Instituto Politécnico de Viseu

Escola Superior de Tecnologia e Gestão



Relatório do Trabalho Final

Sistemas Distribuídos

2021/2022

Licenciatura em Engenharia Informática

Trabalho realizado por:

Pedro Rocha, nº17707

Daniel Tejo, nº17743

Tiago Ferreira, nº18702

Docente: Carlos Cunha Viseu, 2022

# Índice

[Índice 2](#_Toc92664134)

[Introdução 3](#_Toc92664135)

[Requisitos Funcionais 4](#_Toc92664136)

[Requisitos Não Funcionais 5](#_Toc92664137)

[Arquitetura da solução em UML 6](#_Toc92664138)

[Implementação 7](#_Toc92664139)

[Conclusão 8](#_Toc92664140)

# Introdução

Este trabalho foi realizado no âmbito da unidade curricular Sistemas Distribuídos, na licenciatura de Engenharia Informática.

A área de inteligência artificial exige uma grande disponibilidade de recursos computacionais. O CPU é o recurso mais importante nesta área, uma vez que os algoritmos são computacionalmente muito intensivos. Como tal, num sistema distribuído torna-se fundamental rentabilizar a utilização deste recurso nos vários elementos computacionais.  
​  
 Pretende-se desenvolver uma solução distribuída que permita distribuir os pedidos de computação pelos elementos disponíveis do sistema, de forma a garantir o equilíbrio da carga entre os mesmos.  
​  
 Existem vários tipos de elementos no sistema. Os **processadores** são elementos que executam os algoritmos. Os **carregadores** são elementos que armazenam os dados a processar. Os **cérebros** são elementos que armazenam os modelos gerados. Os **estabilizadores** garantem a distribuição equilibrada da carga pelos processadores.

Para realizarmos este trabalho utilizámos o programa IntelliJ IDEA, que é um ambiente de desenvolvimento integrado escrito em Java.

# Requisitos Funcionais

RF01

Ao receber um pedido de processamento do cliente, o serviço **estabilizador** escolhe a instância do **processador** com **mais recursos computacionais** e devolve o seu id na resposta do pedido  
​  
RF02

Ao receber um pedido de processamento do cliente com a script que deverá ser executada, o processador atribui-lhe um **id** e **inicia o processamento** se tiver **recursos disponíveis** (RF03) ou caso contrário, **coloca o pedido em lista de espera**. Em qualquer uma das situações, devolve o **id** do pedido ao cliente.  
​  
RF03

Antes de iniciar o processamento do cliente, o processador deve contactar um dos carregadores para obter os dados necessários para o processamento  
​  
RF04

Ao finalizar a execução de um pedido, o processador deverá enviar o modelo gerado a um dos cérebros, juntamente com o id do processador  
​  
RF05

Ao receber um pedido de um modelo do cliente, o cérebro devolve o modelo associado utilizando o id do pedido de processamento (RF02).

# Requisitos Não Funcionais

NF01

O protocolo **SFTP** é utilizado para a comunicação entre os **processadores** e os **carregadores**  
​  
NF02

Os processadores devem fazer o broadcast de **heartbeats periódicos** com a informação dos seus recursos e tarefas para todos os processadores e estabilizadores. Desta forma, os estabilizadores podem saber quais são os processadores ativos e os recursos que têm disponíveis e os outros processadores podem assegurar a execução das suas tarefas em cenários de falha.  
​  
NF03

Quando um estabilizador deixa de receber heartbeats durante um período de tempo superior a 30 seg, deverá remover o respetivo processador da lista de processadores disponíveis. Posteriormente, o estabilizador notifica o processador com mais recursos, para resumir os pedidos pendentes do processor com falhas. Para obter a lista de tarefas que não foram concluídas pelo processador com falhas, o processador substituto deverá questionar todos os cérebros da existência do modelo resultante do processamento.  
​  
NF04

Quando um estabilizador recebe um heartbeat do tipo "setup", vindo de um processador que não se encontra na lista, adiciona o processador  
​  
NF05

Se um estabilizador falhar, o estabilizador substituto poderá recuperar o estado dos processadores a partir de um dos processadores ativos

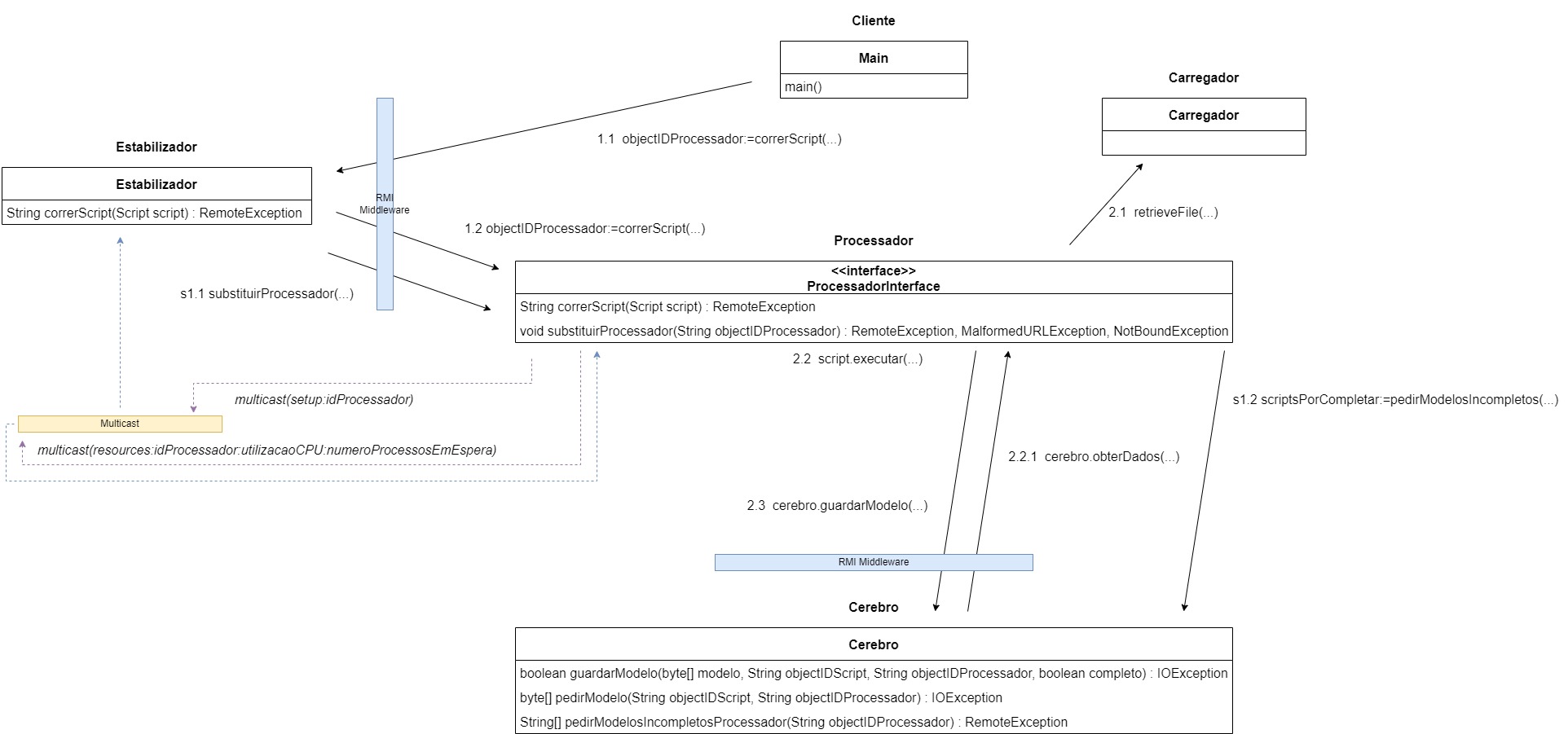
NF06

Os ids dos elementos e dos pedidos devem ser [universais](https://www.baeldung.com/java-uuid)

NF07

O acesso aos recursos partilhados por várias threads têm de ser *thread-safe*

# Arquitetura da solução em UML



# Implementação

# Conclusão

Neste trabalho conseguimos realizar todos os requisitos funcionais e maior parte dos requisitos não funcionais.

Apesar de tudo, concluímos que este trabalho foi realizado com sucesso graças ao esforço, persistência e apoio de todos os elementos do grupo.