

Lucas Afonso Cazarote

**Web-RTsim: Implementação de plataforma
web para o RTsim**

São José do Rio Preto

2019

C386w Cazarote, Lucas Afonso
Web-RTsim: implementação de plataforma web para o RTsim /
Lucas Afonso Cazarote. -- São José do Rio Preto, 2019
50 f.

Trabalho de conclusão de curso (Bacharelado - Ciência da
Computação) - Universidade Estadual Paulista (Unesp), Instituto de
Biociências Letras e Ciências Exatas, São José do Rio Preto
Orientador: Aleardo Manacero Junior

1. Sistemas de Tempo-real. 2. Simulação. 3. Plataforma web. I.
Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca do Instituto de
Biociências Letras e Ciências Exatas, São José do Rio Preto. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

Lucas Afonso Cazarote

Web-RTsim: Implementação de plataforma web para o RTsim

Monografia apresentada ao Departamento de Ciências de Computação e Estatística do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", como parte dos requisitos necessários para obtenção do título de "Bacharel" em Ciência da Computação da UNESP

Universidade Estadual Paulista "Júlio de Mesquita Filho" (UNESP)

Instituto de Biociências, Letras e Ciências Exatas (IBILCE)

Bacharelado em Ciência da Computação

Prof. Dr. Aleardo Manacero Junior

São José do Rio Preto

2019

Lucas Afonso Cazarote

Web-RTsim: Implementação de plataforma web para o RTsim

Monografia apresentada ao Departamento de Ciências de Computação e Estatística do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", como parte dos requisitos necessários para obtenção do título de "Bacharel" em Ciência da Computação da UNESP

Prof. Dr. Aleardo Manacero Junior Lucas Afonso Cazarote

Banca Avaliadora:

Prof. Dr. Rodrigo Capobianco Guido

Profa. Dra. Carina Alexandra Rondini

São José do Rio Preto

2019

*Dedico este trabalho aos meus pais Carlos e Marlene.
À minha irmã Carla.*

Agradecimentos

Agradeço a Deus que durante esses anos de graduação me manteve firme e me deu forças para aguentar todos os momentos difíceis pelos quais passei. À minha família que me deu suporte em todos os momentos e sempre estiveram ao meu lado, à minha mãe Marlene pessoa única e especial a qual devo tudo o que sou e tive a sorte de tê-la como mãe, sempre dedicada e preocupada com o meu bem-estar; ao meu pai Carlos que se arrisca todos os dias nas estradas do Brasil para me proporcionar uma vida digna e a oportunidade de ter uma formação e à minha irmã Carla por sempre estar presente e compartilhar todos os momentos, meu muito obrigado e saibam que eu amo demais vocês.

Agradeço ao Prof. Dr. Aleardo Manacero Junior pela orientação nesse projeto e aporte na busca do conhecimento durante minha vida acadêmica, além da amizade durante todos esses anos; agradeço também a Profa. Dra. Renata Spolon Lobato pela co-orientação durante todos esses anos de laboratório, ajuda nos momentos difíceis, compreensão e amizade.

Meus agradecimentos aos professores com os quais tive aula ou contato durante essa trajetória, em especial, o Prof Dr Norian Marranghello e a Profa Dra Maria Tercília Vilela de Azeredo, os quais me orientaram durante todos os cinco anos que participei da UNATI - Universidade Aberta à Terceira Idade, projeto o qual me auxiliou em vários pontos da minha formação como estudante e pessoa.

Agradeço também a todos os amigos que conheci ao entrar no GSPD, pela amizade, incentivo e apoio, troca de experiências e conhecimentos no laboratório - Gabriel Saraiva, Mario Camara, Renan Miguel, Victor Hugo, Diogo Tavares, Fernanda Peronaglio, Gabriel Covello, Gabriel Bueno, Brunno Navarro, Luís Vinicius, Arthur Jorge, Denison Menezes, Leonardo Santos, Renan Alboy, Cássio Forte, João Antônio, Fábio Amorim, Fernando Kaway, Danilo Costa, Rafael Stabile e Thiago Alexandre.

Por fim, agradeço à UNESP pelo apoio e infraestrutura cedidas para que este trabalho fosse realizado.

*"Sucesso não é o final, fracasso não é fatal:
é a coragem para continuar que conta."
- Winston Churchill*

Resumo

A área de sistemas de tempo-real tem se tornado cada dia mais presente no cotidiano das pessoas, com aplicações como controle de aeronaves em uma área de voo, controle de reatores em usinas nucleares, etc. Contudo, por se tratar de sistemas que precisam atender rigorosamente o prazo de suas tarefas por envolver riscos financeiros ou de segurança física em ambientes reais, utiliza-se de simulações para prever como será o comportamento de tais aplicações. Além disso, nas últimas décadas ocorreu a disseminação da Internet pelo mundo, com o barateamento dos computadores e de serviços de provedores de Internet, o surgimento de dispositivos móveis, incluindo *smartphones*. Isso acarretou em modificações na sociedade em diversos âmbitos. Dentro desse contexto, temos o RTsim (*Real-Time Simulator*), ferramenta utilizada para simular o comportamento de sistemas de tempo-real, atualmente contando com uma versão *desktop*, o que acaba restringindo o seu uso. Assim propõe-se este trabalho, o desenvolvimento de uma aplicação web para o RTsim, chamada de *Web-RTsim*, com o intuito de ampliar e flexibilizar o uso da ferramenta por diferentes dispositivos conectados a rede mundial de computadores.

Palavras-chaves: Simulação, Sistemas de Tempo-Real, Rede de computadores, Plataforma Web.

Abstract

Real-time systems have are increasingly more present in people's daily routines, comprising applications such as flight control of airplanes, control of nuclear reactors, etc. However, since they are systems that need to strictly comply with task's deadlines to avoid either financial or physical security risks in the real world, developers use simulations to predict application's behavior. Beyond this, in the last decades occurred the world dissemination of Internet, brought by lower costs of computers and Internet service providers, along the appearance of smartphones. This implied in the change of the human society at various levels. In this context we have RTsim (Real-Time Simulator), which is a tool used to simulate the behavior of real-time systems, currently working as a desktop version, what restricts its use. Therefore, this work proposes the *Web-RTsim*, in order to expand its use and make it more flexible to be used from different devices connected to the worldwide computer network.

keywords: Simulation, Real-Time Systems, Computer Networks, Web Plataform.

Lista de ilustrações

Figura 2.1 – Interface gráfica do módulo de inserção da fórmula de escalonamento.	24
Figura 2.2 – Arquitetura cliente-servidor	25
Figura 2.3 – Esquematização do funcionamento de um <i>servlet</i>	28
Figura 2.4 – Esquematização do funcionamento de uma <i>Java Server Page</i>	29
Figura 3.1 – Diagrama de classe exibindo as classes de interface e simulação dos algoritmos	31
Figura 3.2 – Enfoque no diagrama de classe exibindo as classes chamadas no exemplo de simulação	31
Figura 3.3 – Chamadas de classe de execução para chamada de escalonamento	32
Figura 3.4 – Diagrama de classe exibindo a interação entre o <i>servlet</i> de simulação e as classes principais do RTsim.	34
Figura 3.5 – Diagrama de classe exibindo a interação entre o <i>servlet</i> de criação de escalo- nadores do RTsim.	34
Figura 3.6 – Layout da página inicial do <i>Web-RTsim</i>	36
Figura 3.7 – Layout da tela inicial do RTsim.	36
Figura 3.8 – Layout da página de simulação do algoritmo Taxa Monotônica.	38
Figura 4.1 – Layout da página de simulação do algoritmo Taxa Monotônica e algumas tarefas inseridas no <i>Web-RTsim</i>	39
Figura 4.2 – Layout da tela de simulação do algoritmo Taxa Monotônica no RTsim e algumas tarefas inseridas	40
Figura 4.3 – Layout da página de simulação do algoritmo Servidor por Deferência no <i>Web-RTsim</i>	41
Figura 4.4 – Layout da página de simulação do algoritmo Servidor por Deferência no <i>Web-RTsim</i> com tarefas aperiódicas	41
Figura 4.5 – Layout da página de simulação do algoritmo Troca de Prioridade no <i>Web-RTsim</i> com semáforo	42
Figura 4.6 – Inserção das tarefas no algoritmo Taxa Monotônica no RTsim	43
Figura 4.7 – Gráfico do esalonamento obtido pelo algoritmo Taxa Monotônica no RTsim	43
Figura 4.8 – Exibição das informações estatísticas do escalonamento obtido pelo algoritmo Taxa Monotônica no RTsim	44
Figura 4.9 – Inserção das tarefas no algoritmo Taxa Monotônica no <i>Web-RTsim</i>	44
Figura 4.10–Gráfico do escalonamento obtido pelo algoritmo Taxa Monotônica no <i>Web-RTsim</i>	45
Figura 4.11–Gráfico do escalonamento obtido pelo algoritmo Taxa Monotônica no <i>Web-RTsim</i>	45

Lista de abreviaturas e siglas

GSPD	Grupo de Sistemas Paralelos e Distribuídos
SPARTS	Simulator for Power Aware and Real-Time Systems
STRESS	Simulator for Hard Real Time Systems
SimSo	Simulator of Mutiprocessor Schedulling with Overheads
AURTSS	AU Real Time Scheduler Simulator
RTsim	Real-Time Simulator
Web	World Wide Web
DS	Deferrable Server
SS	Sporadic Server
GUI	Graphical User Interface
API	Application Programming Interface
JVM	Java Vitual Machine
HTTP	Hypertext Transfer Protocol
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
API	Application Programming Interface
AJAX	Asynchronous JavaScript and XML
XML	Extensible Markup Language
DOM	Document Object Model
JSON	JavaScript Object Notation

Sumário

1	INTRODUÇÃO	12
1.1	Objetivos	12
1.2	Motivação	12
1.3	Justificativa	13
1.4	Metodologia	13
1.5	Exequibilidade	14
1.6	Organização do texto	14
2	REVISÃO BIBLIOGRÁFICA	15
2.1	Sistemas de Tempo-Real	15
2.2	Escalonamento de Tempo-Real	15
2.2.1	Tarefas	17
2.2.2	Algoritmos de Escalonamento	18
2.3	Simuladores de Tempo-Real	20
2.3.1	SPARTS	20
2.3.2	STRESS	20
2.3.3	SimSo	21
2.3.4	Realtss	21
2.3.5	AURTSS	21
2.4	RTsim	22
2.4.1	Breve Histórico	22
2.4.2	Estado Atual	23
2.5	Arquitetura cliente-servidor	24
2.6	Aplicações Web	25
2.7	Java Enterprise Edition	27
2.7.1	<i>Servlets</i>	27
2.7.2	<i>Apache Tomcat</i>	28
2.7.3	<i>Java Server Pages - JSP</i>	28
2.8	Considerações finais	29
3	DESENVOLVIMENTO	30
3.1	Organização da ferramenta RTsim	30
3.2	Organização do <i>Web-RTsim</i>	33
3.2.1	O <i>Servlet</i> de simulação	35
3.2.2	O <i>servlet</i> de criação de escalonadores	35
3.2.3	JSPs na criação de páginas	35

3.3	Interface gráfica	36
3.3.1	Javascript no tratamento de eventos e requisições AJAX	37
3.4	Considerações finais	38
4	TESTES E RESULTADOS	39
4.1	Testes de interfaces gráficas	39
4.2	Geração de um escalonamento utilizando o RT-sim e o <i>web</i>-RTsim . . .	42
4.2.1	Escalonamento realizado no RT-sim	42
4.2.2	Escalonamento realizado no <i>Web</i> -RTsim	44
4.3	Considerações finais	46
5	CONCLUSÕES E TRABALHOS FUTUROS	47
5.1	Conclusões	47
5.2	Trabalhos Futuros	47
	REFERÊNCIAS	49

1 Introdução

Nas últimas décadas a computação tem evoluído rapidamente fazendo-se presente em diversos âmbitos e no cotidiano de milhares de pessoas. Uma das áreas beneficiadas por essa evolução são os sistemas que demandam executar e produzir resultados de forma correta e confiável, dentro de um intervalo de tempo determinado, podendo ser crítico caso ocorra a perda do deadline. Os sistemas de tempo-real (1) são voltados para aplicações que necessitam atender os prazos das tarefas rigidamente, pois são utilizados em áreas consideradas de alto risco, tendo como exemplo, uma torre de controle de aeronaves em uma determinada zona de voo.

Para atender de forma eficiente o maior número de tarefas dentro de seus prazos desenvolveu-se algoritmos de escalonamento, os quais tem como objetivo buscar o melhor cenário para execução das mesmas. Dessa forma existem vários algoritmos propostos que utilizam diferentes abordagens para alcançar esse objetivo. Contudo, conforme dito anteriormente, a verificação do desempenho desses sistemas utilizando o ambiente real é inviável, por trazer riscos financeiros e de segurança a todos os envolvidos, assim, abre-se mão das simulações para a realização de tais testes.

Além disso, a evolução citada inicialmente proporcionou a disseminação do acesso à Internet pelo mundo, conectando milhões de pessoas através dos mais diversos tipos de dispositivos, como: *smartphones*, *laptops*, *consoles para jogos*, entre tantos outros, como mostra Kurose e Ross (2).

1.1 Objetivos

O objetivo deste trabalho é realizar a implementação de uma versão web para a ferramenta de simulação RTsim (3), pois atualmente conta apenas com uma versão *desktop* disponível para *download* na página do grupo o qual desenvolveu e mantém a ferramenta, o GSPD (Grupo de Sistemas Paralelos e Distribuídos) (4). Por se tratar de uma ferramenta utilizada tanto por desenvolvedores de sistema de tempo-real, como por alunos de graduação para o auxílio no aprendizado, o intuito de desenvolver uma versão Web é facilitar o acesso ao usuários, assim, utilizando um dispositivo qualquer conectado à Internet será possível utilizá-la nos mais diversos lugares, até mesmo durante as aulas nas universidades.

1.2 Motivação

Dentre os fatores que motivam esse trabalho estão a necessidade de flexibilizar o uso e tornar mais acessível o RTsim, realizando a implementação de um sistema *web* para a ferramenta. Devido o aumento do acesso a rede de computadores nas últimas décadas e a grande populariza-

ção de diferentes tipos de dispositivos eletrônicos, observou-se uma possibilidade de expandir os campos de uso do simulador, proporcionando ao usuário uma forma prática e rápida de utilizá-la.

1.3 Justificativa

Como referido anteriormente, por gerar riscos e possíveis prejuízos faz-se necessário o uso de simulação no âmbito de sistemas de tempo real, tendo como objetivo verificar sua taxa de êxito no escalonamento de tarefas dentro de seus deadlines. Além disso, como houve uma pluralização e expansão do acesso à Internet, viu-se uma oportunidade de ampliação e flexibilização no uso da ferramenta RTsim acrescentando a possibilidade de acesso utilizando um navegador, justificando assim, a proposição desse trabalho.

1.4 Metodologia

A metodologia adotada para realização e desenvolvimento desse projeto consiste em diversas etapas, sendo elas: estudo sobre o assunto e a ferramenta RTsim, especificação e modelagem da aplicação *web*, implementação e por fim a realização de testes e análise dos resultados obtidos. Para que fosse possível realizar as etapas descritas anteriormente, iniciou-se com o estudo de sistemas de tempo-real, simulação, os algoritmos de escalonamento mais importantes e outras ferramentas de simulação existentes, para isso foram consultados diversos livros sobre os assuntos disponíveis na biblioteca da Universidade Estadual Paulista "Júlio de Mesquita Filho" (UNESP), além do portal de periódicos disponibilizado pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES).

Foi realizada a análise e aprofundamento na arquitetura e funcionamento do RTsim para que fosse possível entender por completo a ferramenta em sua versão *desktop* e conseguir ponderar quais modificações e reestruturações seriam necessárias com finalidade de alcançar o objetivo deste trabalho. Para desenvolver essa versão *web* utilizou-se as linguagens *Java Enterprise Edition* (JavaEE), *HyperText Markup Language* (HTML), *Javascript* e *AJAX*, além do *servlet container Apache Tomcat*, sendo este último o servidor *web* escolhido.

Após o processo de implementação da aplicação *web* foram realizados testes para avaliação e validação de todas as funcionalidades do *Web-RTsim*, inicialmente foi conduzido os testes das interfaces gráficas reformuladas analisando o comportamento em determinadas situações, após isso foi realizado a verificação dos dados inseridos nos formulários e por fim realizados diversos escalonamentos em ambas as versões da ferramenta e analisados se os resultados obtidos foram compatíveis com o esperado.

1.5 Exequibilidade

Para a realização desse trabalho será utilizado os recursos de *hardware* existentes no laboratório do GSPD, fazendo uso do servidor do tipo torre PowerEdge T420 para hospedagem da página, além de todas as linguagens de programação necessárias para alcançar o objetivo final, além de *frameworks* com o intuito de auxiliar no desenvolvimento e composição do sistema. Como todos os itens citados anteriormente já foram adquiridos ou possuem licença de *software* livre, podendo ser utilizados por qualquer pessoa sem custos, chega-se a conclusão de que abre-se mão de todos os requisitos necessários para tornar esse projeto viável dentro do prazo estabelecido.

1.6 Organização do texto

Este texto está dividido em cinco capítulos, além desse capítulo introdutório, o Capítulo 2 contendo a revisão bibliográfica e estado da arte atual, a qual abrange os seguintes tópicos: descrição sobre os sistemas de tempo-real, escalonamentos e conceitos importantes da área, a descrição dos tipos de algoritmos de escalonamento principais, os simuladores de tempo-real, a descrição mais detalhada do Rtsim e seu estado atual. Tem-se como objetivo, apresentar e familiarizar o leitor sobre o assunto tratado, sendo descritos todos os componentes chave para o entendimento acerca do trabalho. O Capítulo 3 onde descreve-se o desenvolvimento e atividades realizadas, enquanto o quarto apresentará os testes realizados para validação da aplicação e os resultados obtidos. Por fim, o quinto capítulo apresentará as conclusões e possíveis trabalhos futuros.

2 Revisão Bibliográfica

Neste capítulo são apresentados os conceitos necessários para a realização e compreensão desse trabalho, serão definidos conceitos essenciais como: sistemas de tempo-real, descrição de escalonadores existentes em tempo-real, a descrição de simuladores utilizados na área, a descrição do RTsim, qual seu estado atual e funcionalidades presentes e também definições sobre plataforma *web*, arquitetura e tecnologias utilizadas.

2.1 Sistemas de Tempo-Real

Com a expansão da computação e crescente uso no cotidiano das mais diversas áreas gerou-se a necessidade de utilização de diversos tipos de sistemas, sendo que cada um deles possui uma função específica a qual irá executar. Para as aplicações que possuem restrições em seu tempo de execução, possuindo prazos determinados para obtenção de resultados de cada tarefa, necessita-se de sistemas que consigam controlar o instante em que inicia-se a execução e que sejam capazes de administrar o cumprimento de todas as tarefas dentro desses prazos estipulados, produzindo resultados a tempo de serem úteis.

Nesse cenário que é possível definir-se um sistema de tempo-real, quando são capazes de gerenciar o tempo de cada tarefa em execução e o cumprimento da maior quantidade de tarefas dentro de seus *deadlines*. Esses sistemas são utilizados para aplicações consideradas críticas, possuindo um tempo rígido para serem atendidas, acarretando consequências graves quando isso não ocorre (5), como por exemplo: controle de reatores em usinas ou de pacientes internados em hospitais. Esses sistemas podem ser classificados de acordo com a necessidade de produzir resultados corretos dentro de seus prazos, sendo eles classificados em (6):

- *Soft Real-Time*: São chamados de sistemas não-críticos de tempo-real quando não há consequências graves na ocorrência de falhas, sendo esses mais tolerantes a atrasos.
- *Hard Real-Time*: São chamados de sistemas críticos de tempo-real quando não admite-se falhas, sendo que qualquer ocorrência pode acarretar consequências catastróficas, como perda de vidas ou prejuízos financeiros.

2.2 Escalonamento de Tempo-Real

A palavra escalonamento refere-se a ação de ordenar tarefas que devem ser executadas. Para indicar a ordem temporal em que ocorrerá essa execução, utilizasse uma escala, elencando quem ocupará o processador do sistema. Essa escala de execução trata-se de uma lista a qual

indica-se ordem em que o processador deve ser ocupado. O elemento responsável por realizar o escalonamento e essa coordenação de ocupação é denominado de escalonador (7).

Considerando-se a importância dos prazos nesse tipo de sistema é possível observar a importância de um bom funcionamento do escalonador, controlando a execução dessas tarefas, para que isso seja possível existem as políticas de escalonamento. Estas políticas são um conjunto de regras as quais devem ser seguidas norteadas a escolha de qual tarefa irá ocupar o processador, quais são as condições para que haja uma ação, além da disponibilidade de recursos e prioridades existentes, possibilitando assim que o escalonador produza resultados corretos dentro de um tempo hábil para que os mesmos sejam utilizados.

Um escalonador deve satisfazer algumas condições (1) descritas a seguir. Em um determinado instante todo processador deve ser alocado a no máximo uma tarefa e toda tarefa é alocada a no máximo um processador por vez. Nenhuma tarefa é alocada antes do seu instante de chegada. Todas as exigências de precedência ou requerimento de recursos devem ser satisfeitas. E por fim, dependendo do algoritmo utilizado pelo escalonador, a soma dos tempos de processamento de uma tarefa deve ser igual ao seu tempo de execução.

Os algoritmos de escalonamento podem ser classificados de acordo com seu comportamento (1), podendo ser classificados em algoritmos dinâmicos e estáticos. Nos algoritmos dinâmicos o conjunto existente de tarefas são ordenados antes de iniciar-se a execução, porém essa ordem inicial pode ser alterada de acordo com as ocorrências no sistema. Já os classificados como estáticos também ordenam o conjunto de tarefas ao início de sua execução, tomando como base um determinado parâmetro, entretanto difere do anterior em um ponto, a ordem pré-definida inicialmente não é alterada durante o decorrer da execução. Para que haja melhor entendimento sobre o assunto, deve-se ter conhecimento sobre determinados conceitos relacionados ao escalonamento, como os que segue:

- **Factibilidade:** um escalonamento é dito factível se for possível que todas as tarefas possam executar e ter seus deadlines atendidos, existindo assim, um escalonamento possível para tal conjunto de tarefas.
- **Preempção:** uma tarefa sofre preempção caso seja interrompida durante a sua execução deixando assim que outra tarefa utilize o processador em seu lugar, isso pode ocorrer devido ao nível de prioridade das mesmas. A tarefa que foi interrompida volta para a fila de espera e poderá, posteriormente, retomar sua execução do ponto em que parou. O responsável pela preempção é o escalonador.
- **Tempo de folga:** quando não há tarefas sendo executadas ou aguardando para executar ocorre um período de tempo em que o processador fica ocioso, sendo esse um parâmetro relacionado a ocupação do processador.

- Fila de pronto: estrutura que são armazenados os dados para a execução de tarefas no sistema quando as mesmas estiverem com o status de pronta para executar.
- Independência: ocorre a independência de uma tarefa quando ela pode ser executada sem que uma outra tarefa precise ter sido executada previamente.
- Tempo de chaveamento: tempo consumido quando ocorre as trocas de tarefas no processador.

2.2.1 Tarefas

Quando se faz referência a processos ou atividades que devem ser executadas pelo escalonador, estas são chamadas de tarefas, uma tarefa é a ação que deve ser executada respeitando suas características de execução e seus parâmetros (7). As tarefas podem ser classificadas de acordo com a sua frequência de execução, pode-se categorizar em:

- Tarefas Aperiódicas: esses tipos de tarefas não possuem seu tempo de ocorrência previamente conhecido, logo, o sistema não sabe quando ela estará na fila de pronto, podem ou não se repetir em períodos iguais, ou seja, suas repetições são irregulares e imprevisíveis.
- Tarefas Periódicas: possuem seu tempo de ocorrência conhecido, respeitando sempre o mesmo intervalo pré-estabelecido de tempo entre as vezes em que ocorre, suas repetições são regulares e previsíveis.
- Tarefas Esporádicas: são tarefas que se repetem em um intervalo mínimo de tempo, mas não há o conhecimento exato do tempo das repetições.

Além disso, as tarefas podem ser caracterizadas a partir de alguns parâmetros específicos, sendo apresentados a seguir:

- Tempo de chegada/Instante de ocorrência: instante de tempo em que a tarefa torna-se pronta para executar
- Período: intervalo de tempo em que a tarefa ocorrerá novamente, tendo relevância apenas para tarefas que são periódicas.
- Carga (tempo de execução): trata-se do tempo necessário para que a tarefa tenha sua execução completa, somente quando executar toda a sua carga será considerada como completa.
- *Deadline*: é um dos parâmetros mais importantes para tarefas de tempo-real, sendo o prazo em que a tarefa deve completar sua execução antes que seja considerada como falha. Existem dois tipos: o deadline relativo (intervalo de tempo entre a chegada da tarefa e seu prazo máximo de término) e deadline absoluto (o exato instante em que a tarefa deve terminar sua execução, sendo a soma entre o tempo de chegada e seu deadline relativo).

2.2.2 Algoritmos de Escalonamento

Os algoritmos de escalonamento são a base da ferramenta de simulação RTsim, logo para compreendê-la faz-se necessário o estudo desses algoritmos. O pacote de algoritmos nativos da ferramenta se dividem em monoprocessados e multiprocessados, sendo:

Cinco algoritmos monoprocessados:

- Taxa Monotônica
- Servidor por Deferência
- Servidor Esporádico
- Topo de Prioridade
- Troca de Prioridade

Três algoritmos para sistemas multiprocessados:

- Míope
- Leilão
- Leilão Focado

A seguir serão apresentados sucintamente os algoritmos de escalonamento nativos da ferramenta.

- **Taxa Monotônica (*Rate monotonic*):** trata-se de um algoritmo para escalonamento com prioridade fixa, sendo assim, são atribuídas prioridades as tarefas no início da execução não alterando sua ordem durante o processo. Tornou-se fundamental nos estudos de tempo-real por ser simples, estável e de fácil entendimento; suas tarefas devem ser periódicas e independentes e sua execução deve ser feita em ambientes monoprocessados (possui apenas uma unidade de processamento) (8).

Para ordenar suas tarefas toma como base os períodos de cada uma, dessa forma, tarefas com menores períodos terão maior prioridade. Por ser preemptivo permite que uma tarefa seja retirada do processador caso apareça uma de maior prioridade na fila de pronto. Para que o algoritmo funcione corretamente, considera-se que os *deadlines* das tarefas sejam equivalentes aos seus períodos, o tempo de chaveamento entre as tarefas é nulo e suas cargas são conhecidas e fixas. Ao adotar essa política de escalonamento, o Taxa Monotônica sempre atenderá os *deadlines* de maior prioridade, assim, em situações de sobrecarga do sistema a tarefa com maior período (menos prioritária) perderá o *deadline*.

- **Servidor por Deferência (*Deferrable Server* (DS)):** trata-se de um algoritmo para tarefas periódicas e aperiódicas em que tarefas aperiódicas possuem maior prioridade.(9) É criada uma tarefa periódica (servidora), com prioridade definida pelo algoritmo de Taxa Monotônica, a qual atende a carga aperiódica, a cada período recarrega-se completamente a carga servidora.
- **Servidor Esporádico (*Sporadic Server*(SD)):** algoritmo muito semelhante com o do *Deferrable Server*, diferindo apenas na política de recarga da tarefa periódica (servidora) (1), está é recarregada apenas o quanto foi utilizado após o período, contabiliza-se apenas a partir do ponto de início da execução da tarefa e não por completo como ocorre no DS.
- **Troca de Prioridade (*Priority Exchange*):** Utilizado em situações que ocorre quando tem-se tarefas com diferentes prioridades compartilhando recursos, acarretando em inversão de prioridade.(10) Uma tarefa, ao interromper a execução de uma outra tarefa que possui maior prioridade, herda a prioridade daquela tarefa até que o recurso utilizado por ambas seja liberado.

Este algoritmo não evita a ocorrência de *deadlocks* (situação em que um conjunto de tarefas estiver esperando por um evento que somente um outro processo desse mesmo conjunto poderá fazer acontecer) (11) devido a não preemptividade da alocação de recursos.

- **Topo de Prioridade (*Priority Ceiling*):** implementação a qual previne a ocorrência de *deadlocks* e ajuda a reduzir o tempo de bloqueio. Para isso, utiliza-se um parâmetro denominado de prioridade máxima, onde sua prioridade é sempre a mais alta possível entre as tarefas que utilizam um recurso compartilhado. Dessa forma, uma tarefa só pode ser interrompida se a prioridade da tarefa que solicita a execução for maior que a prioridade atual do parâmetro (10).
- **Míope (*Myopic*):** algoritmo para ambientes multiprocessados desenvolvido a partir do algoritmo heurístico (12), realiza o escalonamento das tarefas de forma que todos os deadlines sejam atendidos, alocando sempre a primeira tarefa no processador e refazendo o escalonamento para as tarefas restantes. Quando não é possível cumprir todos os *deadlines* realiza-se uma técnica denominada backtracking (pesquisa recursiva em um conjunto de dados para resolver determinado problema).
- **Leilão (*Bidding*):** tem como base o algoritmo do Míope, com a diferença que caso não seja possível atender aos *deadlines* (13), solicita para todas as máquinas do sistema qual o tamanho da folga existente para execução da tarefa, a máquina que possuir o maior valor (maior disponibilidade), recebe a tarefa para ser executada.
- **Leilão Focado (*Focussed Addressing Bidding*):** tem como base o algoritmo do Leilão (13), com a diferença de que a solicitação de oferta é feita para um determinado grupo de processadores pré-selecionados.

2.3 Simuladores de Tempo-Real

Na computação a simulação é amplamente usada e tem grande importância, por meio dela é possível antever o comportamento de sistemas e realizar testes de possíveis situações sem que haja a necessidade de implementação real, evitando gastos desnecessário e possíveis erros de projeto. O cenário em sistemas de tempo-real não é diferente, tendo simulação como algo indispensável, ainda mais quando trata-se de avaliação de sistemas críticos. Serão apresentados a seguir resumidamente, alguns dos simuladores existentes e suas principais funcionalidades, logo após será apresentado o RTsim, ferramenta foco deste trabalho.

2.3.1 SPARTS

O SPARTS (14) foi desenvolvido em Portugal tratando-se de uma ferramenta flexível para simulação, é composta por quatro módulos principais sendo eles:

- *Task Set Generator*: Esse componente tem como finalidade básica permitir que o usuário trabalhe com diferentes conjuntos de tarefas tendo como parâmetros o comportamento e as características do conjunto de tarefas.
- *Job Generator*: Sua finalidade é especificar os parâmetros individuais de cada tarefa do conjunto que foi inserido pelo usuário anteriormente.
- *Job Sequencer*: Recebe as tarefas do módulo citado anteriormente e as organiza para serem executadas de acordo com o seu tempo de execução. Sua saída é enviada para o módulo descrito a seguir.
- *Execution Environment*: Essa etapa é onde ocorrerá a execução da simulação, por meio de dois módulos que interagem entre si (*Hardware Model* e *Scheduler*), as tarefas serão escalonadas de acordo com o algoritmo escolhido.

Por possuir uma estrutura modular pode ser utilizada para diversos propósitos e seus módulos podem ser utilizados separadamente, além disso, o usuário pode escolher entre os algoritmos disponíveis de forma nativa.

2.3.2 STRESS

O STRESS (15) foi desenvolvido na Universidade de York por um grupo de pesquisa especializado em sistemas de tempo-real, este simulador foi desenvolvido para simular o comportamento de aplicações críticas em tempo-real. Possui uma linguagem desenvolvida especificamente para tornar possível o trabalho com a ferramenta, chamada de *STRESS Language*, possui utilização no meio acadêmico como também no setor industrial. Possui interface gráfica a qual permite a entrada de dados e características acerca do sistema a ser simulado. Para que o usuário

possa realizar uma simulação é necessário codificar as características utilizando a linguagem própria anteriormente citada.

2.3.3 SimSo

O SimSo (*Simulation of Multiprocessor Scheduling with Overheads*) trata-se de um simulador para ambientes multiprocessados desenvolvido na Universidade de Toulouse, na França (16). É uma ferramenta *open source*, sendo possível acessar toda documentação e respectivo código na página do projeto. Foi utilizada a linguagem de programação *Python* para o seu desenvolvimento por ser considerada pelos autores como fácil de ser aprendida e utilizada.

Para que a simulação seja realizada é necessário que o usuário codifique manualmente, ou seja, deve-se descrever o escalonador por meio de uma classe programada em *Python*. Possui interface gráfica construída com o intuito de ser intuitiva para as pessoas que a utilizarem e também com propósitos educacionais.

2.3.4 Realtss

Desenvolvido no Instituto de Tecnologia do México, a ferramenta Realtss (17) foi pensada para ser uma ferramenta aberta, podendo ser utilizada pela mais vasta gama de usuários. Possui estrutura modularizada a qual permite a inclusão de outros algoritmos na ferramenta, podendo utilizar-se linguagens como C, C++ e TCL, podendo ser executada em diversos sistemas operacionais.

Tem como objetivo testar as políticas de escalonamento existentes, sendo uma ferramenta de ensino e pesquisa, a simulação é realizada em três etapas, descritas a seguir:

- Definição dos Parâmetros: A interface desta etapa é bem clara e possui diversas opções, o que proporciona ao usuário indicar características das tarefas e escolha o algoritmo que será utilizado na simulação.
- Simulação: Após a conclusão da etapa anteriormente descrita é exibido o comportamento das tarefas por meio de gráficos.
- Análise dos Resultados obtidos: Nessa etapa é fornecido ao usuário informações como: as ocorrências mais importantes durante a simulação e dados de utilização.

Disponibiliza de um conjunto de escalonadores nativos e por possuir interface simples proporciona uma boa utilização e interação.

2.3.5 AURTSS

Diferentemente dos simuladores citados anteriormente, o AURTSS *AU Real Time Scheduler Simulator* é uma ferramenta exclusivamente para plataforma *Web* (18). Desenvolvido na

Índia, seu objetivo inicial era ensinar sistemas de tempo-real e também disponibilizar o uso para profissionais da área. Possui interface intuitiva, construída de modo que qualquer usuário consiga entender quais os parâmetros devem ser inseridos nos campos.

O simulador recebe os dados das tarefas como entrada, permite ativar/desativar preempção e escolhas de escalonamentos além da alocação de recursos para as tarefas. Permite somente o uso dos escalonadores anteriormente implementados na plataforma.

2.4 RTsim

O RTsim (*Real-Time simulator*) é um simulador para escalonamento de tarefas de tempo-real. (3) O projeto de desenvolvimento teve início em 1996, com objetivo de ser uma ferramenta de simulação que permitiria aos desenvolvedores de sistemas de tempo-real testarem seus algoritmos e verificar se eram adequados para a aplicação em questão. Após a codificação de seu protótipo observou-se uma nova aplicação para a mesma, o auxílio no ensino de sistemas de tempo-real, resultando em um novo foco de desenvolvimento.

2.4.1 Breve Histórico

Inicialmente foi implementado utilizando a linguagem C e possuía cinco algoritmos implementados para ambientes monoprocessados, após alguns anos foram adicionados mais cinco para sistemas multiprocessados (19). Possuía também interface gráfica para interação com o usuário, implementada utilizando a biblioteca *Xforms*.

Essa biblioteca atendia a maioria das necessidades, entretanto tornava a portabilidade um grande problema para o uso da ferramenta, pois dependia de gerenciadores de janelas que em sua maioria eram encontradas apenas em sistemas Linux.

Este problema tinha reflexo direto na quantidade de usuários que faziam uso da plataforma, dificultando sua difusão entre os simuladores existentes na época. Para sanar este contratempo a biblioteca *Xforms* foi retirada e foram integradas interfaces com o mesmo funcionamento das existentes anteriormente, contudo agora implementadas utilizando a linguagem Java.

Para que a integração fosse possível foi aplicada uma técnica de comunicação entre as duas linguagens as quais a plataforma havia sido desenvolvida (20), permitindo invocar uma classe Java a partir da linguagem C. Entretanto ainda havia um empecilho, o Java dependia do carregamento da JVM (Máquina Virtual Java) tendo que ser carregada a cada vez que uma janela era aberta.

Assim, foi proposto um novo projeto para reimplementação da ferramenta utilizando integralmente a linguagem Java. Esse processo eliminou os problemas de portabilidade e carre-

gamento anteriormente existentes, tornando a ferramenta apta para ser executada em diversos sistemas operacionais (21). A ferramenta se mantém até os dias atuais nesta mesma linguagem.

2.4.2 Estado Atual

O GSPD (Grupo de Sistemas Paralelos e Distribuídos) é o atual responsável por manter o RTsim e sua versão mais recente está disponível para *download* na página do grupo (4). Pode ser utilizado por profissionais que desenvolvem sistemas de tempo-real e por alunos de Graduação como ferramenta auxiliar no aprendizado.

Essa versão disponibiliza cinco algoritmos monoprocessados, sendo eles: Taxa Monotônica (*Rate Monotonic*), Servidor por Deferência (*Deferrable Server*), Servidor Esporádico (*Sporadic Server*), Troca de Prioridade (*Priority Exchange*), Topo de Prioridade (*Priority Ceiling*); além de três algoritmos para sistemas monoprocessados, sendo: Míope (*Myopic*), Leilão (*Bidding*) e Leilão Focado (*Focussed Addressing Bidding*); os quais estão disponíveis de forma nativa na ferramenta.

Utilizando a biblioteca de algoritmos nativos da ferramenta é possível realizar a sua execução de três maneiras distintas. Após a inserção dos dados de cada tarefa que será escalonada o usuário deverá escolher a forma desejada. As três formas serão descritas a seguir:

- **Análise:** Interface na qual apresentará as estatísticas sobre as ocorrências das tarefas, quantas perderam o deadline, entre outros dados.
- **Plotar Gráfico:** o resultado de saída será um gráfico detalhado sobre o escalonamento resultante, dessa forma, o usuário pode acompanhar as ocorrências das tarefas. Se utilizado um algoritmo multiprocessado, há a separação dos gráficos por processador, além de proporcionar outras opções.
- **Modo Aprendizado e Teste:** Utilizado no auxílio do ensino de sistemas de tempo-real, sendo possível que o usuário insira manualmente o escalonamento e receba a correção do mesmo. Possibilita a exibição dos erros cometidos ao tentar realizar o escalonamento e também fornece ajuda durante o processo de inserção no modo aprendizado.

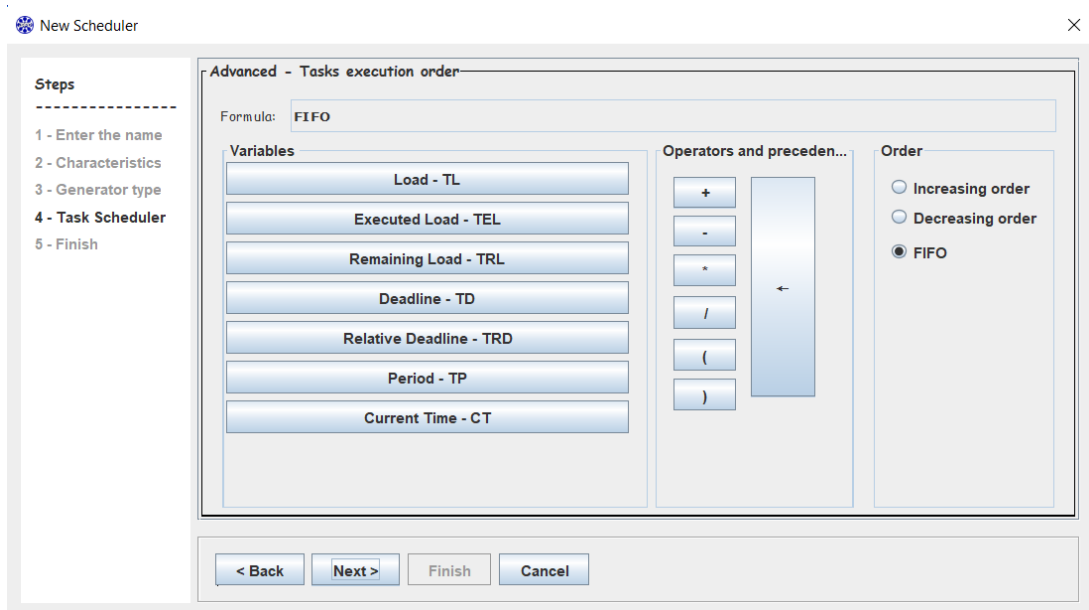
O uso do simulador como ferramenta auxiliar no aprendizado tem-se mostrado muito útil, pois possibilita aos alunos praticar e testar os algoritmos e conteúdos ministrados facilitando a compreensão do funcionamento dos escalonadores anteriormente vistos somente na teoria. A ferramenta produz automaticamente dois arquivos, sendo um com os dados das tarefas de entrada (pode ser carregado posteriormente no RTsim para utilizar as mesmas tarefas) e o rastro da simulação; após plotar o gráfico é possível salvá-lo como imagem para uso posterior.

Recentemente foi acrescentado ao simulador uma forma automatizada de inserção de novas políticas de escalonamento por meio de uma interface gráfica (22), o que torna dispensável

a programação manual ou acréscimo no pacote nativo de algoritmos disponíveis na ferramenta. Para que isso seja possível é necessário que o usuário descreva o comportamento desse escalonador de tarefas por meio de expressões matemáticas e que defina os parâmetros da política de alocação de tarefas.

Os parâmetros os quais precisam ser descritos para realizar a especificação matemática dos escalonadores são: Carga, Período, Carga Executada, Carga Restante, *Deadline*, *Deadline* Relativo e Tempo Atual, os quais já foram definidos em 2.2.1. A Figura 2.1 mostra a interface onde é possível realizar a descrição dos parâmetros citados anteriormente e criar as regras da política de escalonamento. Além disso, é possível escolher a ordenação das tarefas, se ocorrerá de forma crescente ou decrescente. Após a criação da política também há a possibilidade de definir os tipos de tarefas (periódicas ou aperiódicas) e no caso da existência de aperiódicas qual modelo será utilizado no atendimento dessas.

Figura 2.1 – Interface gráfica do módulo de inserção da fórmula de escalonamento.



Fonte: O autor (2019)

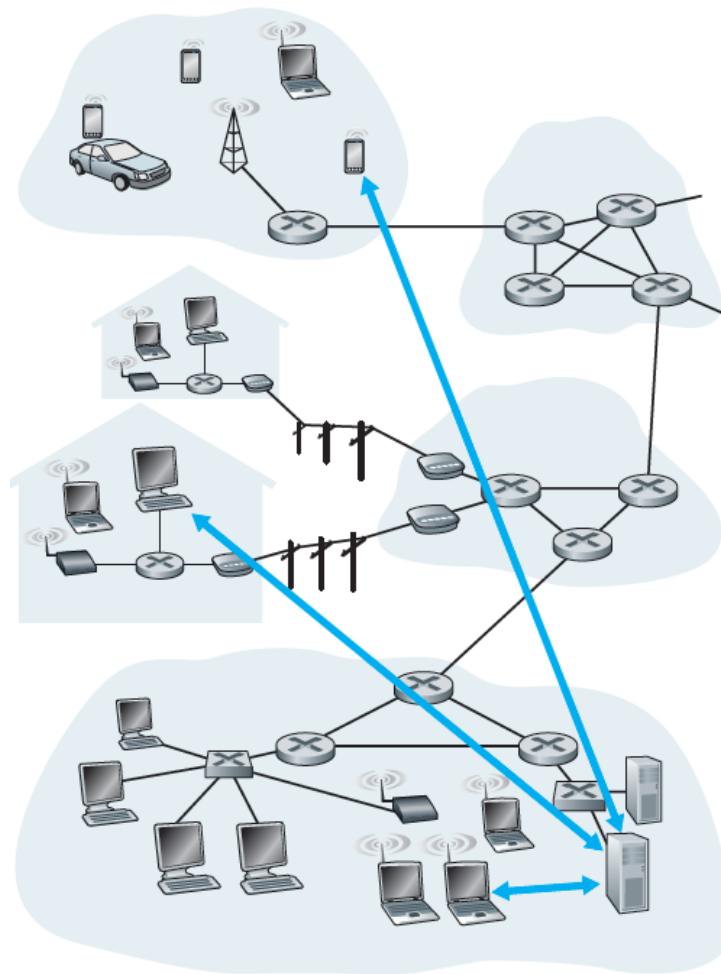
Após essa análise dos conceitos da área de sistemas de tempo-real e seus principais simuladores, faz necessário a formalização dos conceitos principais envolvendo a *Web* e as tecnologias envolvidas nas aplicações atuais. Assim, primeiramente iremos apresentar o funcionamento base da web, o qual faz uso da arquitetura cliente-servidor, descrita a seguir.

2.5 Arquitetura cliente-servidor

Trata-se de uma das arquiteturas mais utilizadas em aplicações modernas de rede. Nessa arquitetura há um hospedeiro sempre em funcionamento, denominado de servidor, o qual atende

as requisições realizadas por diversos outros hospedeiros, esses últimos são denominados de clientes. Tomando como exemplo uma aplicação da *Web*, temos um servidor *Web* o qual está em funcionamento 24 / 7 atendendo sempre as solicitações de navegadores de hospedeiros clientes. Ao receber a requisição de um objeto de um hospedeiro cliente, o servidor *web* responderá enviando o objeto solicitado (2). A seguir temos a 2.2 ilustrando a esquematização dessa arquitetura:

Figura 2.2 – Arquitetura cliente-servidor



Fonte: baseado em (2)

2.6 Aplicações Web

As aplicação *web* são projetadas para serem executadas em servidores web, conforme descrito anteriormente, e enviar sua saída pela Internet. Normalmente esse ambiente permite que vários clientes acessem o servidor simultaneamente, para recuperar dados ou para exibir/modificar dados. O cliente executa o mínimo de processamento de informações, essas informações são armazenadas em suma no servidor, residindo no cliente pequenas porções, chamados de

metadados. Um aplicativo da *web* pode ser visualizado não somente como sendo um cliente e servidor, mas também possuindo camadas, sendo elas:

- **Camada de Apresentação:** Responsável pela exibição e apresentação de informações do lado do cliente.
- **Camada Lógica:** Usada para transformar, consultar, editar e manipular as informações inseridas em formulários nos quais ela precisa ser armazenada ou apresentada.
- **Camada de Dados:** Responsável pela retenção de dados e informações para o aplicativo como um todo.

Todas essas camadas são dependentes da tecnologia trazida na forma da *World Wide Web*, sendo o HTML (*Hypertext Markup Language*) e o protocolo HTTP (*Hypertext Transfer Protocol*), tratando-se do principal protocolo usado para comunicação entre clientes e servidores. Além dessas duas tecnologias citadas nesse parágrafo faz-se uso de outras tecnologias como o CSS (*Cascading Style Sheets*) e o Javascript. Essas três tecnologias: HTML, CSS e Javascript são as principais linguagens para construção de páginas da *web*. A seguir temos a descrição básica do que cada uma delas é e qual seu propósito:

- **HTML:** trata-se de uma das linguagens que utilizamos para desenvolver *websites*, é baseada em marcação, sendo a tradução da sigla "Linguagem de Marcação de Hipertexto", foi padronizada pela primeira vez em 1993 e foi o combustível inicial da *World Wide Web*. É a partir dela que definimos como o conteúdo de uma página será apresentado, para isso é utilizado tags que aparecem entre < > (23).
- **CSS:** criada em 1996 pela W3C (*World Wide Web Consortium*) é uma linguagem de estilização a qual é usada para modificar o estilo de elementos escritos em HTML, sendo possível realizar diversas modificações, como: alterar cor do texto e do fundo, fontes, espaçamentos, etc. Como o HTML não foi projetado para possuir tags as quais ajudariam a formatar a página, apenas escrever a marcação para o site, o CSS surgiu para suprir essa demanda (24).
- **Javascript:** criada em 1995 é uma linguagem de programação interpretada estruturada, originalmente implementada como parte dos navegadores *web*. Tinha o intuito de executar *scripts* do lado do cliente, sem que ele necessitasse passar pelo servidor, porém com a evolução da Internet isso foi modificado e hoje possui diversas funcionalidades, podendo ser usada em ambos os lados da aplicação (*front-end e back-end*). É utilizada para manipulação do comportamento das páginas, por exemplo, ao passar com o mouse em um item do menu e aparecer na tela um submenu, o responsável por mostrar esse submenu é o *javascript*. Além disso, para tornar as páginas mais dinâmicas e com uma capacidade de

resposta mais rápida temos o uso conjunto do AJAX (*Asynchronous Javascript and XML*) (25).

- **AJAX:** é responsável por realizar a troca de pequenas quantidades de informações com o lado servidor nos bastidores da aplicação. Ao realizar essa ação acaba evitando que a página inteira tenha que ser recarregada cada vez que alguma nova informação necessite ser consultada no servidor acarretando em uma maior interatividade, velocidade de processamento e usabilidade da aplicação. Para realizar essa comunicação com o servidor, o AJAX pode utilizar várias tecnologias padronizadas e não padronizadas entre elas temos: XML (*Extensible Markup Language*), DOM (*Document Object Model*), JSON (*JavaScript Object Notation*), entre outras. Nesse trabalho será utilizada o JSON por ser um formato leve de intercâmbio de dados e amplamente utilizado nas aplicações modernas (25).

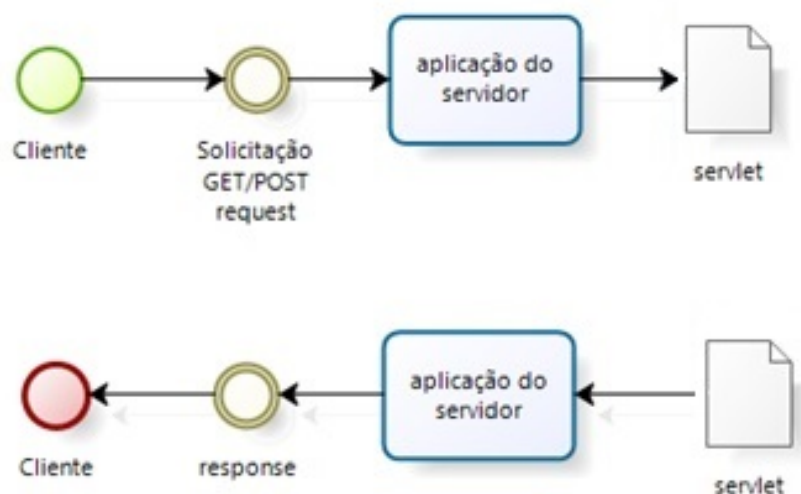
Com o aumento da popularidade da web entre os mais diversos usuários e o surgimento de conteúdos dinâmicos, diferente das aplicações clássicas de texto que se tornaram populares nas décadas de 1970 e 80 (2), viu-se um potencial de expansão de diversas tecnologias, entre elas o Java, dessa forma, iniciou-se o desenvolvimento do Java Enterprise Edition, o qual será melhor descrito a seguir.

2.7 Java Enterprise Edition

O Java EE é uma plataforma ou ambiente para desenvolvimento de aplicações de grande porte e aplicações *web* que possui bibliotecas e funcionalidades que implementam softwares baseados na linguagem Java, ela estende o Java SE, fornecendo uma *API* (*Application Programming Interface*) para mapeamento objeto-relacional, arquitetura multicamada e distribuídas, além dos *web services*. Sendo ela baseada amplamente em componentes modulares que são executadas em um servidor de aplicações. A plataforma JEE possui uma série de tecnologias as quais visam objetivos diferentes, entre elas temos: *servlets*, *java server pages* (*JSP*), *java server faces* (*JSF*), *Java Persistence API* (*JPA*), entre outras, sendo essas as mais conhecidas. Para desenvolver este trabalho utilizaremos as duas primeiras tecnologias, as quais serão descritas em maiores detalhes nas próximas subseções. (26)

2.7.1 *Servlets*

O nome "*servlet*" vem do inglês e dá a ideia de servidor pequeno cujo objetivo é receber as requisições HTTP, processá-las e dar uma resposta ao cliente. As *servlets*, são classes Java desenvolvidas em conformidade com uma estrutura bem definida, as quais são instaladas junto a um servidor de aplicações java (como por exemplo o *Apache Tomcat*), também conhecido como *servlet container*, o qual pode tratar as requisições recebidas de clientes (26). Para exemplificar seu funcionamento segue esquematização na Figura 2.3:

Figura 2.3 – Esquematisação do funcionamento de um *servlet*

Fonte: O autor (2019)

2.7.2 Apache Tomcat

Conforme visto anteriormente, para que os *servlets* e as JSPs desempenhem suas funções faz-se necessário um servidor web especial, sendo esse o *servlet container*, para isso foi escolhido o Apache Tomcat, aplicação que interpreta e processa as duas tecnologias citadas. Foi desenvolvido originalmente pela empresa *Sun* e seu código fonte foi entregue posteriormente à *Apache Software Foundation* e é distribuído como *software* livre, além de possuir suporte a diversas plataformas (27).

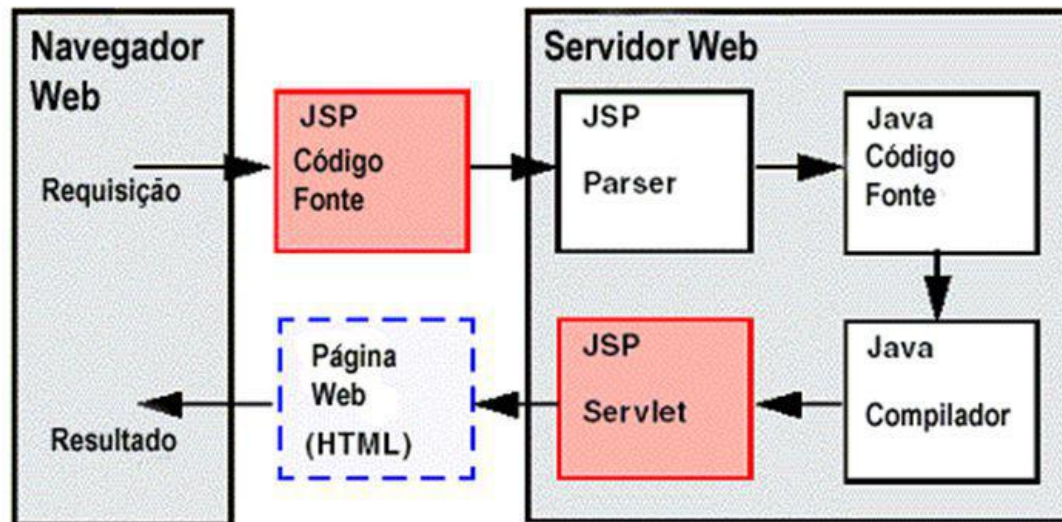
2.7.3 Java Server Pages - JSP

As páginas JSP foram criadas para sanar algumas limitações que surgiram durante o desenvolvimento utilizando somente as *servlets*, ao utilizar um *servlet* a formatação da página HTML se misturava com a lógica da aplicação em si, o que dificultava a alteração dessa formatação; já na JSP essa formatação foi totalmente separada da lógica de programação, o que possibilita a mudança da organização da página *web* sem afetar o restante da aplicação. Assim, uma *JSP* consiste em uma página HTML com alguns elementos especiais, conferindo à página um caráter dinâmico, esses elementos podem tanto realizar um processamento por si, como recuperar o resultado do processamento realizado por um *servlet*, por exemplo, e mostrar esse conteúdo dinâmico junto a página *JSP*.

Outro recurso adicional disponível durante o uso de páginas *JSP* é a recompilação automática, na qual qualquer alteração que seja feita no código é mostrada automaticamente na página *web*, não sendo necessário interromper o funcionamento da aplicação para incorporar as

modificações, como por exemplo, no *layout* da página (26). Para exemplificar o que foi descrito e como ocorre seu funcionamento segue esquematização na Figura 2.4:

Figura 2.4 – Esquematização do funcionamento de uma *Java Server Page*



Fonte: <https://slideplayer.com.br/slide/2263135/>

A utilização de *servlets* e *JSPs* oferece diversas vantagens em relação ao uso de outras tecnologias existentes, as principais vantagens são herdadas da própria linguagem Java como: a portabilidade, facilidade de programação, flexibilidade, além de benefícios adicionais como: escalabilidade, eficiência e recompilação automática, sendo o motivo da escolha realizada.

2.8 Considerações finais

Neste capítulo foram apresentados os conceitos gerais sobre sistemas de tempo-real, algoritmos de escalonamento, algumas ferramentas de simulação existentes e descrição do funcionamento básico do RTsim e seu estado atual. Além de apresentar a base do funcionamento de aplicações na web, qual arquitetura utilizada, como se organizam, tecnologias envolvidas nessas aplicações e exposição das tecnologias utilizadas. Dessa forma, foi possível obter conhecimento sobre o assunto tratado, os quais serão aplicados no decorrer da adaptação para funcionamento em plataforma web, conforme proposto por este trabalho.

3 Desenvolvimento

Este capítulo aborda as atividades desenvolvidas neste trabalho. Na seção 3.1 serão descritos os principais componentes da implementação da versão desktop do RTsim, fundamental para compreender como é o atual funcionamento, e na seção 3.2 serão descritos como foi realizada a readequação e organização da ferramenta *web*, já na seção 3.3 foram descritas as modificações realizadas envolvendo as interfaces gráficas do sistema e na seção 3.4 foram feitas as considerações finais.

3.1 Organização da ferramenta RTsim

Para realizar a adaptação da ferramenta e seus módulos para um ambiente *web* faz-se necessário o estudo aprofundado de sua organização e como foi realizado seu desenvolvimento, para isso, é necessário compreender como a hierarquia de classes é organizada atualmente. Sua estrutura básica possui classes separadas para cada algoritmo de escalonamento implementado de forma nativa, além disso há a separação entre classes de interface de entrada, interface de saída e de simulação do escalonamento.

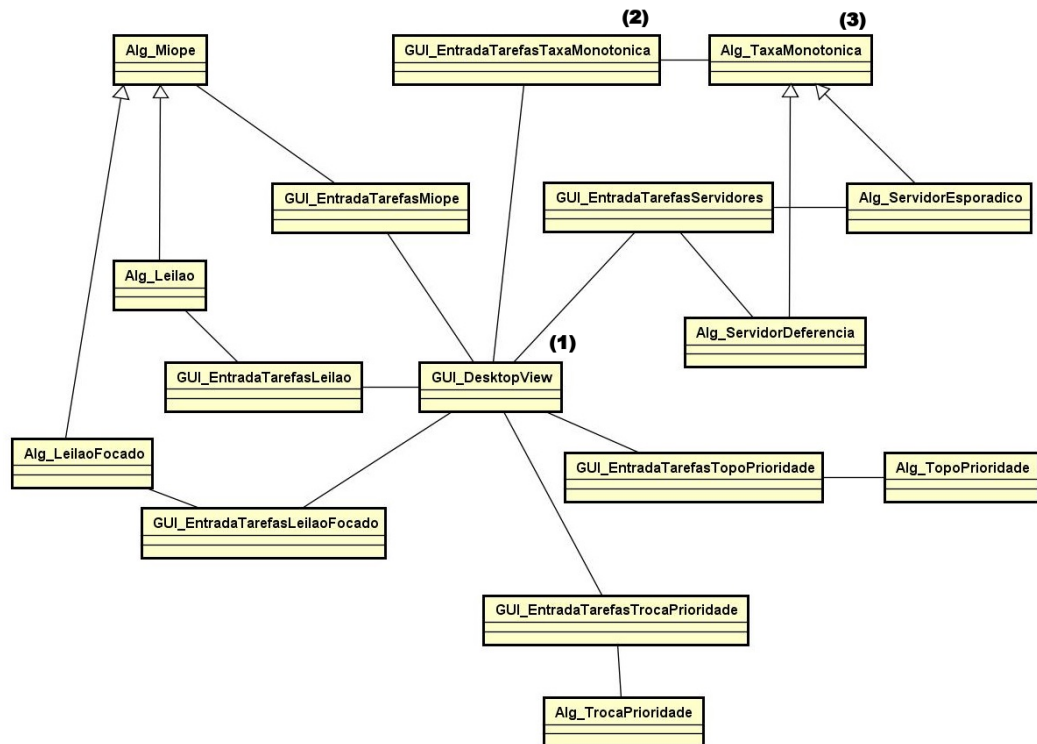
A interface inicial da ferramenta é gerada pela classe principal, a *GUI_DesktopView* e a partir das escolhas do usuário é realizada a chamada de novas classes. Além disso, existem classes específicas para a exibição de gráficos monoprocessados e multiprocessados e a geração das estatísticas.

Os métodos para "Modo Teste" e "Modo Aprendizado" são implementados separadamente tendo cada um deles suas respectivas classes de interface e tratamento. Independente da escolha do tipo de saída que o usuário optar a classe responsável pela execução do algoritmo é a mesma e será sempre executada, gerando assim os dados que serão usados pelas outras classes.

Para cada algoritmo existem duas classes de execução principais, para ilustrar melhor será primeiramente mostrado o diagrama de classes exibindo essas interações, dando uma visão geral do sistema.

Após isso será dado um enfoque ilustrando a parte do diagrama que será utilizado para descrever o comportamento de uma possível simulação. Segue a Figura 3.1 com o diagrama da visão geral das classes:

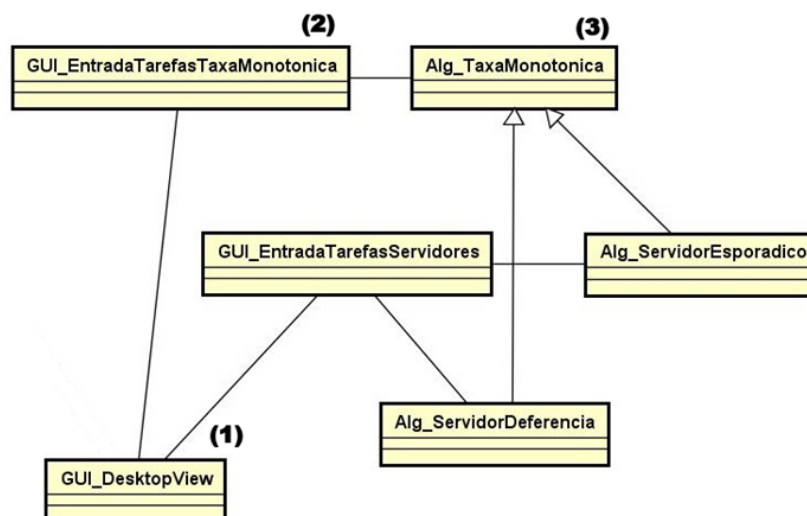
Figura 3.1 – Diagrama de classe exibindo as classes de interface e simulação dos algoritmos



Fonte: O autor (2019)

A seguir temos a Figura 3.2 com o enfoque da Figura 3.1 para melhor visualização, os números entre parenteses da imagem serão utilizados no exemplo.

Figura 3.2 – Enfoque no diagrama de classe exibindo as classes chamadas no exemplo de simulação



Fonte: O autor (2019)

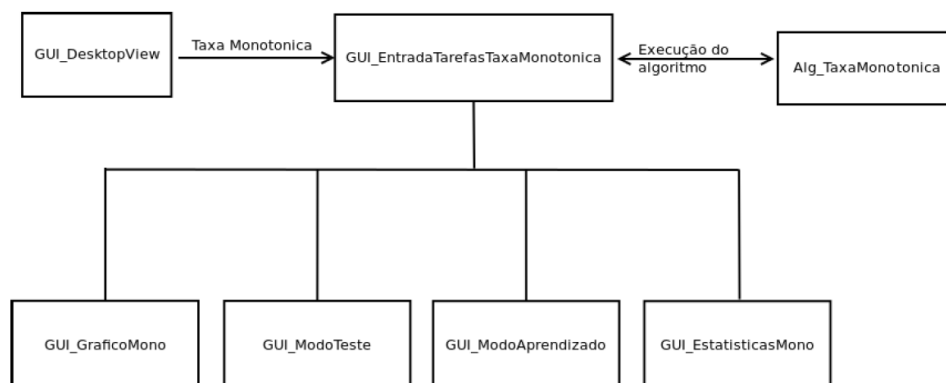
Por exemplo, caso o usuário queira simular o comportamento de determinadas tarefas utilizando o algoritmo Taxa Monotônica ocorrerá a seguinte ordem de chamada de classes:

1. A execução da classe inicial é feita pela classe *GUI_DesktopView* (1), que realiza uma função a partir da seleção do usuário realizada no menu de opções.
2. O usuário seleciona no menu de monoprocessados o algoritmo Taxa Monotônica, e o programa realizará a chamada da classe *GUI_EntradaTarefasTaxaMonotonica* (2). Esta classe é responsável por exibir a janela de entrada do conjunto de tarefas.
3. Quando o usuário entra com os dados das tarefas e solicita o início do escalonamento é feita a chamada da classe principal, a *Alg_TaxaMonotonica* (3), que realiza o escalonamento e gera os dados necessários para as próximas classes.

Além disso, dependendo de qual dentre as quatro formas possíveis de exibição do escalonamento o usuário selecionou, será realizada uma chamada diferente, sendo elas descritas abaixo e ilustradas na Figura 3.3:

- Caso tenha escolhido "Plotar Gráfico", será chamada a classe responsável por desenhar o diagrama de escalonamento, a *GUI_GraficoMono*.
- Caso o usuário escolha o "Modo Teste" ou "Modo Aprendizado", serão chamadas as classes *GUI_ModoTeste* ou *GUI_ModoAprendizado*.
- Se a opção selecionada for para apresentar as estatísticas do escalonamento, a chamada da classe *GUI_EstatisticaMono* ocorrerá.

Figura 3.3 – Chamadas de classe de execução para chamada de escalonamento



Fonte: O autor (2019)

O conjunto de classes que implementam os algoritmos nativos possuem métodos semelhantes, porém implementados de forma diferente por pertencerem a diferentes algoritmos,

cada um possuindo sua particularidade. Os métodos são executados de modo que simulem o escalonamento e produzam resultados para as outras classes.

O método *escalonar()* é o principal utilizado no escalonamento, sendo implementado recursivamente para facilitar a codificação. Este método simula todo o comportamento do escalonador, verificando as ordens de execução, quando necessário realiza-se preempção e é capaz de verificar se poderá executar a tarefa, calcular o tempo de folga, entre outras funcionalidades.

Outros métodos fazem tratamento dos dados que irão passar pelo escalonador, por exemplo, o método que ordena as tarefas por prioridade ou o que determina os tempos de folga do processador, dentre outros.

As classes de interface são mais complexas pela quantidade de itens que elas manipulam e pela quantidade de ações que elas possuem para cada evento. A cada item que será mostrado na tela, como por exemplo um botão, ocorre o tratamento e criação separadamente, sendo unido aos outros e organizado na tela por um outro método da classe.

Do ponto de vista funcional, a manipulação de arquivos de entrada e saída (com dados das tarefas) é realizada pelas interfaces de entrada, sendo responsáveis também por toda a interação com o usuário e validação dos dados de entrada, permitindo o tratamento de erros.

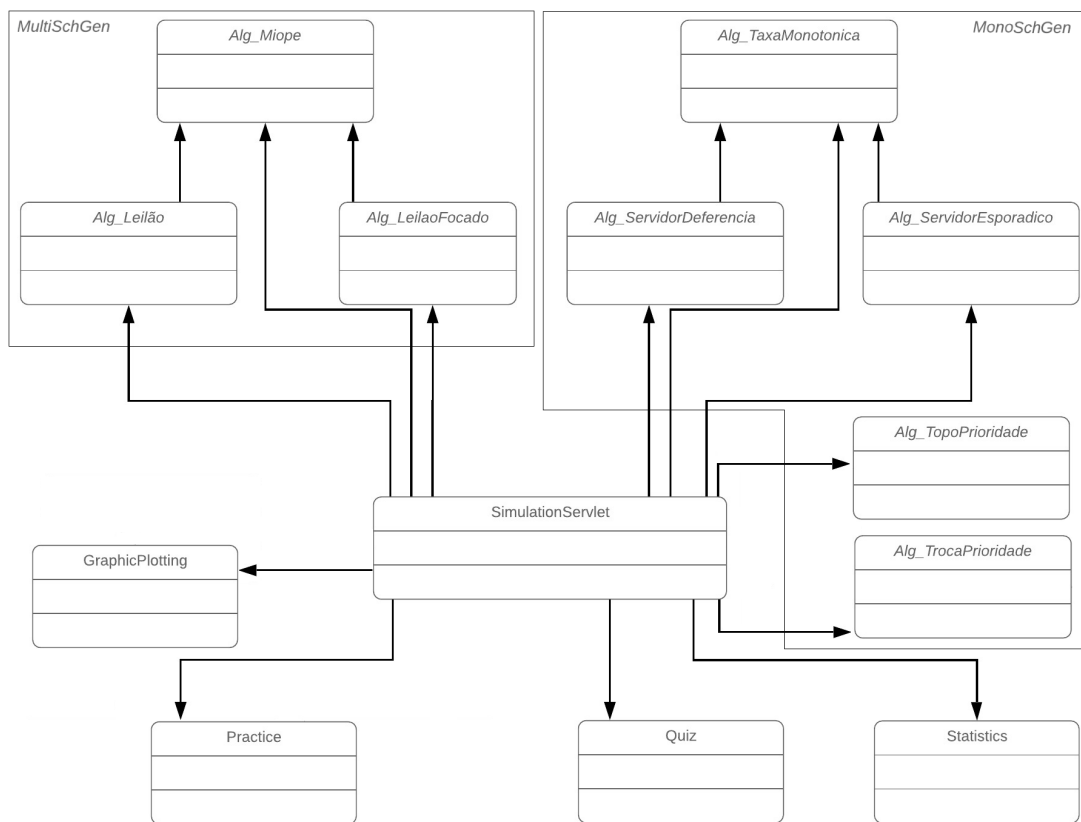
Por ser tratar de uma ferramenta desktop totalmente codificada em Java Standard, todas essas interfaces anteriormente descritas foram desenvolvidas utilizando dois tipos de bibliotecas gráficas dessa linguagem, a *AWT* e a *Swing*, as quais foram desenvolvidas para uso em aplicações desktop, além de todo o processamento das informações ocorrerem na máquina do usuário, não sendo necessário acesso à Internet.

Dessa forma, para migrar a ferramenta para versão *web* é necessário adequar essas interfaces gráficas com tecnologias compatíveis com os navegadores conforme descrito na seção 2.6, além de hospedar a página desenvolvida em um servidor, o qual fará todo o processamento dos dados e devolverá a resposta para o cliente, citado na seção 2.7.2. Assim, na seção a seguir será descrita a organização do *web-RTsim*.

3.2 Organização do *Web-RTsim*

A aplicação foi readequada de tal forma que as classes principais da ferramenta *desktop*, que são: algoritmos de escalonamento e as opções de simulação descritas em 2.4.2, foram aproveitados nessa versão *web*. Entretanto, essas classes são utilizadas de forma diferente da versão anterior, no caso, são utilizadas pelo *servlet* de simulação. A Figura 3.4 ilustra o diagrama de classes de forma resumida:

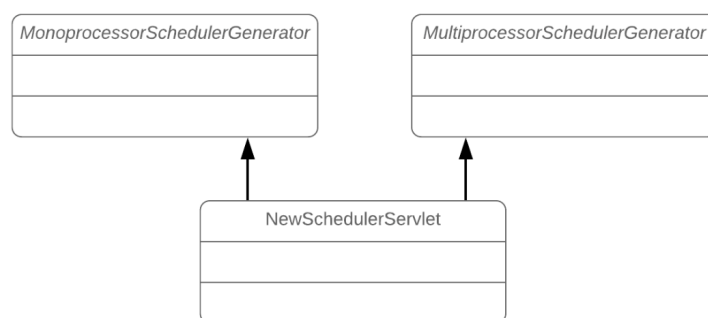
Figura 3.4 – Diagrama de classe exibindo a interação entre o *servlet* de simulação e as classes principais do RTsim.



Fonte: O autor (2019)

A aplicação possuirá um segundo *servlet* o qual terá como objetivo gerenciar a opção de geração de escalonadores que permitem a criação de políticas de escalonamento por meio de expressões matemáticas, ilustrado na figura 2.1. A Figura 3.5 ilustra o diagrama de classes resumidamente:

Figura 3.5 – Diagrama de classe exibindo a interação entre o *servlet* de criação de escalonadores do RTsim.



Fonte: O autor (2019)

3.2.1 O *Servlet* de simulação

O principal objetivo deste *servlet* é responder a requisição HTTP do cliente com a resposta do processamento da simulação de acordo com a opção de simulação selecionada, assim, o navegador exibirá os dados recebidos. A seguir será explicado de forma mais detalhada como esse processo ocorre.

1. A requisição é composta pelo algoritmo de escalonamento, pelo conjunto de tarefas, pela opção de simulação e eventualmente por semáforos.
2. O *servlet* de simulação irá interpretar esses dados da requisição, selecionará o algoritmo solicitado, realizará o escalonamento das tarefas, verificará qual a opção de simulação selecionada.
3. Os resultados do processamento realizado pelo *servlet* serão enviados como resposta para o cliente.

3.2.2 O *servlet* de criação de escalonadores

A principal funcionalidade desse *servlet* é gerar um escalonador como uma classe Java que posteriormente pode ser usada no escalonamento de tarefas, levando em conta todas as informações cedidas pelo usuário durante a criação do mesmo. A seguir será explicado como esse processo ocorre:

1. A requisição é composta por tipo de escalonador (monoprocessado ou multiprocessado), as características do escalonador (prioridade estática ou dinâmica e como tratar as dinâmicas) e a ordem de execução das tarefas (expressão matemática).
2. O *servlet* irá interpretar esses dados da requisição, em seguida realizará a criação do escalonador com base nas características recebidas.
3. Esse escalonador gerado será escrito em um arquivo do tipo ".java" e devolvido como a resposta para o cliente.

3.2.3 JSPs na criação de páginas

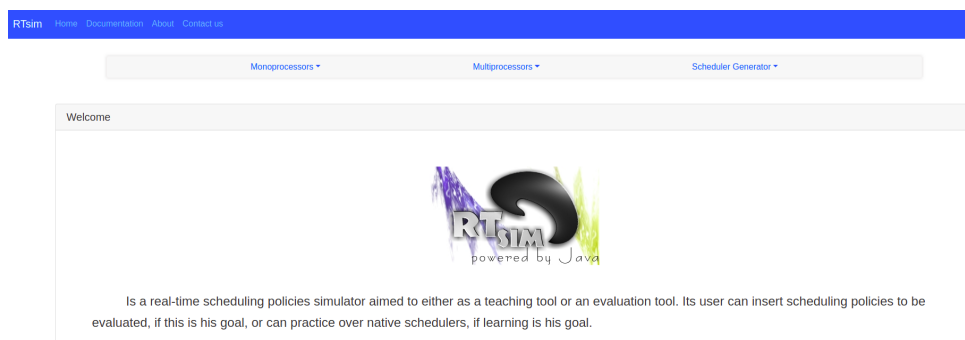
As *JSPs* foram utilizadas na aplicação com o intuito de separar a formatação das páginas, gerar conteúdo dinâmico, além de utilizar informações geradas pelos *servlets*. Além de sanar os problemas e agregar as vantagens descritas na seção 2.7.3.

Nas sub-seções 3.2.1, 3.2.2 e 3.2.3 foram descritas parte da aplicação que atua no lado servidor, a seguir será apresentado a parte da aplicação que atua no lado cliente, composta por: *HTML*, *CSS* e *Javascript*.

3.3 Interface gráfica

A mudança mais drástica realizada nesse trabalho ocorreu envolvendo as interfaces gráficas do RTsim, na versão *desktop* além de formalizar o layout da ferramenta, ela possuía diversas funções como: tratar todos os eventos realizados pelo usuário, verificar possíveis erros de entrada dos dados, entre outras. Nessa versão *web* as interfaces foram desenvolvidas utilizando as linguagens *HTML* e *CSS*, a primeira definindo como o conteúdo da página será exibido e a segunda realizando estilizações para tornar a página mais elegante, funcional e fácil de utilizar. Abaixo temos a Figura 3.6 para demonstrar o *layout* da página inicial do *Web-RTsim*:

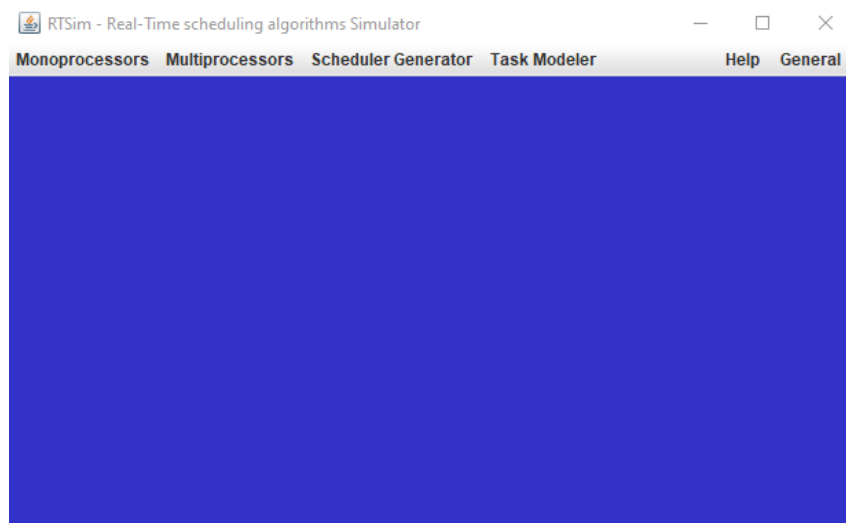
Figura 3.6 – Layout da página inicial do *Web-RTsim*.



Fonte: O autor (2019)

Para realizar uma comparação com a versão desktop, segue a Figura 3.7 com a página inicial do RTsim:

Figura 3.7 – Layout da tela inicial do RTsim.



Fonte: O autor (2019)

3.3.1 Javascript no tratamento de eventos e requisições AJAX

Conforme dito na seção ?? o tratamento de eventos na versão *desktop* era realizada pela própria interface gráfica da ferramenta, o que sanava toda a demanda necessária e possuía ótimo funcionamento, entretanto, nas páginas da *web* o responsável por tratar os eventos, como por exemplo, um clique em um botão de enviar, ou a exibição de um menu é a linguagem de programação Javascript. Além disso, para realizar o intercâmbio de dados com o servidor foi utilizado AJAX e JSON, combinando os dois para evitar o recarregamento de toda a página e ter maior interatividade. Abaixo temos um trecho de código exemplificando o uso dessas três tecnologias:

```
1      $(document).ready(function (event) {
2
3          const optionMap = {
4              "Monotonic Rate": 0,
5              "Deferrable Server": 1,
6              "Sporadic Server": 2,
7              "Priority Exchange": 3,
8              "Priority Ceiling": 4,
9              "Miopic": 5,
10             "Bidding": 6,
11             "Focused Addressing Bidding": 7,
12             "New Scheduler": 8,
13             "Use an existing scheduler": 9
14         }
15
16         $('.dropdown-item').click(function (event) {
17             let optionName = event.currentTarget.text
18             let dropdownOption = optionMap[optionName]
19             tasks = []
20
21             $.ajax({
22                 type: 'POST',
23                 url: '/RTsim_4.0/option',
24                 data: { "option": dropdownOption },
25                 success: function (data) {
26                     $('#form').empty()
27                     $('#form').append(data)
28                     $('#card-title').text('Simulation - ' +
29                                     optionName)
29                 },
30                 error: function (data) {
31                     console.log('error: ', data);
```

```

32         }
33     } ) ;
34 } )
35 } )

```

Esse trecho de código é do arquivo *menu-bar-option.js* o qual realiza o tratamento de um evento realizado pelo usuário, sendo esse evento a opção que foi clicada na barra de menu, entre os algoritmos mono e multiprocessados, sendo que cada opção possui um número relacionado a ela, assim esse número é devolvido ao servidor o qual analisa-a e com base nisso devolve a resposta ao navegador do usuário com o formulário correspondente à simulação desejada, caso ocorra algum erro, retorna uma mensagem de aviso. Por exemplo, se o usuário clicar no algoritmo *Monotonic Rate* o servidor receberá o número 0 como informação, assim ele enviará o formulário Simulation Monotonic Rate, conforme a Figura 3.8:

Figura 3.8 – Layout da página de simulação do algoritmo Taxa Monotônica.

Fonte: O autor (2019)

3.4 Considerações finais

Ao longo deste capítulo foi apresentado as principais modificações realizadas na ferramenta RTsim para torná-la em uma página da web totalmente funcional, a apresentação procurou detalhar as tecnologias adotadas ao simulador e também alterações e adequações em classes anteriormente existentes, as quais eram intimamente ligadas à interface gráfica *desktop*.

4 Testes e Resultados

A validação da ferramenta *web* foi conduzida de modo a verificar se todas as novas interfaces gráficas se comportaram da mesma forma que na versão *desktop* do RTsim, para isso será feita a comparação entre as telas das duas versões da ferramenta. Além disso, para a validação dos resultados adotou-se o critério de gerar possíveis escalonamentos em ambas as versões e verificar se os resultados produzidos são compatíveis com o esperado. Nas próximas seções serão descritos esses testes, começando pela avaliação das novas interfaces gráficas.

4.1 Testes de interfaces gráficas

Para realizar a reformulação das interfaces gráficas da ferramenta de simulação, procurou-se manter o padrão utilizado na versão *desktop* para que usuários acostumados a utilizar a ferramenta não fossem tão impactados pela mudança de plataforma, dessa forma, nessa seção serão mostradas algumas das telas da duas versões da ferramenta, além de exemplificar algumas entradas de dados nos sistemas. Na Figura 4.1 iremos mostrar a interface do algoritmo Taxa Monotônica e algumas entradas de tarefas:

Figura 4.1 – Layout da página de simulação do algoritmo Taxa Monotônica e algumas tarefas inseridas no *Web-RTsim*

Monoprocessors ▾ Multiprocessors ▾ Sched

Simulation - Monotonic Rate

Task properties

Name

Load

Period

Arrival

Registry

(T1, L=100, P=20, Arr=0)
(T2, L=250, P=30, Arr=0)

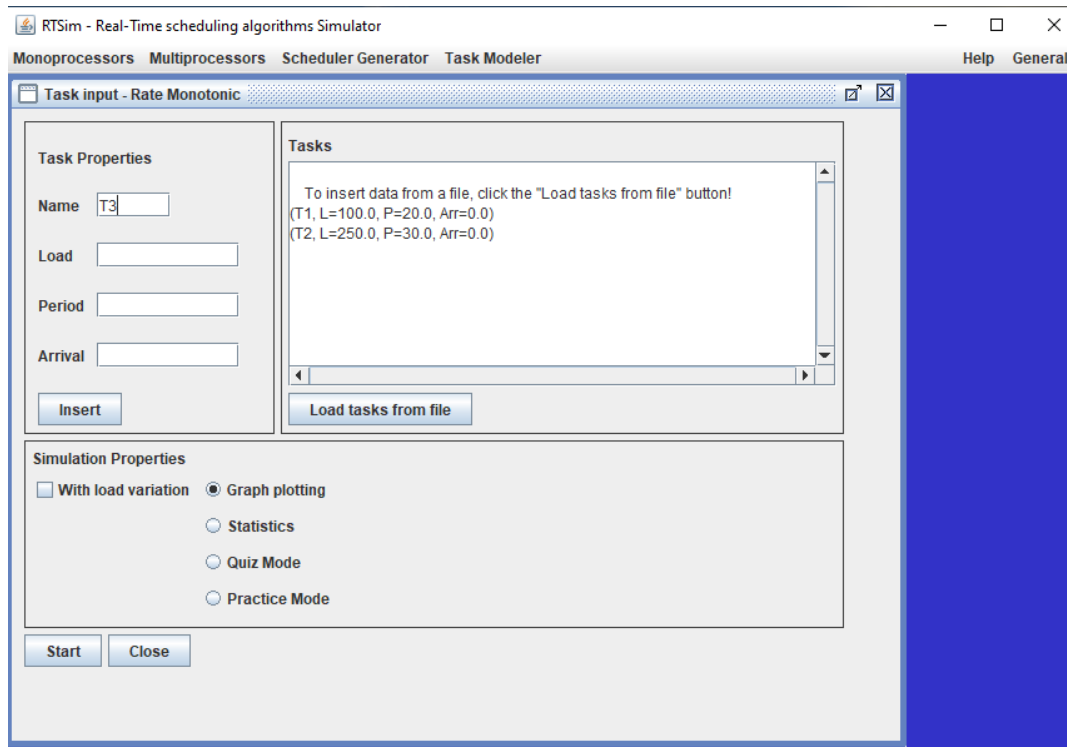
Simulation Options

☐ With load variation ☒ Graph plotting

Fonte: O autor (2019)

Agora na Figura 4.2 temos as mesmas tarefas sendo inseridas na versão *desktop* da ferramenta:

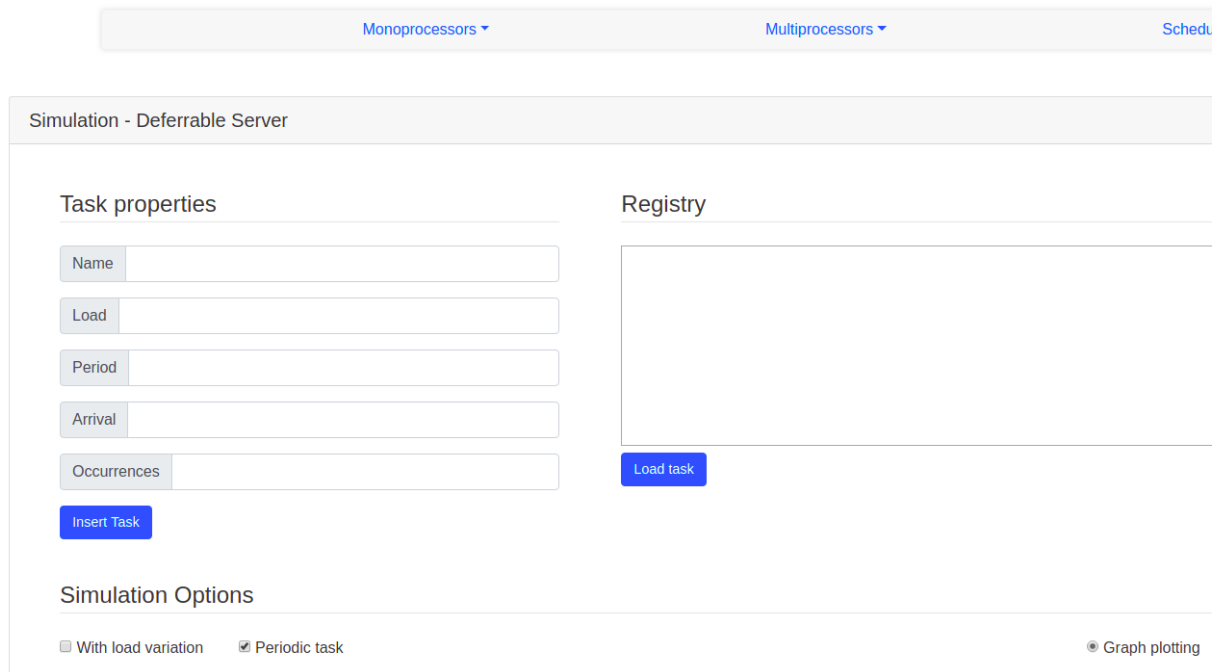
Figura 4.2 – Layout da tela de simulação do algoritmo Taxa Monotônica no RTsim e algumas tarefas inseridas



Fonte: O autor (2019)

Em todos os formulários da versão *web* foi implementado a verificação dos dados de entrada, na versão *desktop* as responsáveis por essa verificação eram algumas classes dentro da interface gráfica, na nova versão o responsável pela verificação desses dados é o próprio Javascript e AJAX, ao receber as informações inseridas no formulário no lado servidor, verifica-se por meio de uma lógica os dados de entrada e caso estejam incorretos, retorna um aviso para o usuário reinserir os dados corretamente.

Pelo fato do algoritmo Servidor por Deferência permitir tarefas aperiódicas e periódicas foi necessário adequar o formulário de entrada de dados, por padrão a opção *Periodic Task* fica selecionada, ao retirar essa seleção os campos de *Period* e *Arrival* são modificados para *Deadline* e *Min. Arrive* respectivamente. Segue abaixo as Figuras 4.3 e 4.4 ilustrando esse tratamento:

Figura 4.3 – Layout da página de simulação do algoritmo Servidor por Deferência no *Web-RTsim*


Monoprocessors ▾ Multiprocessors ▾ Scheduling ▾

Simulation - Deferrable Server

Task properties

Name

Load

Period

Arrival

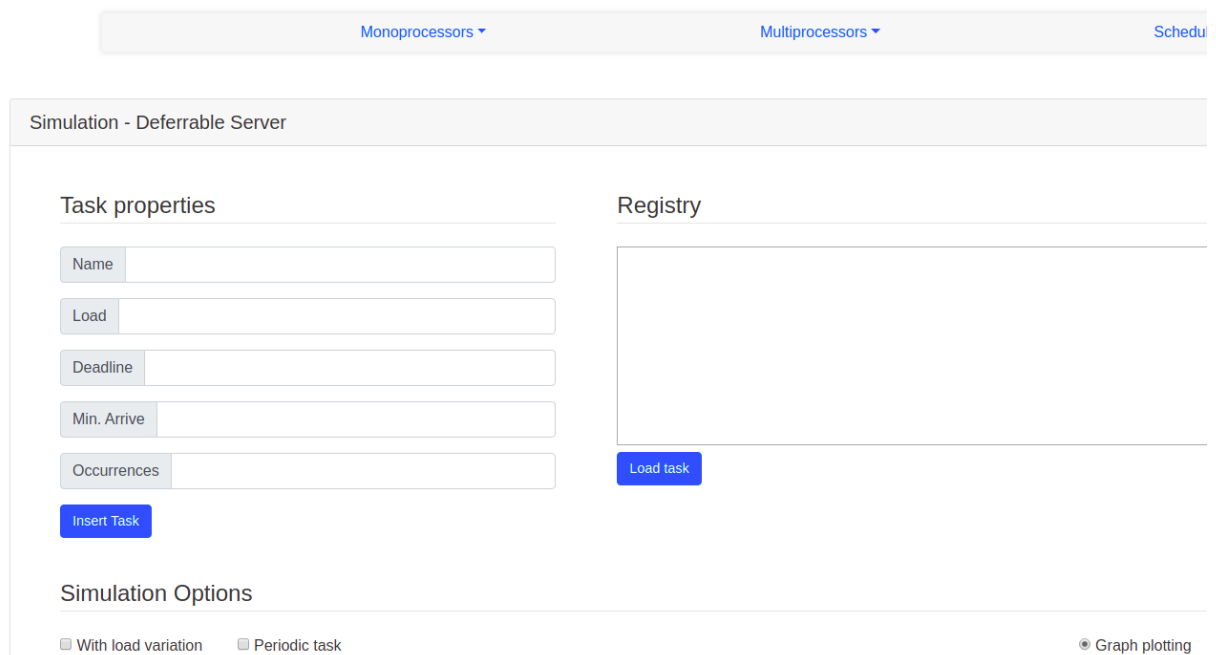
Occurrences

Registry

Simulation Options

☐ With load variation ☒ Periodic task ☒ Graph plotting

Fonte: O autor (2019)

Figura 4.4 – Layout da página de simulação do algoritmo Servidor por Deferência no *Web-RTsim* com tarefas aperiódicas


Monoprocessors ▾ Multiprocessors ▾ Scheduling ▾

Simulation - Deferrable Server

Task properties

Name

Load

Deadline

Min. Arrive

Occurrences

Registry

Simulation Options

☐ With load variation ☐ Periodic task ☒ Graph plotting

Fonte: O autor (2019)

Além disso, os algoritmos Troca de Prioridade e Topo de Prioridade utilizam semáforos

para controlar a entrada em regiões críticas, dessa forma, o formulário desses dois algoritmos são muito parecidos e com alguns campos a mais, segue a Figura 4.5 que ilustra essa página com algumas tarefas e um semáforo inserido:

Figura 4.5 – Layout da página de simulação do algoritmo Troca de Prioridade no *Web-RTsim* com semáforo

The screenshot shows the 'Simulation - Priority Exchange' interface. At the top, there are tabs for 'Monoprocessors', 'Multiprocessors', and 'Scheduler'. The main area is divided into several sections:

- Task properties:** Includes input fields for Name, Load, Period, Arrival, and Priority, along with an 'Insert Task' button.
- Registry:** Displays a list of tasks: (T1, L=150, P=15, Arr=0, Pr=1), (T2, L=260, P=30, Arr=0, Pr=0), and (T3, L=80, P=10, Arr=20, Pr=2). It also shows 'Semaphore added S1'. A 'Load task' button is present.
- Tasks Semaphore:** Includes dropdowns for 'Select a task' and 'Select a semaphore', and input fields for 'Critical section load' and 'Time of CS entrance'.
- Semaphores:** Includes an input field for Name and an 'Insert semaphore' button.
- Simulation Options:** Includes radio buttons for 'Graph plotting' (selected), 'Statistics', 'Quiz mode', and 'Practice mode'.

Fonte: O autor (2019)

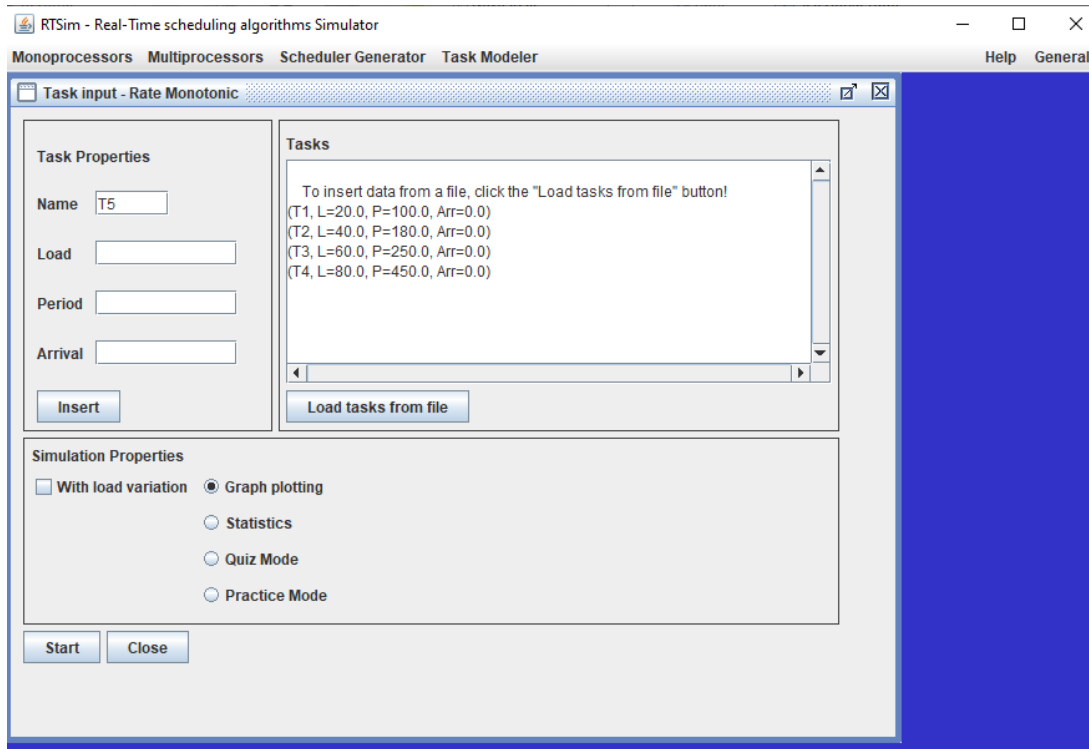
4.2 Geração de um escalonamento utilizando o RT-sim e o *web-RTsim*

Nessa seção será exibido um passo a passo para a realização do escalonamento de quatro tarefas utilizando o algoritmo Taxa Monotônica nas duas versões da ferramenta, apresentando os resultados obtidos na opção de "Plotar gráfico" e "Estatísticas" e ao final dessa exposição será analisado o comportamento dos resultados obtidos.

4.2.1 Escalonamento realizado no RT-sim

Primeiramente será realizado o escalonamento das quatro tarefas em questão utilizando a versão *desktop* da ferramenta, assim, segue a Figura 4.6 da inserção dessas tarefas:

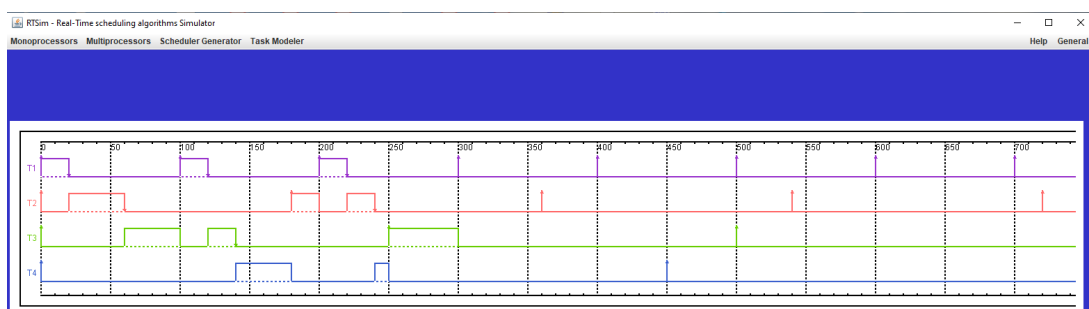
Figura 4.6 – Inserção das tarefas no algoritmo Taxa Monotônica no RTsim



Fonte: O autor (2019)

Após a inserção das tarefas, foi selecionado a opção de "Plotar Gráfico", na Figura 4.7 tem-se o gráfico resultante após a realização do escalonamento:

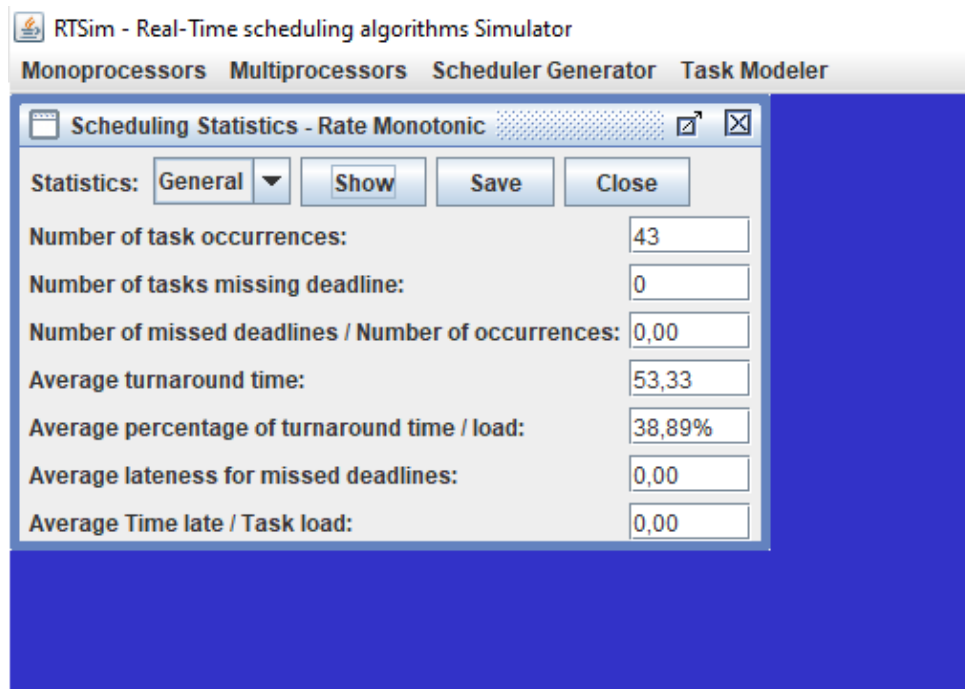
Figura 4.7 – Gráfico do esalonamento obtido pelo algoritmo Taxa Monotônica no RTsim



Fonte: O autor (2019)

Então foi solicitado a exibição dos dados estatísticos gerais do escalonamento realizado, mostra-se na Figura 4.8 a seguir:

Figura 4.8 – Exibição das informações estatísticas do escalonamento obtido pelo algoritmo Taxa Monotônica no RTsim



Fonte: O autor (2019)

4.2.2 Escalonamento realizado no *Web-RTsim*

Serão utilizadas as mesmas quatro tarefas utilizadas no escalonamento mostrado na subseção 4.2.1 para que seja possível comparar os dois resultados obtidos e realizar a análise de consistência e corretude. Assim, segue a Figura 4.9 com a inserção dos dados na página:

Figura 4.9 – Inserção das tarefas no algoritmo Taxa Monotônica no *Web-RTsim*

The screenshot shows the 'Web-RTsim' interface. The 'Simulation - Rate Monotonic' section is active. It includes a 'Task properties' form with fields for Name, Load, Period, and Arrival, and an 'Insert Task' button. The 'Registry' section displays a list of tasks: (T1, L=20, P=100, Arr=0), (T2, L=40, P=180, Arr=0), (T3, L=60, P=250, Arr=0), and (T4, L=80, P=450, Arr=0), with a 'Load task' button. The 'Simulation Options' section at the bottom has a checkbox for 'With load variation' and a radio button for 'Graph plotting'.

Fonte: O autor (2019)

Como pode ser observado na figura 4.10 foi solicitado a forma gráfica de exibição, segue abaixo a figura com o resultado obtido:

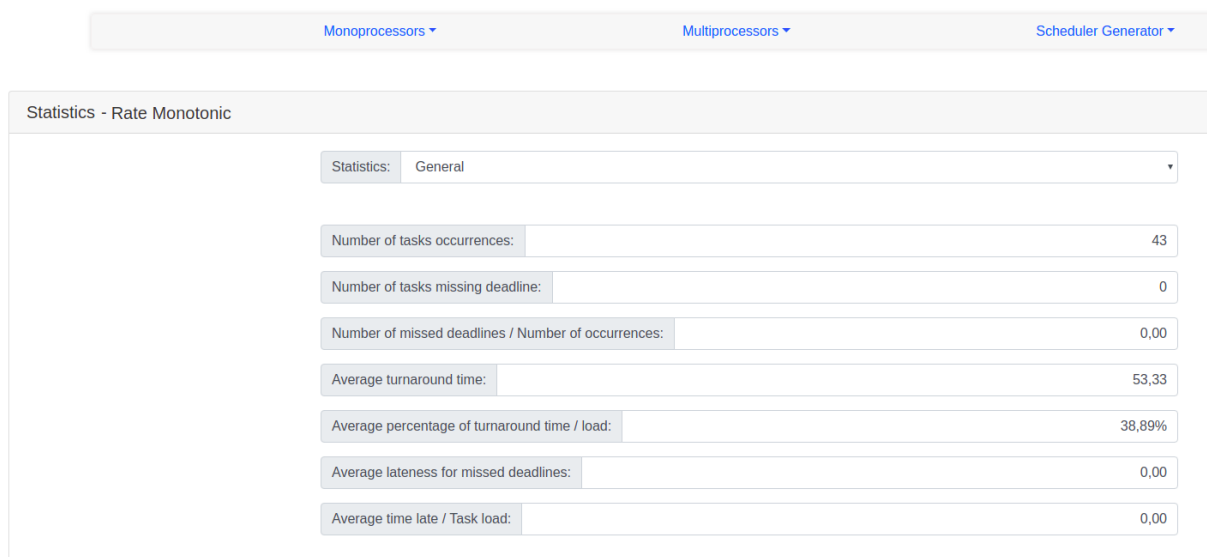
Figura 4.10 – Gráfico do escalonamento obtido pelo algoritmo Taxa Monotônica no *Web-RTsim*



Fonte: O autor (2019)

Após isso, foi solicitado a exibição dos dados estatísticos do escalonamento obtido pela aplicação, assim temos a Figura 4.11:

Figura 4.11 – Gráfico do escalonamento obtido pelo algoritmo Taxa Monotônica no *Web-RTsim*



Fonte: O autor (2019)

Comparando as estatísticas e gráficos obtidos pelas duas ferramentas chega-se a conclusão de que ambas as ferramentas estão se comportando da maneira esperada descrita pela literatura quanto ao algoritmo Taxa Monotônica, dessa forma, pode-se concluir que a aplicação *web* funciona de forma correta, proporcionando dados consistentes.

4.3 Considerações finais

Neste capítulo foram exibidos o funcionamento da ferramenta, com alguns exemplos, algumas interações possíveis com as novas interfaces gráficas, além de testes de funcionalidades da aplicação necessários para validar o funcionamento pleno da versão *web* do RTsim. Assim, foi possível perceber que os objetivos almejados no início desse texto foram alcançados e a nova aplicação apresentou um funcionamento da maneira esperada e correta.

5 Conclusões e Trabalhos Futuros

5.1 Conclusões

Diante do que foi exposto nos capítulos iniciais deste trabalho foi possível perceber a importância de atividades práticas no estudo de sistemas de tempo-real, o RTsim em sua versão anterior a este projeto, permitia ao usuário apenas o uso *offline* da ferramenta de simulação, além disso a máquina do usuário em questão precisaria ter uma versão Java instalada para que pudesse rodar o arquivo do tipo jar e realizar os escalonamentos desejados. O desempenho e o tempo de resposta para a simulação dependia intrinsecamente do *hardware* o qual esse usuário dispunha, podendo refletir no desempenho final obtido durante as simulações.

Com a aplicação *web* desenvolvida, todo o processamento da simulação é computado pelo servidor, o qual costumeiramente possui maior poder computacional do que computadores pessoais, sendo uma das vantagens obtidas nesse processo de migração. Para utilização do *Web-RTsim* o usuário necessitará de acesso a Internet além de um navegador, podendo acessar de diversos dispositivos dentre a gama disponíveis atualmente no mercado, não sendo mais necessário o download da ferramenta nem dependência de outros pré-requisitos de uso.

A aplicação passou por uma reestruturação de suas interfaces gráficas e para que não ocorresse uma mudança abrupta em relação a ferramenta *desktop*, esse desenvolvimento foi baseado no *layout* da versão anterior, de modo que o resultado fosse o mais próximo possível das interfaces do RTsim. Os resultados apresentados no Capítulo 4 permitem perceber que a ferramenta desenvolvida, mesmo após a reestruturação e modificação organizacional, obteve os resultados esperados, tendo consistência nos dados obtidos.

A maior dificuldade encontrada neste trabalho foi a compreensão e estudo de todas as tecnologias necessárias para o desenvolvimento de aplicações para a *web*, além do estudo do funcionamento da versão *desktop* da ferramenta de simulação, a qual possui muitas funcionalidades agregadas.

O principal objetivo desta versão foi facilitar e flexibilizar o uso da ferramenta de simulação em questão, tornando-o mais atrativo para os alunos, professores e desenvolvedores da área de tempo-real.

5.2 Trabalhos Futuros

O RTsim está agora com duas versões uma *desktop* e outra *web*, enquanto essa migração de plataforma ocorria, a versão *offline* recebia novas funcionalidades desenvolvidas por uma aluna de mestrado, dessa forma, não foi possível realizar a adaptação do módulo de interação

entre tarefas desenvolvido por ela, podendo ser realizado a adequação futuramente.

Além disso, pode-se acrescentar a implantação de interface gráfica na versão *web* para realizar a submissão do escalonamento gerado pelo usuário. O qual após análise e aprovação, seria incluso nos escalonadores disponíveis para uso.

Referências

- 1 LIU, J. W. S. *Real-Time Systems*. [S.l.]: Pearson Prentice Hall, 2000.
- 2 KUROSE, J. F.; ROSS, K. W. *Computer Networks A Top-Down Approach*. 6. ed. [S.l.]: Pearson Prentice Hall, 2014.
- 3 MANACERO, JR., A.; MIOLA, M.; NABUCO, V. Teaching real-time with a scheduler simulator. In: *Proc. of the 31st Frontiers in Education Conference*. [S.l.]: IEEE Press, 2001. p. T4D15–19.
- 4 GSPD. Grupo de pesquisa em sistemas paralelos e distribuídos. Página web disponível em <http://www.dcce.ibilce.unesp.br/spd>. 2019.
- 5 STANKOVIC, J. et al. *Real-Time Computin: A Critical Enabling Technology*. Amherst, MA, USA, 1994.
- 6 NISSANKE, N. *Real-Time Systems*. [S.l.]: Prentice Hall, 1997.
- 7 FARINES, J.-M.; FRAGA, J. S.; OLIVEIRA, R. S. *Sistemas de tempo real*. [S.l.]: Escola de computação, 2000.
- 8 LIU, C. L.; LAYLAND, J. W. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)*, v. 20, n. 1, p. 46–61, 1973.
- 9 STROSNIDER, J. K.; LEHOCZKY, J. P.; SHA, L. The deferrable server algorithm for enhanced aperiodic responsiveness in hard real-time environments. *IEEE Transactions on Computers*, v. 44, n. 1, p. 73–91, Jan 1995. ISSN 0018-9340.
- 10 SHA, L.; RAJKUMAR, R.; LEHOCZKY, J. P. Priority inheritance protocols: an approach to real-time synchronization. *IEEE Transactions on Computers*, v. 39, n. 9, p. 1175–1185, Sep. 1990. ISSN 0018-9340.
- 11 TANENBAUM, A. S.; BOS, H. *Modern Operating Systems*. 4th. ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2014. ISBN 013359162X, 9780133591620.
- 12 RAMAMRITHAM, K.; STANKOVIC, J. A.; SHIAH, P. . Efficient scheduling algorithms for real-time multiprocessor systems. *IEEE Transactions on Parallel and Distributed Systems*, v. 1, n. 2, p. 184–194, April 1990. ISSN 1045-9219.
- 13 STANKOVIC, J. A.; RAMAMRITHAM, K.; CHENG, S. Evaluation of a flexible task scheduling algorithm for distributed hard real-time systems. *IEEE Transactions on Computers*, C-34, n. 12, p. 1130–1143, Dec 1985. ISSN 0018-9340.
- 14 NIKOLIC, B.; AWAN, M. A.; PETTERS, S. M. Sparts: Simulator for power aware and real-time systems. In: *2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*. [S.l.: s.n.], 2011. p. 999–1004. ISSN 2324-898X.
- 15 AUDSLEY, N. C. et al. Stress: A simulator for hard real-time systems. *Software: Practice and Experience*, v. 24, n. 6, p. 543–564, 1994. Disponível em: <<https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.4380240603>>.

- 16 CHÉRAMY, M.; HLADIK, P.-E.; DÉPLANCHE, A.-M. SimSo: A Simulation Tool to Evaluate Real-Time Multiprocessor Scheduling Algorithms. In: *5th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*. Madrid, Spain: [s.n.], 2014. p. 6 p. Disponível em: <<https://hal.archives-ouvertes.fr/hal-01052651>>.
- 17 DIAZ, A.; BATISTA, R.; CASTRO, O. Realtss: a real-time scheduling simulator. p. 165–168, Sep. 2007.
- 18 YAASHUWANTH, C.; RAMESH, R. Web-enabled framework for real-time scheduler simulator (a teaching tool). In: *2010 Second International Conference on Computer Research and Development*. [S.l.: s.n.], 2010. p. 826–830.
- 19 KEHDY, R. M. *Simulação de algoritmos de escalonamento de tarefas de tempo-real em sistemas distribuídos*. [S.l.], 1999.
- 20 MIOLA, M. B. *Reengenharia aplicada ao rtsim*. [S.l.], 2001.
- 21 GONÇALVES, T. D. *RTsim 2.0 - Reengenharia e novas funcionalidades*. [S.l.], 2004.
- 22 PERONAGLIO, F. F. *Simulação de Tempo Real: Implementação do método para escalonamento manual na ferramenta RTsim*. [S.l.], 2017.
- 23 FULTON, S.; FULTON, J. *HTML5 Canvas*. 2th. ed. 1005 Gravenstein Highway North, Sebastopol, CA: O'Reilly Media, Inc., 2013. ISBN 9781449334987.
- 24 MCFARLAND, D. S. *CSS: The Missing Manual*. 4th. ed. 1005 Gravenstein Highway North, Sebastopol, CA: O'Reilly Media, Inc., 2015. ISBN 9781491918050.
- 25 ELLIOTT, E. *Programming JavaScript Applications*. 2th. ed. 1005 Gravenstein Highway North, Sebastopol, CA: O'Reilly Media, Inc., 2014. ISBN 9781491950296.
- 26 JENDROCK, E. e. a. *The Java EE 6 tutorial: basic concepts*. 4th. ed. 501 Boylston Street, Suite 900 Boston, MA 02116: Person Education, Inc., 2011. ISBN 9780137081851.
- 27 FOUNDATION, A. S. Introduction to apache tomcat 9. Página web disponível em <http://http://tomcat.apache.org/tomcat-9.0-doc/introduction.html>. 2019.