



# **CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS**

*Recredenciado pela Portaria Ministerial nº 1.162, de 13/10/16, D.O.U nº 198, de 14/10/2016*  
ASSOCIAÇÃO EDUCACIONAL LUTERANA DO BRASIL

Caio Henrique de Sousa

## IMPLEMENTAÇÃO DE AMBIENTE WEB PARA SIMULAÇÃO DE ESCALONAMENTO DE PROCESSOS

Palmas – TO

2017

Caio Henrique de Sousa  
IMPLEMENTAÇÃO DE AMBIENTE WEB PARA SIMULAÇÃO DE  
ESCALONAMENTO DE PROCESSOS

Projeto de Pesquisa elaborado e apresentado como requisito parcial para aprovação na disciplina de Trabalho de Conclusão de Curso II (TCC II) do curso de bacharel em Ciência da Computação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. M. Madianita Bogo Marioti.

Palmas – TO

2017

Caio Henrique de Sousa  
IMPLEMENTAÇÃO DE AMBIENTE WEB PARA SIMULAÇÃO DE  
ESCALONAMENTO DE PROCESSOS

Projeto de Pesquisa elaborado e apresentado como requisito parcial para aprovação na disciplina de Trabalho de Conclusão de Curso II (TCC II) do curso de bacharel em Ciência da Computação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. M. Madianita Bogo Marioti.

Aprovado em: \_\_\_\_/\_\_\_\_/\_\_\_\_

BANCA EXAMINADORA

---

Prof. Me. Madianita Bogo Marioti  
Centro Universitário Luterano de Palmas

---

Prof. Me. Fabiano Fagundes  
Centro Universitário Luterano de Palmas

---

Prof. Me. Jackson Gomes de Souza  
Centro Universitário Luterano de Palmas

Palmas – TO

2017

## **AGRADECIMENTOS**

Primeiramente a Deus por ter me dado essa oportunidade de ingressar na instituição maravilhosa, aos meus pais, irmãos e família que me deram total apoio e suporte, a minha namorada que sempre me apoiou.

A minha segunda mãe Madianita, por seus ensinamentos, seus puxões de orelhas, paciência e nunca desistiu da minha capacidade de estudo. Sem ela não teria conseguido sem seus ensinamentos.

Aos professores Jackson, Fabiano, Cristina, Parcilene, Fernando e Edeilson, que foram muito importantes no meu período acadêmico.

Aos meus amigos Dennis, Alisson, Robson, Murillo, Eugênio e aos demais que não foram citados, que sempre me ajudaram nos momentos difíceis da faculdade e que me deram total apoio. Muito obrigado!

## RESUMO

SOUSA, Caio Henrique de. **Implementação de ambiente web para simulação de escalonamento de processos**. 2017. 41 f. Trabalho de Conclusão do Curso de Ciência de Computação, Centro Universitário Luterano de Palmas, Palmas/TO, 2017.

Para o entendimento do funcionamento de um sistema operacional, é necessário entender o funcionamento básico das quatro áreas de gerência do SO: gerência de memória, gerência de processos, gerência de Entradas e Saídas e gerência de arquivos. Porém, é um conteúdo bastante complexo, pois o funcionamento do sistema operacional ocorre de maneira invisível para o usuário. Para solucionar o problema, existem ferramentas de simulação que apresentam o funcionamento dessas áreas de gerência de um SO. O presente trabalho teve como objetivo desenvolver uma ferramenta *web* que simule os algoritmos de escalonamento de processos da área de gerência de processos, com o intuito de facilitar o entendimento dos conceitos abordados dos algoritmos de escalonamento de processos.

**PALAVRA-CHAVE:** Algoritmos de escalonamento de processos, Simulador, Sistemas Operacionais.

## LISTA DE FIGURAS

Figura 1: Diagrama de Estados de um processo.....	15
Figura 2: Diagrama de tempo de uso da CPU (FIFO).....	18
Figura 3: Diagrama de tempo de uso da CPU (SJF) .....	20
Figura 4: Diagrama de tempo de uso da CPU (Prioridade).....	21
Figura 5: Diagrama de tempo de uso da CPU (Round Robin) .....	23
Figura 6: Tela inicial (I3S) .....	25
Figura 7: Tela de inclusão de processos (I3S).....	26
Figura 8: Tela de parâmetros globais (I3S) .....	26
Figura 9: Tela de simulação inicial (I3S) .....	27
Figura 10: Tela de simulação (I3S) .....	28
Figura 11: Tela inicial (PSSAV) .....	29
Figura 12: Tela de configuração (PSSAV).....	30
Figura 13: Tela de simulação inicial (PSSAV) .....	30
Figura 14: Tela da simulação sendo realizada (PSSAV).....	31
Figura 15: Tela de comparação (PSSAV) .....	32
Figura 16: Tela da média dos algoritmos (PSSAV) .....	32
Figura 17: Tela do diagrama de Gantt (PSSAV).....	33
Figura 18: Tela inicial (Simulador de escalonamento).....	34
Figura 19: Tela de inclusão de processo (Simulador de escalonamento).....	35
Figura 20: Tela final (Simulador de escalonamento) .....	35
Figura 21: Fases para o desenvolvimento do projeto. ....	38
Figura 22: Arquitetura de desenvolvimento da ferramenta .....	41
Figura 23: Diagrama de Classes .....	42
Figura 24: Tela inicial.....	43
Figura 25: Tabela de processos cadastrados (FIFO) .....	45
Figura 26: Simulação em execução (FIFO).....	46
Figura 27: Simulação finalizada (FIFO).....	47
Figura 28: Método de escalonamento FIFO .....	48
Figura 29: Método da tabela resultado .....	49
Figura 30: Tabela de processos cadastrados (SJF) .....	50
Figura 31: Simulação em execução (SJF) .....	51
Figura 32: Simulação finalizada (SJF) .....	52
Figura 33: Método de escalonamento SJF.....	53

Figura 34: Tabela de processos cadastrados (Prioridade) .....	54
Figura 35: Simulação em execução (Prioridade Não Preemptiva).....	55
Figura 36: Simulação Finalizada (Prioridade Não Preemptiva).....	56
Figura 37: Simulação em execução (Prioridade Preemptiva) .....	57
Figura 38: Simulação Finalizada (Prioridade Preemptiva).....	58
Figura 39: Método de escalonamento por Prioridade não preemptiva.....	59
Figura 40: Método de escalonamento Prioridade Preemptiva.....	60
Figura 41: Tabela de processos cadastrados (RR).....	61
Figura 42: Simulação em execução (RR) .....	62
Figura 43: Simulação Finalizada (RR) .....	63
Figura 44: Método de escalonamento RR .....	64

## LISTA DE TABELAS

Tabela 1: Fila de Aptos (FIFO) .....	17
Tabela 2: Fila de Aptos (SJF).....	19
Tabela 3: Fila de Aptos (Prioridade) .....	21
Tabela 4:Fila de Aptos (Round Robin) .....	23
Tabela 5: Paralelo entre as ferramentas de simulação .....	36



## **LISTA DE ABREVIATURAS E SIGLAS**

**CPU** – Central Processing Unit

**E/S** – Entradas e Saídas

**FIFO** – First In, First Out

**RR** – Round Robin

**SO** – Sistemas Operacionais

**SJF** – Shortest Job First

**TI** – Tecnologia da Informação

## SUMÁRIO

<b>1 INTRODUÇÃO .....</b>	<b>12</b>
<b>2 REFERENCIAL TEÓRICO .....</b>	<b>14</b>
2.1 SISTEMA OPERACIONAL .....	14
2.1.1 <i>Gerência de Processos</i> .....	15
2.2 ALGORITMOS DE ESCALONAMENTO DE PROCESSOS .....	16
2.2.1 <i>FIFO (First In, First Out - Primeiro a Entrar, Primeiro a Sair)</i> .....	17
2.2.2 <i>SJF (Shortest Job First - Menor Trabalho Primeiro)</i> .....	19
2.2.3 <i>Prioridade</i> .....	20
2.2.4 <i>RR - Round Robin</i> .....	22
2.3 FERRAMENTAS RELACIONADAS .....	24
2.3.1 <i>I3S</i> .....	25
2.3.2 <i>PSSAV (Process Scheduling Simulation, Analysis, and Visualization)</i> .....	29
2.3.3 <i>Simulador de escalonamento de processos</i> .....	34
2.4 PARALELO ENTRE AS FERRAMENTAS .....	36
<b>3 MATERIAIS E MÉTODOS.....</b>	<b>38</b>
3.1 METODOLOGIA .....	38
3.2 SOFTWARES .....	38
<b>4 RESULTADOS E DISCUSSÃO .....</b>	<b>40</b>
4.1 ARQUITETURA DE SOFTWARE .....	40
4.2 INTERFACE INICIAL DO SISTEMA .....	42
4.3 SIMULAÇÃO DO ALGORITMO DE ESCALONAMENTO <b>FIFO</b> .....	45
4.4 SIMULAÇÃO DO ALGORITMO DE ESCALONAMENTO <b>SJF</b> .....	49
4.5 SIMULAÇÃO DO ALGORITMO DE ESCALONAMENTO PRIORIDADE (NÃO-PREEMPTIVO/ PREEMPTIVO).....	53
4.6 SIMULAÇÃO DO ALGORITMO DE ESCALONAMENTO <i>ROUND ROBIN</i> .....	61
<b>5 CONSIDERAÇÕES FINAIS.....</b>	<b>65</b>
<b>6 REFERÊNCIAS .....</b>	<b>66</b>
<b>7 APÊNDICES .....</b>	<b>68</b>

<b>8 ANEXOS .....</b>	<b>69</b>
-----------------------	-----------

## 1 INTRODUÇÃO

Segundo Silberschatz, Galvin e Gagne (2001), “um sistema operacional (SO) é o programa que atua entre o usuário de um computador e o hardware do computador”, com a finalidade de proporcionar um ambiente no qual o usuário possa realizar a execução de programas de sua demanda, de forma eficiente e satisfatória, fazendo com que os programas do usuário não interfiram nos procedimentos ideais do sistema (SILBERSCHATZ; GALVIN; GAGNE, 2001). Assim, todo o computador é gerenciado pelo sistema operacional e faz com que o usuário não se preocupe com o gerenciamento dos dispositivos.

Em cursos de graduação área de Computação são ofertadas disciplinas voltadas aos estudos de SO, o que é uma recomendação das diretrizes curriculares atribuídas pelo MEC (MEC, 2012). Geralmente, os conteúdos abordados na disciplina de SO são divididos em quatro áreas: Gerenciamento de Processos, Gerenciamento de Memória, Gerenciamento de Entradas e Saídas e Gerenciamento de Arquivos.

Para compreender os conceitos associados a estas áreas, é necessário que o aluno tenha conhecimento do funcionamento de um SO como execução de programas e armazenamento de arquivos. Essas atividades realizadas pelo SO são invisíveis, logo, a dificuldade de entendimento das áreas de gerência é maior.

Entre os conceitos da área de Gerenciamento de Processos, são estudados os algoritmos de escalonamento de processos, que são responsáveis por definir quais processos ganharão a unidade central de processamento (CPU, do inglês *Central Processing Unit*). Relacionado a este tema, os alunos deverão compreender como é o funcionamento da CPU e os estados dos processos a cada momento.

Uma forma atraente de auxiliar no processo de ensino e aprendizagem dos algoritmos de escalonamento é apresentar de forma gráfica o funcionamento dos algoritmos, de maneira que fosse possível para o aluno a visualização dos conceitos apresentados em sala de aula.

Neste contexto, o presente trabalho consistia em implementar uma ferramenta *web* que realizasse a simulação dos algoritmos de escalonamento de processos, apresentando graficamente o funcionamento dos mesmos, de forma que os alunos consigam visualizar o funcionamento dos algoritmos de escalonamento apresentados em sala.

Existem diversas ferramentas de simulação que demonstram o funcionamento dos algoritmos de escalonamento de processos como I3S, PSSAV, etc. Algumas destas

ferramentas foram estudadas e realizado um levantamento dos principais aspectos técnicos, em que contribuiu para a implementação da nova ferramenta *web*.

O presente documento apresenta os conceitos básicos de sistemas operacionais, enfocando os algoritmos de escalonamento de processos, bem como as ferramentas de simulação I3S, PSSAV e Simulador de escalonamento de processos, e metodologias que foram utilizadas para o desenvolvimento do projeto e as simulações dos algoritmos de escalonamento da ferramenta.

## 2 REFERENCIAL TEÓRICO

Nesta seção, são abordados conceitos básicos relacionados aos sistemas operacionais, enfocando os algoritmos de escalonamento de processos. Também serão apresentadas as três ferramentas de escalonamento de processos encontradas na Internet e um paralelo entre as ferramentas a partir dos aspectos técnicos definidos.

### 2.1 Sistema Operacional

Um sistema operacional é um programa que realiza o gerenciamento do hardware do computador, com o objetivo de possibilitar sua utilização em um ambiente computacional, para que o usuário execute de forma eficiente os programas de sua demanda (SILBERSCHATZ; GALVIN; GAGNE, 2001). Ou seja, o SO é o programa que realiza o intermédio entre dispositivos e aplicações do usuário.

O Sistema Operacional é o programa essencial em um sistema de computação, pois sem ele, o usuário seria responsável por realizar o gerenciamento dos periféricos que seus programas precisassem. O Sistema Operacional é dividido em quatro áreas (SILBERSCHATZ; GALVIN; GAGNE, 2001):

- Gerência de Processos: gerência da CPU para alocações de programas à serem executados;
- Gerência de Memória: gerencia o atendimento das solicitações de liberação e alocação de memória, seu objetivo é garantir que os processos não conflitem uns com os outros e que se evite o desperdício do espaço de memória;
- Gerência de Dispositivos E/S: visa simplificar o desenvolvimento de aplicações, de forma que o SO gerencie os dispositivos de E/S, disponibilizando vários serviços aos processos. Esses serviços são associações de um dispositivo a um processo em específico, englobam leitura e escrita de um dispositivo, liberação e fechamento de um dispositivo etc.;
- Gerência de Arquivos: gerência o armazenamento dos dados do sistema operacional e os dados do usuário que utiliza o sistema.

Como o objetivo desse trabalho é a simulação dos algoritmos de escalonamento, que é um conceito da área de gerência de processos, a seguir esse tema é abordado com mais detalhes.

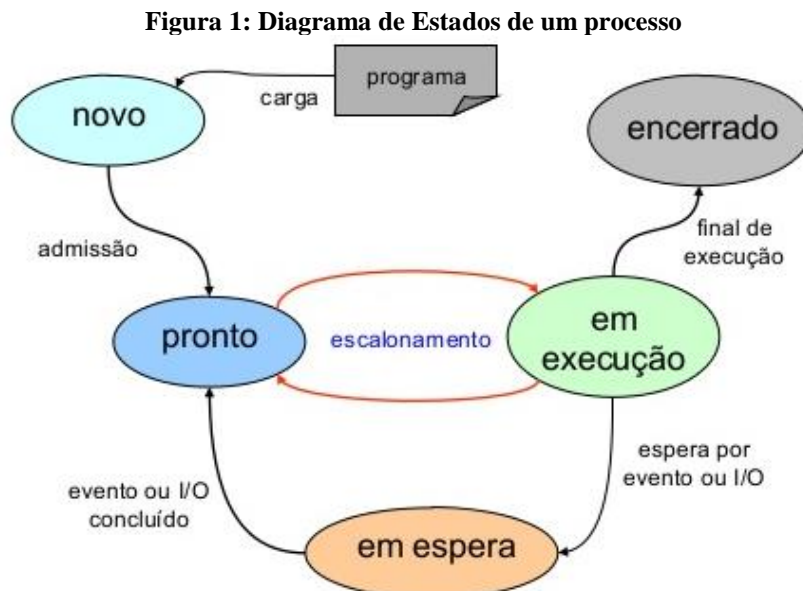
### 2.1.1 Gerência de Processos

A gerência de processos tem como finalidade executar os programas solicitados pelo usuário de forma ágil. No gerenciamento da CPU (*Central Processing Unit*) se trabalha com definições de programas em execução, que são definidos como processos.

Para o suporte desses processos, o SO em geral oferece serviços que são utilizados para um processo operar sobre outros processos. Esses serviços fazem com que os processos mudem de estados durante a execução. Cada processo pode se encontrar nos seguintes estados:

- novo: criação de um novo processo;
- em execução: o processo que está sendo executado;
- em espera ou bloqueado: o processo aguarda a finalização de alguma ocorrência, geralmente uma operação de entrada ou saída;
- pronto ou apto: o processo que está aguardando para ser alocado no processador;
- encerrado: processo que terminou sua execução.

A Figura 1 apresenta os estados em que os processos podem se encontrar:



Fonte: (Massa, 2009)

Como pode ser observado na Figura 1, durante o ciclo de vida de um processo, os estados de um processo ficam alternando entre apto ou pronto, executando na CPU ou realizando uma entrada ou saída (em espera ou bloqueado).

O sistema operacional é responsável pelo escalonamento dos processos, ou seja, para escolha do processo a ser executado na CPU. O mecanismo responsável por realizar a escolha de qual será o próximo processo a ser executado é o escalonador, que é um processo do SO (SILBERSCHATZ; GALVIN; GAGNE, 2004).

Os algoritmos de escolha utilizados pelo escalonador são definidos como algoritmos de escalonamento de processos, que serão abordados na próxima seção.

## 2.2 Algoritmos de escalonamento de processos

O método escalonador de processos é a parte do sistema operacional responsável por decidir qual processo ou *thread* será executado na CPU em cada momento. Esta escolha ocorre quando dois ou mais processos estão na fila de prontos ou aptos. Para a escolha desses processos, são utilizados os algoritmos de escalonamento, que devem escolher quais processos utilizarão a CPU nas situações seguintes (TANENBAUM, 2009):

- Quando um novo processo é criado, sendo necessário decidir em executar o processo pai ou processo filho;
- Quando um processo é finalizado, ou seja, quando deixa de existir, sendo necessário escolher outro processo que está na fila de pronto. Se não possuir nenhum processo na fila de pronto, o sistema gera um processo inoperante que executará na CPU;
- Quando um processo é bloqueado para a realização de E/S, deverá escolher outro processo para ser executado;
- Quando acontece interrupções de E/S, é necessário tomar uma decisão de escalonamento.

Durante a execução, o processo pode sair da CPU quando terminar seu ciclo ou quando o escalonador interromper sua execução. Quando um algoritmo de escalonamento executa um processo e, temporariamente é bloqueado, é chamado de escalonamento preemptivo. Quando o escalonador permite que o processo seja executado até o final de seu ciclo, liberando a CPU espontaneamente, tem-se o escalonamento não preemptivo (TANENBAUM; WOODHULL, 2000).

Existem quatro algoritmos de escalonamento de processos básicos: FIFO (*First In, First Out*), SJF (*Shortest Job First*), Prioridade e Round Robin. Nas seções seguintes serão



apresentados estes algoritmos. Os exemplos de cada algoritmo conceituado nas seções seguintes apresentam as informações a seguir:

- uma tabela com os dados dos processos que iniciaram a sua execução:
  - identificação dos processos, que estão concorrendo pela CPU;
  - tempo de ciclo, que se refere ao tempo que o processo deve ser executado na CPU antes de concluir ou solicitar uma operação de E/S;
  - momento de chegada ou transição, que é o momento em que o processo entra na fila de aptos ao iniciar sua execução ou terminar um ciclo de entrada e saída, ou seja, quando inicializa um ciclo de execução para a utilização da CPU. O primeiro processo a ser representado inicializará com momento de chegada zero;
- um diagrama de tempo:
  - apresenta qual o processo que está sendo executado na CPU em cada momento.

Os exemplos que serão realizados, apresentarão um diagrama de tempo representando a CPU e situação em que os processos concorrem apenas a uma CPU. Em sistemas que possuem mais de um processador, mais de um processo poderiam ser executados simultaneamente, de forma que para cada CPU seria apresentado um diagrama de tempo.

### 2.2.1 FIFO (*First In, First Out* - Primeiro a Entrar, Primeiro a Sair)

O FIFO é um algoritmo de escalonamento não preemptivo, em que o primeiro processo da fila de aptos é escolhido e executa até o fim de seu ciclo de execução. Após a liberação da CPU o processo que está na primeira posição da fila de prontos é escolhido para executar (SILBERSCHATZ; GALVIN; GAGNE, 2001). Quando um processo inicializa seu ciclo de execução o mesmo é inserido no fim da fila.

A seguir é apresentado um exemplo do funcionamento do algoritmo FIFO, no qual três processos concorrem à CPU, ou seja, três processos entram na fila de aptos esperando para serem executados.

**Tabela 1: Fila de Aptos (FIFO)**

<b>Processo</b>	<b>Momento de chegada</b>	<b>Tempo de ciclo</b>
<b>P1</b>	0	20
<b>P2</b>	1	4

<b>P3</b>	2	3
-----------	---	---

Como pode ser observado na Tabela 1, os processos chegam na fila de apto na ordem P1, P2, P3. O processo P1 chegou no momento 0, P2 chegou no momento 1 e P3 chegou no momento 3. O tempo de ciclo é considerado o tempo total que cada processo necessita para realizar sua tarefa e finalizar. O tempo de ciclo de P1 é 20, de P2 é 4 e P3 é 3.

Para representar o que acontece quando se utiliza o algoritmo de escalonamento FIFO, a Figura 2 apresenta o diagrama de tempo de utilização da CPU, na execução dos processos indicados na Tabela 1, mostrando o processo estava sendo executado efetivamente em cada momento.

**Figura 2: Diagrama de tempo de uso da CPU (FIFO)**

Processos	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	
P1	P1	P1	P1	P1	P1	P1	P1	P1	P1	P1	P1	P1	P1	P1	P1	P1	P1	P1	P1									
P2																					P2	P2	P2	P2				
P3																									P3	P3	P3	

Como pode ser observado na Figura 2, de acordo com o funcionamento do algoritmo FIFO não preemptivo, o diagrama de tempo de uso da CPU foi preenchido da seguinte forma:

- no momento 0, o processo P1 foi o primeiro a chegar, logo é o primeiro a ser executado até seu fim na CPU, com o ciclo de 20 tempos;
- no momento 20, o processo P2, é executado até seu fim, com o ciclo de 4 tempos;
- no momento 24, o processo P3 executa na CPU com ciclo de 3 tempos. Caso chegasse um novo processo, este seria inserido no final da fila.

Devido ao seu funcionamento, o algoritmo de escalonamento FIFO é considerado como o mais simples, no que se refere a implementação e compreensão. Esse algoritmo pode causar problemas em sistemas compartilhados, pois um processo grande toma a CPU por muito tempo, prejudicando a execução de processos pendentes. Utilizando o exemplo anterior, suponha que o processo P1 tem ciclo de 100 tempos, os processos P2 e P3 que estão aguardando para serem executados na CPU, esses processos ficarão aguardando na fila de aptos por 100 tempos de ciclo, desta forma, os processos pendentes são prejudicados por ficarem aguardando na fila por muito tempo.

### 2.2.2 SJF (*Shortest Job First - Menor Trabalho Primeiro*)

O SJF é um algoritmo de escalonamento que pode ser categorizado tanto preemptivo quanto não preemptivo. Quando é preemptivo, o processo que está executando perde a CPU quando tem um tempo de execução maior do que o processo que chegou na fila de aptos. Quando não é preemptivo, o processo que chegou continua aguardando na fila de aptos, mesmo que possua um tempo de execução menor que o processo em execução. Nos dois casos, a ordenação da fila de aptos é de ordem crescente de tempo de execução.

Independente da ordem de chegada dos processos, esse algoritmo escolhe o processo que possui o menor tempo de ciclo de execução para ser alocado na CPU (TANENBAUM; BOS, 2016). Assim, o escalonado deve verificar previamente o tempo do ciclo de execução dos processos que estão na fila de prontos.

A seguir é apresentado um exemplo do funcionamento do algoritmo SJF, no qual quatro processos concorrem à CPU, ou seja, chegaram na fila de aptos para serem executados.

**Tabela 2: Fila de Aptos (SJF)**

<b>Processo</b>	<b>Momento de chegada</b>	<b>Tempo de ciclo</b>
<b>P1</b>	0	9
<b>P2</b>	1	4
<b>P3</b>	2	2
<b>P4</b>	4	6

Como pode ser observado na Tabela 2, os processos chegam na fila de apto com ordem P1, P2, P3 e P4. P1 chegou no momento 0, P2 chegou no momento 1, P3 chegou no momento 2 e P4 chegou no momento 4. O processo P1 possui tempo de ciclo de 9, P2 com o tempo de 4, P3 com tempo 2 e P4 com tempo de 6.

Para representar o que acontece quando se utiliza o algoritmo de escalonamento SJF, a Figura 3 apresenta o diagrama de tempo de utilização da CPU, na execução dos processos indicados, que mostra que processo estava sendo executado efetivamente em cada momento.

Será apresentado o algoritmo SJF preemptivo na Figura 3, no qual se o processo que chegar na fila de aptos e for menor do que o processo que está na CPU, o processo em execução será interrompido e voltará para a fila de aptos enquanto o que chegou passará a utilizar a CPU. Se não fosse preemptivo, o processo que chegou seria maior que o processo

que está em execução, logo ficaria aguardando a vez até que o processo que está na CPU conclua.

**Figura 3: Diagrama de tempo de uso da CPU (SJF)**

Processos	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
P1	P1													P1	P1	P1	P1	P1	P1	P1
P2		P2			P2	P2	P2													
P3			P3	P3																
P4								P4	P4	P4	P4	P4	P4							

Como pode ser observado na Figura 3, de acordo com o funcionamento do algoritmo SJF preemptivo, o diagrama de tempo de uso da CPU foi preenchido da seguinte forma:

- o processo P1 foi o primeiro a chegar na fila, logo é o primeiro a ser executado na CPU;
- no momento 1 o processo P2 inicia seu ciclo de 4 tempos e P1, que ainda tem que executar 7 tempos, é retirado da CPU para que P2 execute, por possui tempo de ciclo menor que P1;
- no momento 2 o processo P3 chega na fila com ciclo de 2 tempos e P2, que ainda tem que executar 3 tempos, é retirado da CPU para que P3 execute, por ter tempo de ciclo menor que P2;
- no momento 4 o processo P4 chega na fila com ciclo de 6 tempos e P2, é executado por restar o ciclo com 3 tempos, enquanto o processo P4 tem o ciclo de 6 tempos;
- no momento 7, o processo P4 é executado com tempo de ciclo de 6 tempos, enquanto P1 aguarda na fila com ciclo de 7 tempos;
- no momento 13, o processo P1 é executado na CPU com ciclo de 7 tempos.

O SJF é considerado como um ótimo algoritmo de escalonamento, pois ele atinge o tempo médio de espera mínimo em um grupo de processos. Com a utilização desse algoritmo, processo menores podem obter uma performance mais eficiente (SILBERSCHATZ; GALVIN; GAGNE, 2001).

### 2.2.3 Prioridade

O algoritmo de escalonamento por Prioridade pode ser categorizado tanto como preemptivo quanto não preemptivo. Quando é preemptivo, o processo que está sendo executado na CPU perde a vez quando chega um processo com maior prioridade. Quando não

é preemptivo, o processo chega na fila de aptos e permanece aguardando o processo que está em execução terminar, mesmo que tenha prioridade maior que o processo em execução. A fila de aptos é ordenada em ordem de prioridade, sendo que os mais prioritários são posicionados na frente dos menos prioritários.

Essas prioridades são definidas em números e, apesar de não existir um critério afirmando que 0 é a mais alta ou mais baixa prioridade, normalmente, os Sistemas Operacionais utilizam os menores números para mais altas prioridades (SILBERSCHATZ; GALVIN; GAGNE, 2001).

A seguir é apresentado um exemplo do funcionamento do algoritmo de Prioridade, em que quatro processos concorrem à CPU, ou seja, estão aguardando na fila de aptos para serem executados.

**Tabela 3: Fila de Aptos (Prioridade)**

Processo	Momento de transição	Tempo de ciclo	Prioridade
<b>P1</b>	0	5	4
<b>P2</b>	1	3	1
<b>P3</b>	2	1	2
<b>P4</b>	4	6	3

Como pode ser observado na Tabela 3, os processos chegam na fila de aptos com ordem P1, P2, P3 e P4. O processo P1 chegou no momento 0, P2 chegou no momento 1, P3 chegou no momento 2 e P4 chegou no momento 4. O processo P1 possui prioridade 4 e ciclo de 9 tempos, P2 com prioridade 1 e ciclo de 3 tempos, P3 com prioridade 2 e ciclo de 1 tempo, e P4 com prioridade 3 e ciclo de 6 tempos.

Para representar o que acontece quando se utiliza o algoritmo de escalonamento por Prioridade, a Figura 4 apresenta o diagrama de tempo de utilização da CPU, na execução dos processos indicados, que mostra que o processo estava sendo executado efetivamente em cada momento.

**Figura 4: Diagrama de tempo de uso da CPU (Prioridade)**

Processos	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
<b>P1</b>	P1											P1	P1	P1	P1
<b>P2</b>		P2	P2	P2											
<b>P3</b>					P3										
<b>P4</b>						P4	P4	P4	P4	P4	P4				

Como pode ser observado na Figura 4, de acordo com o funcionamento do algoritmo de Prioridade preemptivo, o diagrama de tempo de uso da CPU foi preenchido da seguinte forma:

- no momento 0, o processo P1 foi o primeiro a chegar na fila de aptos, com ciclo de 5 tempos, logo é o primeiro a ser executado;
- no momento 1, o processo P2 com ciclo de 3 tempos ganha a CPU por ter prioridade maior, logo P1 volta para a fila de aptos;
- no momento 2, o processo P3 chega no final da fila de aptos com ciclo de 1 tempo, porém o mesmo não possui prioridade maior que o processo em execução, no caso P2;
- no momento 4, o processo P4 chega no final da fila de aptos, mas o processo P3 tem prioridade maior, logo P3 ganha a CPU;
- no momento 5, após a conclusão de P3, o processo P4 ganha a CPU, possuindo prioridade maior que P1, que permanece na fila de aptos;
- no momento 11, o processo P1 ganha a CPU por não possuir concorrentes com prioridades maiores, dessa forma o mesmo é executado até o fim de seu ciclo.

O algoritmo de prioridade funciona da mesma forma que o algoritmo SJF, em que os algoritmos com menor tempo de ciclo são os que possuem maior prioridade para a execução. Nesse caso quanto menor for o valor da prioridade, mais prioridade o processo pertencerá.

#### **2.2.4 RR - Round Robin**

O Round Robin (RR) é um algoritmo que pode ser categorizado como preemptivo e, semelhante ao algoritmo FIFO, os processos são inseridos na fila de aptos em ordem de chegada. Porém, no RR atribui-se uma fatia de tempo, denominada *Quantum*, para cada processo utilizar a CPU. Independentemente do tempo de ciclo de cada processo, o algoritmo define uma parcela de tempo de execução para todos. Ao atingir o fim desse tempo, o processo que está na CPU é retirado e o processo seguinte ganha a CPU, formando uma execução circular dos processos, até que todos os processos estejam finalizados (STUART, 2011).

A seguir é apresentado um exemplo do funcionamento do algoritmo Round Robin, no qual quatro processos concorrem à CPU com a utilização do *quantum* para definir o tempo

de execução de cada processo, ou seja, estão aguardando na fila de aptos para serem executados.

**Tabela 4: Fila de Aptos (Round Robin)**

Processo	Momento de chegada	Tempo de ciclo
<b>P1</b>	0	8
<b>P2</b>	1	4
<b>P3</b>	2	2
<b>P4</b>	4	6

Como pode ser observado na Tabela 4, os processos chegam na fila de apto com ordem P1, P2, P3 e P4. P1 chegou no momento 0 com o tempo de ciclo de 8, P2 chegou no momento 1 com tempo de ciclo de 4, P3 no momento 2 com 2 tempos de ciclo e P4 chegou no momento 4 com 6 tempos de ciclo.

Para representar o que acontece quando se utiliza o algoritmo de escalonamento Round Robin, a Figura 5 apresenta o diagrama de tempo de utilização da CPU, na execução dos processos indicados, que mostra que processo estava sendo executado efetivamente em cada momento, considerando o *quantum* ou fatia de tempo de 3 momentos.

**Figura 5: Diagrama de tempo de uso da CPU (Round Robin)**

Processos	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	
<b>P1</b>	P1	P1	P1						P1	P1	P1					P1	P1				
<b>P2</b>				P2	P2	P2									P2						
<b>P3</b>							P3	P3													
<b>P4</b>													P4	P4	P4				P4	P4	P4

Como pode ser observado na Figura 5, de acordo com o funcionamento do algoritmo Round Robin, o diagrama de tempo de uso da CPU foi preenchido da seguinte forma:

- no momento 0, o processo P1 foi o primeiro a chegar na fila, logo é o primeiro a ser executado na CPU;
- no momento 1, o processo P2 chega no fim da fila, aguardando ser executado;
- no momento 2, o processo P3 chega na fila de aptos logo depois de P2;
- no momento 3, o processo P2 ganha a CPU, enquanto que o processo P1 retorna para o final da fila, por atingir seu tempo de *quantum*;
- no momento 4, o processo P4 chega ao fim da fila de aptos logo depois de P1;

- no momento 6, o processo P3 executa até o fim de seu ciclo, pois não atinge o limite de *quantum*;
- no momento 8, o processo P1 novamente ganha a CPU, restando ciclo de 5 tempos;
- no momento 11, o processo P4 ganha a CPU com ciclo de 6 tempos, enquanto que P1 retorna para o fim da fila de aptos por atingir o limite de *quantum*;
- no momento 14, o processo P2 ganha a CPU, enquanto que P4 volta para o final da fila por atingir o limite do *quantum*;
- no momento 15, o processo P1 executa até o fim por não atingir o limite do *quantum*, que era de 2 tempos;
- no momento 17, o processo P4 executa até o fim de seu ciclo.

No Round Robin os processos que sofreram preempção na CPU são movidos para o final da fila, da mesma maneira que os processos novos. A performance desse algoritmo depende do valor definido para o quantum: quanto maior o *quantum*, mais será assemelhado ao algoritmo FIFO, e quanto menor o *quantum*, o algoritmo é intitulado como compartilhamento do processador, que faz ter uma percepção de paralelismo para os usuários.

Na internet podem ser encontradas ferramentas que realizam simulações dos algoritmos de escalonamento de processos. Dentre essas ferramentas, foram escolhidas três para a realização de estudo e levantamento das principais características. A seção seguinte apresenta as ferramentas definidas para a implementação do ambiente.

### 2.3 Ferramentas relacionadas

As ferramentas I3S, PSSAV e Simulador de escalonamento de processos realizam simulações de escalonamento de processos e podem ser encontradas na Internet nos seguintes endereços:

- I3S - <http://lasdpc.icmc.usp.br/~ssc640/pos/i3s/index.php?lang=pt>
- PSSAV - <https://code.google.com/archive/p/pssav/>
- Simulador de escalonamento de processos - <https://sourceforge.net/projects/simuladordeescalamentodeproc/>

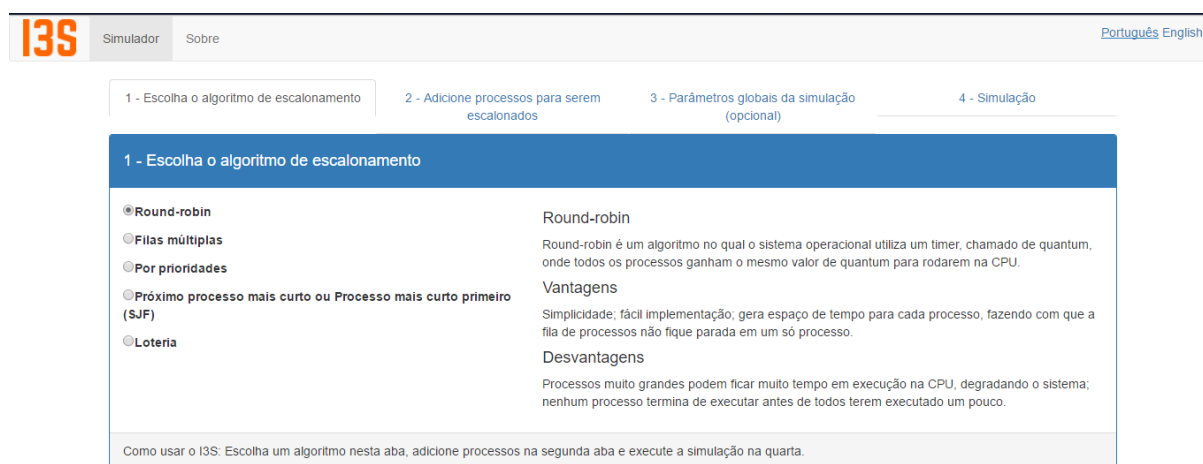


As seções seguintes apresentam, para cada ferramenta, um exemplo de utilização e considerações sobre aspectos técnicos relacionados as mesmas como, por exemplo, processo de instalação, complexidade de utilização e iteração com o usuário.

### 2.3.1 I3S

I3S é uma ferramenta *Web* que realiza simulações de escalonamento de processos, descreve de forma geral os conceitos de cada algoritmo de escalonamento, bem como os parâmetros globais de simulação. A Figura 6 apresenta a tela inicial, que é aberta assim que a ferramenta é executada.

**Figura 6: Tela inicial (I3S)**



© 2017 por Marcelo Koti Kamada, Maria Lydia Fioravanti

Como pode ser observado na Figura 6, a tela inicial do ambiente I3S apresenta as opções de escolha do algoritmo de escalonamento que será definido pelo usuário. Após o usuário selecionar um algoritmo, é apresentada a definição deste, bem como algumas vantagens e desvantagens. Por exemplo, o algoritmo escolhido foi o Round Robin, assim, na tela é apresentada a descrição do algoritmo, as vantagens e desvantagens. Após a escolha do algoritmo, na segunda aba, Figura 7, o usuário deve adicionar os processos que serão utilizados na simulação.

**Figura 7: Tela de inclusão de processos (I3S)**

1 - Escolha o algoritmo de escalonamento    2 - Adicione processos para serem escalonados    3 - Parâmetros globais da simulação (opcional)    4 - Simulação

**2 - Adicione processos para serem escalonados**

Tempo de execução:     Tempo de execução:    
Tempo de processamento na CPU que o processo necessita para terminar

Tipo do processo

Orientado a CPU  
 Orientado a E/S

Nome	Tempo	Tipo
A	4	cpu
B	5	cpu

© 2017 por Marcelo Koti Kamada, Maria Lydia Fioravanti

Como visto na Figura 7, a segunda tela da ferramenta apresenta o formulário para o cadastro dos processos que serão simulados. São oferecidas opções de inserção de tempo de execução e o tipo do processo que será simulado, tanto orientado a CPU (quando os processos não requisitam E/S) quanto orientado a E/S (quando os processos requisitam E/S). Conforme mostra a Figura 8, na terceira tela do ambiente permite definir os parâmetros globais da simulação, sendo opcionais para o usuário.

**Figura 8: Tela de parâmetros globais (I3S)**

1 - Escolha o algoritmo de escalonamento    2 - Adicione processos para serem escalonados    3 - Parâmetros globais da simulação (opcional)    4 - Simulação

**3 - Parâmetros globais da simulação (opcional)**

Quantum:     Quantum:    
Período de tempo que os processos têm direito de executar na CPU cada vez que forem escalonados.

Tempo para troca:

Tempo de uma operação de E/S:

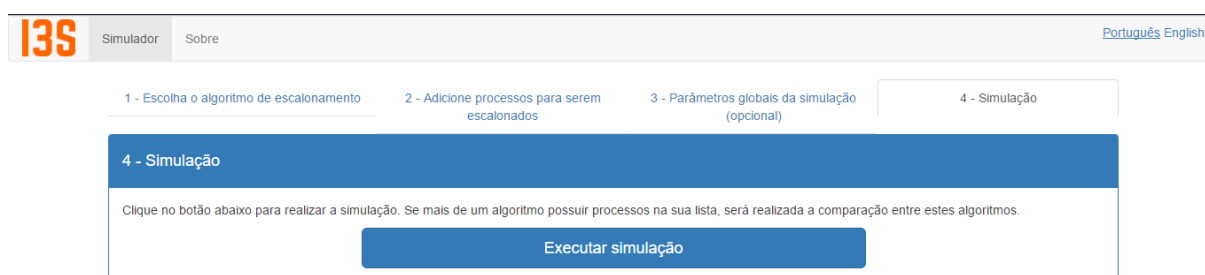
Tempo de processamento até E/S:

© 2017 por Marcelo Koti Kamada, Maria Lydia Fioravanti

Como pode ser observado na Figura 8, considerando que na primeira tela, Figura 6, o usuário selecionou o algoritmo RR, os atributos de entrada definidos para essa tela são: o quantum, explicado na seção 2.2.4; o tempo de troca, tempo necessário para trocar o processo executando na CPU pelo processo escolhido pelo algoritmo de escalonamento; tempo de uma operação E/S; e tempo de processamento até E/S, é o tempo médio que os processos orientados a E/S processam antes de solicitar uma operação E/S.

Após as definições dos parâmetros de simulação, como pode ser observado na Figura 9, é apresentada a opção onde o usuário deve executar a simulação.

**Figura 9: Tela de simulação inicial (I3S)**



A Figura 10 apresenta a tela da simulação sendo realizada, apresentando os passos dos algoritmos de escalonamento de forma textual.

**Figura 10: Tela de simulação (I3S)**

The screenshot shows the I3S simulation interface. At the top, there is a navigation bar with 'I3S', 'Simulador', and 'Sobre' on the left, and 'Português English' on the right. The main area is titled 'Round-robin' and contains several panels:

- Panel 1 (Descrição):** Shows simulation statistics: 'Tempo Total de Execução: 0', 'Utilização da CPU: 0%', 'Trocas de Contexto: 0', 'step = 0', and 'Ação Executada:'.
- Panel 2 (Viewport):** A central black area with white text that reads 'PAUSED' and 'WAITING FOR PROCESS'.
- Panel 3 (Opções):** A list of simulation controls: 'Avançar', 'Voltar', 'Automático', 'Resetar', and 'Página Inicial'.
- Panel 4 (Processos prontos):** A table showing ready processes:
 

Nome	Tipo	Tempo restante
A	cpu	4
B	cpu	5
- Panel 5 (Processos bloqueados):** A table showing blocked processes with columns for 'Nome', 'Tipo', 'Tempo restante', and 'Tempo restante de E/S'.

At the bottom, a footer indicates '© 2017 por Marcelo Koti Kamada, Maria Lydia Fioravanti'.

Como visto na Figura 10, a tela representa o resultado da simulação do escalonamento de processos realizada considerando o algoritmo escolhido pelo usuário. Nesta tela são apresentados: no rótulo 1 apresenta a descrição da simulação a ser realizada; no rótulo 2 mostra a visualização do processo que está sendo executado; no rótulo 3 as opções de simulação para o usuário: avançar, resetar, voltar, automático (realizar a simulação por conta própria) e página inicial; no rótulo 4 apresenta a tabela de processos na fila de prontos ou aptos; no rótulo 5 apresenta a tabela de processos que estão bloqueados.

### 2.3.1.1 Considerações sobre a ferramenta

A ferramenta apresenta o processo de simulação de três dos quatro principais algoritmos de escalonamento de processos: Round Robin, por Prioridade e SJF. Ela segue fielmente aos conceitos dos algoritmos de escalonamento de processos apresentados nos principais livros didáticos de sistemas operacionais.

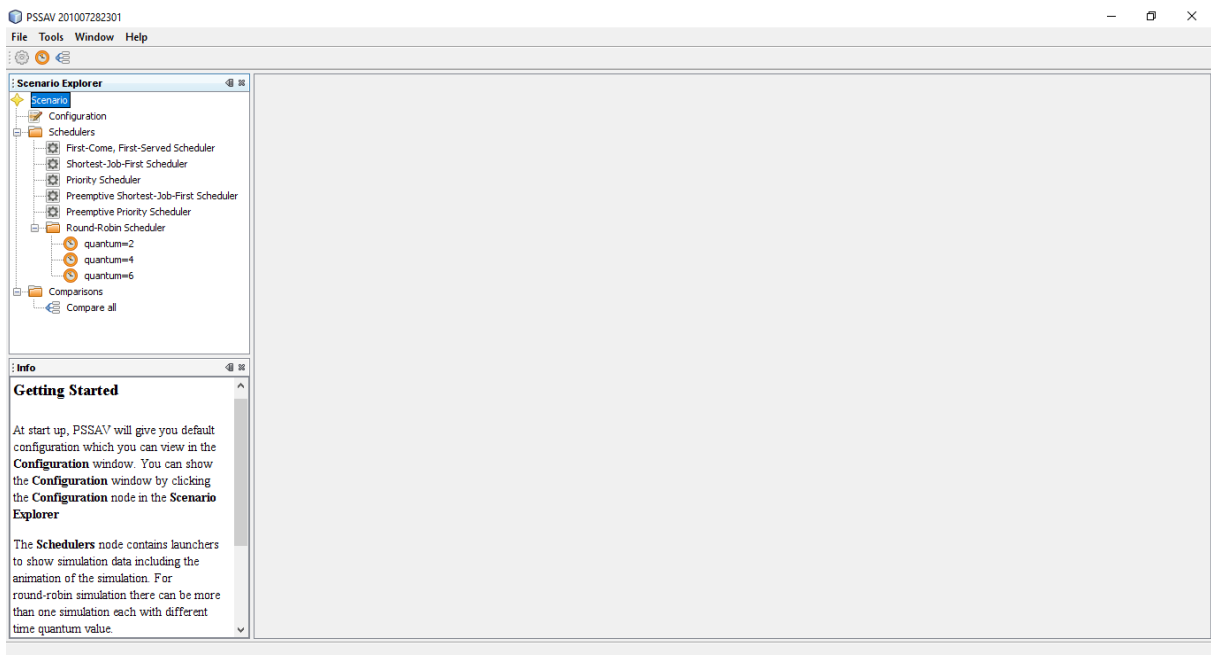
O funcionamento dos algoritmos não é utilizado diagrama de tempo. Por exemplo, a ferramenta não utilizou nenhum diagrama para a realização da simulação, mostrando o passo a passo da simulação do algoritmo de forma de forma descritiva e não visual.

A ferramenta é disponibilizada via *web*, sem necessidade de utilização de arquivos de instalação. A utilização pode ser complicada para usuários iniciantes, pois não utiliza diagrama de tempo para a realização das simulações.

### 2.3.2 PSSAV (*Process Scheduling Simulation, Analysis, and Visualization*)

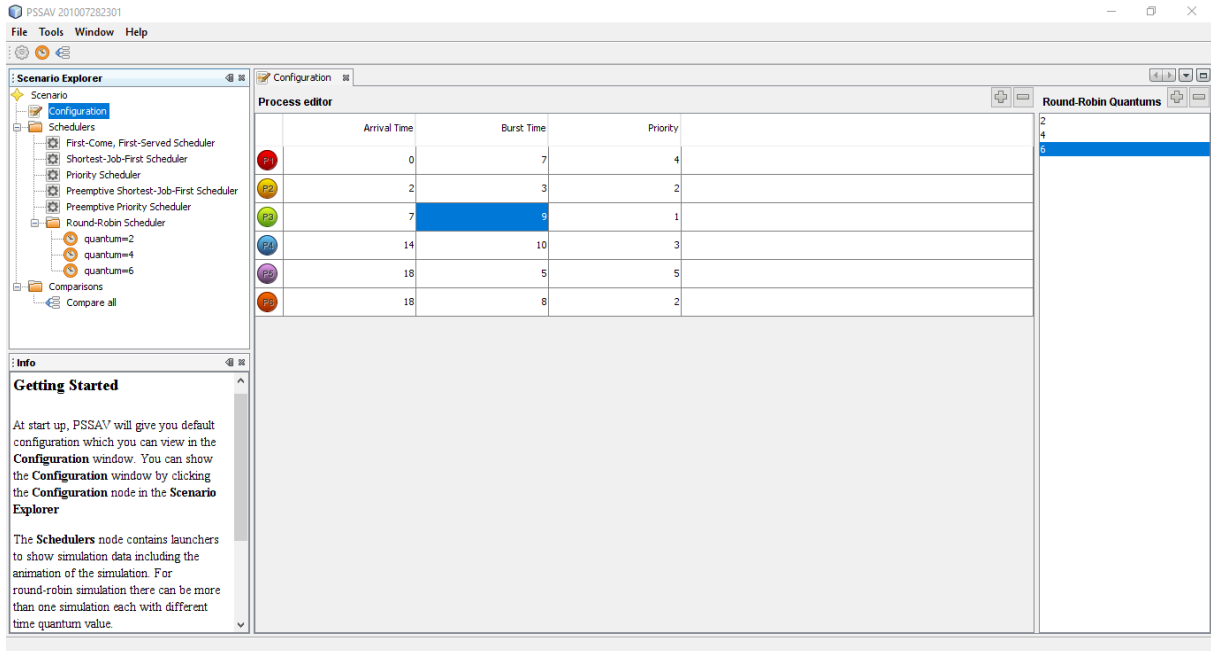
PSSAV é uma ferramenta *Desktop* que realiza simulações de escalonamento de processos e apresenta as simulações de escalonamento de forma gráfica e comparações entre os algoritmos. A Figura 11 apresenta a tela inicial, que é aberta assim que a ferramenta é executada.

Figura 11: Tela inicial (PSSAV)



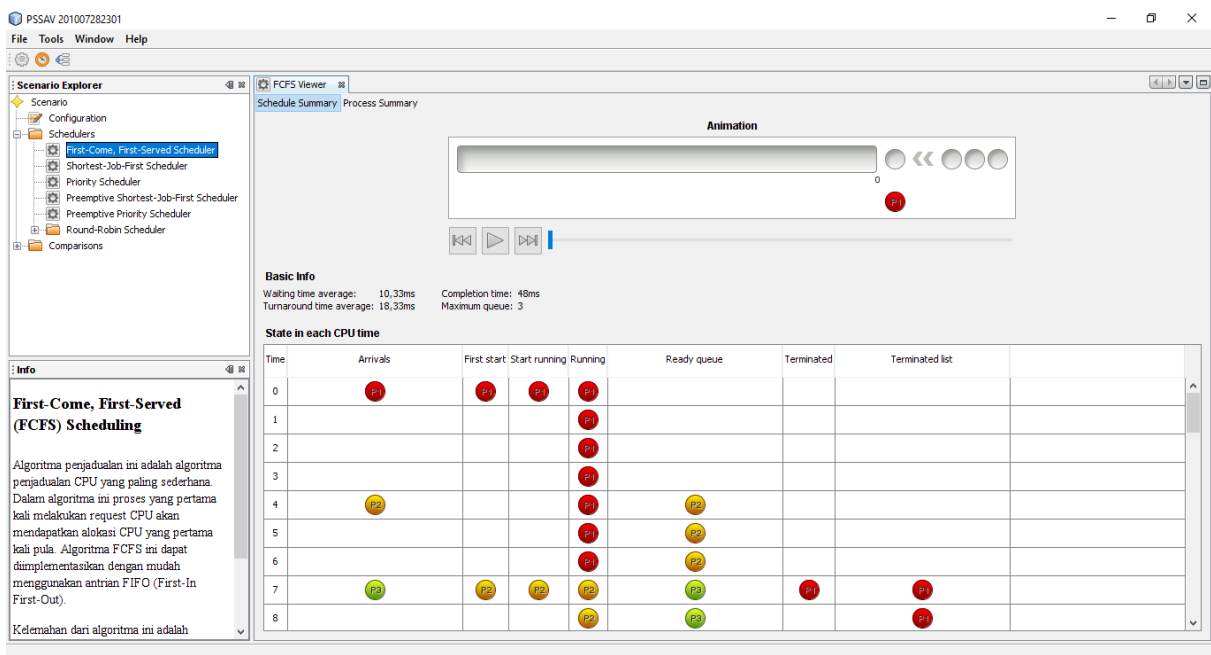
Como visto na Figura 11, na barra vertical apresenta as opções de configuração, os simuladores dos algoritmos de escalonamento e uma opção que realiza a comparação do funcionamento entre os algoritmos. Após a escolha da opção de configuração, é apresentada a tela da Figura 12.

Figura 12: Tela de configuração (PSSAV)



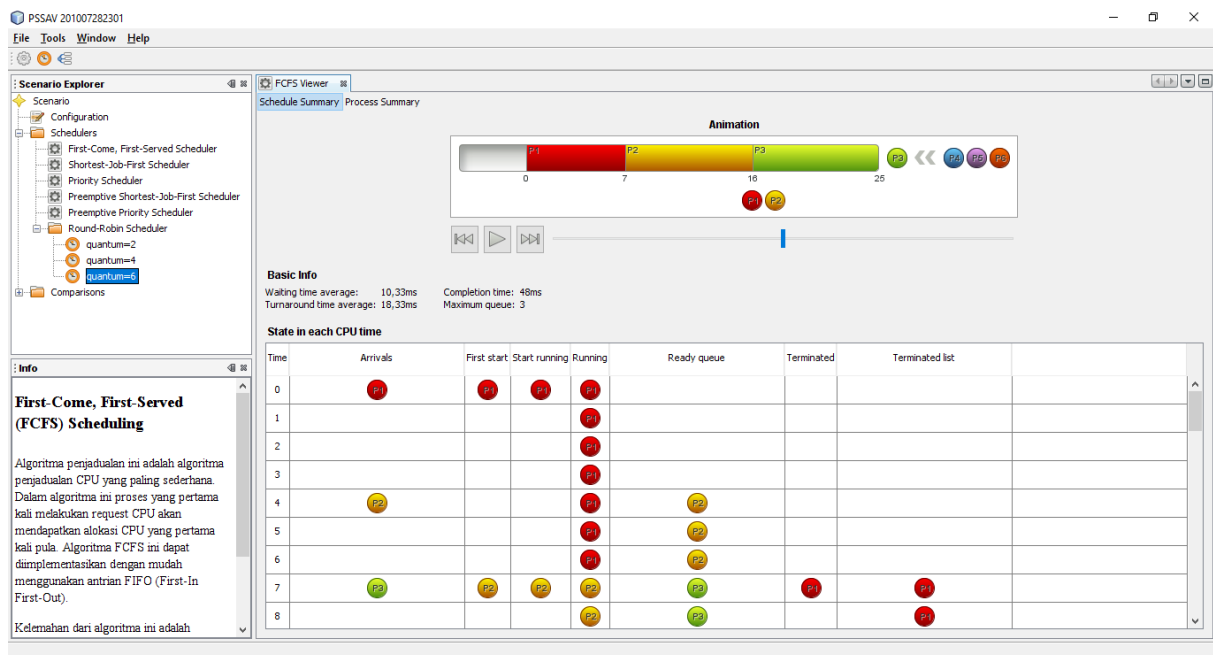
Como visto na Figura 12, a tela apresenta as configurações dos processos e do *Quantum*, em que, com base nesses dados, serão utilizados para realizar a simulação. Esses dados podem ser adicionados, alterados e removidos, tanto para processos, quanto para *Quantum*. Após realizada a configuração, com a escolha de um dos algoritmos, é apresentada a tela da Figura 13, exibindo a tela de simulação inicial.

Figura 13: Tela de simulação inicial (PSSAV)



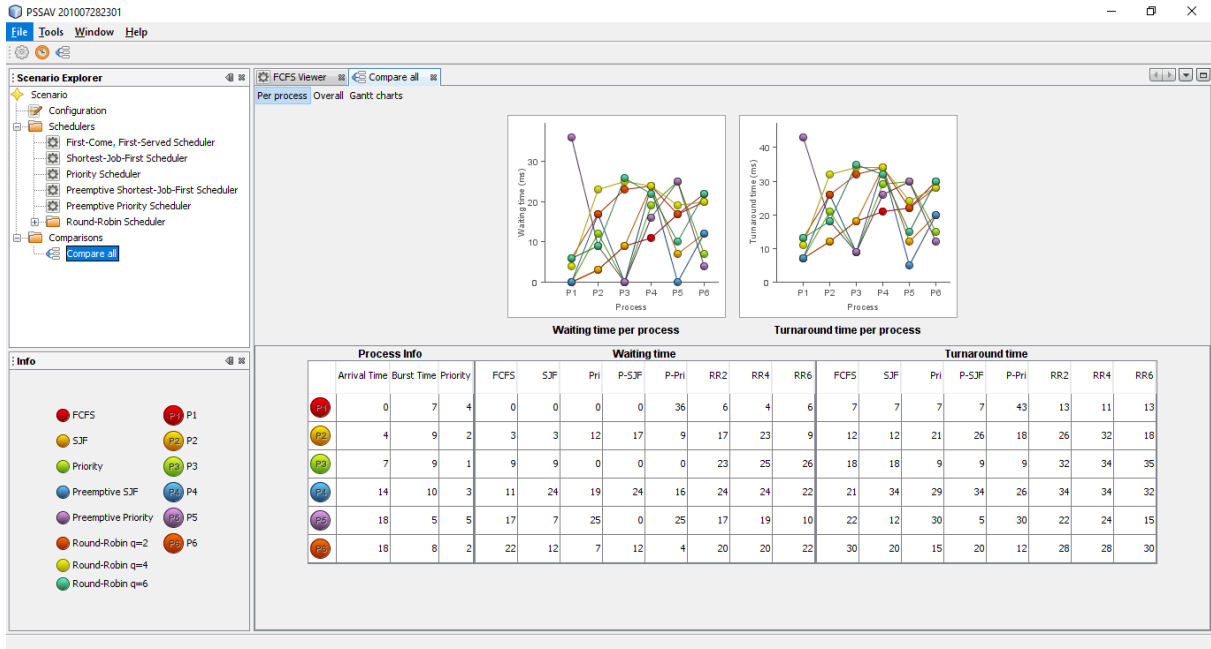
Como visto na Figura 13, é representada a tela inicial antes da realização da simulação, esta tela é a mesma para todos os algoritmos de escalonamento. Apresentando a linha do tempo da CPU, onde o usuário poderá voltar, avançar e pausar. Na mesma figura, a tabela na parte inferior da tela representa o estado de cada momento da CPU. Logo depois de iniciar a simulação, é apresentado a tela da Figura 14, mostrando a simulação sendo realizada.

**Figura 14: Tela da simulação sendo realizada (PSSAV)**



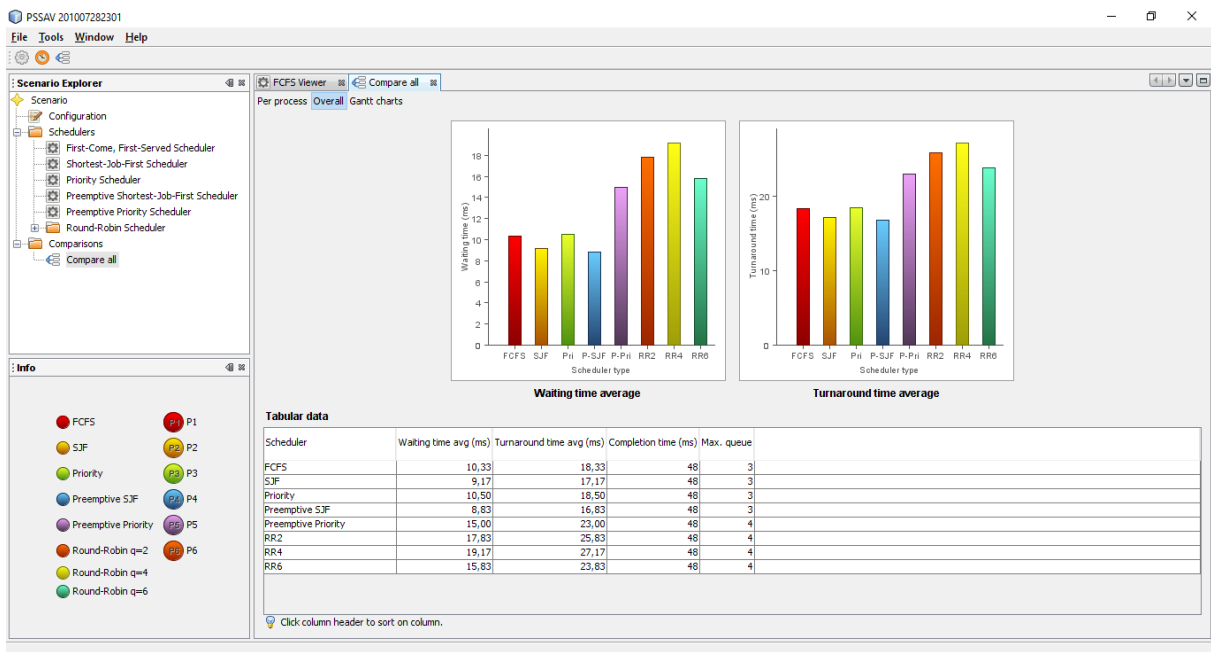
Como pode ser observado na Figura 14, a tela representa a simulação sendo realizada. A animação da simulação é demonstrada utilizando o diagrama de *Gantt*. Nesta tela, apresenta também os processos que estão sendo executados, a fila dos processos que estão aguardando para serem atendidos e os processos que foram concluídos ou finalizados. Ao finalizar a simulação, com a escolha da opção *Compare all* de *Comparisons*, é apresentada a tela da Figura 15, exibindo a comparação entre os algoritmos.

Figura 15: Tela de comparação (PSSAV)



Como visto na Figura 15, a tela apresenta a comparação de todos os algoritmos de escalonamento de processos. Permitindo a visualização dos dados de conclusão como, as informações dos processos, o tempo de espera total e tempo de *turnaround* de cada algoritmo. Após a escolha da opção *Overall*, é representada a tela da Figura 16, exibindo tela de comparação dos algoritmos de escalonamento em geral.

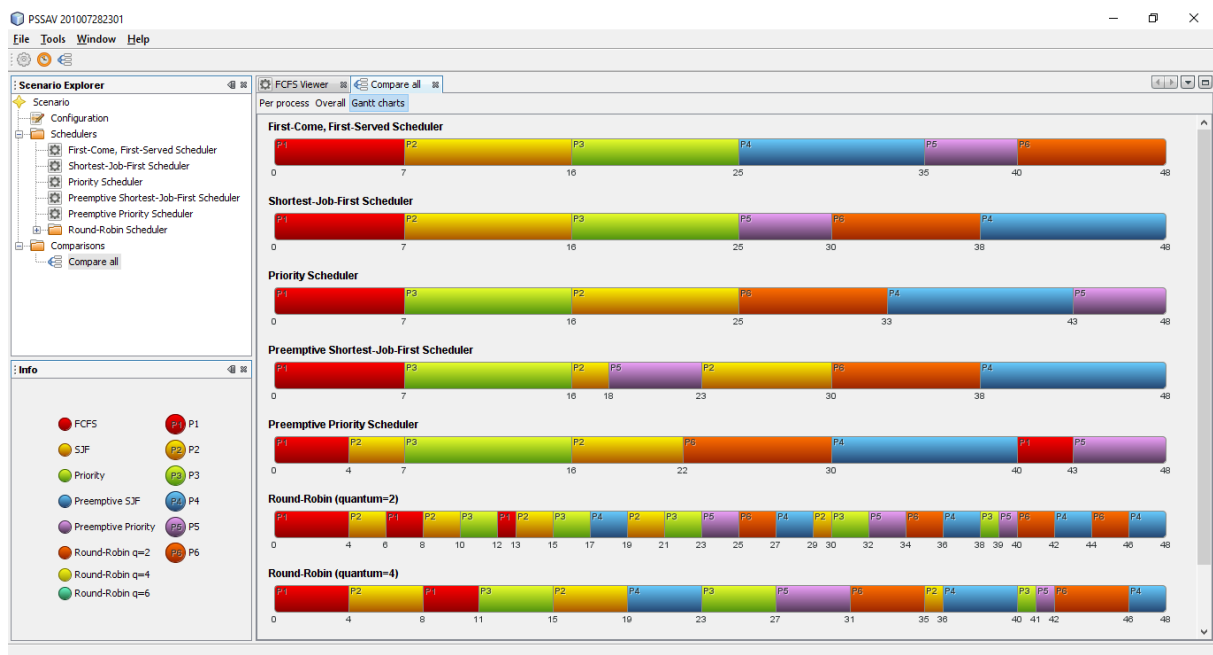
Figura 16: Tela da média dos algoritmos (PSSAV)





Como visto na Figura 16, a tela representa a média de todos os algoritmos de escalonamento. Apresenta tanto a média de tempo de espera, quanto a média de turnaround. Após a escolha da opção *Gantt charts*, é apresentada a tela da Figura 17, mostrando a comparação dos diagramas dos algoritmos de escalonamento.

**Figura 17: Tela do diagrama de Gantt (PSSAV)**



Como visto na Figura 17, a tela apresenta os diagramas de *Gantt* de todos os algoritmos de escalonamento de processos. Assim finalizando as telas do simulador PSSAV.

### 2.3.2.1 Considerações sobre a ferramenta

A ferramenta simula os quatro principais algoritmos de escalonamento de processos de forma visual. Permitindo uma representação gráfica com animações, facilitando no entendimento do funcionamento dos algoritmos, e a comparação entre os algoritmos. Porém, não segue fielmente aos conceitos dos algoritmos de escalonamento de processos apresentados nos principais livros didáticos, pois alguns algoritmos não realizam a preempção nos processos.

O funcionamento do algoritmo é representado graficamente. Por exemplo, foi realizada uma animação com a utilização do diagrama de *Gantt*, apresentando claramente os processos que estavam sendo executados e os que estavam na fila de aptos.

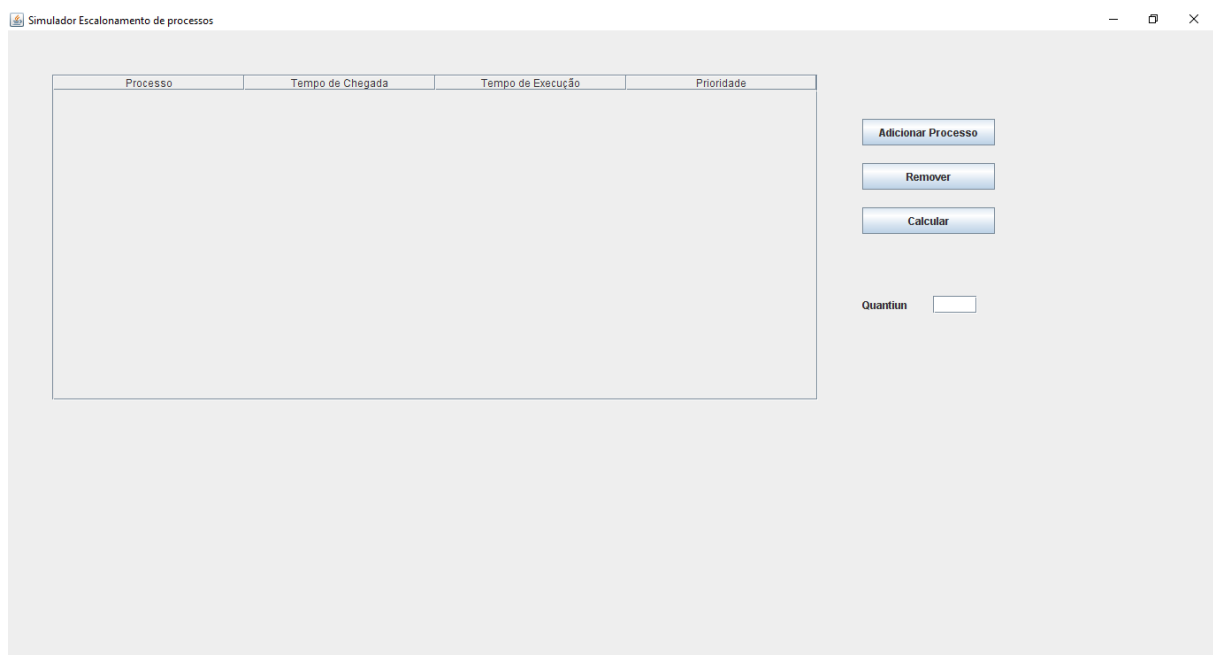
O processo de instalação da ferramenta é padrão como os demais instaladores, porém deverá estar instalado o JDK 6 (*Java Development Kit*) para prosseguir a instalação da ferramenta.

A utilização da ferramenta é simples para o usuário, sendo representado graficamente o funcionamento dos algoritmos. Por exemplo, são apresentadas animações dos diagramas, cada processo possui uma cor diferente, cada processo que chega na fila de aptos é feita animações de como são organizados os algoritmos, a representação do processo que está em execução etc.

### 2.3.3 Simulador de escalonamento de processos

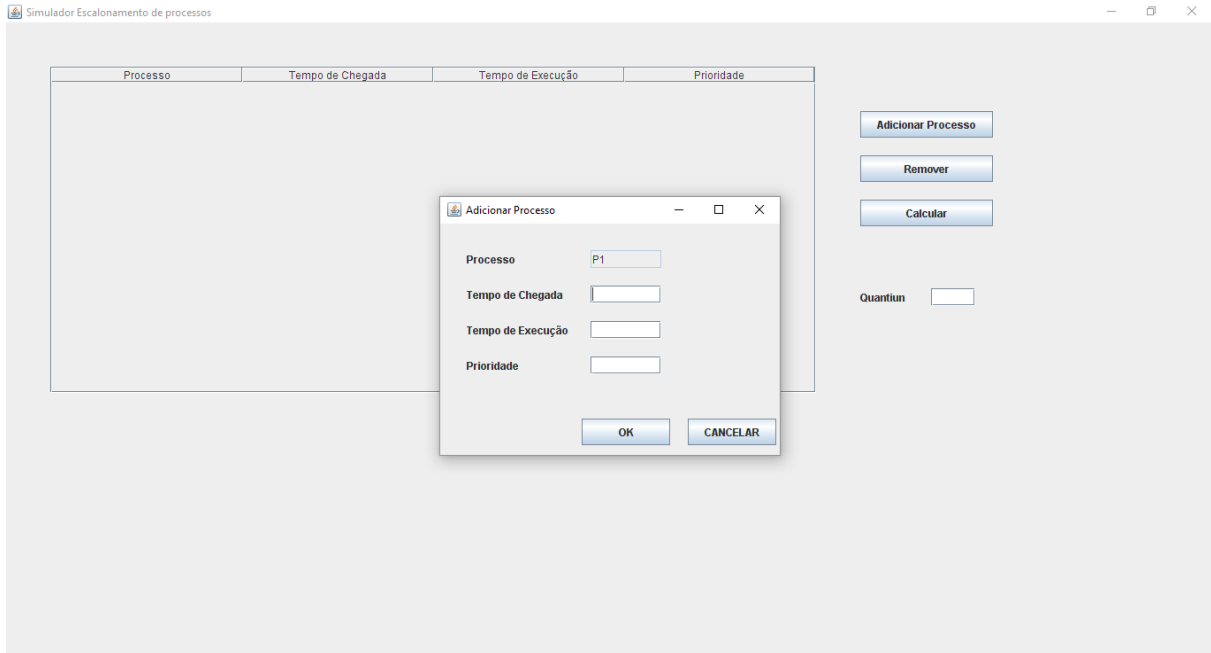
Simulador de escalonamento de processo é uma ferramenta *Desktop* que realiza a simulação dos quatro principais algoritmos de escalonamento de processos. A seguir, na Figura 18, é apresentada a tela inicial da ferramenta.

**Figura 18: Tela inicial (Simulador de escalonamento)**



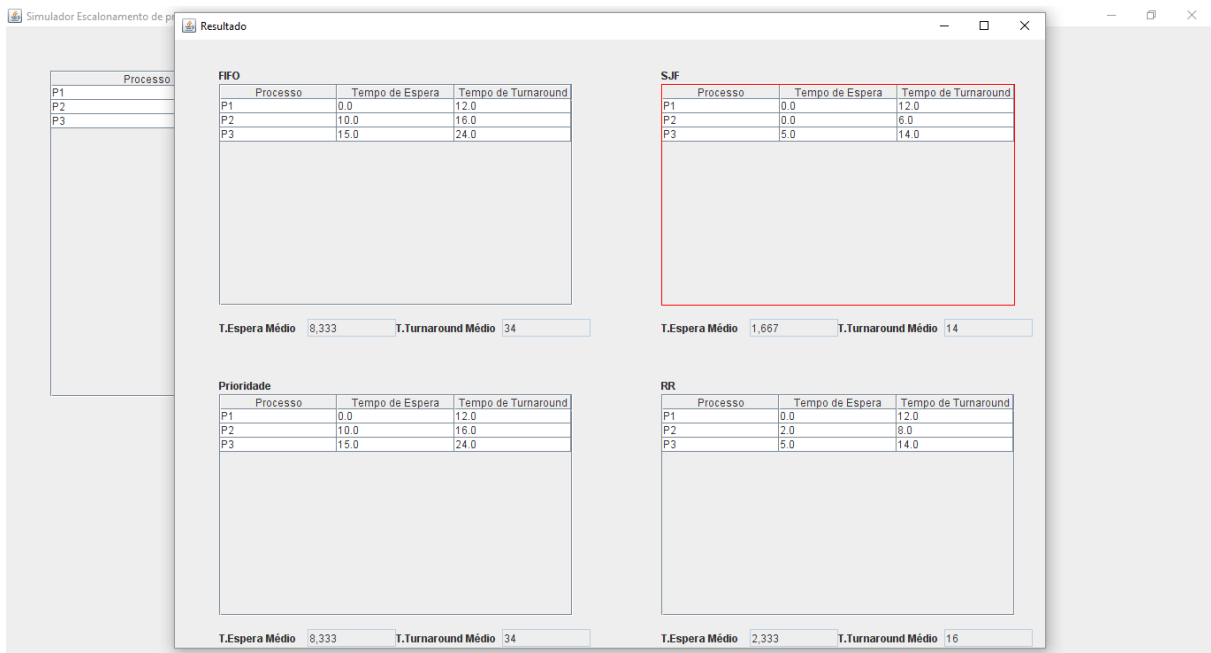
Como pode ser observado na Figura 18, a tela apresenta uma tabela vazia de processos, permitindo as opções de cadastro e exclusão do processo, definição do tempo de *quantum* e calcular as médias dos algoritmos. Com a escolha da opção Adicionar Processo, é apresentada na tela da Figura 19, o cadastro dos processos.

**Figura 19: Tela de inclusão de processo (Simulador de escalonamento)**



Como pode ser observado na Figura 19, a tela apresenta o formulário de inclusão de processo, em que é possível inserir o tempo de chegada, o tempo de execução e a prioridade do processo. Após a escolha da opção calcular, é direcionado para a tela da Figura 20, de resultado dos algoritmos.

**Figura 20: Tela final (Simulador de escalonamento)**



Como visto na Figura 20, esse resultado apresenta uma tabela para cada algoritmo de escalonamento (FIFO, SJF, Prioridades e Round Robin) com as médias de tempo de espera e tempo de turnaround. Assim, finalizando as telas do simulador de escalonamento de processos.

### 2.3.3.1 Considerações sobre a ferramenta

A ferramenta apresenta o processo de simulação dos quatro principais algoritmos de escalonamento de processos. Ela não segue fielmente aos conceitos dos algoritmos de escalonamento de processos apresentados nos principais livros didáticos.

A ferramenta não apresenta de forma gráfica o funcionamento dos principais algoritmos de escalonamento, apresentando apenas a tabela resultado dos mesmos.

Não é necessário a instalação da ferramenta, para o uso da mesma, utiliza-se um executável Java. Sua utilização é bastante simples, porém não interage com o usuário.

## 2.4 Paralelo entre as ferramentas

Esta seção apresenta um paralelo entre as ferramentas I3S, PSSAV e Simulador de escalonamento de processos a partir de alguns aspectos técnicos definidos pela professora Fábria Magali Santos Viera (VIEIRA, 1999). Serão utilizadas notas de 0 a 5, definidas em encontros com prof<sup>a</sup> Madianita Bogo Marioti para a avaliação das ferramentas de simulação de escalonamento de processos, sendo que 0 define-se com a nota mais baixa e 5 a nota mais alta.

A Tabela 5 apresenta o paralelo entre as três ferramentas.

**Tabela 5: Paralelo entre as ferramentas de simulação**

<b>Crítérios</b>	<b>I3S</b>	<b>PSSAV</b>	<b>Simulado de Escalonamento de Processos</b>
<b>complexidade de instalação</b>	0	3	0
<b>complexidade de utilização</b>	4	0	0
<b>fidelidade de conteúdo</b>	5	4	2
<b>plataforma</b>	Web	Desktop	Desktop
<b>interação com o usuário</b>	4	5	0
<b>instruções de forma clara</b>	2	5	3
<b>o software traz orientações sobre as instruções da simulação</b>	5	5	0

<b>o contexto é adequado ao design e são atraentes(interface).</b>	3	5	1
--	---	---	---

Como visto na Tabela 5, são apresentadas as notas de avaliação dos três simuladores de escalonamento de processos. Das ferramentas analisadas, apenas I3S é *web* enquanto as demais são *desktop*. A ferramenta PSSAV destacou-se entre os demais, pois apresenta uma facilidade de uso entre usuário e interface atraente e interativa que ajuda na representação dos algoritmos de forma clara e visual. Na seção seguinte, serão apresentados a metodologia e técnicas que serão utilizadas para o desenvolvimento da ferramenta.

### 3 MATERIAIS E MÉTODOS

Esta seção apresenta o desenho de estudo do trabalho, os softwares que foram utilizados para o desenvolvimento do ambiente, bem como a arquitetura do *software* que foi utilizado na implementação do trabalho.

#### 3.1 Metodologia

O trabalho teve como objetivo implementar um ambiente *web* para simulações dos algoritmos de escalonamento de processos. A Figura 21 apresenta as fases que foram executadas para o desenvolvimento do trabalho.

Figura 21: Fases para o desenvolvimento do projeto.



Para a estruturação desse projeto, o desenvolvimento do projeto foi dividido em três fases:

- **análise dos ambientes:** fase em que foram analisados alguns ambientes de simulação de escalonamento de processos existentes, no qual foi construído um paralelo entre as ferramentas a partir dos aspectos técnicos definidos pela orientadora Madianita Bogo e professora Fábila Magali Santos Viera (VIEIRA, 1999);
- **implementação do ambiente:** com base nas análises obtidas dos ambientes existentes do estudo dos conceitos relacionados aos algoritmos de escalonamento FIFO (First In, First Out), SJF (Shortest Job First), RR (Round Robin) e Prioridade. Na implementação foram utilizadas as linguagens de programação definidas na seção 3.2;
- **testes funcionais:** nesta fase foram realizados testes das funcionalidades de todas as simulações que o ambiente possui.

#### 3.2 Softwares

Para a realização da implementação da ferramenta *web*, foram utilizadas as seguintes linguagens de programação:

- **bootstrap<sup>1</sup>**: framework que possibilita a implementação front-end, parte do software que é executada no navegador (OTTO; JACOB, 2010). Utilizado na construção do layout do ambiente de simulação de escalonamento;
- **angularJS<sup>2</sup>**: framework que realizar estruturação de aplicações web, fornecendo funções para a criação de funcionalidades necessárias para a aplicação. Utilizada na implementação dos métodos dos algoritmos de escalonamento de processos;
- **notify.JS<sup>3</sup>**: plug-in jQuery que fornece notificações prática e personalizada. Foi utilizado para criar notificações no sistema;
- **Google Charts<sup>4</sup>**: biblioteca utilizada na criação dos gráficos de resultado.

Na seção seguinte são apresentados os resultados da implementação do ambiente de simulação com base nas linguagens apresentadas anteriormente.

---

<sup>1</sup> <http://getbootstrap.com/>

<sup>2</sup> <https://angularjs.org/>

<sup>3</sup> <https://notifyjs.com/>

<sup>4</sup> <https://developers.google.com/chart/>

## 4 RESULTADOS E DISCUSSÃO

Neste trabalho foi implementado uma aplicação *web* que realiza simulação dos algoritmos de escalonamento de processos, visando demonstrar de forma gráfica e com animação o funcionamento dos algoritmos FIFO, SJF, por Prioridade e *Round Robin*. A simulação apresenta a animação da construção do diagrama de utilização da CPU, a tabela de resultados e um gráfico que relaciona os valores apresentados na tabela de resultados.

Espera-se que a aplicação auxilie os alunos que da disciplina Sistemas Operacionais, facilitando o entendimento do funcionamento dos principais algoritmos de escalonamento de processos e possibilite que os alunos vejam diversos exemplos diferentes auxiliando no estudo individual, sem a necessidade de ajuda do professor para verificar se os resultados estão corretos.

O simulador foi desenvolvido para ser acessado pela internet, sem precisar que o usuário faça o *download* e instale no computador. A aplicação também trabalha de forma responsiva, possibilitando sua utilização em diversas dimensões de tela, possibilitando seu uso em *smartphones*.

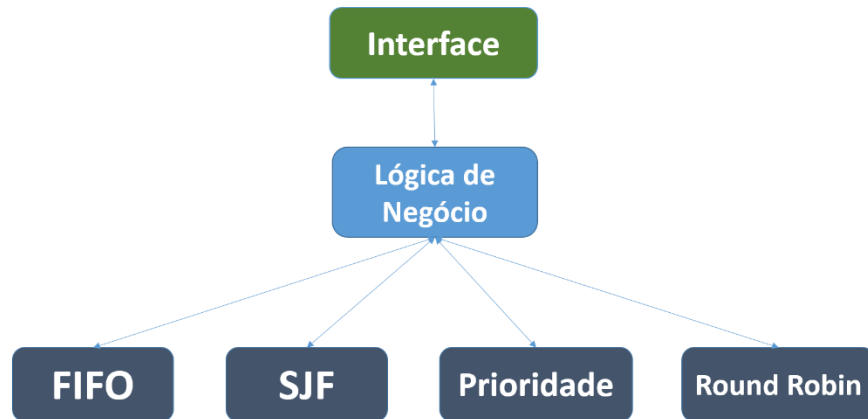
O ambiente de simulação foi projetado levando em consideração os critérios do paralelo das ferramentas de simulações existentes como apresentado na seção 2.4, procurando oferecer melhorias e sanar os problemas encontrados. Porém, a verificação da aplicação será realizada.

### 4.1 Arquitetura de software

Para entender o funcionamento da ferramenta de simulação que foi implementada a Figura 22 apresenta a arquitetura do ambiente.



**Figura 22: Arquitetura de desenvolvimento da ferramenta**

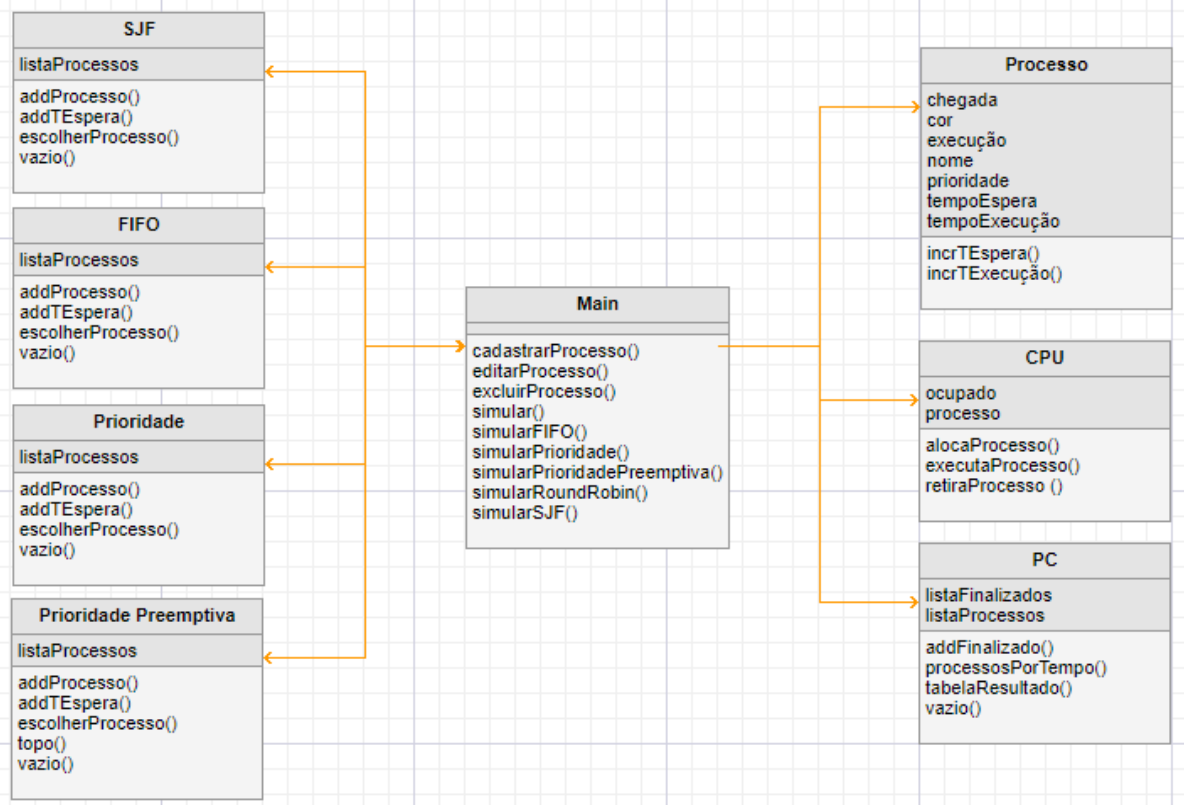


Conforme apresenta a Figura 22, o desenvolvimento da ferramenta foi desenvolvido com base nas seguintes camadas:

- Interface: nessa camada ocorre a interação do usuário com a ferramenta a ferramenta implementada;
- lógica de negócio: é a camada responsável por realizar toda a parte lógica da ferramenta. Nela são abordados os atributos e funcionalidades, além de ser encarregado de realizar as solicitações das classes e métodos dos algoritmos de escalonamento de processos.
- algoritmos de escalonamento de processos: nessa camada, todos os algoritmos possuem seus métodos e classes definidas.

Para detalhar mais sobre a arquitetura do projeto foi desenvolvido um diagrama de classe. A Figura 23 apresenta os métodos e classes utilizados no trabalho.

Figura 23: Diagrama de Classes



Nas seções seguintes, são apresentadas a arquitetura da ferramenta implementada, a interface do ambiente de simulação, exemplos das simulações dos algoritmos de escalonamento e a codificação das principais funcionalidades do sistema.

## 4.2 Interface inicial do sistema

Ao acessar a página do ambiente *web*, o usuário terá acesso a tela inicial do simulador de escalonamento de processos. Para representar as seções do sistema foram numeradas as áreas em que apresentam as opções de configuração e entradas de valores para realizar a simulação. Estas seções estão numeradas por área conforme a Figura 24.

Figura 24: Tela inicial



Para realizar a simulação o usuário deve informar o algoritmo de escalonamento que será simulado, os processos que serão utilizados na simulação e a configuração global caso o usuário utilizar o *Round Robin* para simular. As áreas das seções que o usuário utilizará numeradas na Figura 24 são:

- **Área 1** - escolha do algoritmo de escalonamento: nesta área o usuário deverá informar qual será o algoritmo de escalonamento de processos a ser representado na simulação. Os quatro algoritmos estão listados em uma caixa de combinação, para que o usuário escolha um deles;
- **Área 2** - cadastro de processos: área que permite ao usuário cadastrar os processos que irão concorrer com outros para utilização da CPU durante a simulação. O usuário deve informar, para cada processo, o nome, momento de chegada (transição), tempo de execução e a prioridade, sendo que este último parâmetro é utilizado para caso o usuário for simular o algoritmo de Prioridade. Algumas considerações importantes sobre a inserção de processos são:
  - o primeiro processo a ser cadastrado deverá possuir o tempo de chegada igual a zero, caso o usuário informe um valor diferente o sistema irá notificar o usuário para alterar o valor para 0;

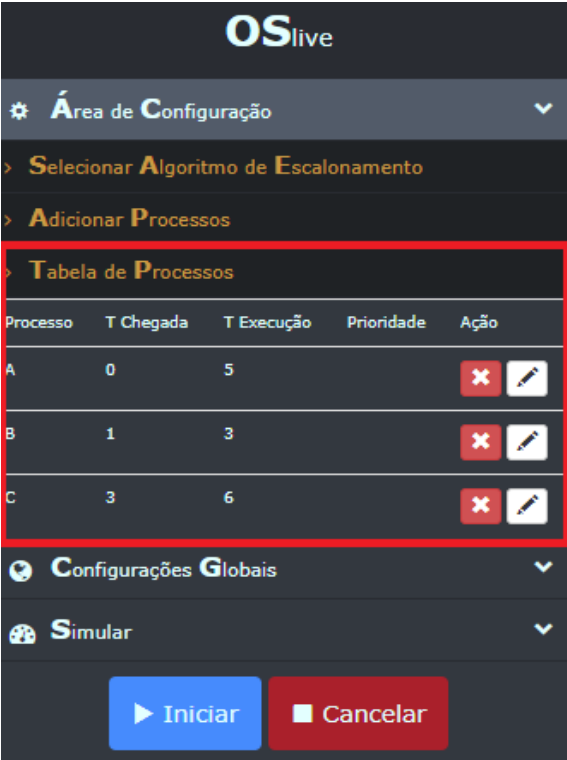
- o tempo de chegada dos processos pode ser inserido fora de ordem, com exceção do primeiro processo, de forma que a aplicação ordena automaticamente durante a execução da simulação;
- não há um limite de quantidade de processos, o usuário poderá cadastrar quantos processos desejar.
- **Área 3** - tabela de processos: nesta área o usuário pode conferir os processos que foram cadastrados em uma tabela, possibilitando a inserção, alteração de dados ou a exclusão do processo, isso pode ser feito antes ou depois da simulação ser realizada;
- **Área 4** - informar o quantum: nesta área, quando for realizar a simulação do algoritmo de escalonamento *Round Robin*, o usuário informa a fatia de tempo de uso da CPU pelos processos, cujo o valor deverá ser acima de zero. Caso o usuário informe um valor abaixo de 1, a aplicação irá notificar o usuário para informar um novo *Quantum*. Quando o usuário utilizar outro algoritmo este campo permanecerá habilitado, porém não será considerado.
- **Área 5** - simular algoritmo de escalonamento configurado: nesta área o usuário pode iniciar ou cancelar uma simulação. São oferecidas ao usuário a opção de simular os algoritmos com ou sem animação:
  - simulação com animação: será apresentada a animação do preenchimento do diagrama de uso da CPU, sendo que o diagrama será montado a cada segundo durante a simulação;
  - simulação sem animação: será apresentado o diagrama de uso da CPU já preenchido, é utilizado caso o usuário queira ver o resultado final do diagrama da CPU sem precisar esperar o preenchimento das colunas de tempo;
  - as tabelas de resultados e os gráficos são apresentados no momento em que o usuário inicia a simulação.







As áreas numeradas na aplicação foram implementadas para serem configuradas em sequência. Nas seções seguintes serão apresentadas as simulações dos algoritmos de escalonamento de processos bem como os trechos de códigos relevantes.

### 4.3 Simulação do algoritmo de escalonamento FIFO

Esta seção apresenta a simulação, com animação, do algoritmo de escalonamento FIFO, quando o usuário insere os processos apresentados na Figura 25. Os processos inseridos foram: processo “A” com tempo de chegada 0 e tempo de execução 5; processo “B” com tempo de chegada 1 e tempo de execução 3; e processo “C” com tempo de chegada 3 e tempo de execução 6.

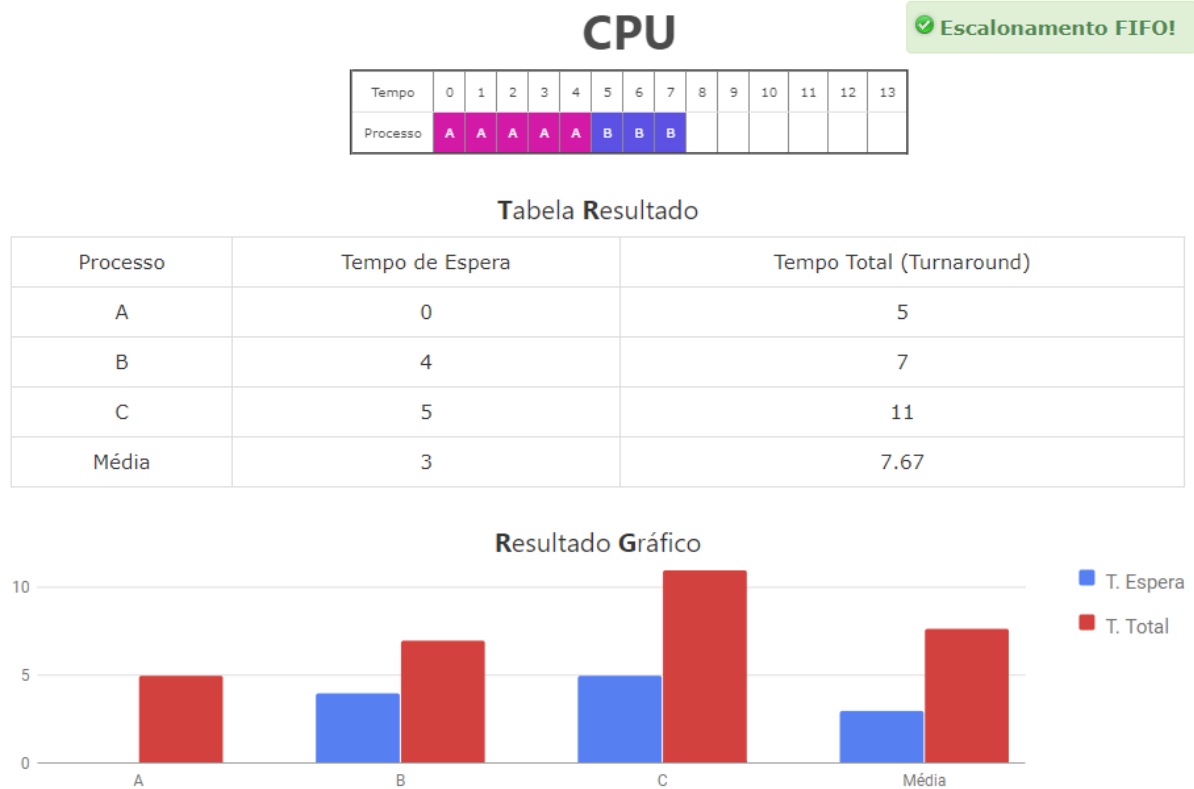
Figura 25: Tabela de processos cadastrados (FIFO)



Processo	T Chegada	T Execução	Prioridade	Ação
A	0	5		 
B	1	3		 
C	3	6		 

O algoritmo FIFO executa os processos em ordem de chegada, como já foi explicado na seção 2.2.1, desse modo, o processo “A” é executado, logo em seguida o processo “B” e, por fim, o processo “C”. Na simulação com animação do algoritmo FIFO, o diagrama de tempo é preenchido de forma sequencial, como mostrado nas Figuras 26 e 27.

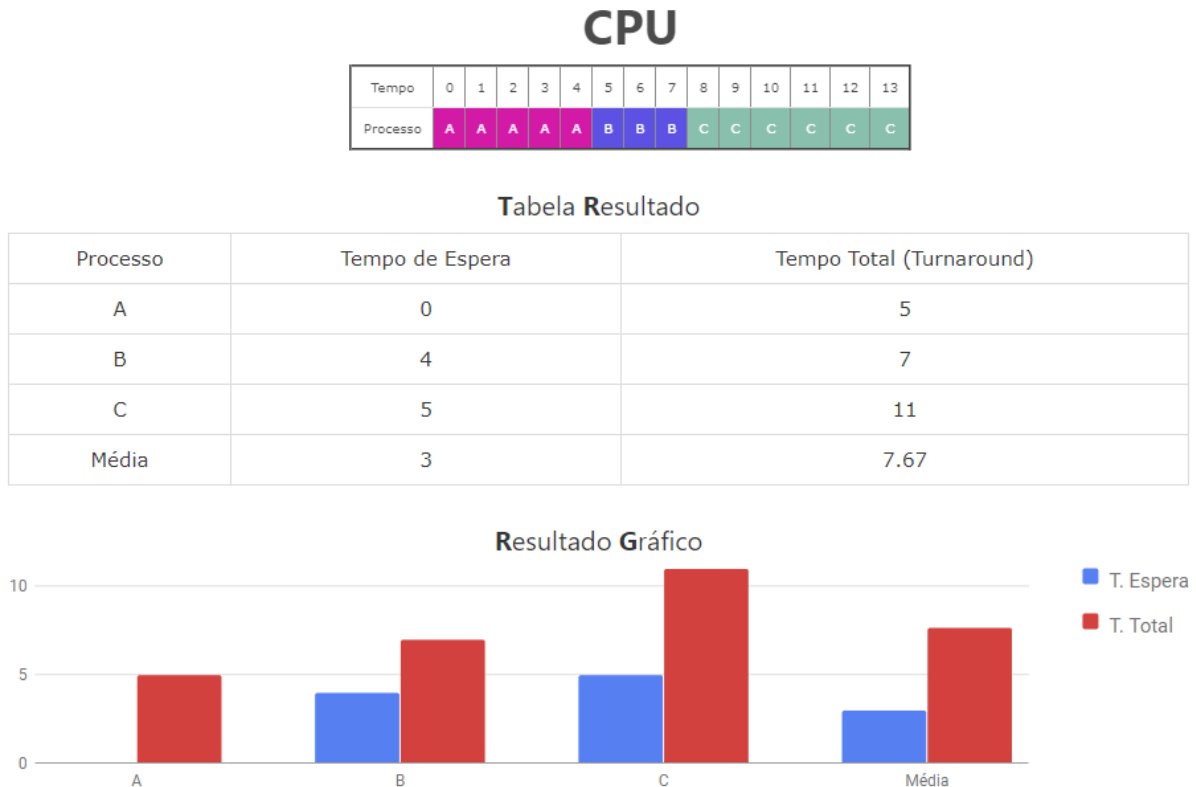
**Figura 26: Simulação em execução (FIFO)**



A Figura 26 apresenta um momento intermediário da simulação, após a execução dos processos “A” e “B”. Nesse ponto da simulação, no diagrama de uso da CPU foi mostrado que: o processo “A” executou do momento 0, que é seu momento de transição, ao momento 4, uma vez que seu tempo de execução foi 5; o processo “B” executou do momento 5, que foi o momento em que a CPU foi liberada pelo processo “A”, ao momento 7, pois seu tempo de execução foi 3.

A Figura 27 apresenta a simulação finalizada. Após a execução dos processos “A”, “B” e “C” apresentados na Figura 26. Neste momento da simulação, no diagrama foi mostrado que: o processo “C” executou do momento 8, que foi quando a CPU foi liberada pelo processo “B”, ao momento 13, considerando que seu tempo de execução foi 6.

**Figura 27: Simulação finalizada (FIFO)**



Ainda na Figura 27, pode ser observado que, logo abaixo do diagrama, os resultados da simulação são apresentados de duas formas:

- **modelo 1** – tabela resultado: tabela que apresenta o tempo de espera e o tempo total de cada processo e, ao final, a média destes tempos;
- **modelo 2** – resultado gráfico: gráfico com a representação dos dados da tabela resultado, contendo para cada processo, uma coluna de cor azul para o tempo de espera e uma coluna de cor vermelha para o tempo total.

Este modelo de resultado apresentado na simulação do algoritmo de escalonamento FIFO é padrão para as simulações de todos os algoritmos.

O método `simulandoFIFO()` é responsável por simular o algoritmo. O método é mostrado na Figura 28.

Figura 28: Método de escalonamento FIFO

```

121 function simulandoFIFO(){
122     var escalonar = new FIFO();
123     var resultado = [];
124     var status = new Array();
125     var aux;
126
127     time =0;
128
129     while(!pc.vazio() || !escalonar.vazio() || cpu.ocupado){
130         aux = pc.processosPorTempo(time);
131
132         while(aux.length >0){
133             escalonar.addProcesso(aux.shift());
134         }
135
136         if(!cpu.ocupado && !escalonar.vazio()){
137             cpu.alocaProcesso(escalonar.escolherProcesso());
138         }
139
140         if(cpu.ocupado){
141             aux = cpu.act();
142             if(!cpu.ocupado)
143                 pc.addfinalizado(aux);
144         }
145         else
146             aux = null;
147
148         escalonar.addTEspera(1);
149
150         if(aux != null)
151             resultado.push({nome:aux.nome, cor: aux.cor});
152         else
153             resultado.push({nome:"-", cor: "#FFFFFF"});
154
155         time++;
156     }
157
158     for(var i=0; i< resultado.length; i++){
159         status.push({tempo:i, nome:resultado[i].nome, cor: resultado[i].cor});
160     }
161
162     return status;
163 }

```

Como apresentado na Figura 28, o método **simulandoFIFO()** realiza a simulação do algoritmo de escalonamento FIFO. O método insere os processos na fila de aptos e verifica se a CPU está ocupada. Caso a mesma esteja ocupada é incrementado um tempo de espera para todos os processos da fila de aptos como mostra na linha 148 e executa o processo que está na CPU como apresentado na linha 141. Se a CPU não estiver ocupada e o possuir processos na fila de aptos, o primeiro da fila é inserido na CPU como visto na linha 137.

O método **tabelaResultado()** da classe **PC** realiza os cálculos para o preenchimento da tabela resultados, a Figura 29 apresenta este método.



Figura 29: Método da tabela resultado

```

31     this.tabelaResultado = function(){
32         var resultado = [];
33         var aux;
34         var espMed=0;
35         var turnMed =0;
36
37         this.listaFinalizados.sort(function(a,b){
38             return a.nome > b.nome;
39         });
40
41         for(i in this.listaFinalizados){
42             aux = this.listaFinalizados[i];
43             espMed += aux.tempoEspera;
44             turnMed += aux.tempoEspera + aux.execucao;
45             resultado.push({nome:aux.nome, espera: aux.tempoEspera, turn: aux.tempoEspera + aux.execucao});
46         }
47
48         espMed = parseFloat((espMed/this.listaFinalizados.length).toFixed(2));
49         turnMed = parseFloat((turnMed/this.listaFinalizados.length).toFixed(2));
50         resultado.push({nome:"Média",espera:espMed, turn:turnMed});
51         return resultado;
52     }
53







```

Conforme apresentado na Figura 29, o método **tabelaResultado()** é utilizado para realizar o cálculo do tempo de espera, tempo total e as médias desses tempos. Na linha 45, foi criada uma lista contendo o nome, o tempo de espera e o tempo total do processo, essa lista é chamada no método principal para ser apresentada na tabela de resultados e no resultado gráfico de resultados. Este método é utilizado para todos os algoritmos de escalonamento, assim, será explicado apenas nessa seção.

#### 4.4 Simulação do algoritmo de escalonamento SJF

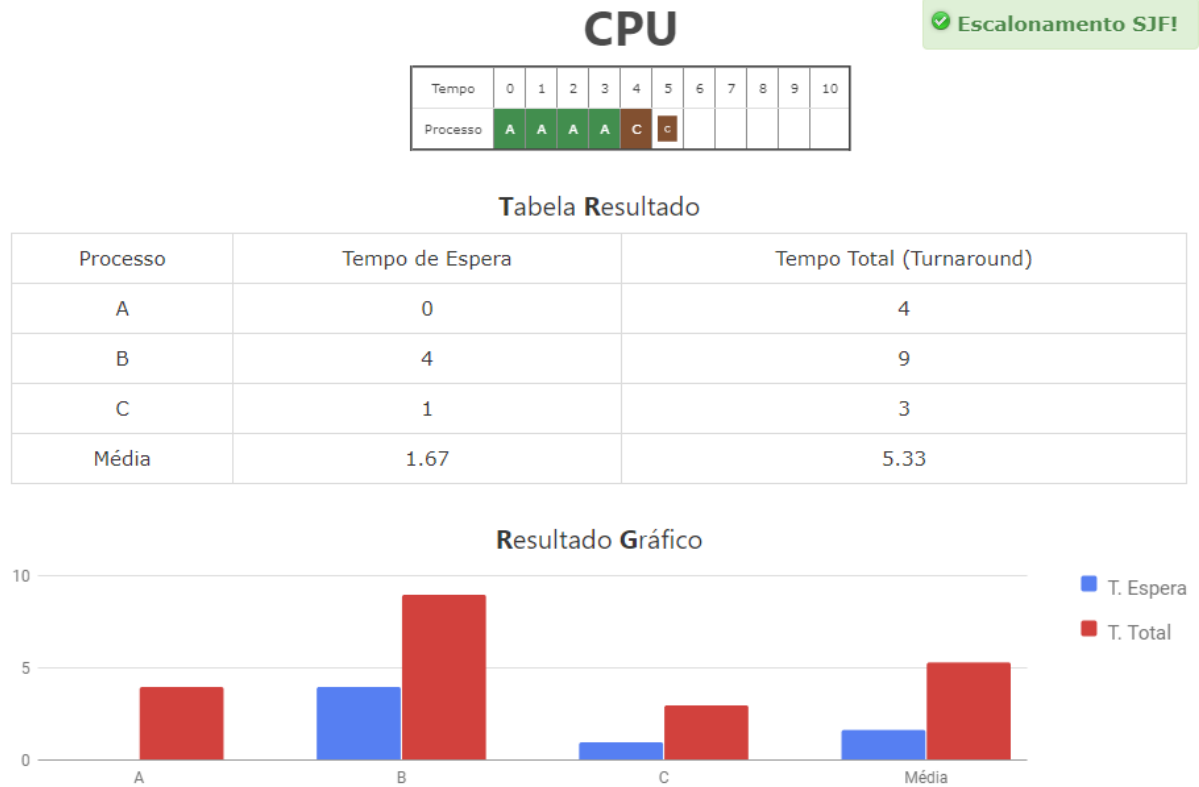
Esta seção apresenta a simulação, com animação, do algoritmo SJF não preemptivo, quando o usuário insere os processos apresentados na Figura 30. Os processos inseridos foram: processo “A” com tempo de chegada 0 e tempo de execução 4; processo “B” com tempo de chegada 2 e tempo de execução 5; e processo “C” com tempo de chegada 3 e execução 2.

Figura 30: Tabela de processos cadastrados (SJF)

Processo	T Chegada	T Execução	Prioridade	Ação
A	0	4		 
B	2	5		 
C	3	2		 

O algoritmo SJF ordena os processos por ordem de execução de forma não preemptiva, ou seja, do menor processo para o maior sem interrupções, como já foi explicado na seção 2.2.2, dessa maneira, o processo “A” é executado, visto que foi o primeiro a chegar na CPU, logo seguinte o processo “C” e por fim o processo “B”. Na simulação com animação do algoritmo SJF, o diagrama de tempo é preenchido da mesma forma que o algoritmo FIFO, como apresentado na Figura 31.

**Figura 31: Simulação em execução (SJF)**



A Figura 31 apresenta o momento intermediário da simulação, após a execução dos processos “A” e “C”. Neste momento da simulação, no diagrama de uso da CPU foi mostrado que: o processo “A” executou no momento 0 ao 3, visto que seu tempo de execução é de 4; o processo “C” executou do momento 4 ao momento 5, visto que o processo possui o tempo de execução menor que o processo “B”.

A Figura 32 apresenta a simulação finalizada. Após a execução dos processos “A” e “C” apresentados na Figura 31. Neste momento da simulação, foi mostrado no diagrama que: o processo “B” executou do momento 6, visto que foi o momento que a CPU foi liberada pelo processo “C”, ao momento 10, considerando que seu tempo de execução foi de 5.

**Figura 32: Simulação finalizada (SJF)**

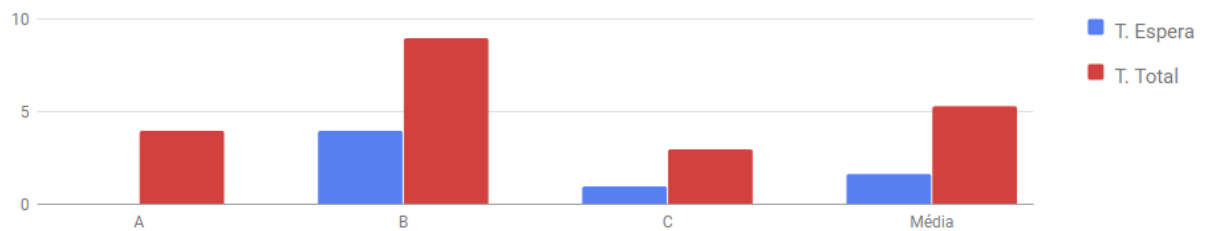
### CPU

Tempo	0	1	2	3	4	5	6	7	8	9	10
Processo	A	A	A	A	C	C	B	B	B	B	B

**Tabela Resultado**

Processo	Tempo de Espera	Tempo Total (Turnaround)
A	0	4
B	4	9
C	1	3
Média	1.67	5.33

**Resultado Gráfico**



Assim como na simulação do algoritmo FIFO, apresentado na seção 4.3, como resultado são apresentados a tabela e o gráfico dos resultados.

O método **simulandoSJF()**, é responsável por simular o algoritmo. O método é mostrado na Figura 33.

Figura 33: Método de escalonamento SJF

```

165 function simulandoSJF(){
166     var escalonar = new SJF();
167     var resultado = [];
168     var aux;
169     var status = new Array();
170
171     time=0;
172     while(!pc.vazio() || !escalonar.vazio() || cpu.ocupado){
173         aux = pc.processosPorTempo(time);
174
175         while(aux.length >0){
176             escalonar.addProcesso(aux.shift());
177         }
178
179         if(!cpu.ocupado && !escalonar.vazio()){
180             cpu.alocaProcesso(escalonar.escolherProcesso());
181         }
182
183         if(cpu.ocupado){
184             aux = cpu.act();
185             if(!cpu.ocupado)
186                 pc.addfinalizado(aux);
187         }
188         else
189             aux = null;
190
191         escalonar.addTEspera();
192
193         if(aux != null)
194             resultado.push({nome:aux.nome, cor: aux.cor});
195         else
196             resultado.push({nome:"-", cor: "#FFFFFF"});
197
198         time++;
199     }
200 }
201
202 for(var i=0; i< resultado.length; i++){
203     status.push({tempo:i, nome:resultado[i].nome, cor: resultado[i].cor});
204 }
205
206 return status;
207 }

```

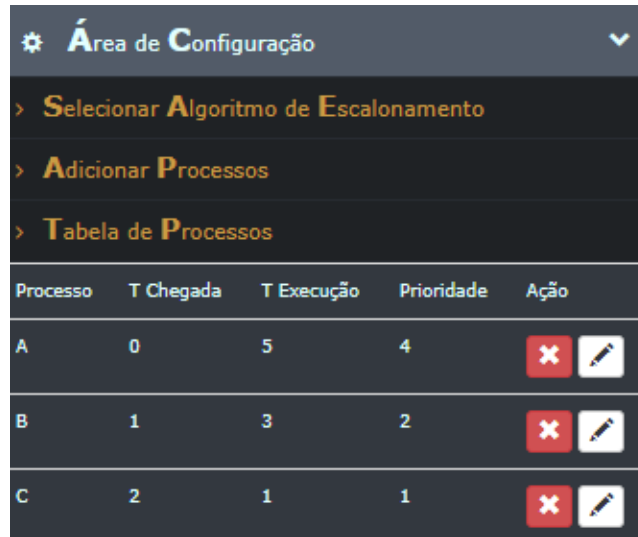
Como apresentado na Figura 33, o método **simulandoSJF()** realiza a simulação do algoritmo de escalonamento SJF. O método executa da mesma forma que o método de simulação do algoritmo FIFO (**simulandoFIFO**) muda apenas a ordenação por tempo de execução dos processos na fila de aptos. Por fim, o preenchimento da tabela de resultados apresentado na seção 4.3.




#### 4.5 Simulação do algoritmo de escalonamento prioridade (não-preemptivo/ preemptivo)

Esta seção apresenta a simulação, com animação, do algoritmo de Prioridade não preemptiva e preemptiva, quando o usuário insere os processos apresentados na Figura 34. Os processos inseridos foram: processo “A” com tempo de chegada 0, tempo de execução 5 e

prioridade 4; processo “B” com tempo de chegada 1, tempo de execução 3 e prioridade 2; processo “C” com tempo de chegada 3, tempo de execução 1 e prioridade 1.

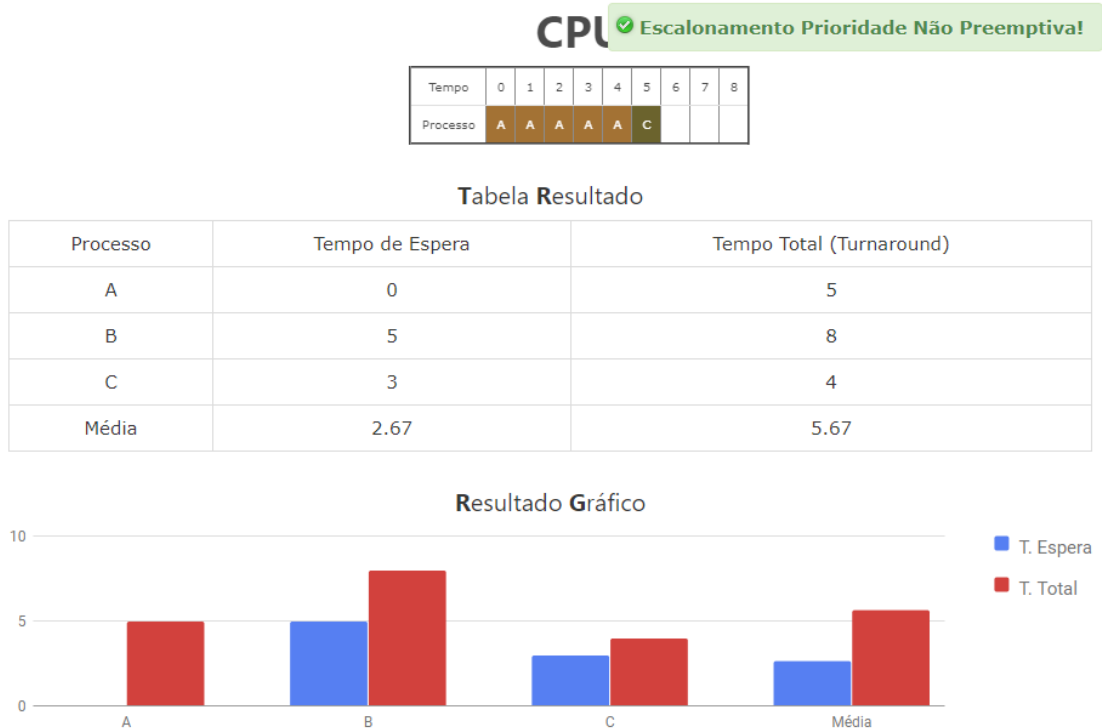
**Figura 34: Tabela de processos cadastrados (Prioridade)**



Processo	T Chegada	T Execução	Prioridade	Ação
A	0	5	4	 
B	1	3	2	 
C	2	1	1	 

O algoritmo de escalonamento por Prioridade executa primeiramente os processos que possui prioridade mais alta como já foi explicado na seção 2.2.3. Caso o usuário utilize o algoritmo de Prioridade não preemptiva, o processo “A” é executado na CPU, depois o processo “C” por ter prioridade maior que o processo “B”. Na simulação do algoritmo de Prioridade não preemptiva, o diagrama de tempo é preenchido de forma sequencial, como mostrado nas Figuras 35 e 36.

**Figura 35: Simulação em execução (Prioridade Não Preemptiva)**



A Figura 35 apresenta o momento intermediário da simulação não preemptiva, após a execução dos processos “A” e “C”. Neste momento da simulação, no diagrama de uso da CPU foi mostrado que: o processo e por último o processo “A” executou do momento 0, que é o momento de transição, ao momento 4, visto que seu tempo de execução foi 5; o processo “C” executou no momento 5, visto que o processo possui prioridade maior que o processo “B”, pois seu tempo de execução foi 1.

A Figura 36 apresenta a simulação finalizada. Após a execução dos processos “A”, “B” e “C”. Neste momento da simulação, no diagrama foi mostrado que: o processo “B” executou do momento 6, que foi o momento em que a CPU foi liberada pelo processo “C”, ao momento 8, visto que seu tempo de execução foi de 3.

**Figura 36: Simulação Finalizada (Prioridade Não Preemptiva)**

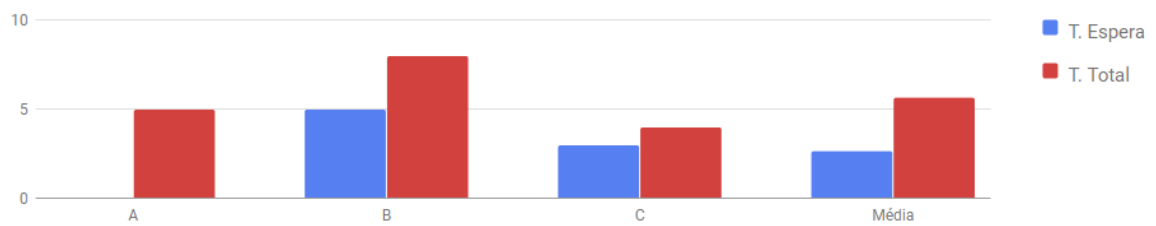
### CPU

Tempo	0	1	2	3	4	5	6	7	8
Processo	A	A	A	A	A	C	B	B	B

**Tabela Resultado**

Processo	Tempo de Espera	Tempo Total (Turnaround)
A	0	5
B	5	8
C	3	4
Média	2.67	5.67

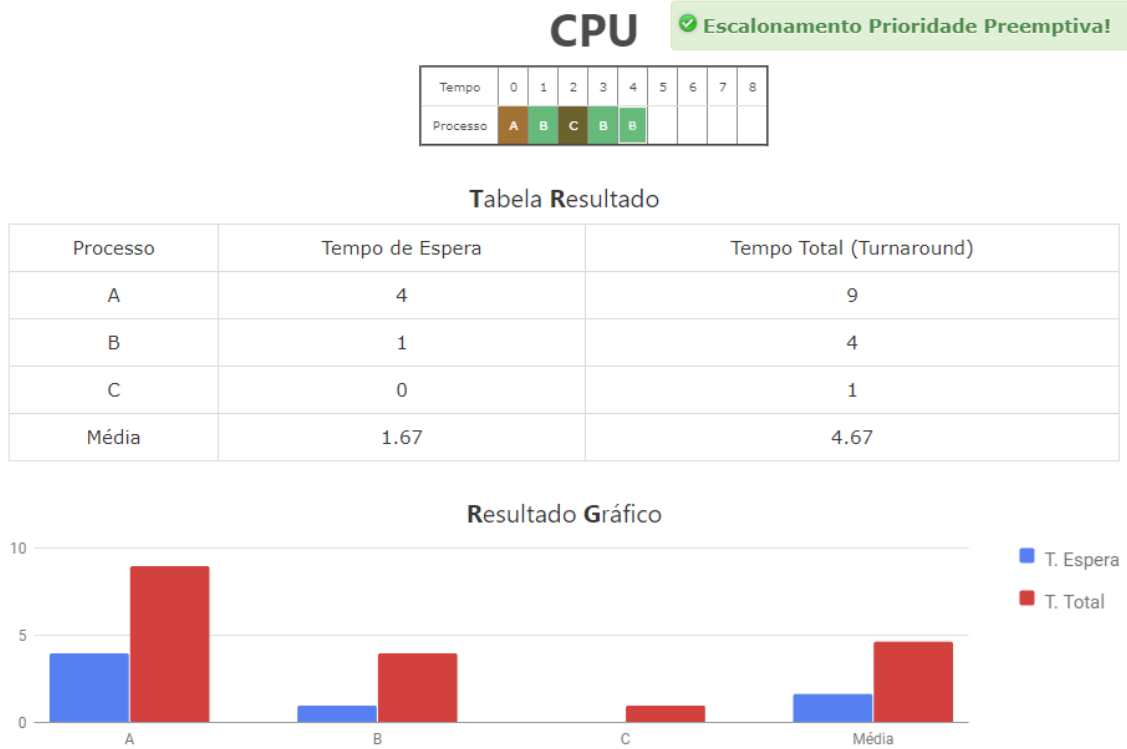
**Resultado Gráfico**



Na simulação do algoritmo de Prioridade preemptiva com os mesmos processos cadastrados, o diagrama de tempo é preenchido de forma sequencial, como apresentado nas Figuras 37 e 38.



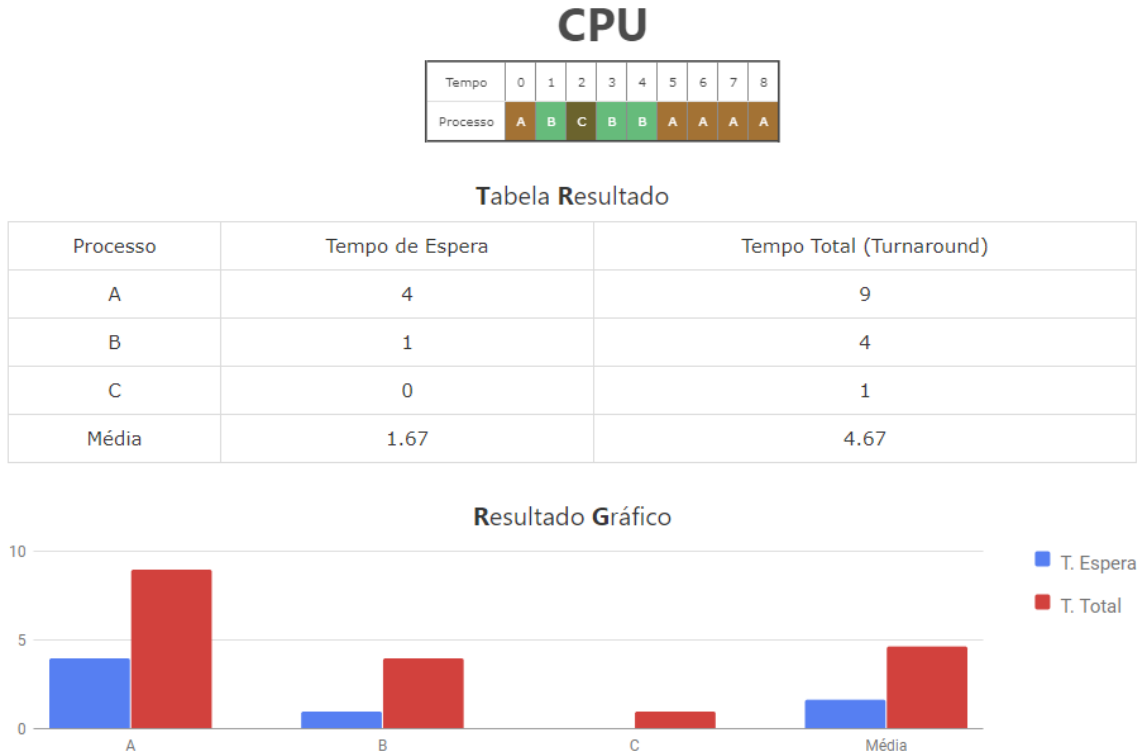
**Figura 37: Simulação em execução (Prioridade Preemptiva)**



A Figura 37 apresenta o momento intermediário da simulação do algoritmo de Prioridade preemptiva, durante a execução dos processos “A”, “B” e “C”. Neste ponto da simulação, no diagrama de uso da CPU foi mostrado que: o processo “A” executou no momento 0, e foi retirado no momento 1 da CPU para que o processo “B” execute, visto que o processo “B” tem prioridade 2; o processo “B” executou no momento 1, e foi retirado no momento 2 da CPU para que o processo “C” execute, considerando que o processo “C” tem prioridade 1 e tempo de execução 1; o processo “B” é executado novamente no momento 3, que foi o momento em que a CPU foi liberada pelo processo “C”, ao momento 4, pois seu tempo de execução foi de 3.

A Figura 38 apresenta a simulação finalizada. Após a execução dos processos “B” e “C” apresentados na Figura 37. Neste ponto da simulação, no diagrama foi mostrado que: o processo “A” executou novamente do momento 5, que foi o momento em que a CPU foi liberada pelo processo “B”, ao momento 8, considerando que seu tempo de execução foi de 5.

**Figura 38: Simulação Finalizada (Prioridade Preemptiva)**



Assim como na simulação do algoritmo FIFO, apresentado na seção 4.3, como resultado são apresentados a tabela e o gráfico dos resultados.

O método `simulandoPrio()`, método responsável por simular o algoritmo de Prioridade não preemptiva. O método é mostrado na Figura 39.

Figura 39: Método de escalonamento por Prioridade não preemptiva

```

209 function simulandoPrio(){
210     var escalonar = new Prioridade();
211     var resultado = [];
212     var aux;
213     var status = new Array();
214
215     time=0;
216     while(!pc.vazio() || !escalonar.vazio() || cpu.ocupado){
217         aux = pc.processosPorTempo(time);
218
219         while(aux.length >0){
220             escalonar.addProcesso(aux.shift());
221         }
222
223         if(!cpu.ocupado && !escalonar.vazio()){
224             cpu.alocaProcesso(escalonar.escolherProcesso());
225         }
226
227         if(cpu.ocupado){
228             aux = cpu.act();
229             if(!cpu.ocupado)
230                 pc.addfinalizado(aux);
231         }
232         else
233             aux = null;
234
235         escalonar.addTEspera();
236
237         if(aux != null)
238             resultado.push({nome:aux.nome, cor: aux.cor});
239         else
240             resultado.push({nome:"-", cor: "#FFFFFF"});
241
242         time++;
243     }
244
245     for(var i=0; i< resultado.length; i++){
246         status.push({tempo:i, nome:resultado[i].nome, cor: result
247     }
248
249     return status;
250 }
251 }

```

Como apresentado na Figura 39, o método **simulandoPrio()** realiza a simulação do algoritmo de escalonamento por Prioridade não preemptiva. O método também executa da mesma forma que o método de simulação do algoritmo FIFO o que muda é apenas a ordenação por prioridade na fila de aptos. E por fim, o preenchimento da tabela de resultados apresentado na seção 4.3.

O método **simulandoPrioPremp()**, método encarregado de simular o algoritmo por prioridade preemptiva. O método é apresentado na Figura 40.

Figura 40: Método de escalonamento Prioridade Preemptiva

```

253 function simulandoPrioPremp(){
254     var escalonar = new PrioridadePreemp();
255     var troca = 0;
256     var resultado = [];
257     var aux;
258     var chaveamento;
259     var status = new Array();
260
261     time=0;
262     chaveamento=0;
263
264     while (!pc.vazio() || !escalonar.vazio() || cpu.ocupado)
265     {
266         aux = pc.processosPorTempo(time);
267
268         while (aux.length > 0)
269             escalonar.addProcesso( aux.shift() );
270
271         if (chaveamento > 0)
272         {
273             resultado.push({nome:"-", cor: "#FFFFFF"});
274             escalonar.addTEspera(1);
275             time ++;
276             chaveamento --;
277             continue;
278         }
279
280         if(!escalonar.vazio()){
281             if(!cpu.ocupado){
282                 cpu.alocaProcesso(escalonar.escolherProcesso());
283             }
284             else if(escalonar.topo().prioridade < cpu.processo.prioridade){
285                 aux = cpu.retiraProcesso();
286                 escalonar.addProcesso(aux);
287                 chaveamento = troca;
288                 continue;
289             }
290         }
291
292         if (cpu.ocupado)
293         {
294             aux = cpu.act();
295             if (!cpu.ocupado)
296             {
297                 pc.addfinalizado(aux);
298                 chaveamento = troca;
299             }
300         }

```

Como apresentado na Figura 40, o método **simulandoPrioPremp()** realiza a simulação do algoritmo por Prioridade preemptiva. O trecho de código responsável por realizar o escalonamento por prioridade de forma preemptiva são entre as linhas 284 a 288.


O preenchimento da tabela de resultados e gráficos de resultados ocorre da mesma forma do algoritmo FIFO apresentado na seção 4.3.

#### 4.6 Simulação do algoritmo de escalonamento *ROUND ROBIN*

Esta seção apresenta a simulação, com animação, do algoritmo de escalonamento Round Robin, em que o usuário insere os processos apresentados na Figura 41. Os processos inseridos foram: processo “A” com tempo de chegada 0 e tempo de execução 5; processo “B” com tempo de chegada 3 e tempo de execução 4; processo “C” com tempo de chegada 4 e tempo de execução 3, considerando que o tempo *Quantum* é 3.

**Figura 41: Tabela de processos cadastrados (RR)**



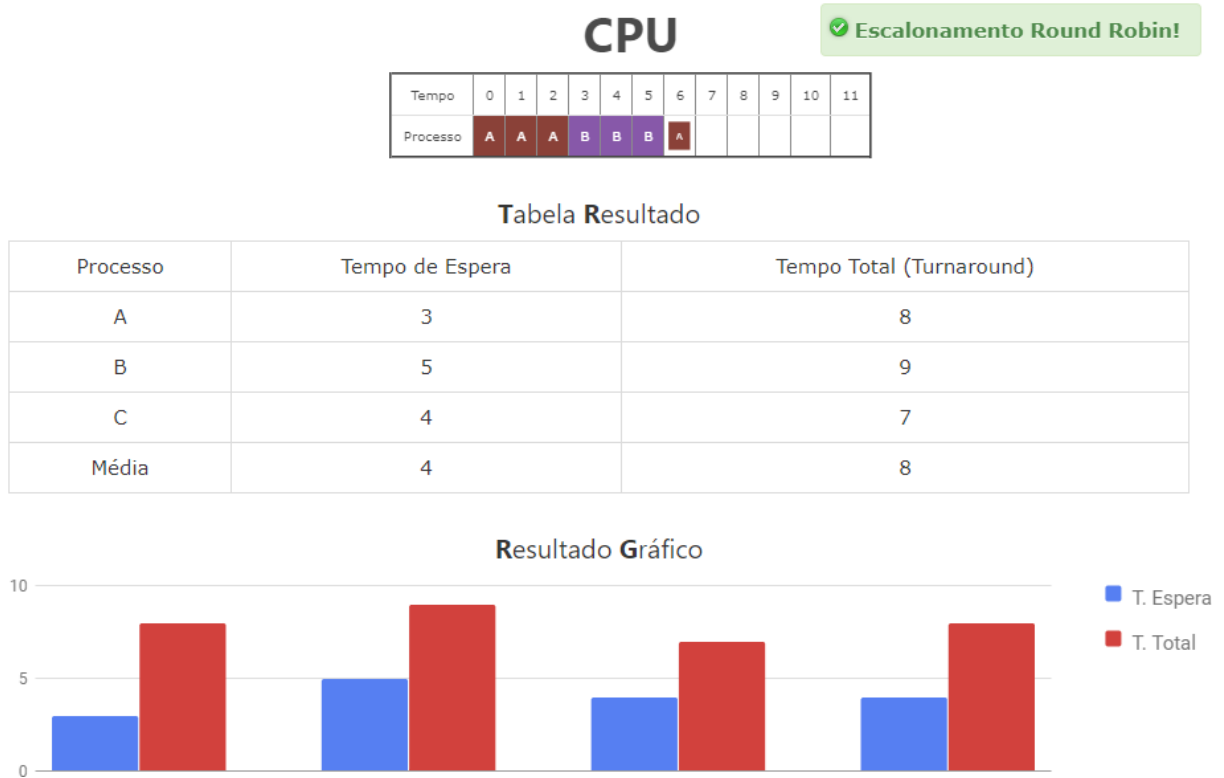
Processo	T Chegada	T Execução	Prioridade	Ação
A	0	5		 
B	3	4		 
C	4	3		 

Configurações Globais

Quantum:

O algoritmo RR executa os processos da mesma forma que o algoritmo FIFO, mas para cada processo é definido uma fatia tempo na CPU chamada *Quantum*, como esclarecido na seção 2.2.4. Desta forma, o processo “A” executa até o limite *quantum*, logo em seguida o processo “B” executa até o limite do *quantum* e, por fim, o processo “C”. Esta execução ocorre de forma circular até que os processos terminem sua execução. Na simulação com animação do algoritmo RR, o diagrama de tempo é preenchido de maneira sequencial, como apresentado nas Figuras 42 e 43.

**Figura 42: Simulação em execução (RR)**



A Figura 42 apresenta um momento intermediário da simulação, durante a execução dos processos “A” e “B”. Neste ponto da simulação, foi mostrado no diagrama de uso da CPU que: o processo “A” executou do momento 0, e foi retirado no momento 3 da CPU para que o processo “B” execute, visto que o *quantum* atingiu seu limite para o processo “A” restando 2 tempos; o processo “B” executou no momento 3, momento em que a CPU liberou para o processo “B”, ao momento 5, em que o limite do *quantum* foi atingido para o processo “B” restando 1 tempo.

A Figura 43 apresenta a simulação finalizada. Durante a execução dos processos “A” e “B”. Neste momento da simulação, no diagrama foi apresentado que: o processo “A” executou no momento 6 até o momento 7, visto que o processo ainda restava 2 tempos e não atingiu o limite do *quantum*; o processo “C” executou no momento 8, até o momento 10, visto que o mesmo não atingiu o limite *quantum*, considerando que seu tempo de execução é de 3; por fim o processo “B” que ainda restava 1 tempo de execução.

**Figura 43: Simulação Finalizada (RR)**

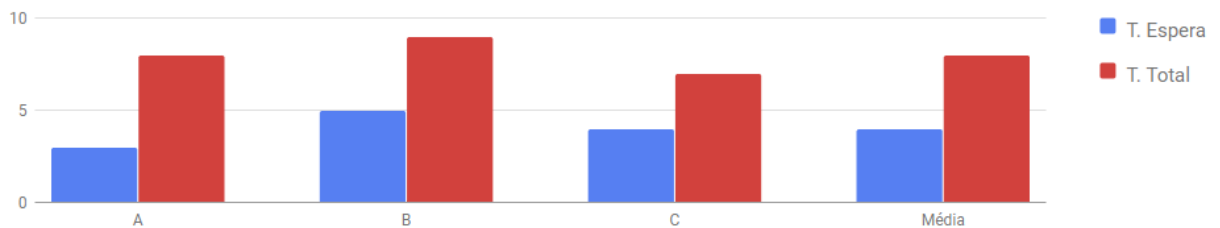
## CPU

Tempo	0	1	2	3	4	5	6	7	8	9	10	11
Processo	A	A	A	B	B	B	A	A	C	C	C	B

**Tabela Resultado**

Processo	Tempo de Espera	Tempo Total (Turnaround)
A	3	8
B	5	9
C	4	7
Média	4	8

**Resultado Gráfico**



Assim como nos demais algoritmos de escalonamento apresentados nas seções anteriores, os resultados são apresentados na tabela de resultados e gráficos de resultados.

O método **simulandoRR(q)** é responsável por simular o algoritmo. O método é apresentado na Figura 44.

Figura 44: Método de escalonamento RR

```

322 function simulandoRR(q){
323     var escalonar = new FIFO();
324     var quantun = q;
325     var troca =0;
326     var resultado = [];
327     var aux,tmp;
328     var chaveamento;
329     var status = new Array();
330
331     tmp =0;
332     time=0;
333     chaveamento=0;
334
335     while (!pc.vazio() || !escalonar.vazio() || cpu.ocupado)
336     {
337         aux = pc.processosPorTempo(time);
338
339         while (aux.length > 0)
340             escalonar.addProcesso( aux.shift() );
341
342         if (chaveamento > 0)
343         {
344             aux = cpu.retiraProcesso();
345             escalonar.addProcesso(aux);
346             chaveamento = 0;
347         }
348
349         if (tmp == 0 && !escalonar.vazio())
350             cpu.alocaProcesso(escalonar.escolherProcesso());
351
352         if (cpu.ocupado)
353         {
354             tmp++;
355             aux = cpu.act();
356             if (!cpu.ocupado)
357             {
358                 pc.addfinalizado(aux);
359                 tmp = 0;
360             }
361         }
362         else
363             aux = null;
364
365         escalonar.addTEspera(1);
366
367         if (aux != null)
368             resultado.push({nome:aux.nome, cor: aux.cor});
369         else

```

Como apresentado na Figura 44, o método **simulandoRR(q)** realiza a simulação do algoritmo de escalonamento *Round Robin*. O método recebe o valor de *quantum* e executa da mesma forma do algoritmo de escalonamento FIFO, a diferença é que o método possui uma variável chamada **tmp**, este tempo é incrementado durante a execução de um processo na linha 354. Quando a variável tmp for igual ao valor *quantum* é realizado a troca do processo na linha 345.

O preenchimento da tabela de resultados e gráficos de resultados ocorre da mesma forma que os algoritmos apresentados nas seções anteriores.



## 5 CONSIDERAÇÕES FINAIS

O presente trabalho teve como objetivo implementar um ambiente *web* para demonstrar o funcionamento dos algoritmos de escalonamentos de processos de forma gráfica com a utilização de animação para auxiliar no processo de aprendizagem dos alunos que cursam a disciplina de Sistemas Operacionais.

A ferramenta foi desenvolvida na plataforma *web* com intenção de eliminar a complexidade de instalação, sem a necessidade de o usuário realizar instalação de programas da ferramenta ou de terceiros para executar as simulações em seu computador.

Ao implementar a ferramenta, buscou-se desenvolver de forma simples e que atendesse as necessidades do usuário na compreensão do funcionamento dos principais algoritmos de escalonamento de processos, em que foram baseados em livros de autores renomados na área de Sistemas Operacionais aumentando a fidelidade do conteúdo. Na ferramenta é possível utilizar apenas configurações essenciais como cadastrar os processos, escolher o algoritmo de simulação e informar o *Quantum*, reduzindo a complexidade de utilização da ferramenta.

A ferramenta que será avaliada no semestre 2018/1 na disciplina de Sistemas Operacionais com base nos seguintes critérios definidos na Tabela 5. Será considerado a interação da ferramenta com o usuário, para perceber se houve resultados positivos na relação entre aluno e ferramenta.

Como trabalhos futuros, pretende-se realizar melhorias que as avaliações jogarem necessárias, adicionar conteúdo na ferramenta para que o usuário pudesse estudar o conteúdo e já realizar os testes na ferramenta e acrescentar outros algoritmos de simulações como:

- algoritmos de múltiplas filas: é o algoritmo que constrói várias filas de aptos em que cada processo é agregado a uma dessas filas, cada uma dessas filas possui uma prioridade associada e o próprio método de escalonamento, é executado primeiro a fila que tiver maior prioridade;
- simular o algoritmo de escalonamento do Linux: algoritmo que divide o tempo do processamento em épocas (*epochs*). No momento da criação de um processo, é definido um *quantum* calculado no começo de uma época. Cada processo pode possuir diferentes valores de *quantum* (OLIVEIRA; CARISSIMI; TOSCANI, 2008).

## 6 REFERÊNCIAS

MEC. **Parecer CNE/CES nº 136/2012, aprovado em 8 de março de 2012.** 2012. Disponível em: <[http://portal.mec.gov.br/index.php?option=com\\_docman&view=download&alias=11205-pces136-11-pdf&category\\_slug=julho-2012-pdf&Itemid=30192](http://portal.mec.gov.br/index.php?option=com_docman&view=download&alias=11205-pces136-11-pdf&category_slug=julho-2012-pdf&Itemid=30192)>. Acesso em: 25 jun. 2017.

MASSA, Ernesto. **Sistemas Operacionais - Gerência de Processos.** 2009. Disponível em: <<https://pt.slideshare.net/computacaodepressao/2009-1-sistemas-operacionais-aula-4-threads-e-comunicacao-entre-processos-15928232>>. Acesso em: 15 abr. 2017.

OTTO, Mark; JACOB. **Começando:** Uma visão geral do Bootstrap. 2010. Disponível em: <<https://v4-alpha.getbootstrap.com/getting-started/introduction/>>. Acesso em: 07 maio 2017.

OLIVEIRA, Rômulo Silva de; CARISSIMI, Alexandre da Silva; TOSCANI, Simão Sirineo. **Sistemas Operacionais.** 3. ed. Porto Alegre: Editora Bookman, 2008. 259 p. Disponível em: <<http://www.inf.ufrgs.br/~asc/livro/secao94.pdf>>. Acesso em: 21 dez. 2017.

SILBERSCHATZ, Abraham; GALVIN, Peter Baer; GAGNE, Greg. **Fundamentos de Sistemas Operacionais.** 6. ed. Teresópolis - Rj: Ltc, 2001. 580 p. Tradução de: Elisabete do Rego Lins.

STUART, Brian L.. **Princípios de sistemas operacionais:** Projetos e aplicações. São Paulo: Cengage Learning, 2011. 655 p. Tradução de: All Tasks.

TANENBAUM, Andrew S.. **Sistemas Operacionais Modernos.** 3. ed. São Paulo: Pearson, 2009. Tradução de: Prof. Dr. Ronaldo A. L. Gonçalves, Prof. Dr. Luís A. Consularo, Luciana do Amaral Teixeira. Disponível em: <<http://ulbra.bv3.digitalpages.com.br/users/publications/9788576052371/pages/89>> [Biblioteca Virtual]. Acesso em: 07 maio 2017.

TANENBAUM, Andrew S.; WOODHULL, Albert S.. **Sistemas Operacionais:** Projeto e Implementação. 2. ed. Porto Alegre: Bookman, 2000. 69 p. Tradução de: Edson Furmankiewicz.

TANENBAUM, Andrew S.; BOS, Herbert. **Sistemas Operacionais Modernos**. 4. ed. São Paulo: Pearson Education do Brasil, 2016. 103 p. Tradução de: Daniel Vieira e Jorge Ritter. Disponível em: <<http://ulbra.bv3.digitalpages.com.br/users/publications/9788543005676/pages/-19>>. Acesso em: 10 maio 2017.

VIEIRA, Fábila Magali Santos. **Avaliação de Software Educativo: Reflexões para uma Análise Criteriosa**. 1999. Disponível em: <[http://s3.amazonaws.com/academia.edu.documents/32016960/Avaliacao\\_de\\_Software\\_Educativo\\_Reflexoes\\_para\\_uma\\_Analise\\_Criteriosa.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1497589881&Signature=je9Hm7lOsMuSh9Fy81g6ull8YgY%3D&response-content-disposition=inline%3B%20filename%3DAvaliacao\\_de\\_Software\\_Educativo\\_Reflexoe.pdf](http://s3.amazonaws.com/academia.edu.documents/32016960/Avaliacao_de_Software_Educativo_Reflexoes_para_uma_Analise_Criteriosa.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1497589881&Signature=je9Hm7lOsMuSh9Fy81g6ull8YgY%3D&response-content-disposition=inline%3B%20filename%3DAvaliacao_de_Software_Educativo_Reflexoe.pdf)>. Acesso em: 15 abr. 2017.

## **7 APÊNDICES**

## **8 ANEXOS**