Graduação em Análise e Desenvolvimento de Sistemas

Trabalho de Conclusão de Curso

Sistema para Automação de Carga de Trabalho

Felipe Fonseca Ribeiro

Marco Paulo Correia da Mota Ollivier

Orientador: Érico Corrêa Torres

Rio de Janeiro

Novembro, 2012

SISTEMA PARA AUTOMAÇÃO DE CARGA DE TRABALHO

Felipe Fonseca Ribeiro

Marco Paulo Correia da Mota Ollivier

Trabalho de Conclusão de Curso apresentado ao Curso de Análise e Desenvolvimento de Sistemas do Instituto Infnet como requisito parcial para a obtenção de grau de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Rio de Janeiro

Novembro, 2012

RIBEIRO, Felipe Fonseca, **OLLIVIER**, Marco Paulo C. da Mota.

Sistema para Automação de Carga de Trabalho / RIBEIRO, Felipe Fonseca, OLLIVIER, Marco Paulo C. da Mota, 2012, número de folhas.

CUTTER

Monografia (Curso de Análise e Desenvolvimento de Sistemas) – Instituto Infnet, RJ.

1. Sistema de Gerenciamento. 2. Automação de carga de trabalho. 3. Gerenciamento de Batch.

CDU - NUMERAÇÃO

FELIPE FONSECA RIBEIRO

MARCO PAULO CORREIA DA MOTA OLLIVIER

PROFESSOR ORIENTADOR: ÉRICO CORRÊA TORRES

SISTEMA PARA AUTOMAÇÃO DE CARGA DE TRABALHO

Trabalho de graduação aprovado para a conclusão do curso de Análise e Desenvolvimento de Sistemas, no Instituto Infnet, pela comissão formada pelos seguintes professores:

EXAMINADORES:

NOME:_____

TITULAÇÃO:_____

NOME:_____

TITULAÇÃO:_____

NOME:_____

TITULAÇÃO:_____

NOME:_____

TITULAÇÃO:_____

NOME:_____

TITULAÇÃO:_____

PARECER FINAL

_____

_____

RESUMO

A pesquisa pretende apresentar uma solução para o gerenciamento de processos automatizados em ambientes corporativos. Esse trabalho visa prevenir falhas de execuções provenientes da falta de recursos de *hardware* do servidor. Esse tipo de falha se deve à possibilidade de diversos processos serem executados simultaneamente no mesmo servidor, concorrendo pela utilização de processador e de memória. Partindo-se do pressuposto de que cada processo necessita de uma quantidade mínima de recursos para ser executado, tem-se como hipótese a utilização da Inteligência Artificial (IA) baseada em conhecimento, para coletar e armazenar o histórico de consumo de cada um dos processos. Com esse histórico, o sistema será capaz de tomar decisões de quando e como executar um determinado processo de acordo com sua necessidade de recursos. Dessa forma, será possível otimizar a execução dos processo agendados, aproveitando o máximo dos recursos do servidor e diminuindo o número de falhas e travamentos das execuções.

Palavras-chave: processos automatizados; consumo de *hardware*; inteligência artificial; gerenciamento de processos;

ABSTRACT

This research intends to propose a solution for managing automated processes on corporate environments. This project aims at the prevention of failures during executions due to the shortage of the server's hardware resources. This kind of failure occurs because of the execution of several processes simultaneously in the same server, competing to use the processor and the memory. Inferring that each process needs a minimal amount of resources to be executed, Artificial Intelligence (AI) based on knowledge is introduced, to collect and store the history of usage of each process. With this history, the system will be capable of deciding when and how to run a specific process according to its need of resources. Thereby, it will be possible to optimize the execution of scheduled processes, taking full advantage of server resources and reducing the number of fails and crashes.

Keywords: automated processes; hardware usage; artificial intelligence; process management

SUMÁRIO

DICIONÁRIO DE SIGLAS

API: *Application Programming Interface*

BI: Business Inteligence

CRM: *Customer Relationship Management*

ERP: *Enterprise Resource Planning*

HTTP: *Hyper Text Transfer Protocol*

IA: Inteligência Artificial

IDE: *Integrate Development Environment*

JEE: Java *Enterprise Edition*

JSE: Java *Standard Edition*

JVM: Java *Virtual Machine*

RBC: Raciocínio Baseado em Casos

SDK: *Software Development Kit*

SO: Sistema Operacional

SQL: *Structured Query Language*

SVN: Subversion

TI: Tecnologia da Informação

UML: *Unified Modeling Language*

XML: *Extensible Markup Language*

LISTA DE FIGURAS E IMAGENS

1. INTRODUÇÃO

A automação está cada vez mais presente em nosso dia-a-dia, seja o vidro-elétrico do carro, ou a cafeteira programada para preparar o café pela manhã. E quando falamos em informática isso é muito mais comum.

Servidores de *e-mail*, aplicações *web*, bancos de dados, servidores de arquivos, ERP[1], CRM[2], CMS[3] são alguns exemplos dos diversos *softwares* que auxiliam as empresas a atingirem os seus objetivos. E manter todos esses serviços funcionando perfeitamente não é uma tarefa fácil. Rotinas de *backup*[4], replicação, transformação de dados, transferência de informações, e outras diversas rotinas se fazem necessárias. No entanto, realizar essas atividades de forma manual gera um alto risco para a empresa, pois é mais suscetível a falhas. E alguns desses serviços quando saem de funcionamento podem provocar um prejuízo à organização.

Uma forma de automatizar essas tarefas é a utilização de pequenas rotinas onde uma sequência de atividades seria previamente estabelecida. Isso é possível através da utilização de arquivos *batch*[5]. E a execução desses arquivos pode ser feitas de forma manual, ou seja, por um profissional, ou de forma automática, utilizando ferramentas de agendamento.

Em pesquisas realizadas nas empresas *Target Solutions* e Calandra Soluções, foi identificada a necessidade de execução de mais de três rotinas para gerenciamento das suas atividades internas. E o motivador da nossa pesquisa foi a necessidade encontrada em um projeto de BI (*Business Intelligence*) na Calandra Soluções onde era necessário o agendamento de quatro tarefas para serem executadas durante a madrugada, e por falta de recursos de *hardware* essas

[1] *Enterprise resource planning*: Sistema para integração de gestão empresarial

[2] *Customer Relationship Management*: Sistema para gerenciamento de gestão com o cliente

[3] *Content Management System*: Sistema para gerenciamento de conteúdos

[4] Termo utilizado na computação para cópia de segurança

[5] Arquivo de *scripts* para a execução automatizada de comandos interpretados por um Sistema Operacional.

atividades não puderam ser concluídas antes do inicio da próxima tarefa, causando o travamento do servidor.

Tendo em vista a necessidade de controlar as execuções de tarefas levando em consideração a disponibilidade dos recursos de *hardware*, o presente trabalho tem como tema o gerenciamento de cargas de trabalho automatizadas.

Este trabalho parte do seguinte problema de pesquisa: Como gerenciar as execuções de tarefas automatizadas evitando a sobrecarga do ambiente de processamento? E como executar o maior número de atividades no menor tempo possível?

Defende-se a hipótese de que uma atividade consome, em média, a mesma quantidade de memória e de processador em cada uma das suas execuções. Portanto, ao alimentar uma base histórica de execuções, armazenando esses consumos, é possível prever se é ou não possível executar a atividade naquele momento. Com a constante execução dessas tarefas, através de técnicas de IA (Inteligência Artificial) será possível melhorar gradualmente o desempenho da aplicação.

O objetivo geral do trabalho é desenvolver um *software* para o gerenciamento de cargas de trabalho para executar o maior número de atividades no menor tempo possível, afim de diminuir o número de falhas por falta de recursos de *hardware*. Para isso, pretende-se agendar e disparar a execução das tarefas, além de armazenar o histórico de consumo de cada uma das atividades e criar estatísticas referente às execuções.

Como metodologia utilizou-se a pesquisa aplicada, tendo os dados coletados em entrevistas nas empresas *Target Solutions* e Calandra Soluções. Além de leitura de livros e de materiais online, que estão devidamente mencionados nas Referências.

2. JUSTIFICATIVA

A automação de processos está a cada dia mais presente no cotidiano de empresas devido ao aumento de demanda de atividades feitas utilizando algum tipo sistema computacional (envio de *e-mails*, *backups* de arquivos, migração de dados entre Bancos de Dados, renderizações de imagens etc.), porém muitas vezes o gerenciamento dessa demanda ainda é feito da forma inadequada.

Diante da premissa apresentada, foi feito um levantamento em três empresas: a *Target Solutions*, empresa que presta soluções em TI (Tecnologia da Informação) voltadas para o ramo de telecomunicações; a Calandra Soluções, empresa que presta soluções também em TI; e por fim a CETIP empresa do ramo financeiro que possui um grande núcleo de desenvolvimento de *software*. Através desse levantamento, puderam-se comprovar três cenários relevantes que serviram de motivação para o desenvolvimento de uma solução que atendesse as necessidades que serão apresentadas a seguir.

No primeiro cenário, pôde-se verificar que em alguns casos os processos automatizados são feitos utilizando aplicativos de agendamento que atendem a uma necessidade básica, ou seja, apenas executam a operação na hora marcada. Em alguns casos, inclusive, o aplicativo já era nativo do Sistema Operacional. Porém nesses casos, notou-se que caso ocorresse algum tipo de execução simultânea entre vários processos, poderia ocorrer uma sobrecarga de consumo de *hardware* e consequentemente interromper o processamento por falta de recursos.

No segundo cenário, verificou-se que a execução dos processos era controlada por uma ferramenta especializada no gerenciamento de execução de processos, mostrando-se, inclusive, ser bem completa visto que atende necessidades mais complexas como, por exemplo, o gerenciamento de execução dos processos de acordo com os recursos disponíveis. Dentro das funcionalidades que foram apresentadas, pôde-se notar a ausência de um controle de liberação de processos de acordo com os recursos disponíveis no servidor, o que seria uma funcionalidade interessante, visto que diminuiria o custo operacional no caso de eventuais manutenções fora do horário de expediente. Mas o fator que mais foi levado em

conta nesse cenário é o fato da ferramenta ser importada, o que proporciona uma dificuldade relativa ao suporte e treinamento.

No terceiro cenário, nota-se o que pode vir a ser considerado o pior de todos, onde não existe um agendamento da execução do processo, dependendo assim da permanência de um profissional executando e monitorando a execução dos processos. Nesse caso o risco de existir falha humana é iminente podendo comprometer a execução e o desempenho dos processos.

Uma vez otimizado o gerenciamento da execução dos processos, toda a comunidade (tanto equipe técnica quanto cliente/usuário final) será beneficiada com diminuição nas falhas provenientes de falta de monitoramento e a redução de tempo de execução visto que o gerenciamento das execuções agilizará o processo de execução das tarefas.

Tendo em vista o desenvolvimento de uma nova solução, a contribuição científica apresentada será através do resultado final do projeto. A importância da utilização de boas práticas durante o processo de concepção do *software* deixará para a posteridade uma gama utilizações de padrões de projetos e de utilização de *frameworks*[6] contextualizadas no domínio do problema que foi durante a pesquisa.

[6] Conjuntos de bibliotecas empacotadas para um objetivo específico, visando assim a sua reutilização em outras aplicações.

3. OBJETIVOS

3.1. OBJETIVOS GERAIS

Pretende-se desenvolver um *software* para a execução do maior número de processos no menor tempo possível, afim de diminuir o número de falhas por falta de recursos de *hardware*.

3.2. OBJETIVOS ESPECÍFICOS

- Agendar a execução de tarefas (conjunto de processos);
- Disparar as tarefas e monitorar a execução dos processos;
- Armazenar o histórico de consumo de cada processo;
- Criar estatísticas referentes à execução de cada processo;
- Utilizar as estatísticas armazenadas para otimizar a execução das tarefas;

4. HIPÓTESES

Com base em testes e análises efetuadas pelos desenvolvedores deste trabalho, defende-se a hipótese de que uma atividade consome, em média, a mesma quantidade de memória e de processador em cada uma das suas execuções. Algumas atividades podem sofrer variações, porém essas variações são bem definidas, ou seja, pode ser identificados o consumo mínimo e o consumo máximo.

A partir desses fatos, sugere-se a utilização da técnica de Inteligência Artificial chamada Raciocínio Baseado em Casos (RBC)[7]. Com isso, a cada vez que um processo for executado, será armazenado em uma base de dados, as informações referentes àquela execução, tais como: os horários de início e de término; o menor e o maior consumo de memória; o consumo total de memória; o menor e o maior consumo de processador; e o consumo total de processador. Com essas informações históricas armazenadas será possível prever se o processo pode, ou não, ser executado naquele momento. Sendo assim, a cada nova execução do processo, a base histórica vai ficando mais rica e consequentemente as previsões ficam mais precisas, melhorando gradativamente o desempenho do *software*.

Foi elaborado um algoritmo para a execução das tarefas[8], que funciona da seguinte maneira: Quando uma tarefa é enviada para a execução, o sistema busca os seus processos e cada um deles com o seu respectivo histórico; Em seguida é feita uma análise no histórico para descobrir o quanto de *hardware* será necessário para executar cada um dos processos; Os processos são enfileirados para a execução; Enquanto houver processo na fila, os passos a seguir serão executados; O sistema identifica o primeiro processo da fila e realiza algumas verificações, afim de descobrir se ele pode ser executado naquele momento. Para facilitar a

[7] O Raciocínio Baseado em Casos (RBC) é uma técnica utilizada para solucionar problemas de forma automática. O RBC tem por característica principal a reutilização de informações e conhecimentos em situações anteriores que são similares ao novo problema.

[8] No contexto deste trabalho, tarefa é um conjunto de processos a ser executados. Dentro de uma tarefa, é possível definir as dependências entre os processos.

manutenção dessas regras foi utilizado um padrão de projeto de *software* chamado *Chain of Responsability*[9]. Esse padrão de projetos, torna a inserção de novas regras mais fácil. Atualmente, criamos duas regras: Validação de dependências e Validação de recursos de *hardware*. Na regra de validação de dependências é verificado se o processo atual possui alguma dependência entre os processos da tarefa. Se possuir, o sistema verifica se essas dependências já foram executadas. Na regra de validação de recursos de *hardware*, o sistema identifica a situação atual do servidor, ou seja, a quantidade de memória disponível e o percentual de uso do processador. Com base nessas informações e com a análise feita anteriormente sobre o histórico de execução do processo, o sistema verifica se o servidor possui os recursos necessários para que este processo possa ser executado. Se o processo atual não atender alguma das regras, ele volta para o final da fila. Já se todas as regras foram atendidas, o sistema dispara uma *Thread*[10] para a execução desse processo no Sistema Operacional(SO). Paralelamente, também é disparada uma *Thread* para monitorar a execução desse processo no SO, medindo o consumo de *hardware*. Para esse monitoramento foi utilizado um padrão de projetos chamado *Observer*[11], que auxiliou na notificação à *Thread* monitor de que o processo foi executado. Quando a execução do processo é concluída, as dados referentes ao consumo de *hardware* são armazenados na base de dados; Esses passos são repetidos até que todos os processos da tarefa sejam executados. Dessa forma, evitamos que algum processo entre em execução até que o servidor possua os recursos necessários, reduzindo os erros por falta de recursos de *hardware*.

Além disso, o sistema disponibiliza um painel de controle onde é possível monitorar os processos que estão em execução e que estão aguardando para entrar em execução. Esse painel fornece uma visualização gráfica, em tempo real, do consumo de memória e processador do servidor. Com essa funcionalidade, os administradores do sistema podem acompanhar o estado atual do servidor.

[9] Para ler sobre o padrão de projetos *Chain of Responsability*, vide o capítulo 5 - Pressuposto Teórico.

[10] *Thread* é um fluxo independente dentro de um programa. *Threads* permitem um sistema executar diversas tarefas, independente uma da outra.

[11] Para ler sobre o padrão de projetos *Observer*, vide o capítulo 5 - Pressuposto Teórico.

O sistema disponibiliza ainda uma área onde é possível analisar as estatísticas de execução de cada um dos processos, ou seja, quantidade de execuções, de falhas, consumo médio de memória e consumo médio de processador.

5. PRESSUPOSTOS TEÓRICOS

5.1. INTELIGÊNCIA ARTIFICIAL

A Inteligência Artificial (IA) tem como um dos seus objetivos o desenvolvimento de ferramentas computacionais que simulam o comportamento intelectual dos seres humanos.

Segundo Rich e Knight (1994),

> “Inteligência Artificial (IA) é a área da ciência da computação orientada ao entendimento, construção e validação de sistemas inteligentes, isto é, que exibem de alguma forma, características associadas ao que chamamos inteligência.”

A palavra “inteligência” é originária do latim inter (entre) elegere (escolher), onde trazem a ideia de que a “inteligência” permite ao ser humano realizar escolhas entre duas opções distintas. A palavra “artificial” também vem do latim artificiale (algo não natural), ou seja, produzido pelo homem.

IA tem o objetivo de modelar o modo de pensar dos seres humanos em processos computacionais.

> “O estudo de como fazer computadores realizarem tarefas que no momento as pessoas fazem melhor.” [5]

Além de armazenar e manipular dados, um sistema IA é capaz também de adquirir, representar e manipular conhecimento. Com a manipulação é possível deduzir ou inferir novos conhecimentos a partir do conhecimento que já existe e utilizar métodos de representação e manipulação para solucionar problemas complexos que na maioria dos casos não são quantitativos.

5.1.1. Raciocínio Baseado em Casos

Raciocínio Baseado em Casos (RBC) é uma técnica de IA utilizada para solucionar problemas de forma automática. O RBC tem como característica principal a reutilização de informações e conhecimentos em situações anteriores que são similares ao novo problema. Um sistema RBC resolve problemas,

ajustando soluções que foram utilizadas anteriormente para a resolução de problemas.

A qualidade de um sistema RBC está diretamente relacionada à sua base de casos, ou seja, à sua experiência, além da sua capacidade de adaptação e de avaliação.

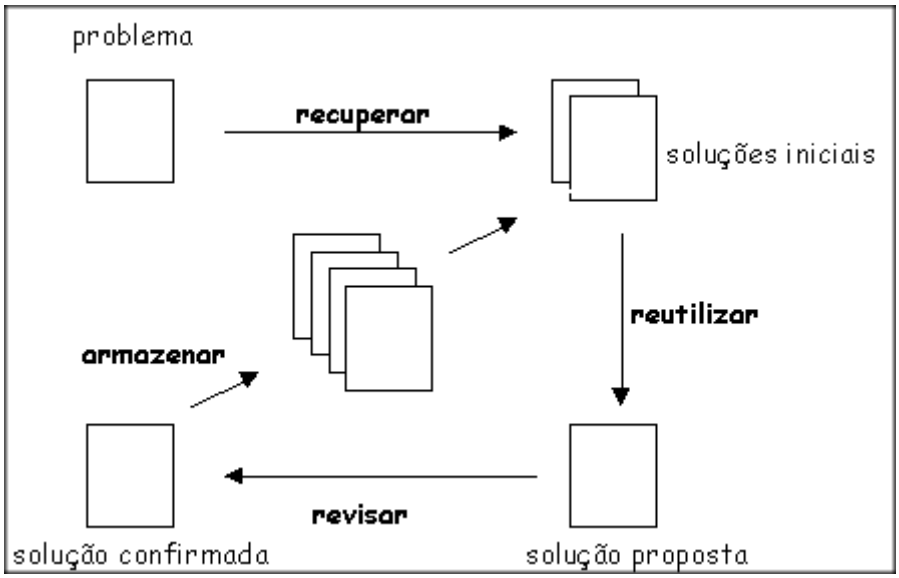

Figura 1: Raciocínio Baseado em Casos

O ciclo de funcionamento do RBC possui quatro etapas:

- **Recuperação** de casos similares ao problema atual. Ela é feita a partir das características mais relevantes para a solução do novo caso;
- **Reutilização** da solução encontrada pelo caso similar;
- **Revisão** da solução recuperada para ajustar ao problema atual, avaliando as diferenças entre o caso recuperado e o caso atual;
- **Retenção** do caso atual e da sua solução para recuperações posteriores.

5.2. SISTEMAS DISTRIBUÍDOS

Com a evolução da computação, foi possível alcançar rapidamente um nível satisfatório, no que se diz respeito a processamento de informações. Paralelo a isso, a evolução das redes de computadores possibilitou que a troca dessas mesmas informações fosse feita de forma rápida e eficiente. Pode-se comprovar essa

afirmação voltando um pouco no tempo. Uma máquina que inicialmente conseguia executar apenas uma instrução por segundo e tinha um custo elevado passou a executar um bilhão de instruções por segundo e seu custo foi reduzido consideravelmente. Somada à popularização do computador, veio também a popularização das redes e, por consequência, da internet, permitindo assim que um público vasto (de grandes empresas a usuários domésticos) pudesse trocar centenas de informações em milissegundos. Visto que se passou a ter uma vasta capacidade de transferência e de processamento computacionais, por que não executar uma mesma informação em vários lugares diferentes (otimizando o processo de execução através da execução de uma única tarefa em vários núcleos computacionais) e transferir essas informações através de uma rede de computadores? Pensando nisso surgiu o que é conhecido nos dias de hoje como Sistemas Distribuídos.

Na computação, um Sistema Distribuído (SD) pode ser qualquer sistema que esteja sendo executado paralelamente em mais de um ponto de processamento através de uma rede, tendo como objetivo executar uma tarefa em comum. De acordo com Tanenbaum [7] é uma coleção de computadores independentes que se apresenta ao usuário como um sistema único e coerente.

Um SD tem como características principais:

- Componentes (computadores, em geral) independentes entre si que executam uma mesma tarefa em comum;
- A maneira que ele é executado é transparente, ou seja, o consumidor/cliente não sabe como ele processa suas tarefas, visualizando assim uma aplicação única;
- Sempre está disponível para o consumidor. Mesmo que algum componente apresente problemas, ele tem que continuar sua execução.

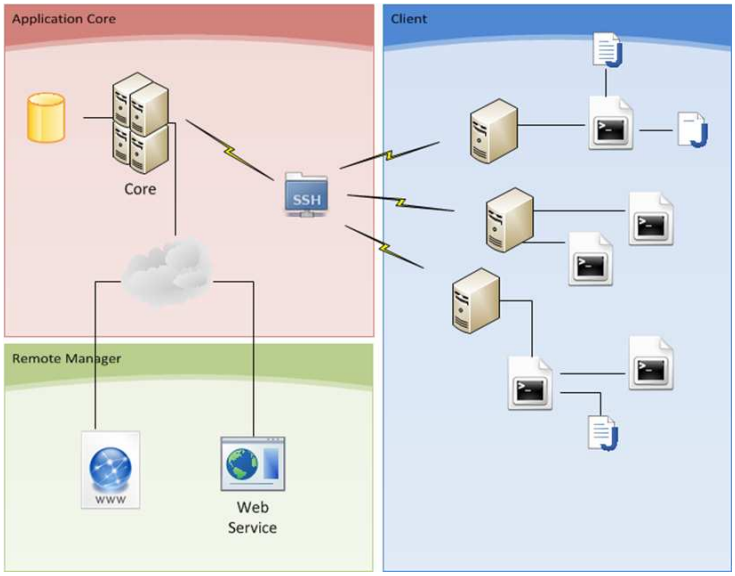

Figura 2: Contexto do projeto dentro de um ambiente distribuído.

5.2.1. Arquitetura Cliente/Servidor

A arquitetura cliente servidor é uma das mais conhecidas no mundo da informática. Nessa arquitetura as informações ficam centralizadas em um computador (servidor) e outras máquinas acessam essas informações via rede (clientes).

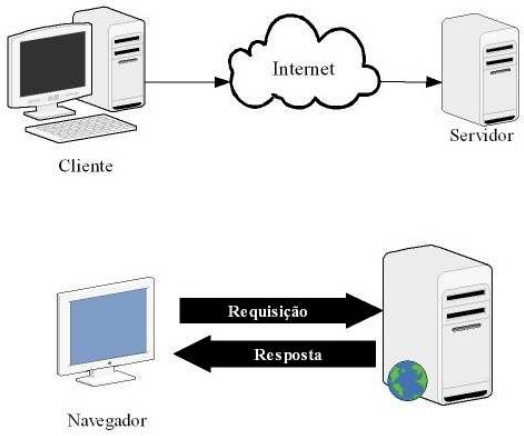

Figura 3: Arquitetura Cliente-Servidor

5.3. JAVA

De acordo com Deitel [2], em torno de 1991, a Sun Microsystems financiou um projeto de pesquisa que acabou proporcionando uma linguagem de programação baseada em C++. Inicialmente a linguagem chamava-se Oak (nome dado em

homenagem a uma árvore que existia perto do escritório de Sun), porém como já existia uma linguagem de programação com esse nome, foi dado o nome de Java (devido ao nome do local onde era fabricado um tipo de grão de café importado).

Ainda de acordo com Deitel, o projeto, ainda em sua fase de pesquisa, passou por um período muito turbulento, pois a evolução do mercado de eletrônicos não evoluiu da maneira que a Sun pensou. Porém, em 1993, com o surgimento da *Web*, a Sun viu uma oportunidade de colocar o Java em um estado de destaque, implementando, assim, uma série de bibliotecas dinâmicas que auxiliariam a utilização da linguagem em aplicações voltadas para a internet.

Alguns outros fatores influenciaram a escolha do Java como a linguagem desse sistema, como por exemplo;

- A existência de uma plataforma sólida e consolidada como a JVM facilita a implementação de programas em Java;
- Também devido à existência da JVM, existe a interoperabilidade entre plataformas, fazendo com que uma única aplicação consiga ser executada nos principais Sistemas Operacionais que estão no mercado;
- Sua vasta quantidade de bibliotecas existentes facilita o desenvolvimento de aplicações para diferentes plataformas, desde celulares até servidores de grande porte;

5.4. PADRÕES DE PROJETO

O software desenvolvido possui muitas regras e validações, o que deixou alguns pontos do código-fonte muito complexos. E visando evoluções futuras, foram utilizados alguns padrões de projetos afim de tornar a manutenção do código mais fácil.

Segundo Erick Gamma,

> “Todos *(padrões de projetos)* podem ser implementados em linguagens orientadas a objetos comuns, embora possam exigir

um pouco mais de trabalho que soluções *ad hoc*[12]. Porém, o esforço adicional invariavelmente traz dividendos na forma de flexibilidade.” [8]

5.4.1. Singleton

Garante que uma classe tenha somente uma instância e fornece um ponto global de acesso a essa instância. [8].

Como o objetivo desse trabalho é gerenciar os recursos de *hardware* do servidor, este sistema não pode ser uma carga pesada para o servidor. Portanto, utilizamos o padrão de projetos *Singleton* em algumas classes para evitar a criação de vários objetos desnecessariamente.

Para auxiliar no gerenciamento desses objetos *Singleton*, foi utilizado um *framework* chamado Spring utilizando a funcionalidade de injeção de dependências[13].

5.4.2. Chain of Responsability

O *Chain of Responsability* é um padrão de projeto comportamental que tem como objetivo evitar o acoplamento do remetente de uma solicitação ao seu destinatário, dando a mais de um objeto a chance de tratar a solicitação. Encadeia os objetos receptores e passa a solicitação ao longo da cadeia até que um objeto a trate. [8]

No presente trabalho, esse padrão foi utilizado para as regras de verificação da execução dos processos (dependências e disponibilidade de recursos). Com isso, a inclusão de novas regras e a manutenção das existentes ficou mais fácil.

[12] *Ad hoc* é uma expressão latina que significa "para esta finalidade" ou "com este objetivo". Geralmente se refere a uma solução destinada a atender a uma necessidade específica ou resolver um problema imediato - e apenas para este propósito, não sendo aplicável a outros casos.

[13] Injeção de dependências é um padrão de desenvolvimento que é utilizado quando existe a necessidade de um baixo acoplamento entre partes de um sistema. Para isso, um *container* se responsabiliza por instanciar e injetar nos componentes as suas dependências que foram declaradas.

5.4.3. Observer

Define uma dependência um-para-muitos entre objetos, de modo que, quando um objeto muda de estado, todos os seus dependentes são automaticamente notificados e atualizados. [8]

O *observer* foi utilizado nesse sistema para a comunicação entre a *thread* de execução e a *thread* de monitoramento. Dessa forma, quando a execução de um processo é terminado, a *thread* de execução envia uma notificação a *thread* de monitoramento para que armazene no banco de dados as informações adquiridas.

5.5. FRAMEWORKS

Cada vez mais os *frameworks* estão ficando mais importantes e populares nos projetos. Isso se deve pela facilidade de reutilização de funcionalidades.

> “Um framework é um conjunto de classes cooperantes que constroem um projeto reutilizável para uma determinada categoria de software.” [8, pág. 41]

Para a construção deste software, foram utilizados os *frameworks* a seguir.

5.5.1. Spring

O Spring é um *framework* para a linguagem Java e é baseado em Injeção de Dependências e Inversão de Controles. Em projetos orientados a objetos, o Spring reduz o acoplamento entre as classes. Ele possui um *container* que é responsável por instanciar as classes e injetar as dependências de cada classe que foi configurada em um arquivo XML ou através de anotações. [12]

5.5.2. Quartz

O *Quartz Scheduler* é um *framework* para agendamento de tarefas, através dele é possível fazer o controle de quando o processo entrará em execução.

Utilizando uma *cron expression*[14] é possível definir o momento específico ou repetição de momentos em que um processo entrará em execução.

5.5.3. Sigar

Sigar é uma API de acesso às informações do sistema independentemente da plataforma. Ela dá suporte à Windows, Linux, Solaris, MacOSX, FreeBSD, AIX e HP-UX. Através dela é possível recuperar as seguintes informações: [14]

- Memória do sistema, cpu, *swap*, *logins*;
- Variáveis de ambiente, estado do servidor, arquivos abertos;
- Interfaces de rede, informações de configuração.

Para o presente trabalho, a utilização dessa API foi fundamental, uma vez que através das classes nativas do Java só é possível obter as informações da JVM.

5.5.4. Hibernate

> "Historically, Hibernate facilitated the storage and retrieval of Java domain objects via Object/Relational Mapping. Today, Hibernate is a collection of related projects enabling developers to utilize POJO-style[15] domain models in their applications in ways extending well beyond Object/Relational Mapping."[16] [9]

Dentro do contexto deste projeto, o *framework Hibernate*, foi de extrema importância visto que havia a necessidade de se fazer o mapeamento entre o modelo relacional do banco de dados e o modelo orientado a objetos na parte da aplicação.

---

[14] expressão textual que define informações temporais que serão utilizadas para definir um agendamento.

[15] *Plain Old Java Object*: classe Java em sua construção mais simples (Atributos, métodos de acesso e construtores.)

[16] Historicamente, *Hibernate* facilitou o armazenamento e recuperação de objetos de domínio Java via mapeamento objeto-relacional. Hoje, o *Hibernate* é uma coleção de projetos relacionados, permitindo que os desenvolvedores utilizem modelos POJO estilo de domínio em suas aplicações em formas que se estendem bem além de mapeamento objeto-relacional.

Através do *Hibernate*, também foi possível manter um controle transacional mais seguro, visto que este *framework* auxilia no processo de persistência das informações, garantindo assim uma maior segurança em relação a integridade das informações.

5.5.5. JSF

> "*Java Server Faces* é o *framework* de aplicações *Web Java* oficial da *Sun Microsystems*[17] Esse *Framework* foi desenhado para simplificar o desenvolvimento de aplicações *Web*, através do conceito baseado em componentes. Na prática, a utilização de JSF torna fácil o desenvolvimento através de componentes de interface de usuário, possibilitando a conexão desses componentes a objetos de negócio de forma simplificada." [10]

Dentro do projeto o JSF foi responsável pelo controle de navegação das páginas da área administrativa, Através do JSF foi possível fazer controle de sessão web, controlar o fluxo de navegação do projeto, auxiliar no andamento correto do ciclo de vida da aplicação e facilitar a interação do usuário com a aplicação.

5.5.6. Primefaces

O *Primefaces* é uma biblioteca que deve ser adicionada ao JSF para auxílio da parte visual de aplicações *web*. Através do *Primefaces*, é possível fazer com que formulários fiquem mais completos facilitando, por exemplo, a criação de validações. Outra funcionalidade do *Primefaces* é o recurso de gráficos. Através dessa funcionalidade foi possível criar a parte de estatísticas de processamentos com mais facilidade.

[17] Devido à compra da *Sun Microsystems* hoje em dia quem dá suporte ao *framework* é a *Oracle*

6. METODOLOGIA

6.1. METODOLOGIA DE PESQUISA

A metodologia de pesquisa utilizada neste trabalho foi a pesquisa aplicada. Foi utilizada esta metodologia, pois o objetivo deste trabalho é apresentar uma solução para o problema de falta de recursos de *hardware* durante a execução de cargas de trabalho. Além de defender a hipótese de que o RBC pode resolver esse tipo de problema.

Para levantar dados para essa pesquisa, foi realizada entrevistas com profissionais de três empresas diferentes (Calandra Soluções, *Target Solutions* e CETIP). Nessas entrevistas foram verificadas as suas rotinas automatizadas, como elas eram disparadas, a quantidade de rotinas em execução ao mesmo tempo, como o servidor se comportava durante as execuções, como os erros eram tratados e como essas rotinas eram gerenciadas.

Além das entrevistas realizadas nas empresas acima, também foram utilizados livros e materiais *online* sobre Inteligência Artificial e Sistemas Operacionais. Estes livros estão devidamente mencionados nas Referências.

6.2. METODOLOGIA DE GERENCIAMENTO

Para o gerenciamento da equipe de desenvolvimento do *software*, foram utilizadas algumas técnicas do paradigma de desenvolvimento Ágil, como por exemplo: Criação de *sprints* de desenvolvimento[18] e controle de tarefas executadas através de *cards*. Com essa metodologia tornou a equipe auto-gerenciável, criando pequenas entregas mas que agregavam valor ao sistema.

6.2.1. Metodologia de versionamento

Para o controle das versões do projeto será utilizada uma ferramenta de versionamento chamada de *Subversion* (SVN). Através do SVN é possível gerenciar o que já foi feito através do histórico que é armazenado quando um código é enviado para o servidor de gerenciamento.

[18] Tarefa ou fração a serem entregues ao cliente.

Foi utilizada uma ferramenta *online* chamada Assembla que provê um servidor SVN gratuito. Além do versionamento, essa ferramenta disponibiliza um módulo de controle de atividades para auxiliar no gerenciamento.

6.3. METODOLOGIA DE MODELAGEM

Para a parte de modelagem do sistema, foram utilizadas algumas ferramentas da UML como: Casos de Uso, Diagrama de Caso de Uso, Diagrama de Classes e Diagrama de Seqüência. Assim, se obterá uma visão gráfica e textual de como é o sistema como um todo.

Alguns outros diagramas, como por exemplo, o Diagrama Arquitetural também foram utilizados para expor a maneira como o projeto foi idealizado e projetado.

Como auxilio na modelagem do sistema, foram utilizados alguns livros, tais como: BOOCH, Grady; RUMBAUCH, James; JACOBSON, Ivar, 2005; EVANS, Eric, 2009; e ARMAN, Craig, 2007.

6.4. METODOLOGIA DE DESENVOLVIMENTO

Durante o processo de desenvolvimento do projeto foi adotado pela equipe alguns paradigmas ágil, dessa maneira a evolução do projeto foi iterativa e incremental, fazendo com que sua modelagem fosse refinada de acordo com o desenvolvimento.

A linguagem de desenvolvimento utilizada foi Java na sua versão 6. A escolha do Java se deu por permitir a execução do sistema tanto em ambientes Windows como em ambientes Unix, que são os principais ambientes de servidores. Além disso, o Java possui uma quantidade muito grande de frameworks que facilitam o desenvolvimento em geral.

“Padrões de projetos de softwares permitem descrever fragmentos de projeto e reusar idéias de projeto, ajudando desenvolvedores a se nivelar com a experiência de outros” (LARMAN, 2007). Visto isso, um dos cuidados tomados pela equipe de desenvolvimento foi implementar o projeto de forma que seu acoplamento seja baixo, facilitando assim a criação de novos módulos no futuro,

além de aumentar a possibilidade de reuso ,em outros projetos, dos módulos construídos.

Outros frameworks também foram utilizados no desenvolvimento deste sistema. O Spring Framework foi usado para injeção de dependências. O Maven foi utilizado para o controle do ciclo de vida do projeto e das dependências de bibliotecas externas. O Quartz Framework foi utilizado para o agendamento de execução das tarefas. Como servidor de aplicação foi utilizado o Tomcat 6.

O banco de dados utilizado foi o PostgresSql por ser gratuito e suportar o volume de dados utilizado no sistema. Além disso, ele é leve e não necessita de especialistas para dar manutenção. E para a persistência de dados foi utilizado o framework Hibernate. Que auxilia no acesso aos dados e que facilita a troca de banco de dados, caso seja necessário.

O desenvolvimento do sistema foi dividido em duas partes. Na primeira fase do desenvolvimento foi implementado o núcleo do sistema, onde se encontra a inteligência responsável pela execução das atividades de acordo com o estado atual da máquina. Já na segunda fase foi implementado o cadastro de cargas de trabalho, o agendamento das atividades e os gráficos.

6.5. METODOLOGIA DE TESTES

Paralelo ao processo de desenvolvimento, alguns tipos de teste também serão implementados, garantindo uma maior qualidade do resultado final esperado.

Para testes mais específicos de código, serão feitos Testes Unitários, onde cada método das classes de negócio será testado. Estes testes serão feitos através do *framework* JUnit.

Para testes de interface, serão feitos Testes Funcionais, onde será simulada a interação do usuário com a aplicação. Estes testes serão feitos através da integração do *framework Selenium* com o JUnit.

7. CONCLUSÃO

O sistema projetado fundamentou-se na necessidade da automatização de processos computacionais que otimizasse o tempo de execução dos mesmos. A partir da observação dos principais problemas encontrados por corporações que utilizam esse tipo de recurso, foi desenvolvida uma alternativa que funcionasse de maneira a solucionar esses problemas.

Ao ser desenvolvido na plataforma Java, garantiu-se a interoperabilidade entre plataformas, ou seja, a possibilidade de sua execução nos principais Sistemas Operacionais utilizados em servidores (Windows e Linux).

O controle remoto dos recursos disponíveis no servidor e dos processos através da web garante aos administradores do sistema uma maior flexibilidade, pois poderão acessar e gerenciar as atividades de qualquer lugar.

A interface gráfica foi idealizada de forma a ser amigável, ou seja, facilmente operável, além de possibilitar a visualização de estatísticas referentes à execução das tarefas. Este recurso busca facilitar o entendimento detalhado do desempenho computacional e proporcionar sua gerência remota.

Além disso, através da utilização dos padrões de projetos mencionados e de boas práticas de programação, garantiu-se que o sistema não fosse um peso para o servidor, necessitando apenas 150MB de memória. Essa foi uma grande preocupação durante o desenvolvimento e atingiu um resultado satisfatório.

Através da utilização dos conceitos de Inteligência Artificial, também foi possível fazer com que o sistema alimentasse uma base histórica e a cada nova execução as análises ficaram mais refinadas e as estimativas mais precisas. Com isso, a utilização dos recursos do servidor foram utilizados com mais inteligência, garantindo que tanto a memória quanto o processador fossem aproveitados ao máximo sem interferir na execução das tarefas.

Foram realizados diversos tipos de testes: grande quantidade de processos, processos com muitas dependências, processos que consumiam muita memória, processos que consumiam muito processador, diversas tarefas em execução ao mesmo tempo. E durante esses testes ficou comprovado a teoria de que os

processos consomem em média a mesma quantidade de recursos durante as suas execuções. Observou-se também que em todos os testes com muitos processos agendados, a utilização do processador manteve-se em torno de 90% e o consumo de memória quase em 100% e sem nenhuma falha, interrupção ou travamento por falta de recursos de *hardware*. Pode-se concluir com isso que o processador e a memória do servidor foi utilizado da melhor forma possível pelo sistema fazendo com que o maior número de processos fossem executados no menor tempo possível.

Por fim, conclui-se que a redução das falhas em execuções de processos computacionais, estabelecida pelo sistema desenvolvido, bem como o controle automatizado de recursos de *hardware* e *software*, possibilita às organizações uma maior flexibilidade no controle dos seus procedimentos. Visto que todas as soluções apresentadas são viáveis e comprovadamente eficazes pôde-se verificar que a proposta apresentada conseguiu gerar flexibilidade no controle de procedimentos, suprindo assim a demanda por um *software* único de gerenciamento.

8. REFERÊNCIAS

[1] BOOCH, Grady; RUMBAUCH, James; JACOBSON, Ivar. **UML: Guia do Usuário**. 2ª ed. Rio de Janeiro: Elsevier, 2005.

[2] DEITEL, Paul; DEITEL, Harvey. **Java™: Como Programar**. 8ª ed. São Paulo: Pearson Prentice Hall, 2010.

[3] EVANS, Eric. **Domain-Driven Design: Atacando as Complexidades no Coração do Software**. 2ª ed. Rio de Janeiro: Alta Books, 2009.

[4] LARMAN, Craig. **Utilizando UML e Padrões**. 3ª ed. Porto Alegre: Bookmark, 2007.

[5] RICH, Elaine; KNIGHT, Kevin. **Inteligência Artificial**, 1994.

[6] RIESBECK, C. K.; SCHANK, R. C. **Inside Case-Based Reasoning**. 1989. Lawrence Erlbaum Associates, Cambridge, MA.

[7] TANENBAUM, Andrew S. **Distributed Systems: Principles and Paradigms**, 2002.

[8] GAMMA, Erich; HELM, Richard; JOHNSON, Ralph; VLISSIDES. **Padrões de projeto: soluções reutilizáveis de software orientado a objetos** - Porto Alegre: Bookman, 2000.

[9] Hibernate - JBoss Community. **Relational Persistence for Java and .Net.** <http://www.hibernate.org/>. Acessado em: 23/11/2012.

[10] GONÇALVES, Edson. **Dominando Java Server Faces utilizando Spring 2.5, Hibernate e JPA** - Rio de Janeiro: Ciência Moderna, 2008.

[12] Spring Framework. **Spring Framework | SpringSource.org** <http://www.springsource.org/spring-framework/>. Acessado em: 23/11/2012.

[13] Quartz Scheduler. **Quartz Scheduler | Overview** <http://quartz-scheduler.org/overview/>. Acessado em: 23/11/2012.

[14] SIGAR API. **SIGAR API (System Information Gatherer and Reporter) | Hyperic** <http://www.hyperic.com/products/sigar/>. Acessado em: 23/11/2012.

9. BIBLIOGRAFIA COMPLEMENTAR

[1] BLOCH, Joshua. **Java Efetivo.** 2ª ed. Rio de Janeiro: Alta Books, 2008.

[2] MARTIN, Robert C. **Código Limpo: Habilidades Práticas do Agile Software**. 1ª ed. Rio de Janeiro: Alta Books, 2011.

[3] NEMETH, Evi; SNYDER, Garth; HEIN, Trent R. **Manual Completo do Linux: Guia do Administrador**. 2ª ed. São Paulo: Pearson Prentice Hall, 2007.

[4] NEVES, Julio Cezar. **Programação Shell Linux**. 7ª ed. São Paulo: Brasport, 2008

[5] Núcleo de Computação Eletrônica. **Visão Geral Sobre Inteligência Artificial**. <http://www.nce.ufrj.br/GINAPE/VIDA/ia.htm>. Acessado em: 10/04/2012.

[6] PREISS, Bruno R. **Estruturas de dados e algoritmos: Padrões de Projeto Orientados a Objetos com Java™**. 6ª ed. Rio de Janeiro: Elsevier, 2000.

[7] PUGA, Sandra; RISSETTI, Gerson. **Lógica de Programação e Estruturas de Dados**. 2ª Ed. São Paulo: Pearson Prentice Hall, 2008.

[8] ROBBINS, Arnold; BEEBE, Nelson H. F. **Classic Shell Scripting**. 1ª ed. USA: O'Reilly, 2005

[9] SAMPAIO, Cleuton. **Java Enterprise Edition 6: Desenvolvendo Aplicações Corporativas**. 1ª ed. São Paulo: Brasport, 2011.

[10] SIMÕES, Anabela; COSTA, Ernesto. **Inteligência Artificial: Fundamentos e Aplicações**. 2ª ed. São Paulo: FCA, 2008

[11] SZWARCFITER, Jayme Luiz. **Grafos e Algoritmos Computacionais**. 2ª ed. Rio de Janeiro: Campus LTDA, 1983.

[12] WALLS, Craig; BREIDENBACH, Ryan. **Spring Em Ação**. 2ª Ed. Rio de Janeiro: Alta Books, 2008.

[13] WAZLAWICK, Raul Sidnei. **Metodologia de Pesquisa para Ciência da Computação**. 1ª ed. Rio de Janeiro: Elsevier, 2009.

10. ANEXOS

10.1. DOCUMENTO DE VISÃO

Grande parte das empresas utiliza diferentes tipos de software para gerenciar atividades. Para que essas rotinas funcionem perfeitamente, é necessária a existência de uma equipe de TI altamente capacitada. Mas, como a execução das rotinas é realizada manualmente, fica suscetível a erros. Além disso, quando possuímos vários processos, com dependências entre si e cada um com necessidades de hardware diferentes. Juntando isso com processos rodando em paralelo, muitas atividades falham por falta de recursos de hardware. Visto isso foi identificada a oportunidade de criar um sistema que gerenciasse de maneira automatizada essa demanda.

O sistema tem como objetivo gerenciar os processos automatizados de uma organização, agendando e executando essas cargas de trabalho de acordo com a disponibilidade dos recursos de hardware do servidor.

O usuário responsável pelas atividades deverá se autenticar no sistema. Ele poderá visualizar todos os processos e tarefas que estão sob sua responsabilidade, além de poder criar novos e editar ou excluir os existentes.

Cada atividade de um processo pode ser dependente de outras atividades, portanto ela só poderá ser executada se as suas dependências forem concluídas com sucesso.

O usuário poderá agendar a execução dos seus processos e também poderá executá-las imediatamente.

Antes de cada execução de um processo, o sistema irá verificar se é possível executá-la naquele momento, fazendo uma relação entre o histórico de execuções e os recursos de hardware disponíveis no momento.

10.2. LISTA DE REQUISITOS

- Gerenciar Processos;
- Rodar em sistemas operacionais baseados em Unix;
- Controlar a execução dos processos remotamente via web;
- Ser desenvolvido na linguagem Java;
- Conectar-se remotamente com outras máquinas da rede;
- Controle de acesso.

10.3. LISTA DE REQUISITOS DETALHADA

10.3.1. Requisitos Funcionais

- Gerenciar Processos;
    - Cálcular préviamente o consumo dos recursos de hardware;
    - Executar os processos;
    - Armazenar estatísticas de execuções dos processos;
    - Parar os processos;
    - Cadastrar de processos;
    - Agendar Execução;
    - Definir prioridade entre as execuções dos processos;
- Rodar em sistemas operacionais baseados em Unix;
    - Executar nativamente em distribuições Linux;
- Controle remoto via web;
    - Apresentar graficamente estatísticas dos processos ativos;
    - Listar processos em execução;
    - Disparar uma execução remotamente.

    - Parar uma execução remotamente.
- Controle de acesso
    - Cadastro de usuários;
    - Login;

10.3.2. Requisitos Não Funcionais

- Ser desenvolvido na linguagem Java;
- Conectar-se remotamente com outras máquinas da rede.

10.4. LISTA DE CASOS DE USO

- CAU001: Manter Usuário;
- CAU002: Autenticar Usuário;
- CAU003: Manter Processos;
- CAU004: Manter Tarefas;
- CAU005: Manter Dependências;
- CAU006: Executar Processos;
- CAU007: Parar Processos;
- CAU008: Executar Tarefa;
- CAU009: Listar Processos Ativos.

10.5. DIAGRAMA DE CASOS DE USO

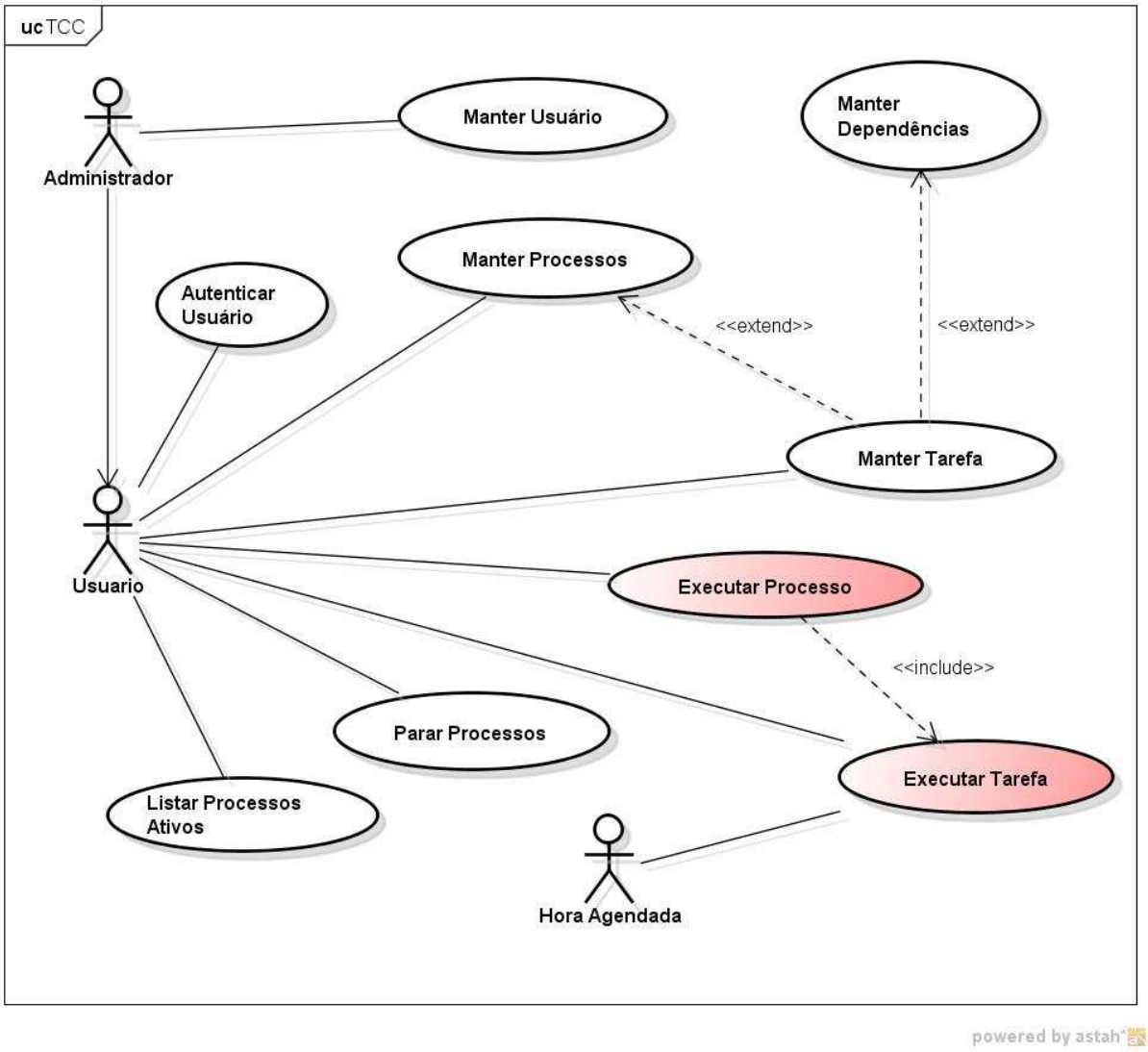

Figura 4: Diagrama de Casos de Uso

10.6. DESCRIÇÃO DE CASOS DE USO

10.6.1. Manter Usuário

Descrição	Este caso de uso visa descrever o processo de gerenciamento dos usuários do sistema.
Atores Envolvidos	Administrador.
Pré-condições	Efetuar a autenticação no sistema com o perfil de administrador.
Pós-condições	Manter atualizado o cadastro de usuários.

Cadastrar			
Ator		**Sistema**	
1	Seleciona a opção cadastrar novo usuário.		
		2	Exibe o formulário para o preenchimento das informações: (CPF, Nome, Tipo de Perfil, Login e Senha).
3	Fornece as informações solicitadas.		
		4	Valida as informações. **[E01 – Campos obrigatórios não preenchidos]** **[E02 – Usuário já cadastrado]**
		5	Salva no banco de dados.
Sequências de Exceção			
E01 – Campos obrigatórios não preenchidos			
		1	Exibe mensagem informando quais campos obrigatórios não foram preenchidos.
		2	Retorna ao passo 2.
E02 – Usuário já cadastrado			
		1	Exibe mensagem alertando que o CPF informado já está cadastrado em outro usuário.
		2	Retorna ao passo 2.
Sequências Alternativas			

Editar			
Ator		**Sistema**	
1	Após selecionar o usuário a ser alterado, seleciona a opção “editar usuário”.		
		2	Exibe o formulário preenchido com as informações do usuário selecionado para as devidas alterações.
3	Altera as informações.		
		4	Valida as informações. **[E03 – Campos obrigatórios não preenchidos]**
		5	Atualiza o usuário no banco de dados.
Sequências de Exceção			
E03 – Campos obrigatórios não preenchidos			
		1	Exibe mensagem informando quais campos obrigatórios não foram preenchidos.
		2	Retorna ao passo 2.
Sequências Alternativas			

Excluir			
Ator		**Sistema**	
1	Após selecionar o usuário a ser excluído, seleciona a opção “excluir usuário”.		
		2	Solicita uma confirmação para exclusão do usuário.
3	Confirma a exclusão. **[A041 – O usuário cancela a exclusão]**		
		4	Exclui o usuário do banco de dados.
		5	Exibe uma mensagem confirmando a exclusão do usuário.
Sequências de Exceção			
Sequências Alternativas			
A01 - O usuário cancela a exclusão			
		1	Exibe uma mensagem informando que nenhum usuário foi excluído.

Pesquisar			
Ator		**Sistema**	
1	Seleciona a opção pesquisar usuários.		
		2	Exibe o formulário de busca com uma caixa de texto.
3	Preenche o campo.		
		4	Busca os usuários no banco de dados com o filtro no informado nos campos: CPF, Nome e Login. **[E04 – Nenhum Usuário Encontrado]** **[A02 – O usuário não preenche o campo de busca]**
		5	Exibe a listagem de usuários.
Sequências de Exceção			
E04 – Nenhum Usuário Encontrado			
		1	Exibe uma mensagem informando que nenhum usuário foi encontrado.
Sequências Alternativas			
A02 – O usuário não preenche o campo de busca			
		1	Busca todos os usuários do sistema.
		2	Retorna ao passo 5.

10.6.2. Autenticar Usuário

Descrição	Este caso de uso visa descrever o processo de autenticação no sistema.
Atores Envolvidos	Usuário.
Pré-condições	Para efetuar a autenticação, o usuário deverá ter sido cadastrado previamente no sistema.
Pós-condições	Obtém acesso ao sistema.

Autenticar			
Ator		**Sistema**	
1	O Usuário informa o login e a senha.		
		2	Verifica se os campos foram preenchidos. **[E01 – Campos não preenchidos]**
		3	Busca no banco de dados o usuário através do login e da senha que foram informados. **[E02 – Usuário não encontrado]**
		4	Guarda as informações do usuário autenticado na sessão.
Sequências de Exceção			
E01 – Campos não preenchidos			
		1	Exibe mensagem informando que os campos são obrigatórios.
		2	Retorna ao passo 1 do fluxo principal.
E02 – Usuário não encontrado			
		1	Exibe mensagem login/senha inválidos.
		2	Retorna ao passo 1 do fluxo principal.
Sequências Alternativas			

10.6.3. Manter Processos

Descrição	Este caso de uso visa descrever o processo de gerenciamento dos processos do sistema.
Atores Envolvidos	Usuário.
Pré-condições	N/A
Pós-condições	Manter atualizado o cadastro de processos.

Cadastrar			
Ator		**Sistema**	
1	Seleciona a opção “cadastrar novo processo”.		
		2	Exibe o formulário para o preenchimento das informações do processo.
3	Fornece as informações solicitadas.		
		4	Valida as informações. **[E01 – Campos obrigatórios não preenchidos]**
		5	Salva no banco de dados.
Sequências de Exceção			
E01 – Campos obrigatórios não preenchidos			
		1	Exibe mensagem informando quais campos obrigatórios não foram preenchidos.
		2	Retorna ao passo 2.
Sequências Alternativas			

Editar			
Ator		**Sistema**	
1	Após selecionar o processo a ser alterado, seleciona a opção “editar processo”.		
		2	Exibe o formulário preenchido com as informações do processo selecionado para as devidas alterações.
3	Altera as informações.		
		4	Valida as informações. **[E02 – Campos obrigatórios não preenchidos]**
		5	Atualiza o processo no banco de dados.
Sequências de Exceção			
E02 – Campos obrigatórios não preenchidos			
		1	Exibe mensagem informando quais campos obrigatórios não foram preenchidos.
		2	Retorna ao passo 2.
Sequências Alternativas			

Excluir			
Ator		**Sistema**	
1	Após selecionar o processo a ser excluído, seleciona a opção “excluir processo”.		
		2	Verifica se o processo pode ser excluído [**RN001]**. **[E03 – O processo não pode ser excluído]**
		3	Solicita uma confirmação para exclusão do processo.
4	Confirma a exclusão. **[A01 – O usuário cancela a exclusão]**		
		5	Exclui o processo do banco de dados.
		6	Exibe uma mensagem confirmando a exclusão do processo.
Sequências de Exceção			
E03 – O processo não pode ser excluído			
		1	Exibe uma mensagem informando o motivo pelo qual não permite o processo ser excluído.
Sequências Alternativas			
A01 – O usuário cancela a exclusão			
		1	Exibe uma mensagem informando que nenhum processo foi excluído.

Pesquisar			
Ator		**Sistema**	
1	Seleciona a opção pesquisar processos.		
		2	Exibe o formulário de busca com uma caixa de texto.
3	Preenche o campo.		
		4	Busca os processos no banco de dados com o filtro informado em todos os campos. **[E04 – Nenhum processo encontrado]** **[A02 – O usuário não preenche o campo de busca]**
		5	Exibe a listagem de processos.
Sequências de Exceção			
E04 – Nenhum processo encontrado			
		1	Exibe uma mensagem informando que nenhum processo foi encontrado.
Sequências Alternativas			
A02 – O usuário não preenche o campo de busca			
		1	Busca todos os processos do sistema.
		2	Retorna ao passo 5.

Regras de Negócio	
RN001	Um processo só pode ser excluído se ele não estiver em execução e se não estiver agendado.

10.6.4. Manter Tarefas

Descrição	Este caso de uso visa descrever o processo de gerenciamento de tarefas.
Atores Envolvidos	Usuário.
Pré-condições	N/A
Pós-condições	Manter atualizado o cadastro de tarefas e seus agendamentos de execuções.

Cadastrar			
Ator		**Sistema**	
1	Seleciona a opção “nova tarefa”.		
		2	Exibe o formulário para o preenchimento das informações da tarefa e um formulário para busca dos processos cadastrados.
3	Fornece as informações solicitadas e seleciona os processos envolvidos naquela tarefa. **[A01 – Cadastrar novo processo]**		
4	Seleciona a opção “salvar”.		
		5	Valida as informações. **[E01 – Campos obrigatórios não preenchidos]**
		6	Salva no banco de dados.
		7	Exibe um fluxograma ilustrando a execução dos processos.
8	**[A02 – Definir dependências entre os processos]**		
Sequências de Exceção			
E01 – Campos obrigatórios não preenchidos			
		1	Exibe mensagem informando quais campos obrigatórios não foram preenchidos.
		2	Retorna ao passo 3.
Sequências Alternativas			
A01 – Cadastrar novo processo			
1	Seleciona a opção de cadastrar novo processo.		
		2	**[*Extends* – Manter Processos –**

			Cadastrar]
		3	Retorna ao passo 3.
A02 – Definir dependências entre os processos			
1	Seleciona a opção para definir as dependências entre os processos.		
		2	Exibe formulário com todos os processos selecionados para aquela tarefa.
3	Seleciona o processo desejado e dispara o comando para visualizar as dependências.		
		4	**[*Extends* – Manter Dependências - Visualizar]**

Editar			
Ator		**Sistema**	
1	Após selecionar a execução a ser alterada, seleciona a opção “editar tarefa”.		
		2	Exibe o formulário preenchido com as informações da tarefa selecionada para que o usuário possa alterá-la e exibe um formulário para busca dos processos cadastrados.
3	Fornece as informações solicitadas e seleciona os processos envolvidos naquela tarefa. **[A01 – Cadastrar novo processo]**		
5	Seleciona a opção “salvar”		
		5	Valida as informações. **[E02 – Campos obrigatórios não preenchidos]**
		6	Atualiza as informações no banco de dados.
		7	Exibe um fluxograma ilustrando a execução dos processos.
8	**[A02 – Definir dependências entre os processos]**		
Sequências de Exceção			
E02 – Campos obrigatórios não preenchidos			
		1	Exibe mensagem informando quais campos obrigatórios não foram preenchidos.
		2	Retorna ao passo 3.
Sequências Alternativas			
A01 – Cadastrar novo processo			

1	Seleciona a opção de cadastrar novo processo.		
		2	**[*Extends* – Manter Processos – Cadastrar]**
A02 – Definir dependências entre os processos			
1	Seleciona a opção para definir as dependências entre os processos.		
		2	Exibe formulário com todos os processos selecionados para aquela tarefa.
3	Seleciona o processo desejado e dispara o comando para visualizar as dependências.		
		4	**[*Extends* – Manter Dependências - Visualizar]**

Excluir			
Ator		**Sistema**	
1	Após selecionar a tarefa a ser excluído, seleciona a opção “excluir tarefa”.		
		2	Verifica se a tarefa pode ser excluída [**RN002]**. **[E01 – A tarefa não pode ser excluída]**
		3	Solicita uma confirmação para exclusão da tarefa.
4	Confirma a exclusão. **[A01 – O usuário cancela a exclusão]**		
		5	Exclui a tarefa do banco de dados.
		6	Exibe uma mensagem confirmando a exclusão da tarefa.
Sequências de Exceção			
E01 – A tarefa não pode ser excluída			
		1	Exibe uma mensagem informando o motivo pelo qual não permite a exclusão da tarefa.
Sequências Alternativas			
A01 – O usuário cancela a exclusão			
		1	Exibe uma mensagem informando que nenhuma tarefa foi excluída.

Pesquisar			
Ator		**Sistema**	
1	Seleciona a opção pesquisar tarefas.		
		2	Exibe uma lista com as tarefas cadastradas e um formulário de busca.
3	Preenche os campos e seleciona a opção “pesquisar”.		
		4	Busca as tarefas no banco de dados com os filtros informados. **[E01 – Nenhuma tarefa encontrada]** **[A01 – O usuário não preenche os campos de busca]**
		5	Exibe a listagem das tarefas encontradas.
Sequências de Exceção			
E01 – Nenhuma tarefa encontrada			
		1	Exibe uma mensagem informando que nenhuma tarefa foi encontrada.
Sequências Alternativas			
A01– O usuário não preenche os campos de busca			
		1	Busca todas as tarefas do sistema.
		2	Retorna ao passo 5.

Regras de Negócio	
RN002	Uma tarefa só poderá ser excluída se ele não estiver em execução.

10.6.5. Manter Dependências

Descrição	Este caso de uso visa descrever o processo de gerenciamento das dependências entre os processos em uma determinada execução.
Atores Envolvidos	Usuário.
Pré-condições	N/A
Pós-condições	Manter atualizada as dependências entre os processos.

Visualizar			
Ator		**Sistema**	
1	Após selecionar o processo , seleciona a opção “definir dependências”.		
		2	Exibe uma lista com os processos que devem ser executados antes do processo selecionado.
Sequências de Exceção			
E01 – Nenhuma dependência selecionada			
		1	Exibe uma mensagem informando que o processo selecionado não possui dependências definidas.
Sequências Alternativas			

Cadastrar			
Ator		**Sistema**	
1	Seleciona a opção “nova dependência”.		
		2	Exibe uma lista com os demais processos cadastrados. **[E01 – Nenhum processo disponível]**
3	Escolhe um processo e seleciona a opção “adicionar”.		
		4	Salva no banco de dados.
Sequências de Exceção			
E01 – Nenhum processo disponível			
		1	Exibe mensagem informando que não existem processos disponíveis.
Sequências Alternativas			

Excluir			
Ator		**Sistema**	
1	Após selecionar a dependência a ser removida, seleciona a opção “excluir”.		
		2	Solicita uma confirmação para exclusão da dependência.
4	Confirma a exclusão. **[A01 – O usuário cancela a exclusão]**		
		5	Exclui a dependência do banco de dados.
		6	Exibe uma mensagem confirmando a exclusão da dependência.
Sequências de Exceção			
Sequências Alternativas			
A01 – O usuário cancela a exclusão			
		1	Exibe uma mensagem informando que nenhuma dependência foi excluída.

10.6.6. Executar Processos

Descrição	Este caso de uso visa descrever o processo de execução de um processo do sistema.
Atores Envolvidos	Usuário / Hora Agendada.
Pré-condições	O processo só poderá ser executado se não houver dependências ainda não concluídas.
Pós-condições	Processo executado e estatísticas de execução do processo armazenadas.

Executar Processo			
Ator		**Sistema**	
1	Após selecionar o processo, envia comando para executar o processo.		
		2	Verifica se é possível executar o processo naquele momento de acordo com os recursos de hardware disponíveis e levando em consideração a análise do histórico de execuções. **[E01 – Não é possível executar o processo no momento]** **[RN003]**
		3	Executa o processo e realiza a leitura de consumo de hardware durante toda a execução.
		4	Armazena as estatísticas da execução do processo.
		5	Atualiza o status do processo na tarefa.
Sequências de Exceção			
E01 – Não é possível executar o processo no momento			
		1	**[RN004]**
		2	Retorna ao passo 2 do fluxo principal.
Sequências Alternativas			

Regras de Negócio	
RN003	TODO: Escrever regra de negócio para definir se um processo pode ou não ser executado.
RN004	Se um processo não puder ser executado no momento por falta de recursos de hardware, o sistema deverá aguardar 30 segundos e então fará uma nova verificação.

10.6.7. Parar Processos

Descrição	Este caso de uso visa descrever o processo para a interrupção processos que estiverem em execução no momento.
Atores Envolvidos	Usuário.
Pré-condições	Existirem processos em execução.
Pós-condições	Interrupção do processo selecionado.

Parar Processo			
Ator		**Sistema**	
1	Após listar os processos ativos e selecionar o processo ao qual se deseja interromper, seleciona a opção “parar processo”		
		2	Solicita confirmação para parar o processo selecionado.
3	Confirma a interrupção do processo. **[A01 – Usuário cancela a interrupção]**		
		4	Interrompe a execução do processo.
		5	Atualiza o status do processo no banco de dados e armazena um registro no log.
Sequências de Exceção			
Sequências Alternativas			
A01 – Usuário cancela a interrupção			
		1	Exibe mensagem informando que nenhum processo foi interrompido.

10.6.8. Executar Tarefa

Descrição	Este caso de uso visa descrever o processo de execução de uma tarefa.
Atores Envolvidos	Usuário / Hora Agendada.
Pré-condições	N/A
Pós-condições	Todas os processos da tarefa executados.

Executar Tarefa			
Ator		**Sistema**	
1	Após selecionar o a tarefa, envia comando para executar a tarefa.		
		2	Busca no banco de dados as informações da tarefa.
		3	Realiza uma análise do histórico de execução de cada um dos processos envolvidos.
		4	Cria uma sequência de execução dos processos levando em consideração os recursos de hardware disponíveis no momento, a análise histórica dos processos e as dependências definidas.
		2	Executa cada um dos processos. **[*Extends* – Executar Processo]**
		5	Atualiza o status da tarefa.
Sequências de Exceção			
Sequências Alternativas			

10.6.9. Listar Processos Ativos

Descrição	Este caso de uso visa descrever o processo para a exibição do processos que estiverem em execução no momento.
Atores Envolvidos	Usuário.
Pré-condições	N/A
Pós-condições	Exibir listagem de processos ativos.

Listar Processos Ativos			
Ator		**Sistema**	
1	Seleciona a opção “listar processos ativos”		
		2	Busca no banco de dados os processos que estiverem em execução. **[E01 – Nenhum processo ativo]**
		3	Exibe listagem de processos.
Sequências de Exceção			
E01 – Nenhum processo ativo			
		1	Exibe mensagem informando que nenhum processo está em execução no momento.
Sequências Alternativas			

10.7. DIAGRAMA DE CLASSES

10.7.1. Pacote Modelo

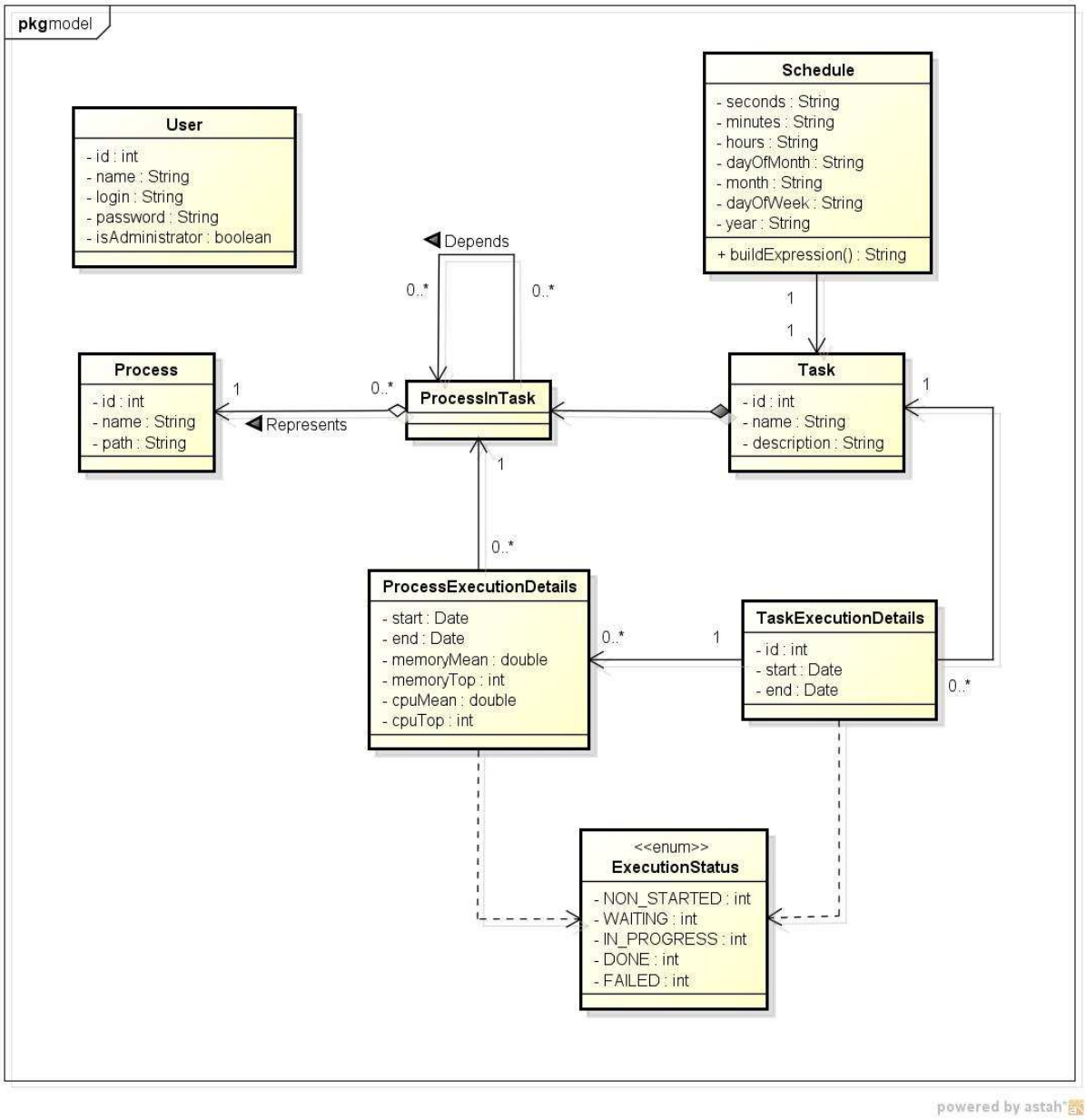

Figura 5: Diagrama de Classes

10.8. DIAGRAMA DE SEQUÊNCIA

10.8.1. Executar Tarefa

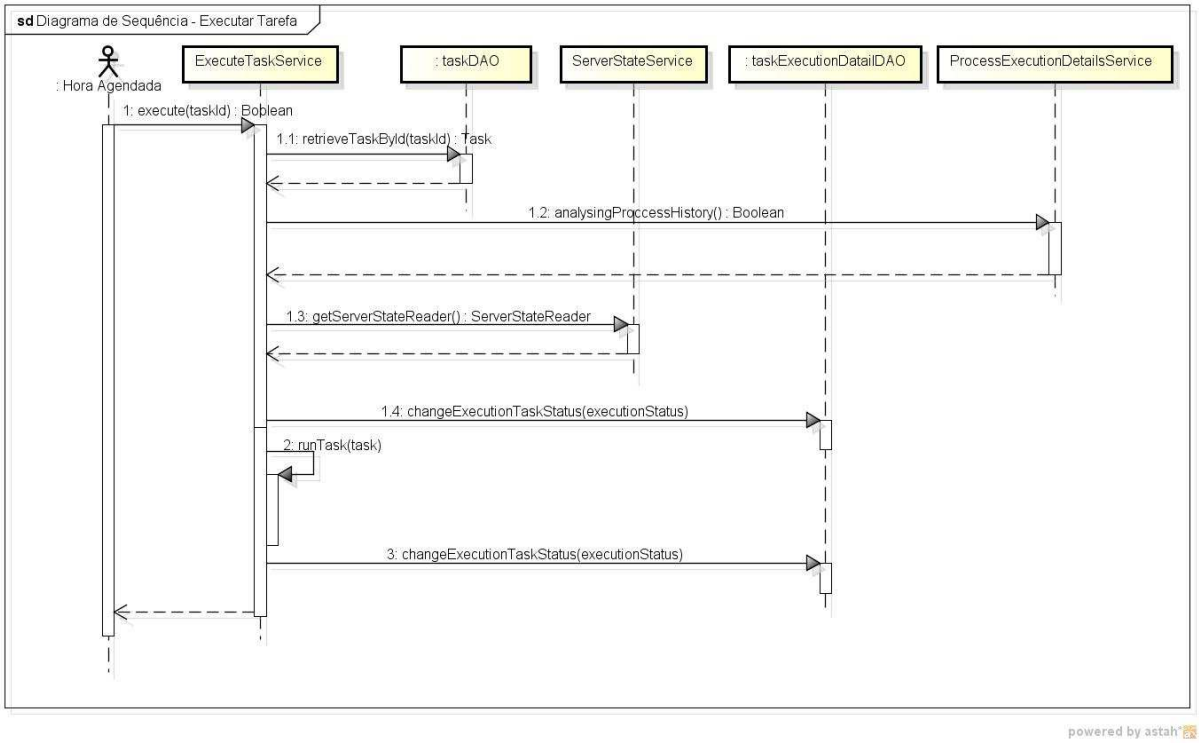

Figura 6: Diagrama de Sequência - Executar Tarefa

10.8.2. Executar Processo

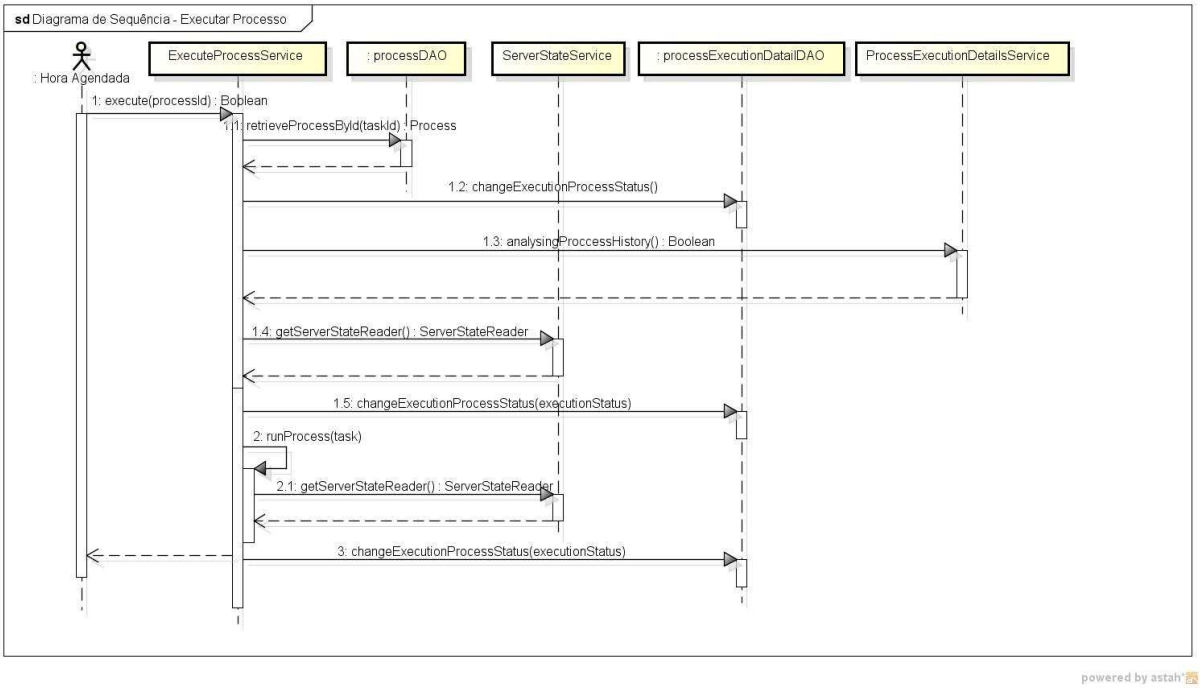

Figura 7: Diagrama de Sequência - Executar Processo

10.9. DIAGRAMA DE ESTADO

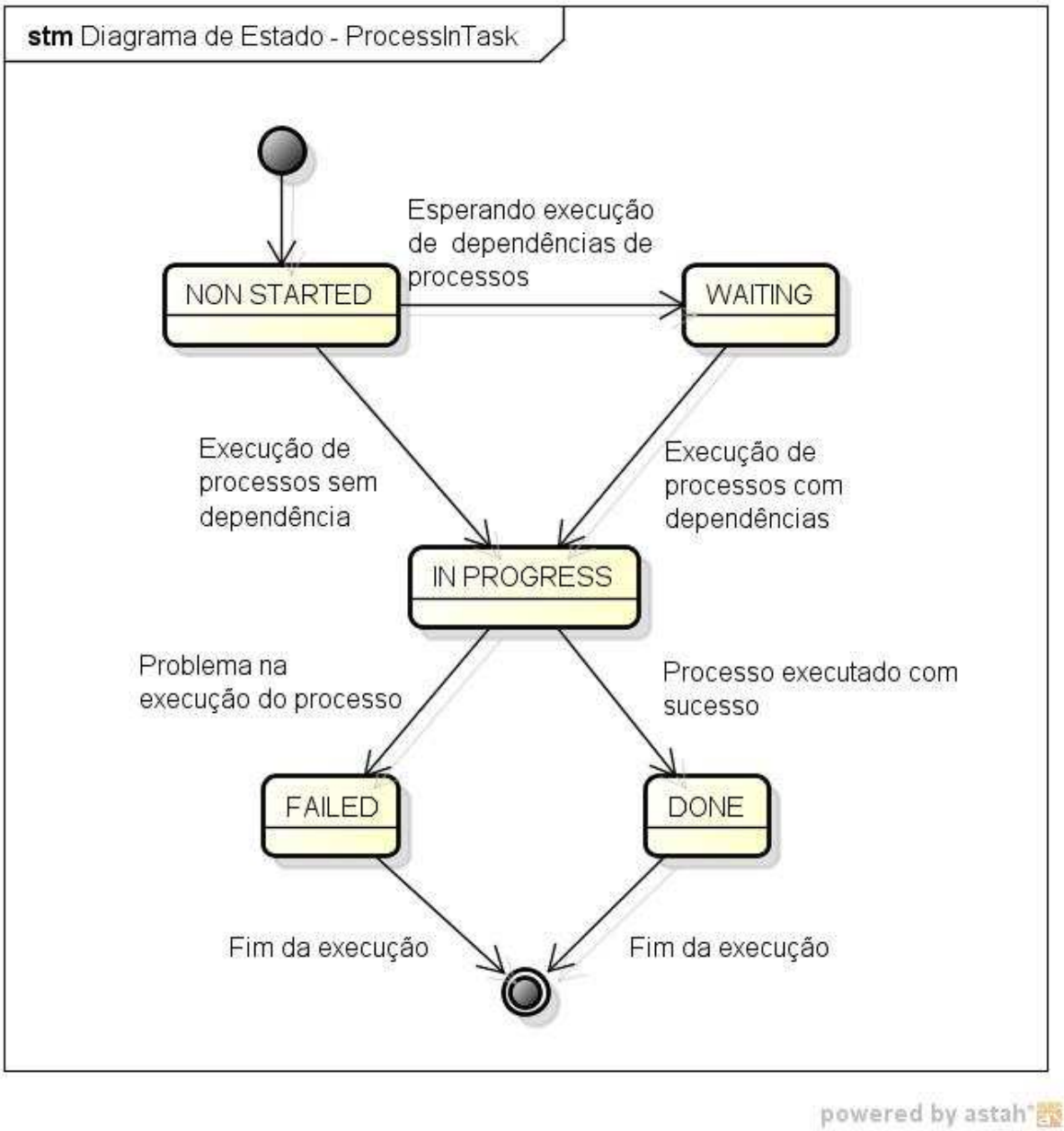

Figura 8: Diagrama de Estado - Process In Task

10.10. DIAGRAMA DE ATIVIDADES

10.10.1. Executar Tarefa

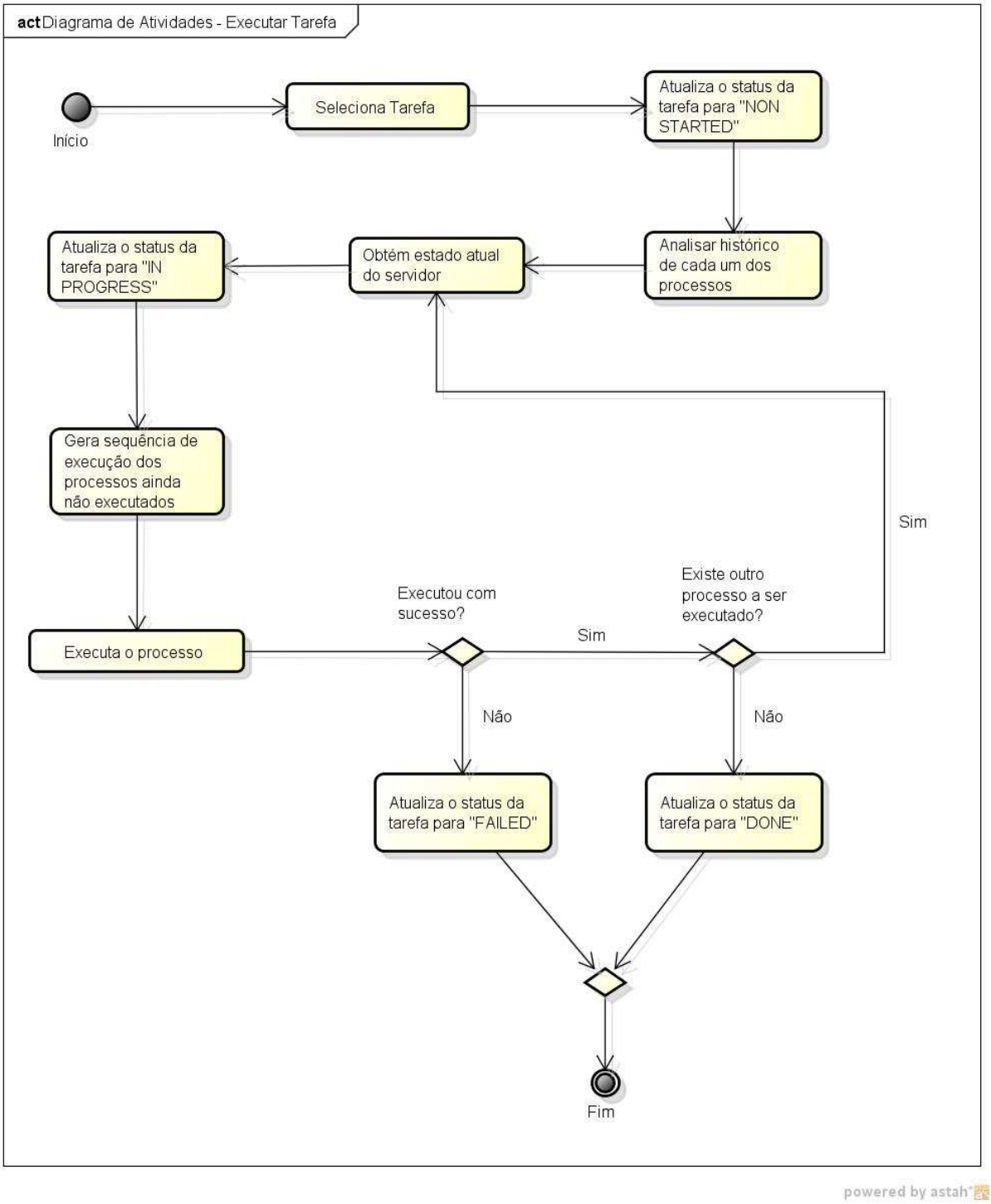

Figura 9: Diagrama de Atividades - Executar Tarefa

10.10.2. Executar Processo

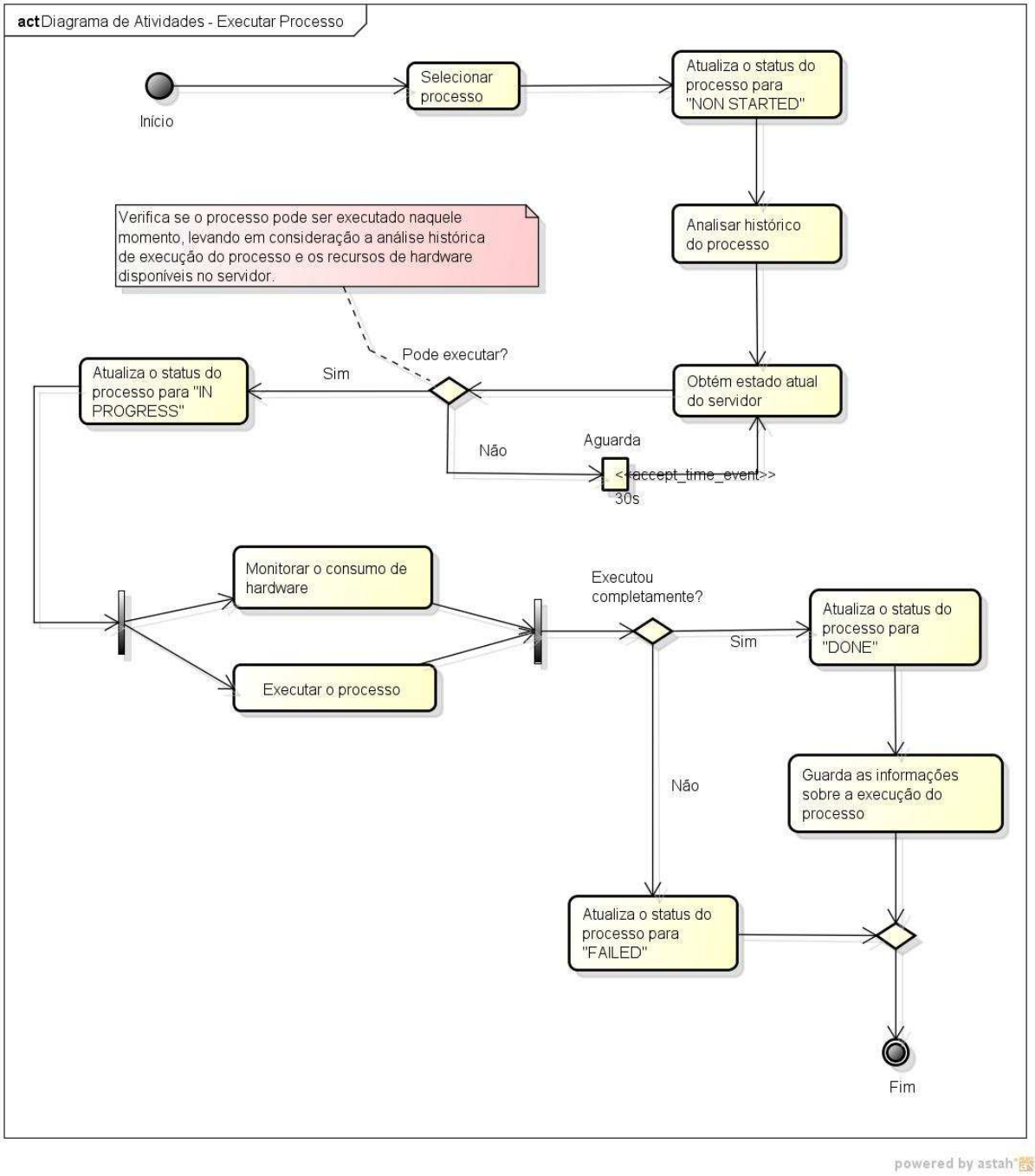

Figura 10: Diagrama de Atividades - Executar Processo

10.11. DIAGRAMA ARQUITETURAL

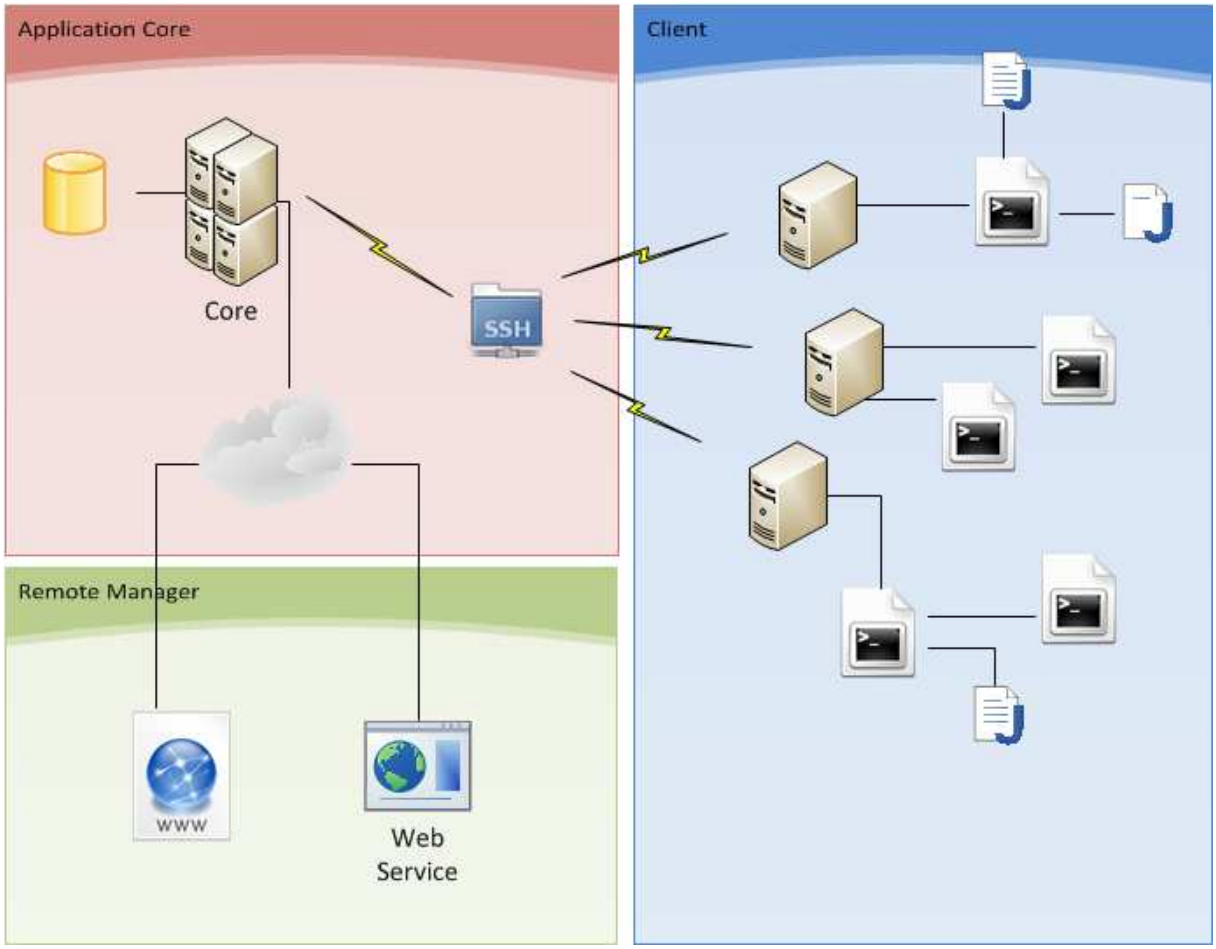

Figura 11: Diagrama Arquitetural