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

Agosto, 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

Agosto, 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

Concepção e implementação de um sistema para gerenciamento de processos automatizados em ambientes distribuídos, responsável pelo agendamento e execução de tarefas computacionais. Através de pesquisa realizada em empresas que possuem processos automatizados, foram observadas falhas provenientes da demanda excessiva de execuções simultâneas no servidor, bem como tempo elevado de execução de processos. Como solução para tais problemas, será desenvolvido um sistema que funcione como um agendador de tarefas, bem como controle os recursos disponíveis no ambiente onde a tarefa será executada. Por meio de técnicas de Inteligência Artificial (IA), o sistema consegue tomar decisões de quando e como executar determinada atividade, sempre levando em conta sua importância dentro de uma sequencia pré-estabelecida de processos e sua demanda por recursos do servidor. Com isso será possível otimizar o processo, diminuindo o tempo de execução e eliminando falhas provenientes dessa demanda.

ABSTRACT

Design and implementation of a system for managing automated processes in distributed environments, responsible for scheduling and running computational tasks. While researching companies that use automated processes, we have observed errors originated from the excessive demand of simultaneous executions on the server, as well as an elevated amount of time spent on the execution of processes. A system that works as a task manager, and also controls the resources available on the environment where the tasks will be held, will be developed as a solution for such problems. Using Artificial Intelligence (AI) techniques, the system can decide about when and how to run certain activities, always taking into consideration their importance within a predetermined sequence of processes and its demand for server resources. Thus, it will be possible to optimize processes by decreasing their runtime and by eliminating errors caused by this demand.

SUMÁRIO

1.**INTRODUÇÃO** 11

2.**JUSTIFICATIVA** 14

3.**OBJETIVOS** 15

3.1.OBJETIVOS GERAIS 15

3.2.OBJETIVOS ESPECÍFICOS 15

4.**HIPÓTESES** 16

5.**PRESSUPOSTOS TEÓRICOS** 17

5.1.INTELIGÊNCIA ARTIFICIAL 17

5.1.1.Raciocínio Baseado em Casos 17

5.2.SISTEMAS DISTRIBUÍDOS 18

5.2.1.Arquitetura Cliente/Servidor 19

5.3.JAVA 19

6.**METODOLOGIA** 21

6.1.METODOLOGIA DE GERENCIAMENTO 21

6.1.1.Metodologia de versionamento 21

6.2.METODOLOGIA DE MODELAGEM 21

6.3.METODOLOGIA DE DESENVOLVIMENTO 21

6.4.METODOLOGIA DE TESTES 22

7.**CONCLUSÃO** 23

8.**REFERÊNCIAS** 24

9.**ANEXOS** 25

9.1.DOCUMENTO DE VISÃO 25

9.2.LISTA DE REQUISITOS 26

9.3.LISTA DE REQUISITOS DETALHADA 26

9.3.1.Requisitos Funcionais 26
9.3.2.Requisitos Não Funcionais 27
9.4.LISTA DE CASOS DE USO 27
9.5.DIAGRAMA DE CASOS DE USO 28
9.6.DESCRIÇÃO DE CASOS DE USO 29
9.6.1.Manter Usuário 29
9.6.2.Autenticar Usuário 34
9.6.3.Manter Processos 36
9.6.4.Manter Tarefas 41
9.6.5.Manter Dependências 48
9.6.6.Executar Processos 52
9.6.7.Parar Processos 54
9.6.8.Executar Tarefa 56
9.6.9.Listar Processos Ativos 58
9.7.DIAGRAMA DE CLASSES 59
9.7.1.Pacote Modelo 59
9.8.DIAGRAMA DE SEQUÊNCIA 60
9.8.1.Executar Tarefa 60
9.8.2.Executar Processo 61
9.9.DIAGRAMA DE ESTADO 62
9.10.DIAGRAMA DE ATIVIDADES 63
9.10.1.Executar Tarefa 63
9.10.2.Executar Processo 64
9.11.DIAGRAMA ARQUITETURAL 65

DICIONÁRIO DE SIGLAS

API: *Application Programming Interface*

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*

SQL: *Structured Query Language*

SVN: Subversion

UML: *Unified Modeling Language*

XML: *Extensible Markup Language*

LISTA DE FIGURAS E IMAGENS

Figura 1: Diagrama de Casos de Uso 28

Figura 2: Diagrama de Classes 59

Figura 3: Diagrama de Sequência - Executar Tarefa 60

Figura 4: Diagrama de Sequência - Executar Processo 61

Figura 5: Diagrama de Estado - Process In Task 62

Figura 6: Diagrama de Atividades - Executar Tarefa 63

Figura 7: Diagrama de Atividades - Executar Processo 64

Figura 8: Diagrama Arquitetural 65

1. INTRODUÇÃO

Hoje em dia, grande parte das empresas funciona com a ajuda de vários *softwares*, como por exemplo, ERP, CRM, CMS, servidor de e-mail, aplicações *web*, bancos de dados, servidor de arquivos, entre outros. Para manter todos esses serviços funcionando perfeitamente é necessária uma equipe de TI capacitada. Isso gera um custo muito alto. E como o trabalho manual é mais suscetível a falhas, muitas vezes alguns desses serviços saem de funcionamento, podendo provocar um prejuízo à organização. Outro problema maior ainda é a inclusão de novos recursos e a integração com outros *softwares*.

Uma possibilidade de gerenciar uma gama tão diversificada e vasta de aplicações e de tarefas é a utilização de pequenas rotinas onde uma sequência de atividades seria previamente estabelecida. Isso é possível através da utilização de arquivos batch[1]. Porém mesmo com a utilização de um arquivo desse tipo, a manutenção ainda é trabalhosa, custosa e dependente de uma equipe altamente treinada, para que alguma chamada não perca sua devida referência de execução.

Pode-se ver anteriormente que a utilização de arquivos batch já facilitou bastante a execução de rotinas, porém em um ambiente onde várias pequenas tarefas são necessárias, a criação de mais arquivos em batch acaba sendo inevitável, o que acaba fazendo com que o primeiro problema identificado volte e faça com que a utilização das rotinas em batch não tenha tanta produtividade.

Visto que, apesar da grande quantidade de arquivos batch, essa ainda é a maneira mais viável de se executar rotinas em sequência, pode se fazer uso de uma ferramenta para agendamento de rotinas. Dessa maneira seria possível definir quando e quais rotinas seriam executadas, ajudando inclusive na organização das rotinas. Porém, ainda há a necessidade de intervenção de um profissional especializado, pois algumas rotinas são mais importantes, ou simplesmente dependem de outras.

[1] Arquivo de scripts para a execução automatizada de comandos interpretados por um Sistema Operacional

Outro problema que pode ser verificado é que durante a execução dessa quantidade considerável de tarefas pode ser gerada uma sobrecarga em seu ambiente de processamento. Levando em conta que os recursos são finitos, se faz necessária uma inteligência que conseguisse analisar algumas variáveis - como prioridade, consumo de recursos e dependência entre elas – para que a execução das rotinas e a integridade do Sistema e do *Hardware* não sejam comprometidas.

Com base nos problemas apresentados anteriormente, conclui-se a necessidade de uma aplicação que conseguisse gerenciar uma quantidade vasta de rotinas previamente agendadas, levando em conta sua importância de execução, sua dependência hierárquica e os recursos disponíveis. Visto isso, pensou-se no desenvolvimento de um *software* para gerenciamento de carga de trabalho que conseguisse agendar, executar, interromper, pausar e criar novas rotinas de uma empresa, além da possibilidade de coletar informações de forma gráfica e bem inteligível, facilitando assim a manutenção e a interação desse tipo de atividade.

Uma das características definidas é a criação de um pacote de execuções em estrutura de Árvore[2], através da qual será possível criar dependências entre os processos, garantindo, dessa forma, que um processo só será iniciado quando o anterior for executado com sucesso. Com a constante execução desse pacote, será possível gravar um histórico das execuções e através de técnicas de IA será possível melhorar gradualmente o desempenho da aplicação.

Um dos grandes diferenciais de outras aplicações existentes no mercado é a criação de estatísticas a partir das execuções anteriores, permitindo o gerenciamento dos processos de acordo com os recursos de *hardware*. Assim, é possível a identificação dos níveis de memória e processamento exigidos/consumidos pala tarefa atual. Caso não existam recursos disponíveis, o sistema age de forma a tornar possível sua execução sem ocasionar problemas decorrentes da falta dos mesmos.

Além dessas grandes funcionalidades, o sistema permite a visualização de forma gráfica dos processos ativos, e fornece uma série de relatórios que

[2] Em Estrutura de dados, Árvore é um grafo que seja acíclico e conexo. (SZWARCFITER, 1983, p. 43).

transformam os números em informações úteis para o administrador, proporcionando a possibilidade de intervenção manual na execução das tarefas.

Como a ideia do sistema é manter a gerência constante e ininterrupta das atividades agendadas, é necessário que a interface esteja disponível de forma remota para que a pessoa responsável possa intervir a qualquer momento. Em resumo, é imprescindível o desenvolvimento de uma versão *web* e possivelmente, de uma versão *mobile*.

2. JUSTIFICATIVA

Nos dias de hoje as empresas precisam cada vez mais da automação de processos, porém muitas vezes gerenciá-los ainda é feito de forma manual, complexa e demorada. Diante desse problema, foi feita uma pesquisa e pôde-se chegar à conclusão que não existem muitas soluções disponíveis no mercado para a resolução desse caso específico, o que gerou uma motivação para o desenvolvimento de um software que atenda a essas especificações.

Através de pesquisa em grandes empresas, verificou-se que já existe uma ferramenta no mercado, mas além de ser uma ferramenta importada (o que proporciona uma dificuldade relativa ao suporte e treinamento), ainda apresenta um custo de aquisição alto. Também foi verificado que a mesma ferramenta não possui 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.

Levando em conta que será preciso trabalhar com informações referentes ao Sistema Operacional e ao *hardware*, foi decidido que será usada a plataforma Java, pois assim o sistema não fica restrito a apenas uma plataforma (Windows ou Unix), podendo proporcionar uma maior flexibilidade para o cliente. Outros fatores motivaram o uso dessa tecnologia, como, por exemplo, a robustez da plataforma, sua forte comunidade de desenvolvedores, sua vasta opção de *frameworks*[3] existentes no mercado e sua facilidade de comunicação com o Sistema Operacional e com o *hardware* por conta da existência da JVM[4].

Visto a necessidade de uma ferramenta com o propósito específico que, além de conter novas funcionalidades, seja de fácil manuseio e que apresente um custo mais acessível, percebeu-se que seria uma oportunidade de construir uma ferramenta nesses moldes.

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

[4] Ambiente onde o *bytecode* Java pode ser executado.

3. OBJETIVOS

3.1. OBJETIVOS GERAIS

O objetivo deste trabalho é desenvolver um software que gerencie os recursos de *hardware* e *software* para a execução de processos agendados hierarquicamente em servidores remotos utilizando técnicas de Inteligência Artificial.

3.2. OBJETIVOS ESPECÍFICOS

- Liberação de processos de acordo com a disponibilidade de recursos na máquina;
- Agendamento de processos levando em conta a data/hora, dependências e prioridade;
- Criar estatísticas referentes à execução de cada processo;
- Utilizar as estatísticas armazenadas para otimizar a execução das tarefas;
- Envio de notificações aos responsáveis pelos processos informando o início, término e possíveis falhas;
- Controle remoto (web/mobile) dos recursos disponíveis no servidor e dos processos;

4. HIPÓTESES

O número de falhas nas execuções das cargas de trabalho por falta de recursos de *hardware* é quase zero quando esses processos são gerenciados por sistemas automatizados. Neste caso, só é possível ocorrer uma falha quando o processo executado se comportar de maneira incomum ou inesperada.

Este sistema permite que o profissional que anteriormente era responsável pelo gerenciamento e execução dessas atividades fique livre para realizar outras tarefas dentro da organização.

5. PRESSUPOSTOS TEÓRICOS

5.1. INTELIGÊNCIA ARTIFICIAL

O termo “Inteligência Artificial (IA)” surgiu no encontro de Dartmouth que aconteceu em 1956. Desde então, diversas correntes de pensamento em IA têm estudado formas de estabelecer comportamentos “inteligentes” nas máquinas.

“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.” (RICH & KNIGHT, 1994).

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.

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

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.

Segundo Reisbeck e Schank (1989), 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. O RBC possui três fases: recuperação dos casos, definição do problema atual, adaptação.

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. Em 50 anos de computação, uma máquina que inicialmente conseguia executar apenas uma instrução por segundo e custava dez bilhões de dólares passou a executar um bilhão de instruções por segundo e seu custo foi reduzido para aproximadamente mil dólares. 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 (p.2, 2002) 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.

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).

5.3. JAVA

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).

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 disseminação da linguagem no mundo da informática, 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;

Recentemente adquirida pela Oracle, a linguagem hoje se encontra em sua sétima versão.

6. METODOLOGIA

6.1. METODOLOGIA DE GERENCIAMENTO

Para o gerenciamento da equipe, serão utilizadas algumas técnicas do paradigma Ágil, como por exemplo: Criação de *sprints* de desenvolvimento[5], controle de tarefas executadas através de *cards*, entre outras técnicas, fazendo com que a equipe consiga ser auto-gerenciável.

6.1.1. 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.

6.2. METODOLOGIA DE MODELAGEM

Para a parte de modelagem do sistema, serão 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.

6.3. METODOLOGIA DE DESENVOLVIMENTO

A linguagem de desenvolvimento utilizada será Java por permitir a execução do sistema tanto em Windows como no Linux. O banco de dados utilizado será o PostgresSql por ser gratuito e suportar o volume de dados utilizado no sistema. Será usado o *framework* Hibernate para a persistência de dados.

Na primeira fase do desenvolvimento será implementado o núcleo do sistema, onde estará a IA responsável pela execução das atividades de acordo com o estado atual da máquina.

[5] Uma fração de tarefas a serem desenvolvidas

Na segunda fase será implementado o cadastro de cargas de trabalho e o agendamento das atividades.

6.4. 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 mais funcional e eficiente.

O sistema atende às necessidades do mercado. Ao ser concebido baseado na plataforma Java, visa-se a interoperabilidade entre plataformas, ou seja, a possibilidade de sua execução nos principais Sistemas Operacionais. Outra forte tendência, o controle remoto (*web*/*mobile*) dos recursos disponíveis no servidor e dos processos, foi previsto para uma aplicação futura.

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.

Através de um conceito bem definido de Inteligência Artificial, também foi possível fazer com que o sistema analisasse execuções passadas e otimizasse cada vez mais o desempenho das tarefas, levando em conta seu consumo (em relação a *hardware*), o tempo de execução e suas respectivas dependências.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

9. ANEXOS

9.1. DOCUMENTO DE VISÃO

Nos dias de hoje, 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.

9.2. 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.

9.3. LISTA DE REQUISITOS DETALHADA

9.3.1. 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;

9.3.2. Requisitos Não Funcionais

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

9.4. 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.

9.5. DIAGRAMA DE CASOS DE USO

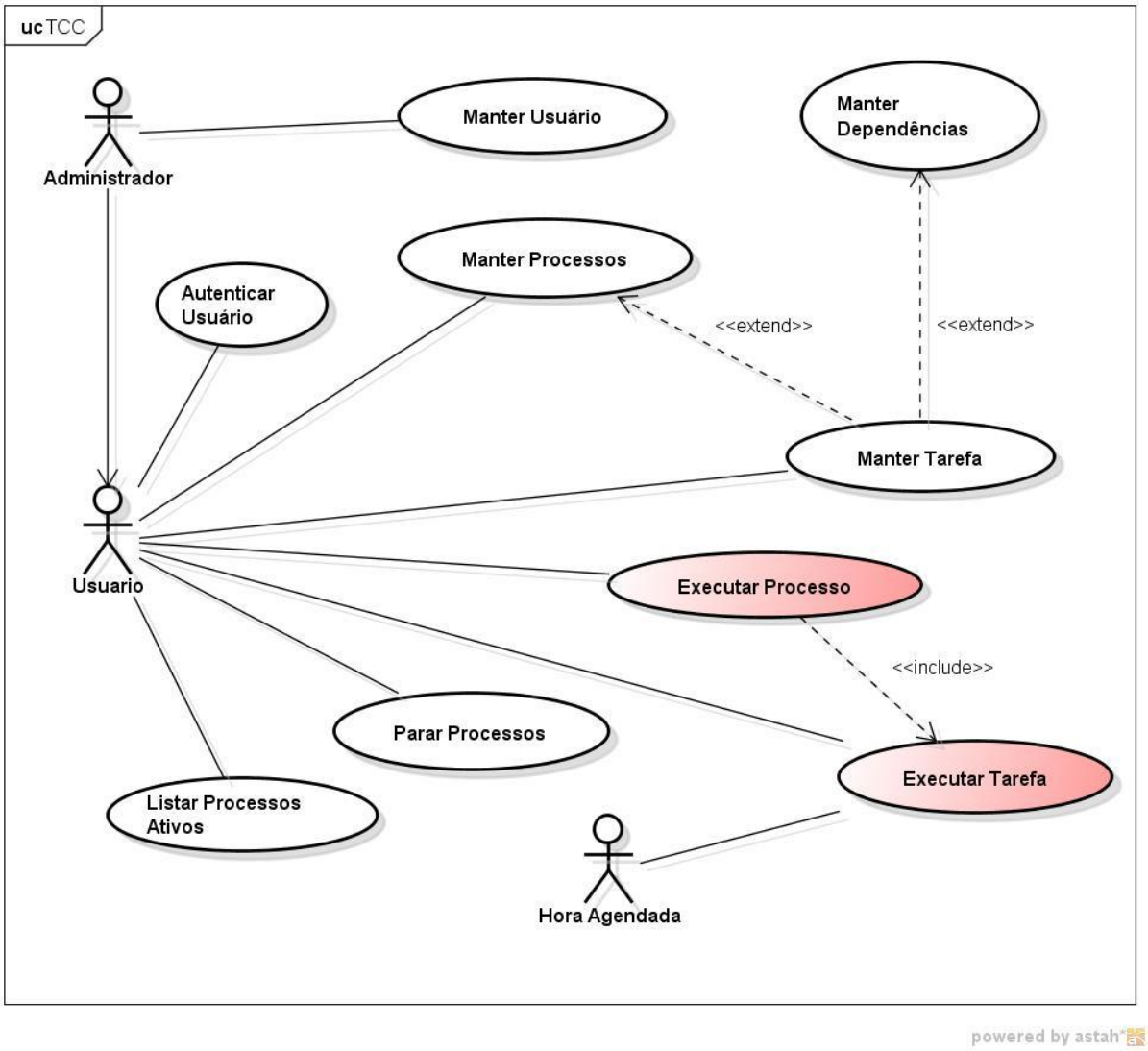

Figura 1: Diagrama de Casos de Uso

9.6. DESCRIÇÃO DE CASOS DE USO

9.6.1. 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.

9.6.2. 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			

9.6.3. 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.

9.6.4. 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.

9.6.5. 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.

9.6.6. 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.

9.6.7. 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.

9.6.8. 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			

9.6.9. 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			

9.7. DIAGRAMA DE CLASSES

9.7.1. Pacote Modelo

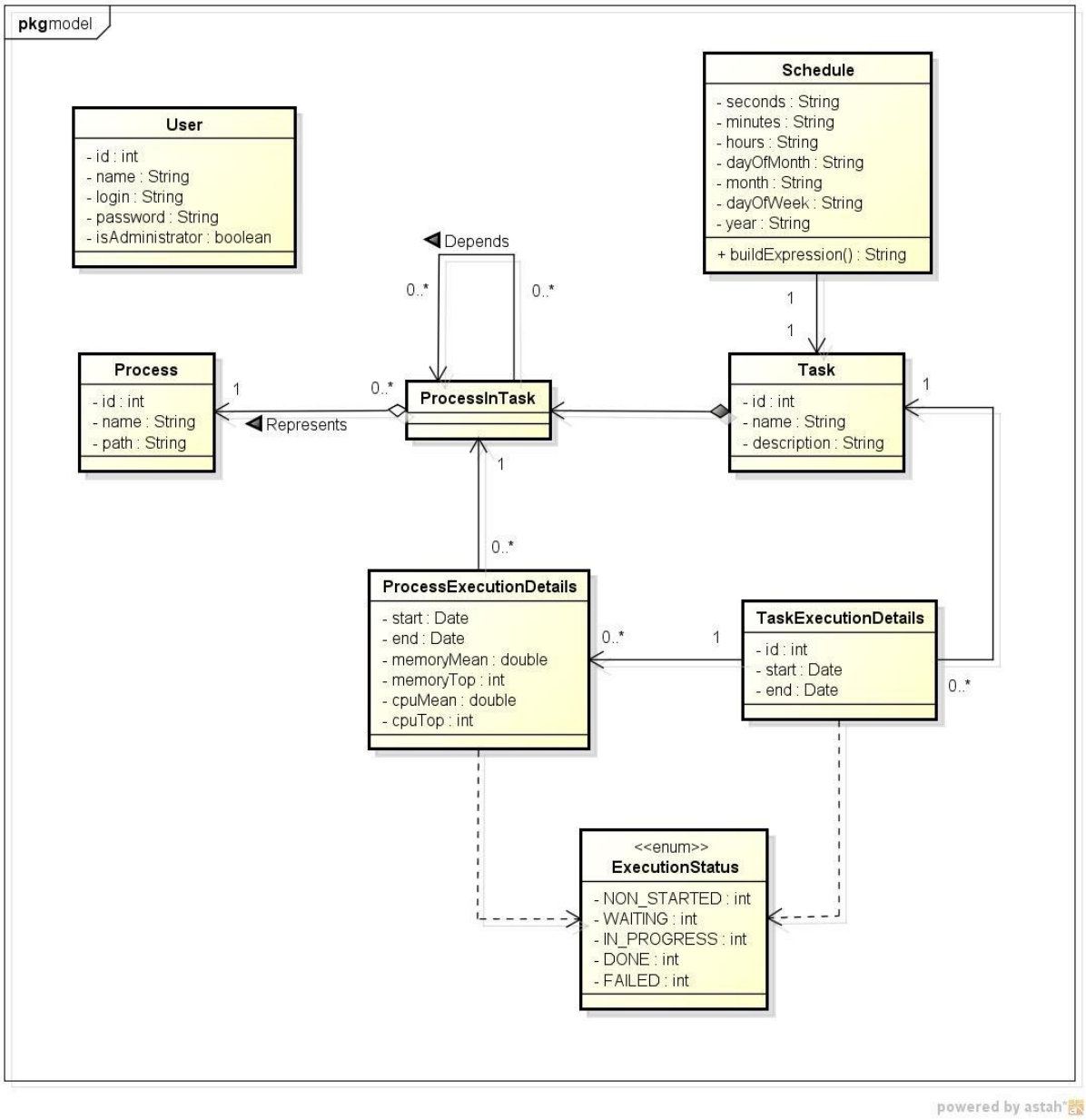

Figura 2: Diagrama de Classes

9.8. DIAGRAMA DE SEQUÊNCIA

9.8.1. Executar Tarefa

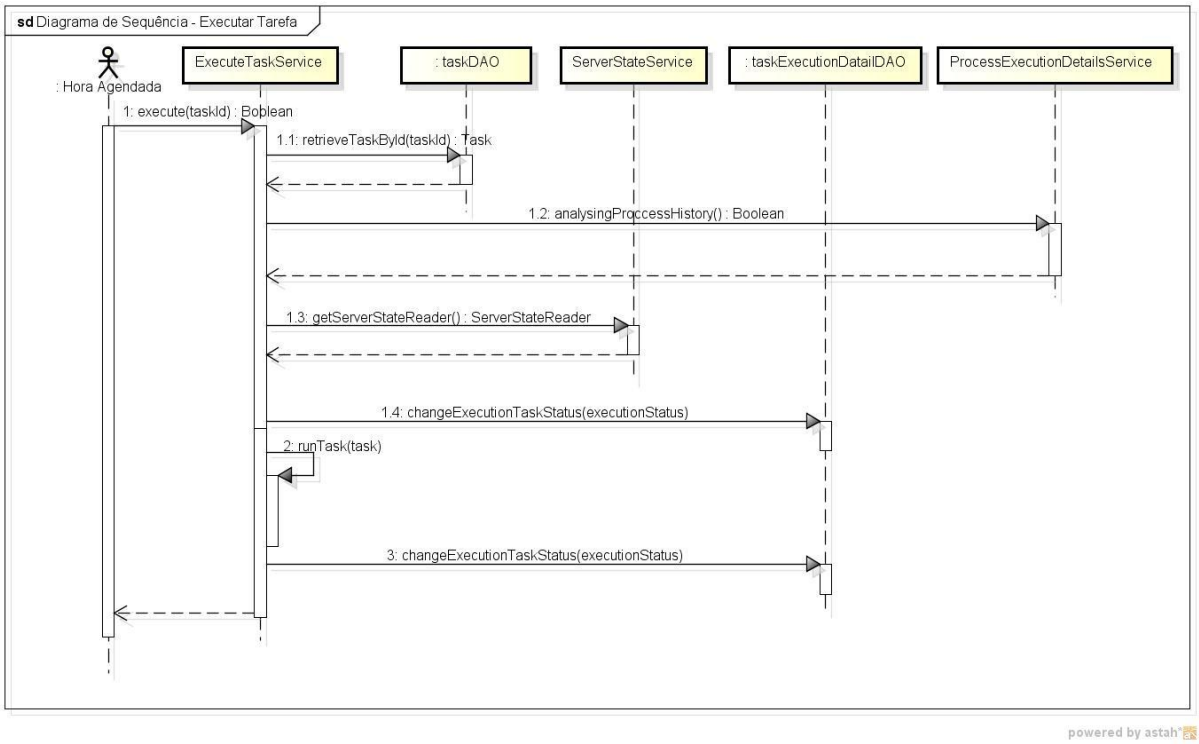

Figura 3: Diagrama de Sequência - Executar Tarefa

9.8.2. Executar Processo

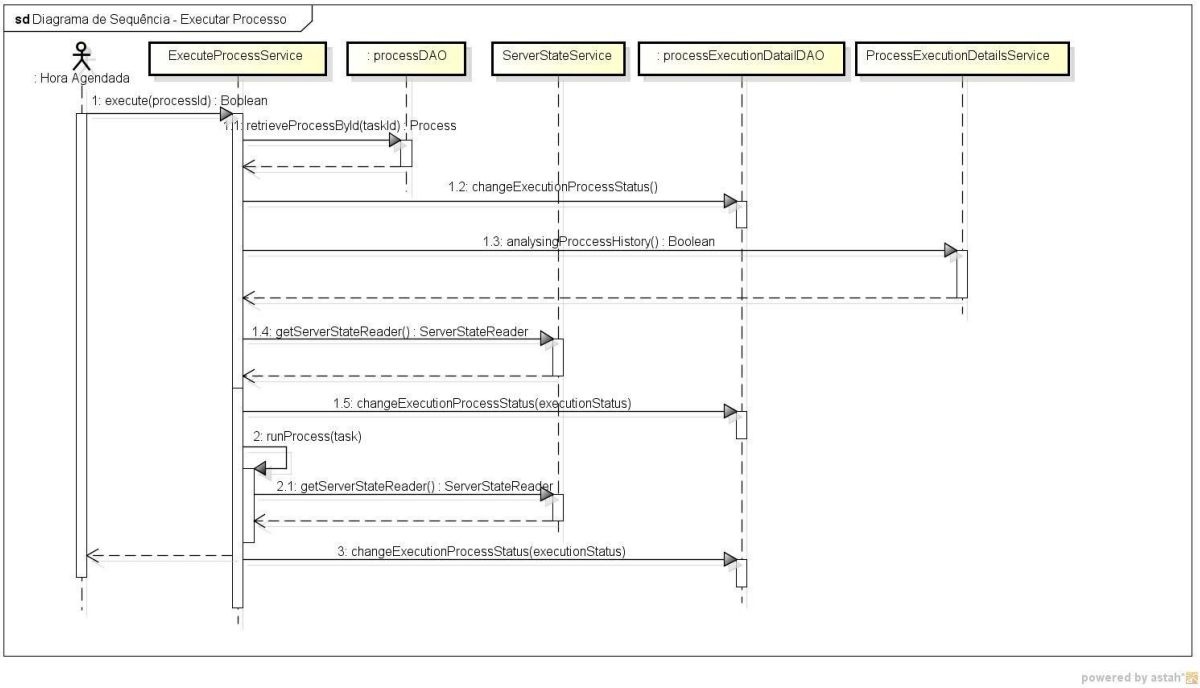

Figura 4: Diagrama de Sequência - Executar Processo

9.9. DIAGRAMA DE ESTADO

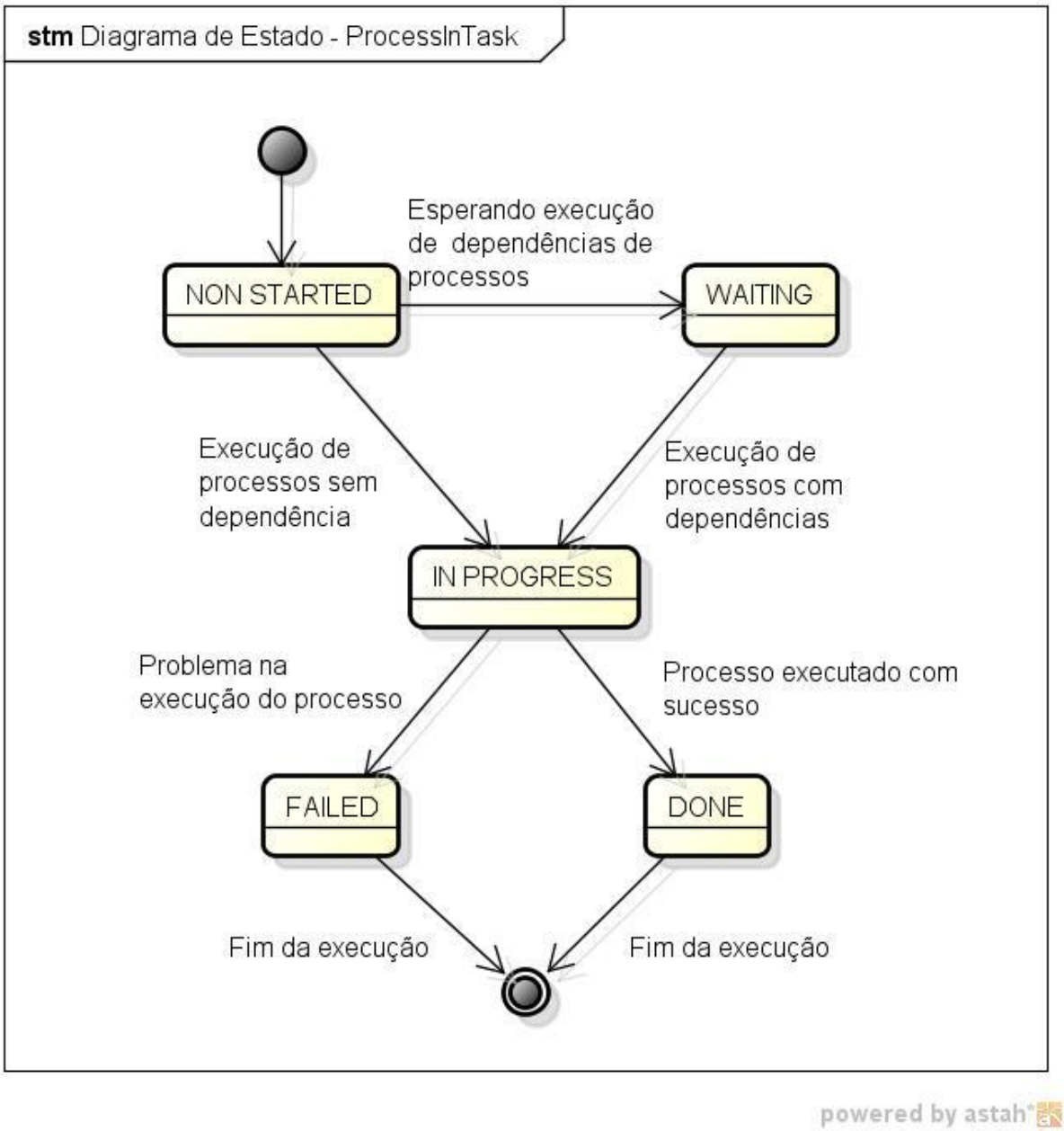

Figura 5: Diagrama de Estado - Process In Task

9.10. DIAGRAMA DE ATIVIDADES

9.10.1. Executar Tarefa

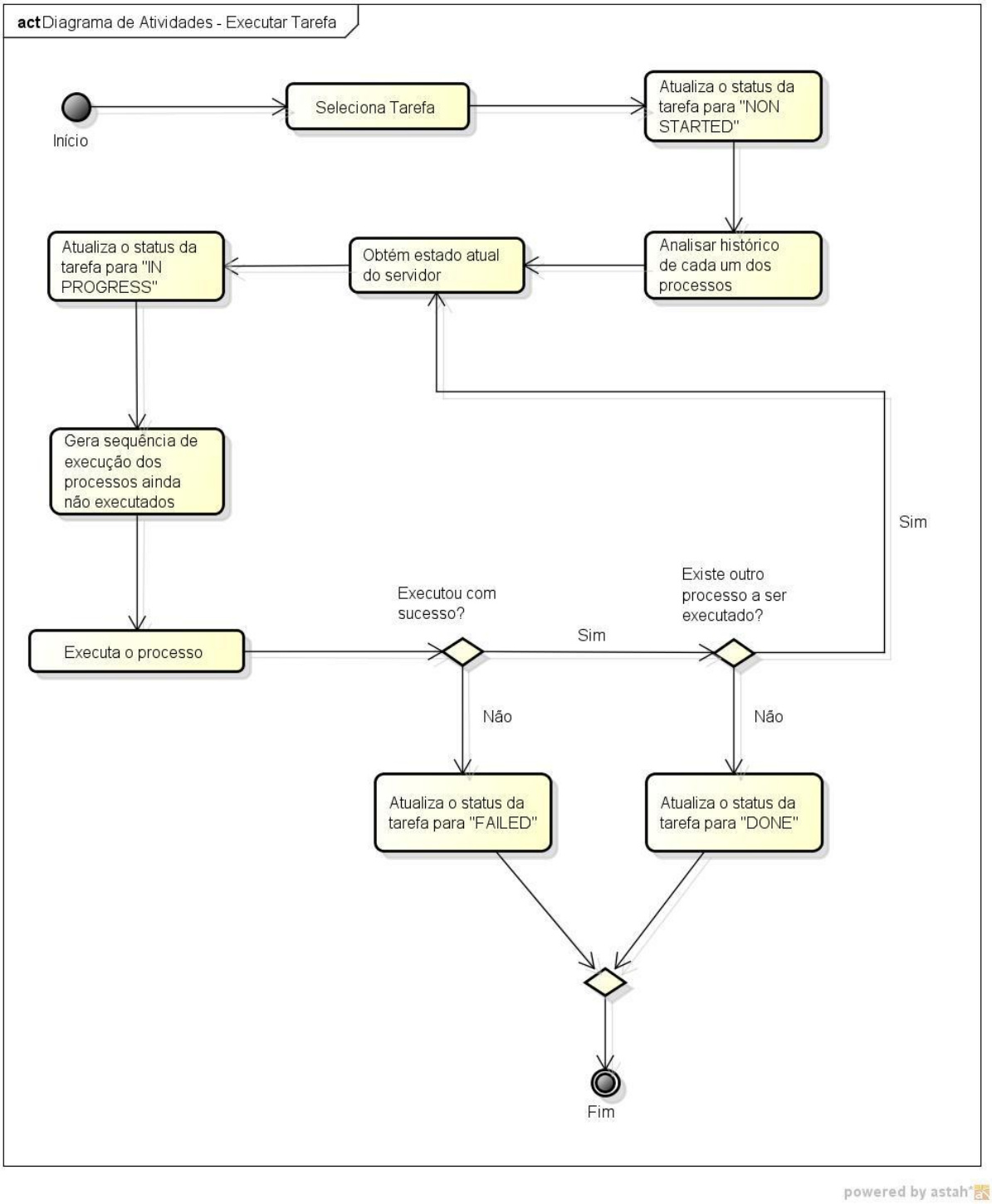

Figura 6: Diagrama de Atividades - Executar Tarefa

9.10.2. Executar Processo

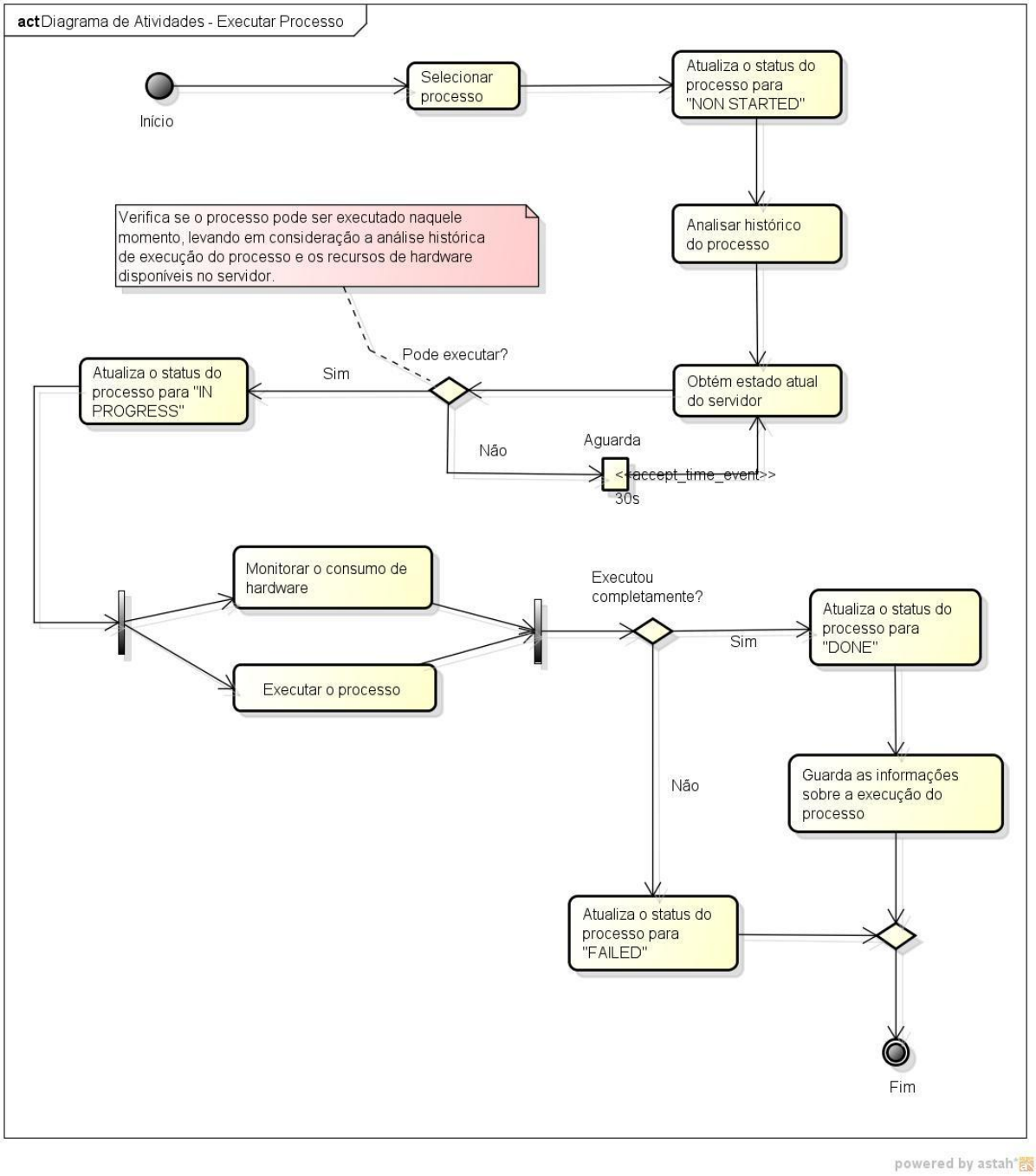

Figura 7: Diagrama de Atividades - Executar Processo

9.11. DIAGRAMA ARQUITETURAL

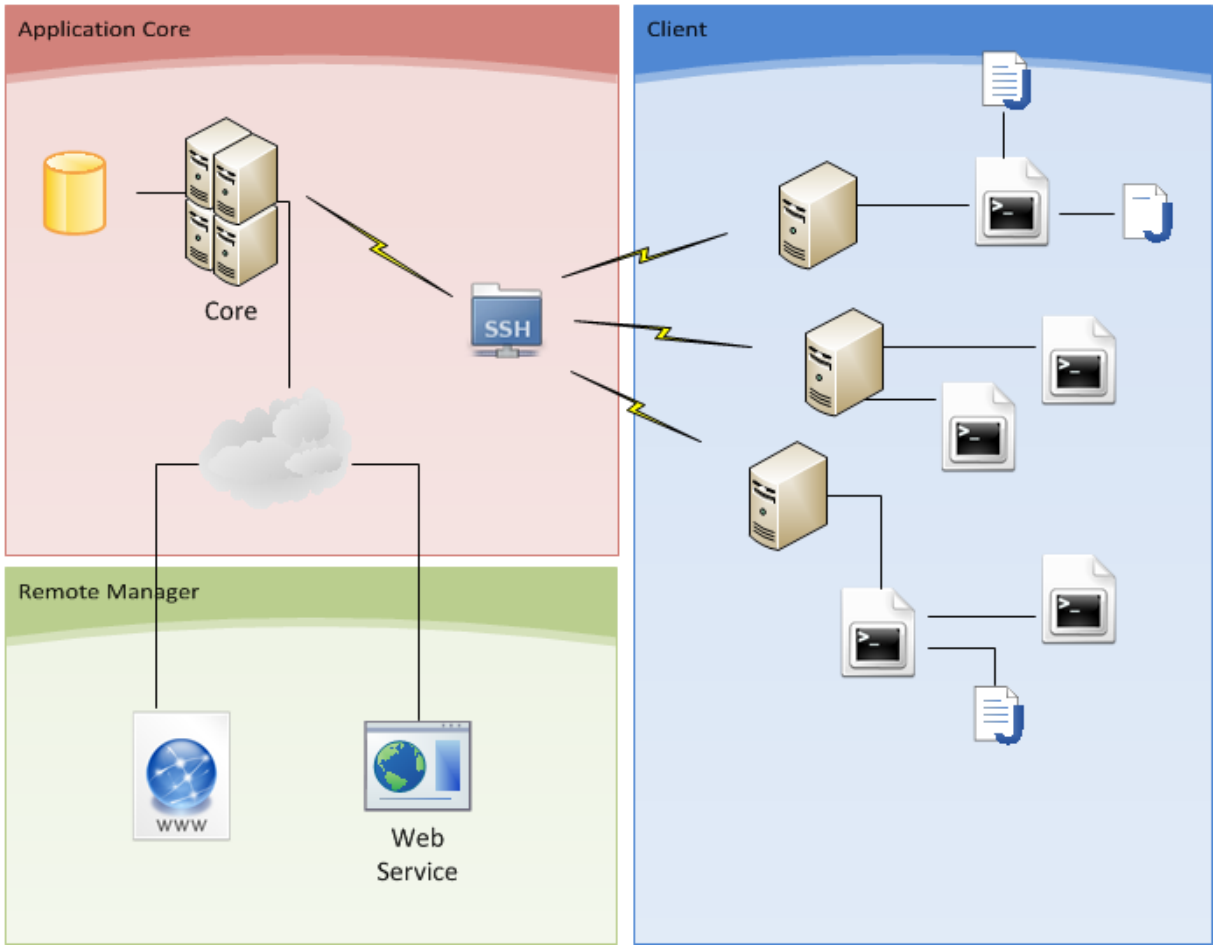

Figura 8: Diagrama Arquitetural