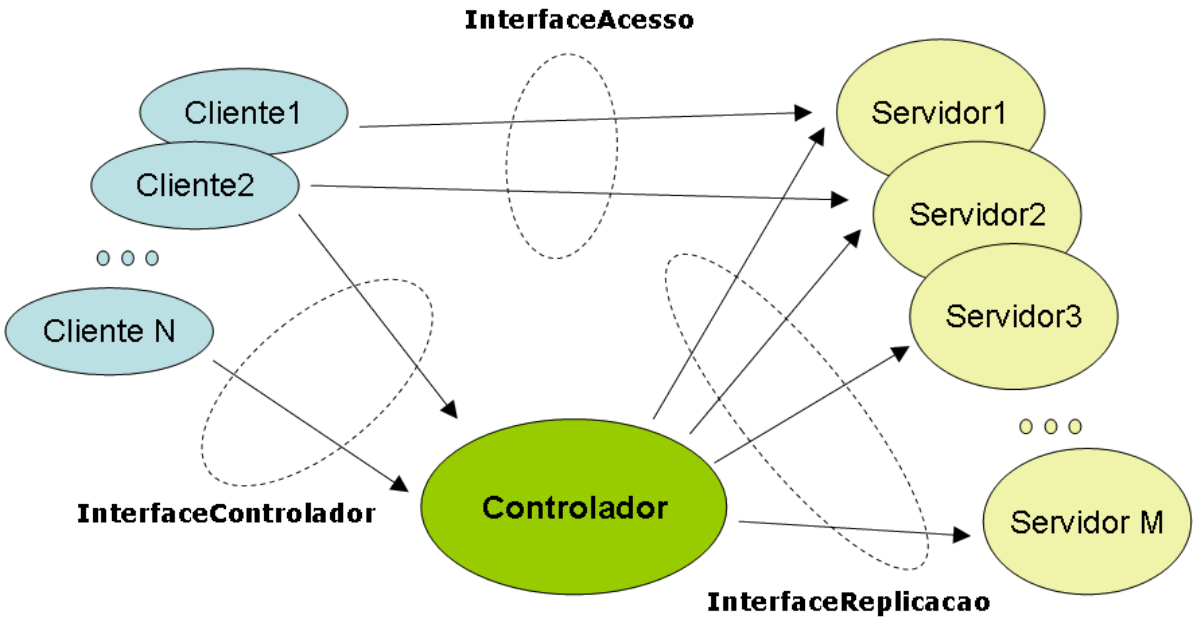
TRABALHO PRÁTICO
Armazenamento Distribuído de Objetos

Especificação

Sua empresa foi contratada para desenvolver um sistema eficiente e seguro de armazenamento de objetos quaisquer, em um ambiete OO (Orientado à Objetos). O arquiteto de soluções definiu que este sistema deveria ser implementado na forma de um sistema distribuído, com armazenamento dos objetos em memória (sem persistência em disco), com alta disponibilidade e alta performance. Como o prazo para desenvolvimento era curto, foi aprovado junto ao cliente ser aceitável a existência de um ponto crítico de falha, identificado como Controlador do Sistema, de forma a simplificar o desenvolvimento da solução. Também foi escolhida como linguagem de desenvolvimento o Java 5, sem o uso dos recursos de anotação.

A comunicação entre os elementos do sistema distribuído deveria utilizar a biblioteca de *middleware* RMI, de forma a tornar o desenvolvimento mais transparente em relação à comunicação, acelerar o desenvolvimento, e permitir grande flexibilidade na representação e transporte dos objetos.

A seguinte arquitetura foi especificada:



Os clientes, através da InterfaceControlador, poderão executar as seguintes operações no sistema:

`void armazena(String nome, Object obj)` *armazena o objeto obj identificado por nome.*

`String procura(String nome)` *busca o objeto identificado por nome e retorna o endereço do primeiro servidor disponível que contenha o objeto*

A string link (retornada pelo método) identifica a tupla {object_id, servidor, serviço} remota. Sua sintaxe é:

`#@rmi://SERVIDOR/SERVIÇO` *(sem nenhum espaço em branco)*

Onde:

- valor numérico (ID) identificando o arquivo (ex: 13, 89, 35222, etc) SERVIDOR - caminho do servidor que contém o objeto SERVIÇO - nome do serviço a ser acessado (no caso do RMI, nome do objeto remoto).

Exemplos:

```
13@rmi://computador1/Servidor
37479@rmi://computador2/Servidor
777@rmi://computador2/ServidorAlternativo
```

Cabe ao cliente interpretar (fazer o *parsing*) deste ***link***, de forma a extrair o ID e o URL (rmi://SERVIDOR/SERVIÇO), instanciar o serviço remoto representado pela URL, e chamar o método recupera(ID) neste serviço remoto para obter o objeto desejado.

`String[] lista()` *retorna uma lista de strings representando TODOS os objetos armazenados no sistema*.

`void apaga(String nome)` *apaga o objeto identificado por nome do sistema*.

Pela InterfaceAcesso, o cliente pode executar a seguintes operação no sistema:

`Object recupera(String nome)` *recupera um objeto à partir de um link obtido pelo método procura*

A recuperação do objeto foi separada de sua identificação (procura) de forma a garantir alta disponibilidade dos dados, evitar as perdas por queda de servidores, e permitir maior escalabilidade devido à possibilidade de múltiplas recuperações em paralelo dos objetos. Para isso, o Controlador replica cada um dos objetos armazenados em vários servidores: no momento da chamada ao método armazena(), o Controlador grava o objeto em todos os servidores disponíveis descartando os que porventura não responderem ou estiverem indisponíveis. Caso nenhum servidor esteja disponível, o Controlador deve retornar um erro.

Já na interface entre o Controlador e os Servidores, a InterfaceReplicacao, as seguintes operações estão disponíveis:

`void replica(int id, Object obj)` grava uma cópia do objeto obj no servidor
`void apaga(id)` apaga um objeto à partir de seu identificador numérico id

Optou-se por identificar os objetos nos servidores com o uso de um ID numérico único, para evitar inconsistências entre as possíveis diversas versões das réplicas de um mesmo objeto armazenado.

Para todos os métodos acima, as seguintes *Exception* são possíveis, além das exceções do RMI:

`NenhumServidorDisponivelException` **= *O Controlador não foi capaz de gravar nenhuma replica do objeto***
`ObjetoNaoEncontradoException` **= *O objeto identificado por*** *nome* ***ou*** *id* ***não foi encontrado***