assim, quando notificada por uma mensagem ter sido publicada, verifica o seu conteúdo junto da classe *ShatBadWords*, esta responsável por manipular os repositórios em memória relativos a termos impróprios e idioma utilizado. Caso a mensagem contenha um termo impróprio é enviada uma mensagem informativa para o tópico e registada na *ShatLogQueue* informação acerca da ocorrência.

No que respeita à administração dos termos impróprios, bem como, de qual o idioma a ser utilizado, foi implementado o *WebService*, materializado na classe *ShatAdmin*,que manipulará o repositório.

A classe *ShatBadWords* é, conforme indicado, responsável por manipular os repositórios em memória. Desta forma, foi delegado ao contentor a gestão da concorrência no acesso aos seus métodos, estando os mesmos classificados quanto ao tipo de operação a realizar sobre as estruturas de dados.

A aplicação cliente foi implementada com recurso a uma *Applet* através da classe *ShatAppletClient*. Esta mantém a estrutura utilizada nos trabalhos anteriores sendo que utiliza uma instância de *ShatClient* para comunicar com o contentor.