**INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA**

Engenharia Informática e de Computadores

**Infra-Estruturas de Sistemas Distribuídos**

**2º Trabalho Prático**

**Serviços JINI**

**Desenvolvido por:**

Ricardo Neto (#26657)

**Análise do Problema**

O trabalho proposto tem o objectivo de permitir consolidar os conhecimentos relativos aos serviços JINI estabelecendo um conjunto de requisitos a serem implementados com recurso aos mecanismos estudados.

O objectivo final é semelhante ao do trabalho anterior com a principal diferença de que a tradução de mensagens deverá agora ser disponibilizada sob a forma de um serviço.

**Solução Proposta**

De acordo com os requisitos do enunciado, foi implementada a arquitectura em baixo mostrada, no sentido de cobrir as funcionalidades necessárias ao serviço.



Diagrama - Diagrama de classes dos Serviços

A classe *ShatServer* é a responsável por instanciar o serviço e de registá-lo junto do RMI *Registry*, sendo o mesmo activado de imediato devido a não ser um objecto *Activatable*. Devido à configuração de arranque de *ShatServer* referir o *Codebase* local como uma *classpath* válida, são descarregados via http os binários dos tipos utilizados.

O serviço *MessageService* será responsável por agregar objectos *IMailbox* chamando o método *receiveMessage* de cada um deles por cada chamada a *multicastMessage*. Quando ocorre uma excepção ao notificar uma *mailbox* de uma nova mensagem o objecto que a representa é descartado. No que respeita à disponibilização de tradutores, foram feitos testes no sentido de criar uma condição para que essa responsabilidade fosse delegada a outras *mailboxes* registadas, contudo, não foi adoptada essa solução devido às dificuldades encontradas, sendo todos os tradutores disponibilizados pelo serviço.

Relativamente ao cliente foi implementada uma Applet estando definidas várias configurações de arranque no sentido de simular diferentes clientes com diferentes nomes e idiomas.



Diagrama - Diagrama de classes do Cliente

A classe ShatAppletClient será a responsável por obter um *proxy* para o serviço de mensagens e de registar o seu objecto IMailbox.Devido a implementar IMailboxListener a Applet, após se subscrever como observadora do seu objecto *Mailbox*, é notificada acerca da chegada de uma mensagem por cada callback feito no servidor ao objecto registado. Desta forma, torna-se também necessário que a implementação de IMailbox seja também um objecto que permita chamadas remotas.