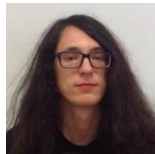




Fundamentos de Sistemas Distribuídos  
Livraria e Banco - Transações distribuídas e distribuição  
transparente  
MIEI

André Diogo



A75505

Diogo Pimenta



A

December 23, 2017

# Contents

<b>1</b>	<b>Introdução</b>	<b>2</b>
<b>2</b>	<b>Arquitetura</b>	<b>3</b>
2.1	Visão Geral da Arquitetura . . . . .	3
2.2	Serviço de Nomes e Gestor de Transações . . . . .	5
2.3	Modularidade e classes auxiliares . . . . .	5

# Chapter 1

## Introdução

Este relatório tratará de oferecer uma visão arquitetural do projeto realizado, além de considerações e justificações sobre a implementação escolhida para os diversos componentes.

A descrição da API do programa estará embebida no código fonte no formato javadoc.

## Chapter 2

# Arquitetura

### 2.1 Visão Geral da Arquitetura



Como mostra a figura, o nosso sistema está dependente de um processo que atua como serviço de nomes e gestor de transações conhecido por todos os outros processos.

Para garantir a consistência do sistema, caso este serviço falhe, ele mantém logs que refletem o seu estado atual, para poder ser reiniciado num estado são.

Temos assim ao nossa dispor a possibilidade de correr múltiplas lojas, bancos e clientes, sendo que estes se ligam apenas quando necessário recorrendo ao serviço de nomes para obterem a localização na rede do interveniente que requerem.

## **2.2 Serviço de Nomes e Gestor de Transações**

Para permitir uma configuração de rede razoavelmente dinâmica torna-se necessário ter um serviço de nomes capaz de registar servidores (quer de bancos quer de lojas) ativos, prontos para servir os clientes.

Como é necessário também ter um coordenador de transações para implementar o Two-phase commit, o serviço de nomes tratará também de o fazer no mesmo processo.

## **2.3 Modularidade e classes auxiliares**

Para permitir uma implementação bastante modular e genérica do código envolvido na manutenção de objetos distribuídos e conexão de rede criaram-se classes auxiliares dedicadas à gestão de referências distribuídas (DistObjManager), encapsuladora de funções comuns de leasing de objetos e garbage collection, assim como importação de Stubs e exportação de Servants (usou-se o termo Skeleton para estes na base de código); e de gestão de conexões (Server e Stub), encapsuladoras de funções comuns de gestão de ligações, registo de handlers e registo de callbacks.

Estas classes abstratas tentam deferir os detalhes mais específicos (classes concretas envolvidas) para implementações mais concretas ligadas à lógica de negócio dedicada às livrarias e bancos. Tentam ser o mais Plug 'n Play possível, seguindo uma abordagem de inversão de controlo.