SWSO - Simulador Web de Sistemas Operacionais

Ramon Almeida Oliveira *
Instituto Federal da Bahia
Rua Emídio dos Santos, S/N
Barbalho, Salvador Bahia
ramon.oliveirah@gmail.com

Antonio Carlos dos Santos Souza †
Instituto Federal da Bahia
Rua Emídio dos Santos, S/N
Barbalho, Salvador Bahia
acsantossouza@gmail.com

RESUMO

Este trabalho apresenta uma ferramenta web para auxiliar o ensino-aprendizagem da disciplina de sistemas operacionais. A ferramenta tem como objetivo simular o funcionamento de um Sistema Operacional, explorando os conceitos de gerência de disco, memória e processos de forma integrada e interativa. Além disso, o software foi desenvolvido para permitir que qualquer pessoa com um mínimo de conhecimento na linguagem Java possa baixar o código da aplicação e implementar novos algoritmos de forma simples.

Palavras Chave

Ferramenta de ensino-aprendizagem, simulador, sistemas operacionais, gerência de disco, memória e processos

1. INTRODUÇÃO

A disciplina de Sistemas Operacionais está presente na maioria dos cursos de computação, ela normalmente é apresentada para os alunos nos primeiros semestres e tem o objetivo de fazer o estudante entender como funcionam os Sistemas Operacionais e tudo que é feito por eles para abstrair os detalhes de hardware para o usuário, gerenciando os diversos componentes de um sistema computacional para fornecer aos programas uma interface mais simples [12].

O estudo de Sistemas Operacionais é complexo, e demanda do aluno alta capacidade de abstração, principalmente quando envolve assuntos como algoritmos de escalonamento, gerência de processos e memória virtual. Esta complexidade, associada ao fato que o aluno muitas vezes está no início da vida acadêmica, torna muito complicado o ensino da disciplina.

Desse modo, para que a aprendizagem seja mais fácil são necessárias aulas dinâmicas, com forte interação entre o professor e o aluno, dando aos alunos a possibilidade de construir o próprio conhecimento. Entretanto existe uma carência de boas ferramentas de software disponíveis para promover tal interação e facilitar esse aprendizado.

O SWSO(Simulador Web de Sistemas Operacionais) tem o objetivo de oferecer um software confiável para a simulação do funcionamento de um sistema operacional. O simulador visa oferecer ao professor uma ferramenta online de apoio às aulas, com o foco nos seguintes conceitos: Gerência de Disco, Gerência de Memória(Memória Virtual) e Gerência de Processos.

O presente trabalho tem a proposta de aumentar o nível de interação entre professor e aluno durante as aulas. Esta interação aconterá porque, com o uso da ferramenta, o aluno vai poder monitorar a simulação que está sendo realizada pelo professor. Esse monitoramento poderá ser feito utilizando qualquer computador com acesso à Internet.

Outro diferencial a ser destacado é que a simulação é feita por completo, ou seja, mesmo que o foco da aula seja gerência de memória, será possível a qualquer momento visualizar a simulação do funcionamento dos algoritmos de gerenciamento de disco ou processos. Isso é importante para que o aluno passe a integrar tudo o que foi aprendido durante as aulas de sistemas operacionais [4, 7].

Este trabalho está estruturado da seguinte maneira: A seção 2 apresenta o referencial teórico necessário para a construção do simulador. A seção 3 contém os trabalhos correlacionados ao tema proposto. A seção 4 apresenta o modelo proposto, com seus devidos diagramas de classes, bem como requisitos funcionais e não funcionais e os pontos de mais relevância para o desenvolvimento do software. Na quinta seção são apresentados os resultados obtidos. Por último, na sexta seção é feita a conclusão e as sugestões para futuros trabalhos.

2. REFERENCIAL TEÓRICO

Apesar de toda a complexidade envolvida no seu funcionamento, os sistemas operacionais são apenas rotinas executadas em um sistema computacional, semelhantes às aplicações convencionais [6].

Entretanto, diferente dos programas comuns que normalmente têm inicio meio e fim bem definidos, as suas rotinas executam de forma concorrente e em função de eventos as-

*Aluno do curso de Análise e Desenvolvimento de Sistemas
†Doutor em Ciência da Computação e Professor do Curso de Análise e Desenvolvimento de Sistemas

síncronos. As principais funções de um sistema operacional são: Estender a máquina e gerenciar recursos [12].

Como máquina estendida, realizam o papel de fornecer para os usuários uma interface mais simples e agradável para operações de hardware complexas [6, 12]. Assim, o usuário não precisa conhecer os detalhes sobre a comunicação com os diferentes hardwares (Monitores de vídeo, impressoras, unidades de CD/DVD, Discos Rígidos).

Como gerenciador de recursos, o sistema operacional funciona como um mediador de conflitos. Ele tem o papel de coordenar as requisições de diferentes programas e usuários ao mesmo recurso de hardware [12].

Por exemplo, se diferentes aplicações tentam utilizar a impressora ao mesmo tempo, será papel do sistema operacional coordenar essas diferentes requisições, de modo que todos consigam realizar a tarefa de imprimir, sem que o conteúdo enviado por uma aplicação influencie no enviado por outra.

2.1 Sistemas de Arquivos

Todo processo em execução é capaz de armazenar informações dentro da sua área de endereçamento. No entanto, para alguns programas existe a necessidade de armazenar além dos limites de endereçamento virtual, isto acontece por três diferentes razões: É necessário armazenar uma quantidade grande de informação; a informação deve continuar existindo mesmo depois do término do processo; vários processos devem ser capazes de manipular a mesma informação [12].

Uma vez que habitualmente a memória principal é volátil e não suporta uma quantidade grande de dados, torna-se necessária a utilização de uma memória secundária para o armazenamento persistente. Por este motivo, as informações são salvas em uma unidade de armazenamento chamada de arquivo, os arquivos são informações logicamente relacionadas e podem representar instruções ou dados. A parte do sistema operacional responsável pelo gerenciamento dos arquivos é o sistema de arquivos.

O sistema de arquivos é o aspecto mais visível de um sistema operacional para a maioria dos usuários [6]. No uso diário do computador, o usuário está constantemente criando, movendo, apagando e modificando arquivos. Processos estão frequentemente manipulando arquivos de forma concorrente no mesmo dispositivo.

Assim, o sistema de arquivo existe para mediar o acesso concorrente e facilitar a leitura e escrita dos arquivos para os usuários, além disso, é responsável por tornar a manipulação dos arquivos uma tarefa uniforme e independente dos dispositivos físicos utilizados para armazenar a informação.

Para isso, o sistema de arquivo fornece um conjunto de operações que podem ser realizadas em um arquivo, sendo as mais comuns: criação, gravação, leitura, exclusão, reposicionamento e truncamento [9].

Nos sistemas operacionais atuais, os arquivos são identificados pelo seu nome, que é normalmente uma sequência de caracteres. Além do nome, os arquivos possuem outros atributos que variam nos diferentes sistemas operacionais, sendo os mais comuns: identificador, tipo, localização, tamanho, tempo, data e identificação do usuário.

Uma vez criados e definidos os seus atributos, um arquivo torna-se independente do processo, usuário e sistema operacional que o criou. Podendo ser facilmente copiado para outro dispositivo [9].

2.2 Gerenciamento de Memória

A memória é um recurso essencial dentro de um sistema computacional, afinal é nela que são armazenadas as informações de todos os programas. A memória ideal teria que ser infinitamente grande, extremamente rápida, não volátil e de baixo custo[12].

No entanto, apesar do grande avanço tecnológico nas últimas décadas, esta memória ainda não existe. Assim, o gerenciamento de memória continua sendo um dos fatores mais importantes no projeto de um sistema operacional, embora atualmente a memória seja muito mais rápida e barata se comparada a outros momentos no passado.

A parte do sistema operacional responsável pela organização e estratégias de gerenciamento da memória é chamada gerenciador de memória, cabe a ele as tarefas de alocar e liberar memória conforme demanda dos processos, manter o controle das partes da memória disponíveis e das que estão em uso e gerenciar a troca de processos entre a memória e o disco [2].

Na época em que a computação era monoprogramada, o papel do gerenciador de memória era muito mais simples do que atualmente. Por exemplo, para alocar um processo na memória era necessário apenas verificar se havia espaço suficiente, caso não houvesse, o programa não era executado e o problema era repassado para o usuário, que o solucionava colocando mais memória ou diminuindo o tamanho do software.

Porém, com a utilização da multiprogramação, a memória passou a ser compartilhada por diversos processos. Assim, alguns outros fatores foram adicionados ao gerenciamento da memória, tornando a tarefa de gerenciar a memória uma das mais complexas de um projeto de sistemas operacionais [12].

As estratégias de gerenciamento de memória buscam otimizar a utilização da memória principal, elas estão divididas em três grupos: estratégias de busca, de posicionamento e de substituição.

A primeira é utilizada para determinar quando a próxima porção do processo será transferida para a memória principal; A segunda o local da memória em que serão armazenados os dados que estão chegando; e a terceira define, para casos em que a memória principal esteja muito cheia, qual o processo que deverá ser retirado da memória para dar espaço aos que estão chegando [2].

2.2.1 Memória Virtual

O modelo de gerenciamento de memória utilizando a técnica de memória virtual é uma solução amplamente utilizada pelos sistemas operacionais atuais, esta técnica combina me-

mória principal e secundária, a fim de simular a existência de uma memória muito maior do que a que realmente existe. Deste modo, torna-se possível que um número maior de processos compartilhem a mesma memória principal [6].

Outra vantagem na utilização de memória virtual, é que mesmo a memória principal estando totalmente ocupada, ainda é possível iniciar novos processos. Por último, com o uso de memória virtual é admissível a execução de programas muito maiores que a memória real disponível.

Semelhante ao que ocorre com os "arrays" em linguagens de alto nível, para que a técnica de endereçamento virtual funcione, é necessário abstrair a real localização dos dados na memória, para isto toda vez que um processo tenta acessar um endereço virtual é necessária a tradução para um endereço real, conforme ilustrado na Figura 1.

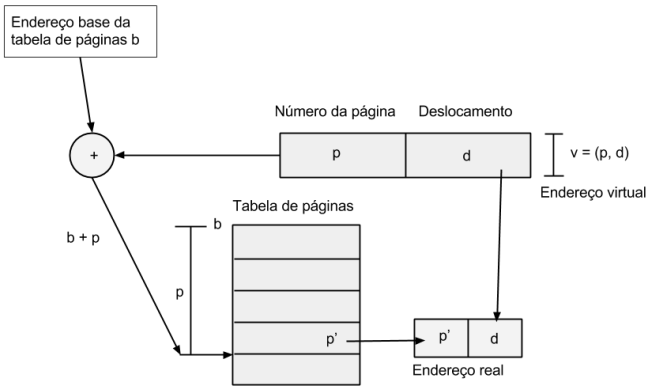

Figura 1: Tradução por mapeamento direto
[2]

O processo de tradução é denominado mapeamento, e para que isso ocorra sem prejudicar o desempenho do sistema, é necessário suporte em hardware por parte da unidade de gerenciamento de memória(*Memory Management Unit* - MMU).

Para que seja possível a tradução dinâmica dos endereços, é necessário manter mapas de tradução, informando se o conteúdo buscado no endereço virtual encontra-se na memória principal e qual é a sua real posição.

Todavia a técnica de mapeamento seria inviável caso o mapeamento fosse feito endereço por endereço, porque a quantidade de informações de mapeamento seria maior que a quantidade de memória. Por este motivo, a maioria das implementações atuais mapeiam blocos de endereços.

A decisão sobre o tamanho dos blocos não é simples, e requer a análise de alguns fatores. Um deles é a quantidade de memória gasta com o mapeamento. Caso o tamanho dos blocos seja pequeno será utilizado muita memória para o mapeamento, por outro lado, blocos grandes tendem a ter parte da sua capacidade não preenchida, gerando fragmentação interna e também aumentando o tempo de transferência do armazenamento secundário para memória principal [2].

Outro fator a ser levado em consideração é a utilização de blocos de tamanho fixo ou variado. Nas implementações com blocos de tamanho fixo, o bloco é chamado de página e a organização da memória de paginação, quando os blocos variam no tamanho são denominados segmentos e a organização de memória virtual de segmentação [2].

2.2.2 *Memória Virtual por Paginação*

A paginação é a técnica de gerência de memória na qual o espaço de endereçamento virtual e real são divididos em páginas. Habitualmente, as páginas do espaço virtual são chamadas de páginas virtuais e as paginas no espaço real de *frames* ou molduras.

Cada página virtual possui uma entrada na tabela de páginas (ETP) e cada processo uma tabela de páginas própria, assim é possível traduzir um endereço virtual em real(Figura 2) [2, 6].

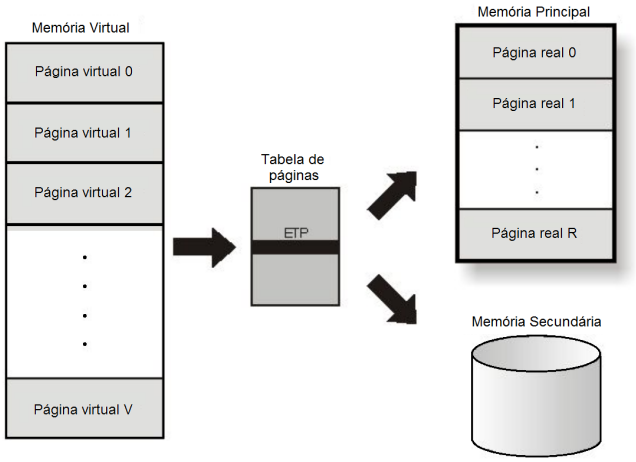

Figura 2: Tabela de páginas
[6]

Um endereço virtual é formado pelo número da página virtual (NPV) e pelo deslocamento, o NPV funciona como um índice para a tabela de página, e o deslocamento serve para indicar a posição do endereço em relação ao início da página. Combinando-se o endereço da página e o deslocamento consegue-se chegar ao endereço real.

Entretanto, como a memória virtual pode ser maior que a real, em alguns casos a página referenciada pode não estar na memória real. Nestes casos será gerada uma exceção para o sistema operacional(*page fault*), indicando para o sistema que uma página referenciada não se encontra na memória principal.

Toda vez que ocorrer um *page fault*, o sistema operacional será responsável por executar uma operação de E/S chamada de *Page in*, a qual vai carregar a página da memória secundária para a memória real [2].

A quantidade de *page faults* gerados por um processo em um determinado espaço de tempo, é chamada de taxa de paginação do processo, esta taxa depende da política de gerência de memória implantada. Quando ocorre um *page fault* o processo em execução é interrompido e passa para o estado de espera, enquanto aguarda o termino da operação de E/S.

Taxas de paginação muito elevadas acabam comprometendo o desempenho do sistema [2].

Políticas de Busca de Páginas. As políticas de busca de página são responsáveis por determinar quando uma página deve ser carregada na memória principal, sendo fundamentais para manter baixas as taxas de paginação dos processos. Quanto às estratégias de busca, a paginação pode ser de dois tipos: Por demanda ou Antecipada [6].

Na paginação por demanda, quando o processo executa pela primeira vez, apenas a primeira página do processo é carregada, e a partir desse momento as páginas só são carregadas quando referenciadas, o uso desta política evita a carga de páginas que nunca serão utilizadas, permitindo potencialmente a carga de mais processos ao mesmo tempo na memória virtual.

Embora tenha suas vantagens, a paginação por demanda tem seus problemas, nesta abordagem sempre que um processo referencia uma página nunca antes utilizada, obrigatoriamente terá que ser feita uma operação de E/S para adicionar a página na memória. Para os casos em que o processo já tem muitas paginas residentes na memória, esta operação de E/S faz com que uma quantidade significativa da memória fique inutilizável, aguardando a conclusão do processo de carga da nova página.

No intuito de tentar diminuir o tempo de espera, os projetistas de sistemas operacionais implementam mecanismos de paginação antecipada. Nestes o sistema operacional tenta prever quais serão as próximas páginas a serem referenciadas pelo processo, caso a decisão seja acertada o tempo total de execução dos processos tendem a reduzir [2].

No entanto, a paginação antecipada deve ser implementada com cautela, uma vez que a carga de muitas páginas indesejadas ou estratégias que consumam muito processamento podem piorar o tempo de execução dos processos, causando um efeito contrário ao desejado.

Políticas de alocação de páginas. As politicas de alocação de páginas determinam quantas páginas os processos podem manter na memória principal. Basicamente podem ser por alocação fixa ou variável. Na primeira, o número máximo de molduras utilizadas por um processo é determinado no início da execução, normalmente com base no tipo de aplicação que será executada.

Já na alocação variável, o número máximo de molduras utilizadas pelo processo varia de acordo com a taxa de paginação e ocupação da memória. Apesar da alocação variável parecer uma solução mais interessante, a necessidade de monitoramento constante dos processos gera um *overhead*, o qual deve ser avaliado antes de ser empregada a estratégia.[6]

Políticas de substituição de páginas. Quando há a necessidade de carregar uma nova página e toda memória principal já está ocupada, é necessário decidir qual moldura de página será desocupada para dar lugar à nova página, para a realização desta tarefa são necessárias as politicas de alocação de páginas [2, 6].

Ao escolher uma página para ser substituída, as politicas de substituição devem optar por páginas que não serão referenciadas num futuro mais próximo, entretanto como não é possível prever o fluxo dos processos, não há como ter exatidão.

Deste modo, as politicas de substituição trabalham com algumas heurísticas na tentativa de aumentar a acertibilidade e diminuir o número de *page faults* gerados por consequência da substituição equivocada de uma página [2, 9].

Ao escolher uma página para ser substituída, o gerenciador de memória deve observar se aquela página foi ou não modificada pelo processo durante a execução, caso sim, a página deve ser copiada para uma área especifica, de outro modo, não é necessário, pois o conteúdo da página está exatamente igual no armazenamento secundário[2, 6, 12].

Algoritmos de substituição de páginas. Abaixo serão enumerados alguns dos principais algoritmos de substituição de páginas:

1. **Algoritmo de Substituição de Página Ótimo -** Neste algoritmo admite-se que cada página contenha um rótulo com a quantidade de instruções que serão executadas antes daquela página ser referenciada, assim o gerenciador poderia escolher a página com maior rótulo. Deste modo, o algoritmo adiaria ao máximo a próxima ocorrência de falta de página. Contudo, como já vimos na seção anterior, não é possível prever o fluxo dos processos, sendo portanto impossível a implementação deste algoritmo.[12]

2. **Algoritmo de Substituição de NRU -** O algoritmo de substituição de paginas não usadas recentemente(NRU) baseia-se na ideia de que uma página que não foi referenciada recentemente, não será referenciada no futuro próximo.

   Para a implementação deste algoritmo, a cada entrada na tabela de páginas, necessariamente deve haver dois bits, um deles para informar se página foi referenciada, e o outro para informar se foi modificada. O algoritmo NRU baseia-se nestes dois bits para decidir qual página substituir.

   Primeiro, ele divide as opções em quatro classes:

   - Classe 0 - não referenciada e não modificada;
   - Classe 1 - não referenciada e modificada;
   - Classe 2 - referenciada e não modificada;
   - Classe 3 - referenciada e modificada.

   A página a ser substituída é escolhida aleatoriamente dentro da classe de menor ordem.

3. **Algoritmo de Substituição de FIFO -** No algoritmo de substituição de página primeira a entrar, primeira a sair(first-in, first-out – FIFO), o sistema operacional mantém uma lista encadeada de páginas, a

medida que uma nova página chega ela é adicionada no final da lista, quando ocorrer a necessidade de substituição é retirada a primeira página lista. Ou seja, a mais antiga na memória [12]. A Figura 3 ilustra o funcionamento deste processo.

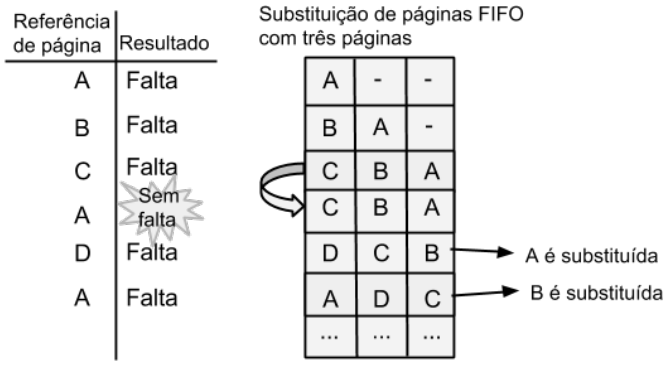

Figura 3: Substituição de página FIFO [2]

Este é um algoritmo de fácil implementação, pouco oneroso, mas raramente é utilizado, pois o algoritmo não evita a retirada de páginas antigas que são frequentemente utilizadas [2, 6, 9, 12].

4. **Algoritmo de Substituição de MRU -** O algoritmo de substituição de paginas menos recentemente utilizada – MRU seleciona a moldura menos recentemente utilizada para ser substituída (Figura 4). Apesar de possível e de apresentar resultados próximos do desempenho teórico do algoritmo ótimo, este algoritmo é pouco utilizado devido ao alto custo de implementação[12].

5. **Algoritmo de Substituição de NFU -** No algoritmo de substituição de paginas não frequentemente utilizada(NFU) a decisão de retirar uma página para ser substituída é tomada com base na quantidade de vezes que uma página foi referenciada. Para isso cada página mantém um contador que deve ser atualizado cada vez que a página é referenciada, quando necessário, a página menos referenciada é a escolhida para ser substituída [2, 6]. Embora pareça uma boa estratégia, o algoritmo acaba retirando muitas vezes as páginas que acabaram de chegar na memória em detrimento de outras que possuem muitas referencias, mas que não serão utilizadas no futuro. Por isso, este algoritmo é raramente utilizado [2, 6].

2.3 Gerência de Processos

Os computadores atuais são capazes de executar varias tarefas ao mesmo tempo, no entanto, com exceção dos computadores com mais de uma CPU, só é possível executar um programa por vez. Assim, o sistema operacional alterna múltiplas vezes em um curto espaço de tempo entre os programas, causando uma ilusão de paralelismo.

Para isso, os programas, incluindo o próprio sistema operacional, são organizados em processos sequenciais. Portanto,

Figura 4: Substituição de página MRU [2]

no estudo de sistemas operacionais, processo é programa em execução acompanhado dos valores atuais do contador de programa, registradores e variáveis[12].

Cada processo tem a ilusão de ter uma CPU própria, mas na realidade a CPU troca constantemente de um processo para outro(multiprogramação). Quando cada processo vai executar, o contador de processo dele é carregado no contador real da CPU, já o contador de programa do processo que está deixando a CPU é salvo na memória até que chegue novamente a sua vez de utilizar a CPU.

2.3.1 Estados dos Processos

Ao longo do processamento o processo pode passar por três diferentes estados:

Execução(running) - O processo está em *Execução* quando está sendo processado pela CPU, em sistemas com apenas uma CPU somente um processo estará em execução por vez.

Pronto(ready) - São considerados no estado *Pronto* todos os processos fora da CPU mas aguardando apenas a sua vez para executar.

Espera ou Bloqueado (wait) - Os processos em *Espera* são aqueles que estão esperando por uma entrada ainda não disponível, como por exemplo o término de uma operação de entrada/saída.

Os processos podem mudar de estado em função de eventos voluntários ou involuntários. Podemos observar na Figura 5 que existem quatro possíveis mudanças de estado:

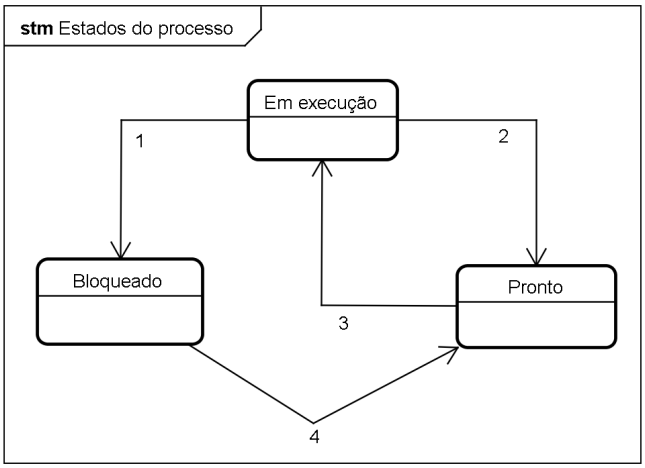

Figura 5: Estados dos processos

1. Em execução -> Bloqueado: transição causada por um evento voluntário gerado pelo próprio processo quando descobre que não pode prosseguir.

2. Execução -> Pronto: transição causada por um evento disparado pelo escalonador de processos(involuntário). Ocorre normalmente quando termina a fatia de tempo disponibilizada pelo sistema operacional para o processo, sendo portanto a vez de um outro processo executar.

3. Pronto -> Execução: ocorre quando um processo da fila de pronto é escolhido para executar. É uma transição causada por um evento involuntário gerado pelo escalonador de processos. A politica de escalonamento é um dos principais tópicos do estudo de sistemas operacionais e será detalhada na seção 2.3.2 deste trabalho.

4. Bloqueado -> Pronto: a transição ocorre quando a operação esperada é concluída, ou o recurso aguardado é liberado. Ou seja, quando o evento externo esperado pelo processo é concluído.

2.3.2 *Escalonamento de Processos*

Nas situações em que existem vários processos disponíveis para serem executados (Pronto) é necessário estabelecer critérios para a escolha de qual deles irá assumir o uso da CPU. Estes critérios compõem a política de escalonamento, tratada por alguns autores como gerência de processador[6]. Alguns critérios de escalonamento de processos são de extrema relevância para a implementação de um sistema operacional, são eles [6, 12]:

- Utilização do processador
- *Throughput*
- Tempo de processador
- Tempo de espera
- Tempo de turnaround
- Tempo de resposta

De modo geral, a política de escalonamento deve otimizar o *throughput* e a utilização do processador e diminuir o tempo de resposta, turnaround e espera. No entanto, cada sistema operacional tem um destes critérios mais importante que outros[12].

As politicas de escalonamento ainda levam em consideração a possibilidade do sistema operacional interromper um processo em execução e substituí-lo por outro. Para os sistemas nos quais isto pode acontecer, o escalonamento é chamado de *preemptivos*. Quando não há interrupções por parte do sistema operacional diz-se que o escalonamento é *não-preemptivos*[6].

Abaixo serão enumerados alguns dos principais algoritmos de escalonamento de processo:

1. **First-in-first-out(FIFO) -** neste algoritmo o primeiro processo a chegar ao estado de pronto é selecionado para a execução. Como é um algoritmo não preemptivo, o processo só deixa o processador quando entra em espera, ou finaliza sua execução. Caso um processo seja bloqueado, o primeiro da fila assume o processador. Todos os processos quando saem do estado de espera voltam para o fim da fila de pronto.

2. **Menor Job Primeiro -** algoritmo com escalonamento do tipo não preemptivo, nele supõe-se que o tempo de execução de todos os processos é conhecido, com isso o algoritmo de escalonamento escolhe o processo menor JOB para ser executado.

3. **Próximo de menor tempo restante -** este algoritmo é a variação preemptiva do Menor Job Primeiro. Neste algoritmo é avaliado o tempo de execução restante, assim, caso um novo processo chegue na fila de pronto e tenha um custo de execução restante menor, o processo atual sofrerá preempção e o novo processo irá assumir a CPU.

   com escalonamento do tipo não preemptivo, nele supõe-se que o tempo de execução de todos os processos é conhecido, com isso o algoritmo de escalonamento escolhe o processo menor JOB para ser executado.

4. **Circular -** algoritmo com escalonamento do tipo preemptivo, nele cada processo tem um tempo limite para uso continuo do processador, chamado de *timeslice*. Quando este tempo se esgota, o processo volta para o fim da fila e dá a vez para outro processo.

5. **Prioridade -** algoritmo de escalonamento preemptivo, neste algoritmo o escalonamento é realizado com base em um valor associado aos processos denominado prioridade. Processos com a mesma prioridade são escalonados utilizando o algoritmo FIFO. Caso um processo de maior prioridade chegue na fila de pronto, o algoritmo atual sofre preempção para dar lugar ao recém chegado. No algoritmo por prioridade não existe preempção por tempo.

3. TRABALHOS CORRELATOS

Esta seção apresenta alguns Simuladores de Sistemas Operacionais, descrevendo de forma sucinta o seu funcionamento, bem como, fazendo as devidas comparações com o SWSO.

3.1 Sosim

O Sosim é um Simulador para o ensino de Sistemas Operacionais, foi desenvolvido durante uma tese de mestrado do professor Luis Paulo Maia [5].Ele permite simular os conceitos implementados em sistemas operacionais multiprogramados, basicamente é feita a simulação da gerência de processo e gerência de memória(com memória virtual). Com este simulador é possível visualizar a percentagem de memória física livre, as trocas de páginas entre a memória principal e secundária, além das informações sobre os processos.

O software foi desenvolvido para o uso em computadores de mesa, o programa é um executável que roda no sistema operacional *Windows*. O sistema é baseado em janelas, tendo uma janela principal e varias outras que podem estar ou não visível (a configuração da exibição é feita na janela principal).

Durante a simulação é permitida a configuração de alguns parâmetros dos gerenciadores. Por exemplo, no gerenciador de processos é permitido escolher a fatia de tempo, a frequência de *clock*, o tempo de espera de IO e alterar a prioridade dos processos. A visão das Janelas relacionadas à gerência de processos no Sosim é exibida na Figura 6.

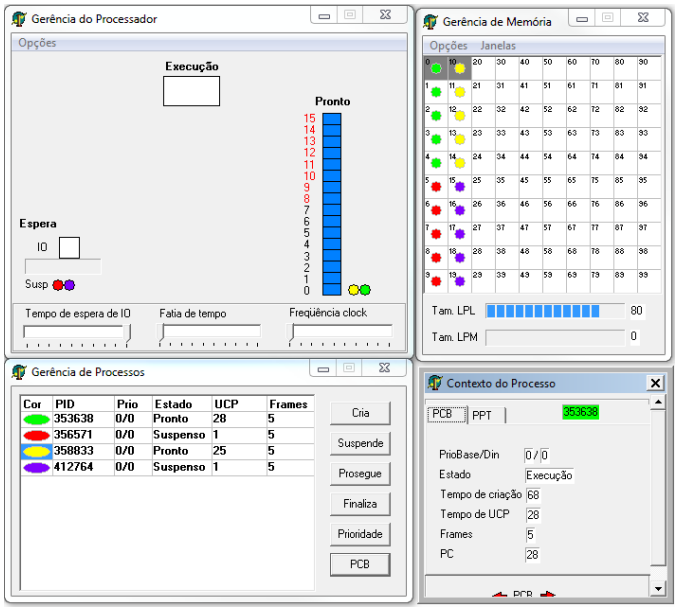

Figura 6: Sosim

Embora permita a configuração dos parâmetros supracitados durante a simulação, o Sosim possui uma interface um pouco confusa, em alguns momentos é necessário ter diversas janelas abertas. Além disso, a simulação não dá a oportunidade da troca dos algoritmos que estão sendo utilizados. Por exemplo, a interface consegue apresentar bem o processo de substituição de páginas, entretanto só há a possibilidade de simular substituição com o algoritmo FIFO.

3.2 SimulaRSO

O SimulaRSO é um simulador web desenvolvido por estudantes de graduação da Universidade Católica de Santos, ele faz a simulação de escalonamento de processos, de disco e de paginação de memória. Além disso com ele é possível alterar os algoritmos ou fazer comparações entre eles. A interface também é uma característica a ser destacada nesse software, é de utilização simples, existindo a possibilidade de apresentação gráfica da simulação realizada.

No módulo de escalonamento de processo se consegue escolher de 1 à 20 processos e para cada um deles determinar a chegada, a prioridade e custo. Também, para alguns algoritmos é possível determinar o tempo de corte e de troca de contexto, como podemos observar na Figura 7.

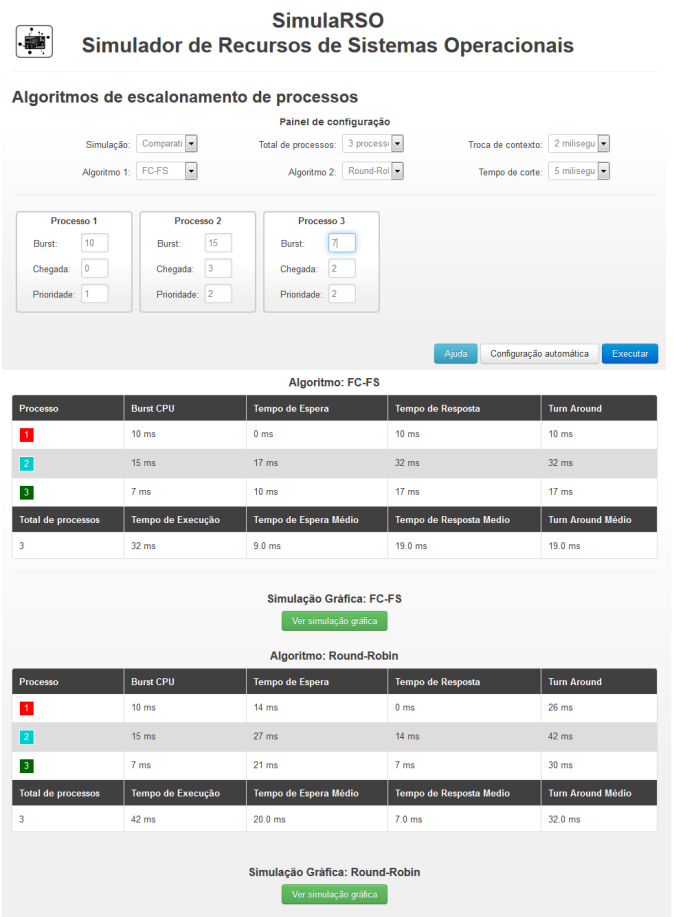

Figura 7: SimulaRSO

Já na simulação de paginação de memória o sistema permite escolher as estratégias de substituição, o número máximo de molduras de páginas na memória, quantas páginas serão referenciadas e a sequência em que elas serão referenciadas;

O módulo de escalonamento de disco, assim como os outros citados, também permite a comparação entre dois algoritmos diferentes, tendo como entrada para a simulação o setor inicial, a quantidade de requisições e o setor para cada requisição; e apresenta como saída a movimentação total do cabeçote de leitura, bem como o gráfico desta movimenta-

ção.

Apesar das funcionalidade informadas, o SimulaRSO acaba não fazendo simulações de um sistema operacional como um todo, nem simula o funcionamento do sistema operacional em execução. O simulador apenas exibe/compara diferentes algoritmos baseado em alguns parâmetros passados, não havendo integração entre os diferentes módulos.

3.3 SSOG

Simulador de sistemas operacionais com suporte para a gerência de processos e memória, possuí interface desktop e foi desenvolvido usando a linguagem de programação Java SE. O SSOG não utiliza o conceito de memória virtual e também não faz simulação de disco. Entretanto com ele é possível configurar a prioridade dos processos no momento da criação, além disso é possível visualizar a percentagem da memória que está sendo utilizada e a quantidade de processo que estão carregados[3]. A Figura 8 mostra a tela de criação de processos e a Figura 9 a tabela de processos do SSOG.

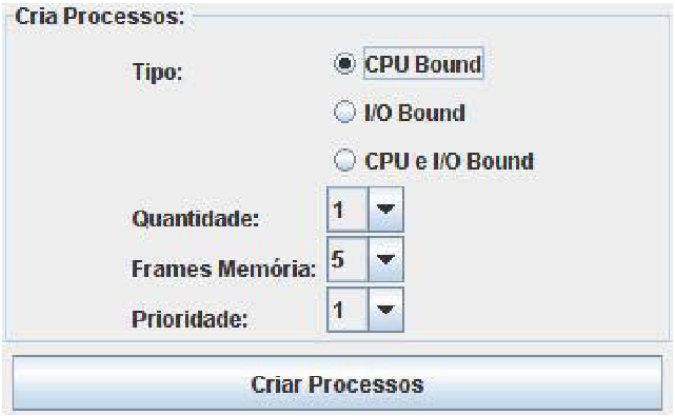

Figura 8: SSOG Criação de Processos

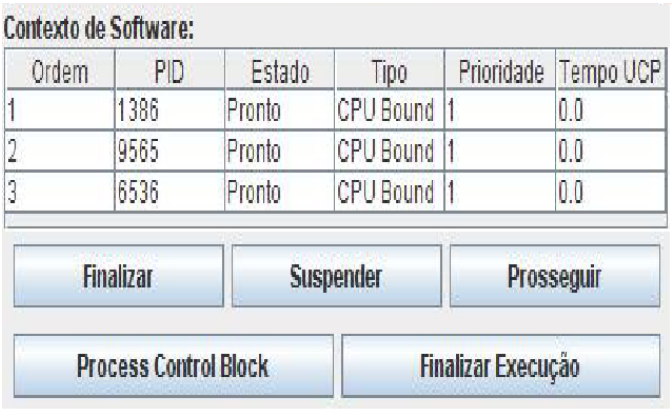

Figura 9: SSOG Tabela de Processos

Assim como os correlacionados anteriores o SSOG não faz simulação de escalonamento de disco, nem dos principais conceitos da gerência de memória. O controle do tempo também não foi implementado neste algoritmo.

4. MODELO PROPOSTO

O SWSO procura unir os aspectos mais relevantes dos trabalhos correlacionados citados na seção 3. Ele tem a capacidade de alterar o *timeslice* e prioridade vista no Sosim, também é possível configurar os algoritmos assim como o SimulaRSO. Outro aspecto importante é a praticidade de funcionar na web, também característica positiva do SimulaRSO.

Entretanto, além de absorver os traços positivos dos trabalhos correlacionados, este trabalho possuí características não vistas em nenhum dos trabalhos estudados. Ele é o único que faz de maneira integrada a gerência de disco, processo e memória. Também é o único no qual é possível controlar a passagem do tempo.

Nesta seção será apresentada a documentação e os pontos de mais relevância para o desenvolvimento do software com seus devidos diagramas de classes e sequências. Também serão listadas as tecnologias utilizadas para a construção da aplicação, por último as principais funcionalidades serão detalhadas.

O desenvolvimento da aplicação foi realizado em 3 etapas, primeiramente foi desenvolvido o protótipo da aplicação, para facilitar o reconhecimento e compreensão dos requisitos.

Após a prototipagem, os requisitos funcionais e não-funcionais identificados foram documentados utilizando documentos de texto e diagramas de casos de uso.

Uma vez definidos os requisitos, foi iniciada a etapa de implementação. Nesta etapa, foi criada a camada de abstração da parte física do computador, o núcleo do sistema operacional, e as telas para as entradas do usuário e apresentação dos resultados da simulação.

4.1 Requisitos do software

No processo de desenvolvimento de um *software*, a compreensão completa dos requisitos é fator fundamental para alcançar uma alta qualidade. Mesmo que o software tenha sido bem codificado, caso ele tenha sido mal especificado, terá grande chance de ser mal sucedido e desapontar o seu usuário final[1].

Entende-se por requisitos:

> "Uma especificação de uma característica ou propriedade que um sistema deve possuir ou fazer, assim como sua restrição de operação".

[10]

Os requisitos podem ter diversas classificações: requisitos funcionais, não funcionais, de negócio e etc. Para a elaboração deste trabalho foram escritos requisitos funcionais e não funcionais. Abaixo a definição de cada um deles:

1. Requisitos Funcionais - Definem as funções de um software ou parte dele. São o conjunto de entradas, o comportamento e sua saída. Os Requisitos funcionais

envolvem lógicas de trabalho, manipulação, cálculos e processamento de dados, entre outros. [8, 11]

2. Requisitos Não-Funcionais - são requisitos de desempenho, usabilidade, confiabilidade, segurança, disponibilidade, manutenção e tecnologias envolvidas, todos eles relacionados ao uso da aplicação. São características mínimas de um software de qualidade, e na maioria das vezes fica a cargo do desenvolvedor optar por atender esses requisitos ou não. [8, 11] No caso do SWSO, a maioria deles foram identificados ainda na fase de elicitação de requisitos.

4.1.1 Requisitos funcionais

Os principais requisitos funcionais do SWSO podem ser vistos na Figura 10. Logo em seguida, será feito um breve resumo de cada um deles.

Identificador	Requisito Funcional	Categoria
RF01	Criar Máquina Virtual	Usuário
RF02	Configurar algoritmos	Usuário
RF03	Manter Arquivos	Usuário
RF04	Simular busca e escalonamento de Disco	Usuário
RF05	Manter Processo	Usuário
RF06	Escalonar Processo	Usuário
RF07	Alocar Memória	Usuário
RF08	Simular Substituição de Páginas da Memória	Usuário

Figura 10: Requisitos Funcionais

RF01. Criar Máquina Virtual - Para dar início à simulação o usuário deverá criar uma máquina virtual. Na criação da máquina virtual deverão ser informados alguns parâmetros, como por exemplo: tamanho da memória, quantidade de bytes por página e quantidade de bytes por setor. A máquina deverá ter um nome único na aplicação, que servirá para que outros clientes possam acessá-la, em modo de visualização.

RF02. Configurar algoritmos - O sistema deve permitir configurar os algoritmos da simulação. Na criação da máquina virtual devem ser informados os algoritmos de escalonamento de disco, escalonamento de processo e substituição de páginas. E durante a simulação deverá ser possível alterá-los.

RF03. Manter Arquivos - Deverá ser possível fazer *upload* de arquivos de texto. No momento da criação do arquivo deverá ser informada uma cor(esta cor servirá para facilitar a distinção do arquivo na simulação). Os arquivos enviados para o servidor serão tratados como arquivos executáveis do sistema operacional, e ocuparão espaço no disco virtual da simulação. O sistema deve permitir a exclusão dos arquivos, na exclusão o espaço em disco deverá ser liberado.

Para ser válido o arquivo deve conter 4 algarismos Hexadecimais por linha, que é equivalente à 16 bits, que foi o tamanho definido para a palavra do nosso sistema computacional.

RF04. Simular busca e escalonamento de Disco - O sistema deve permitir que o usuário simule o processo de busca no disco, mostrando a movimentação do cabeçote de leitura na forma de gráfico para os diferentes algoritmos. Toda vez que um arquivo for inserido ou referenciado pelo sistema, seja no carregamento inicial, ou em qualquer etapa da simulação o gráfico de movimentação deverá ser atualizado.

Também deve ser possível simular a solicitação de requisições a diversos cilindros, para isso o usuário deverá enviar a lista de cilindros que deseja consultar e o simulador fará o gráfico com o comportamento do disco para atender essas requisições.

RF05. Manter Processo - Será possibilitado pelo sistema a criação, remoção e mudança de estado de um processo. Os processos serão carregados por meio de uma interface na qual o usuário informará a partir de qual arquivo ele deseja criar o processo. Os processos poderão ser finalizados e suspensos pelo usuário. No momento da criação de um processo a prioridade dele deverá ser informada pelo usuário.

RF06. Escalonar Processo - Quando a fatia de tempo do processo atual estiver terminado, o sistema será responsável por escolher qual o próximo processo sairá da fila para ser executado. Para isso deverá ordenar a fila de pronto de acordo com um algoritmo de escalonamento selecionado pelo usuário.

RF07. Alocar Memória - O sistema deverá alocar um espaço na memória da máquina virtual sempre que um processo for criado. Além disso, quando necessário carregar uma ou mais páginas que ainda não estejam na memória deve ser possível encontrar um espaço vazio para alocar a(s) página(s). A alocação de memória deve ser fixa, e o número máximo de molduras ocupadas pelos processos deve ser definido no início da execução de cada processo.(Dúvida: Por processo ou da simulação?)

RF08. Simular Substituição de Páginas da Memória - O sistema deve permitir que o usuário simule o processo de substituição de páginas da memória. O usuário deve ter a possibilidade de escolher o algoritmo de substituição de páginas.

4.1.2 Requisitos não-funcionais

Ainda na definição dos requisitos, foram identificados os requisitos não funcionais da aplicação,os principais deles estão listados na Figura 11.

Identificador	Requisito Não-Funcional	Categoria
RNF01	A aplicação web deve funcionar em um browser atual.	Usabilidade
RNF02	Deve ser desenvolvido usando a plataforma Java EE.	Software
RNF03	As dependências do projeto e o *build* da aplicação devem ser gerenciados pelo Apache Maven.	Software
RNF04	A aplicação deve rodar no servidor de aplicação Jboss 7.1.	Ambiente

Figura 11: Requisitos Não-Funcionais

O SWSO foi desenvolvido utilizando a linguagem de programação Java, com o suporte da plataforma Java EE (Enterprise Edition) possibilitando a integração com o *framework*

JSF(Java Server Faces). Além do JSF foi utilizado o Primefaces, html5 e javascript.

Uma vez definida a utilização da plataforma Java EE foi necessário escolher e configurar um servidor de aplicação para o desenvolvimento e teste do software, a opção foi o *JBoss Application Server*, por ser gratuito, open source e ser atualmente um dos mais utilizados no mundo.

4.2 Implementação

A implementação do SWSO foi realizada em 4 etapas. As etapas de desenvolvimento em ordem cronológica foram: criação da máquina virtual, construção do módulo de gerência de disco, construção do módulo de gerência de processos e construção do módulo de gerência de memória.

4.2.1 Máquina Virtual

A primeira etapa da construção foi a implementação da máquina virtual, ela foi implementada de forma isolada, para garantir o seu funcionamento independente das classes que simulam o sistema operacional, e também de toda a camada de apresentação.

Para isso, foi necessário, com o uso da orientação a objetos, criar toda a arquitetura de um computador, abstraindo cada um dos mais importantes componentes de um sistema computacional em classes Java.

As classes e a organização desta implementação podem ser vistas no diagrama de classes da Figura 12. As principais classes estão listadas abaixo, seguidas de uma breve explicação:

CentralProcessingUnit. - Esta classe abstrai a unidade central de processamento(CPU) de um sistema computacional. Ela tem na sua composição importantes outras classes, como por exemplo: *ArithmeticLogicUnit, ControlUnit, InstructionDecoder, Registers*; Cada uma delas implementando um papel semelhantes aos componentes de hardware homônimos.

O método mais importante da *CentralProcessingUnit* é o *execute*. O algoritmo simulado pelo método execute é uma abstração da tarefa realizada por um processador no momento da execução. A implementação do *execute* pode ser vista no **Algoritmo 01**.

Algoritmo 01 - Método que simula a execução do processador

```
public void execute() {
   if(registers.getProgramCounter().realValue() != -1) {
      Word instruction =
           controlUnit.seekInstruction(registers,
           memoryManagementUnit);
      int tipoInstrucao =
           controlUnit.decode(instruction,
           instructionDecoder);
       switch (tipoInstrucao) {
         case 1:
            controlUnit.execute(arithmeticLogicUnit);
            break;
         case 2:
            controlUnit.seekOperators();
            break;
         case 3:
            controlUnit.storeResults();
            break;
      }
   }
      cpuTime++;
}
```

A demonstração do **Algorimo 1** tem também o objetivo de ressaltar que, apesar de não estar implementada, a decodificação da instrução é possível de ser feita sem afetar outros pontos do sistema, as tarefas necessárias para concretizar mais esta funcionalidade envolve a alteração em apenas classes de arquitetura.

RandomAccessMemory. - Esta classe abstrai a memória principal da nossa máquina virtual, menos complexa que a CPU, ela basicamente serve para guardar um *array* de ByteSWSO (objeto usado como abstração de um Byte). Os principais métodos dela são relacionados à leitura e escrita neste *array*, são eles: *alloc* e *getWord*. O primeiro recebe um conjunto de dados e a posição real inicial para começar a escrita e o segundo atua de forma contrária, recebe um endereço e retorna o valor guardado neste endereço.

MemoryManagementUnit. - Classe responsável pela tradução de endereços lógicos para endereços físicos. Como podemos ver no **Algoritmo 1**, no primeiro passo da execução a unidade de controle realiza a busca da instrução, nesta busca três passos são realizados, sendo que o primeiro deles é acionar a unidade de gerenciamento de memória (MMU) para obter a instrução na posição indicada pelo contador de programa. A MMU recebe uma posição virtual e a partir desta descobre em qual página se encontra esta instrução, logo após consulta a tabela de página do processo, que retorna a ETP (Entrada na Tabela de Páginas) relacionada à página.

Neste ponto é verificado o bit de validade na ETP, caso a página esteja na memória principal encontra-se o deslocamento e é retornada a instrução contida na posição da memória real encontrada; Caso o bit de validade seja falso, indica que a página não se encontra na memória física e será lançada a exceção *PageFault*, que indica uma falta de página.

HardDisk. - Esta classe abstrai a memória secundária, assim como a memória principal sua função é armazenar informações, entretanto, a estrutura para o armazenamento é um pouco diferente. O HardDisk, guarda um *array* de pratos, que guarda um *array* de trilhas, e em cada trilha há também um *array* de setores; os objetos que representam esta estrutura respectivamente são *Plate, Track e Sector*, como pode ser visto na Figura 12.

Além das abstrações dos componentes de hardware supracitados, foi criada uma classe chamada *CoreVirtualMachine*, esta classe tem o papel de agregar as principais classes da

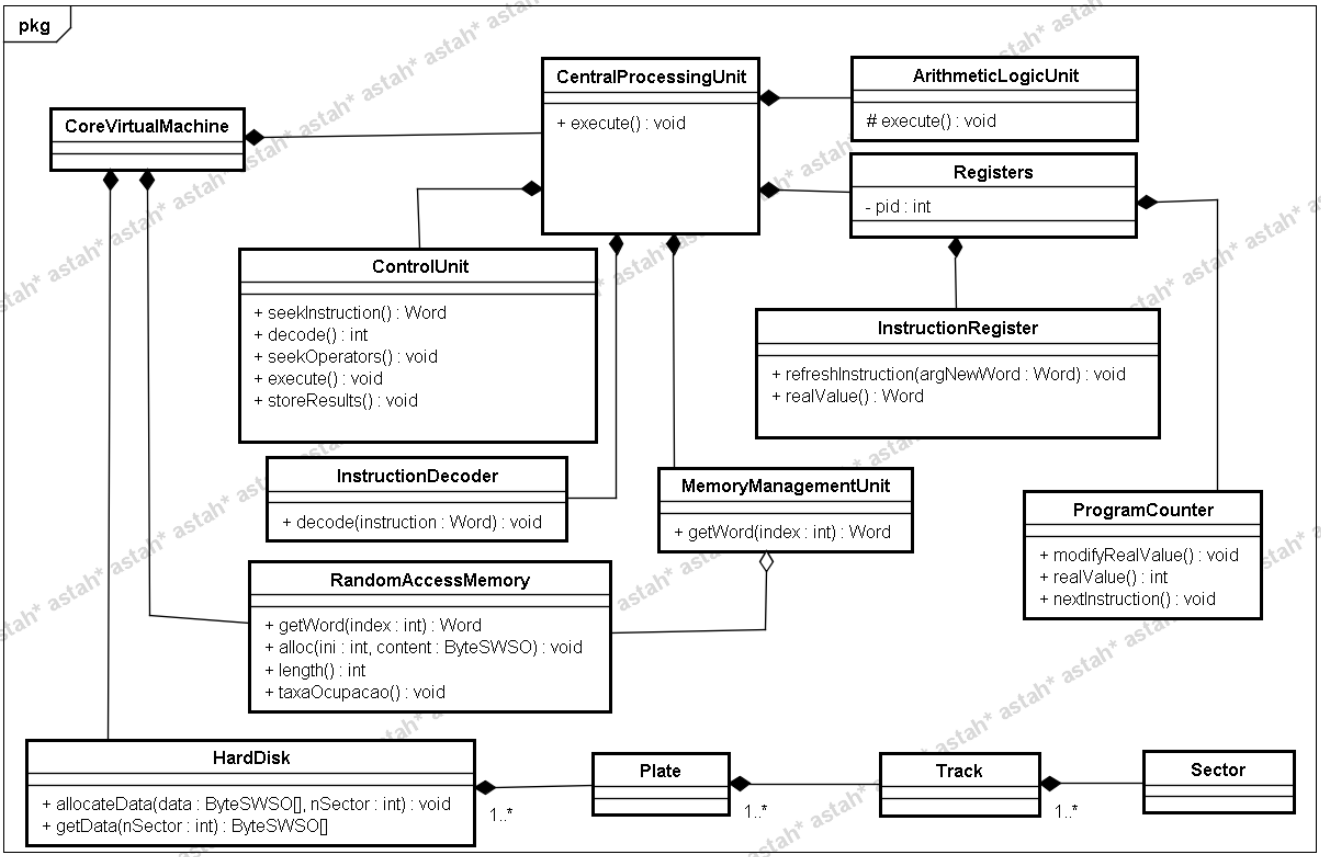

Figura 12: Diagrama de Classe da Máquina Virtual

máquina virtual. Veremos que é por meio desta classe que o *Kernel* do Sistema Operacional consegue acessar e realizar as ações necessárias nas abstrações dos componentes de hardware.

Também foram criadas as classes *ByteSWSO, Page, File e Word.* Elas respectivamente abstraem bytes, paginas, arquivos e palavras. Sendo muito importante para a trafegar dados dentro da aplicação.

4.2.2 Sistema de arquivos e gerência de disco

Para fazer a simulação da gerência de disco, foi necessário implementar um módulo de sistemas de arquivos. Entretanto, como não fazia parte dos requisitos iniciais, e nem é foco deste trabalho, os únicos aspectos explorados foram a alocação de arquivos e a gerência de espaços vazios.

Por isso a implementação do sistema de arquivos ficou bastante simples. Para a alocação de arquivos foi feita a escolha pelo método de alocação indexada e o mapa de bits foi usado para gerenciar os espaços vazios.

A implementação do sistema de arquivos é feita pela classe FileSystem, o papel desta classe é servir de intermediária para acesso ao disco(HardDisk). Ao ser recebido pelo servidor o arquivo é transformado em um array de BytesSWSO, este array juntamente com o nome do arquivo passa a compor outra abstração chamada de FileInput. O objeto FileInput gerado é enviado para o sistema operacional.

Além do *FileInput* o sistema operacional recebe um algoritmo de escalonamento de disco, que obrigatoriamente deve implementar a interface IDiskScheduler. E em posse desses dois parâmetros, o executa os seguintes passos:

1. Descobre quantos setores serão necessários para armazenar o arquivo;
2. Busca no disco uma lista de setores vazios suficientes para armazenar o arquivo;
3. Transforma esta lista de setores em uma lista de requisições de gravação;
4. Ordena a lista de requisições utilizando o objeto DiskScheduler que foi recebido por parâmetro;
5. Adiciona o arquivo no disco rígido;
6. Atualiza o mapa de bits;
7. Adiciona o arquivo numa lista de arquivos;

No método de alocação, o escalonamento de disco é feito no momento que a lista de requisições é ordenada(Passo 4). O algoritmo de escalonamento é passado para o método de

alocação a cada requisição, uma vez que o sistema permite a troca do algoritmo mesmo depois do início da execução.

A implementação do algoritmo foi encapsulado em um objeto que assume o compromisso com a interface IDiskScheduler. Desta forma é possível definir novas implementações sem afetar a classe FileSystem, funcionando da melhor maneira possível, pois o algoritmo de escalonamento é independente do sistema de arquivo utilizado. Sendo portanto um exemplo da aplicação do padrão de projeto Strategy. O **Algoritmo 2** apresenta a interface IDiskScheduler;

Algoritmo 2 - Interface IDiskScheduler.

```
public interface IDiskScheduler {
   int[] escalonar(int[] queue, int initialCylinder);
}
```

4.2.3 Gerência de processos

Para a implementação da gerência de processo foi criado um modulo de gerência de processos, a implementação desse módulo é responsável pela criação dos processos, criação e escalonamento da fila de pronto e coordenar as trocas de contexto.

A criação de um processo ocorre quando o usuário do simulador seleciona um dos arquivos executáveis, que foram carregados na etapa de criação de arquivos. Neste momento, o sistema envia um objeto da classe "File"(Objeto criado para representar um arquivo) para o módulo de gerência de processo.

Em posse do arquivo, o gerenciador de processos cria um novo processo, com os seguintes atributos: tamanho(em bytes), quantidade de instruções, nome(utiliza o mesmo nome do arquivo), identificador do processo(pid) e tempo de início. Importante ressaltar que os processos não têm tamanho fixo, o tamanho do processo é determinado pelo tamanho do arquivo.

Após a criação do processo, este é adicionado na fila de pronto, que é representada por uma estrutura de dados do tipo "Lista encadeada". O escalonamento de processos será feito por meio de ordenamento desta lista, utilizando um algoritmo de escalonamento.

Habitualmente na implementação de um sistema operacional, apenas um algoritmo de escalonamento é utilizado. No entanto, por se tratar de um simulador, foi dada a possibilidade de alterar o algoritmo durante a utilização do simulador.

Para permitir a troca do algoritmo de escalonamento durante a simulação, bem como facilitar novas implementações de algoritmos de escalonamento, toda a lógica foi encapsulada em um objeto que assume compromisso com a interface IProcessesScheduler, que pode ser vista no **Algoritmo 3**.

Algoritmo 3 - Interface IProcessesScheduler.

```
public interface IProcessesScheduler {
   void escalonar(LinkedList<Process> listaPronto);
   boolean isPreemptivo();
   boolean isPrioridade();
}
```

A execução do escalonamento ocorre em diversas situações, que dependem do algoritmo de escalonamento que está sendo utilizado. Para os algoritmos não preemptivos, o escalonamento só ocorre quando o processo que está executando termina sua execução.

Já para os algoritmos preemptivos, o escalonamento ocorre quando é esgotada a fatia de tempo do processo que está em execução, quando ele é bloqueado ou no término de cada execução.

A implementação da troca de contexto neste simulador acontece depois da rotina de escalonar os processo, quando é escolhido um novo processo para assumir o processador. Nesta situação o gerenciador de processos guarda o valor contido no registrador de instrução e no contador de programa. Em seguida atualiza os mesmos registradores com o valor do processo que entrará em execução.

O tempo para a realização da troca de contexto não foi considerado, neste simulador consideramos que a troca ocorre instantaneamente, não representando fielmente a realidade. Entretanto, outras todas as outras funções do sistema operacional demandam algum tempo, mas por fins didáticos decidimos desconsiderar.

A classe responsável por implementar estas regras é a "ProcessManager", ela é a representação do gerenciador de processos. Esta classe compõe a classe "KernelOperatingSystem", que é a abstração do núcleo do sistema operacional. A classe "KernelOperatingSystem"é responsável por fazer toda a comunicação com as classes da máquina virtual.

4.2.4 Gerência de memória

A implementação da gerência de memória é feita a partir do módulo de gerência de memória, entretanto, para simular os principais aspectos de gerência de memória, outros módulos do sistema precisaram ser ajustados.

As alterações aconteceram na implementação da máquina virtual, na classe "MemoryManagementUnit" e no gerenciador de processo, pois no momento da criação do processo já é necessário acionar a funcionalidade de alocação de memória. As alterações serão melhor explicadas posteriormente, quando será detalhada a implementação das funcionalidades de gerência de memória do simulador.

A primeira das funções do gerenciador de memória a ser chamada pelo simulador é a função "allocateProcess", ela é responsável por dividir o processo em páginas, baseado na quantidade de instruções. É também dela a responsabilidade de determinar quantas páginas do processo serão carregadas inicialmente na memória.

No entanto, a escolha da quantidade de páginas depende da configuração feita pelo usuário. Caso o usuário tenha

optado por uma politica de busca por demanda, o simulador carregará apenas a primeira página do processo que está sendo criado. As outras páginas serão carregadas a medida que forem referenciadas.

Para os casos em que a política de busca antecipada foi selecionada, a função avalia a politica de alocação. Se a politica de alocação for fixa, a função checa a quantidade de páginas que um processo pode ter na memória e já carrega esta quantidade, neste caso, a quantidade é definida nas configurações de gerência de memória. Outra possibilidade é a política de alocação fixa por processo, para esta, a função carrega um número de páginas determinado na criação do processo.

A terceira opção é a politica de alocação variável, ela funciona exatamente igual a fixa por processo. Entretanto, ao longo da simulação existe a possibilidade de alterar a quantidade de páginas que um processo pode ter na memória. Na implementação de um sistema operacional isto é feito baseado em algum critério preestabelecido; no entanto, neste simulador a opção de alterar esta quantidade foi disponibilizada para o usuário.

Outra função do gerenciador de memória é "allocatePage", está função é chamada de dois pontos do simulador. O primeiro é de dentro da função "allocateProcess", quando é necessário carregar uma ou mais páginas do processo. O outro ponto, é quando ocorre uma falta de página.

A função "allocatePage" recebe dois parâmetros, o identificador do processo e o número da página que se deseja alocar na memória principal. Através do identificador ela busca a tabela de páginas do processo e encontra a ETP (entrada na tabela de página) referente a página solicitada.

Em seguida, encontra-se um frame disponível na memória principal e todo o conteúdo do disco é copiado para a memória principal. O código da função "allocatePage" pode ser visto no **O Algoritimo 4**.

Algoritmo 4 - Função "allocatePage".

```
public void allocatePage(int pid, int pageNumber) {
   PageTable pageTableProcess = pageList.get(pid);
   ETP etp =
        pageTableProcess.getListaEtp().get(pageNumber);
   if (etp.getBitV() == '0') {
      int realPosition;
      try {
         realPosition = realMemory.foundFreePosition();
      } catch (MemoryFullException e) {
         realPosition = searchPageToReplace();
      }
      realMemory.blockPosition(realPosition, pid);
      etp.setPpr(realPosition);
      copiarDoDiscoParaRAM(etp.getAllocatedSectors(),
           realPosition);
      etp.setBitV('1');
   }
}
```

Nesta mesma função, caso a memória esteja cheia ou tenha se esgotado a quantidade de páginas máxima que um processo pode ter na memória, é chamada a função "searchPageToReplace", responsável por encontrar uma página para ser substituída. A política de substituição é determinada pela implementação do algoritmo de substituição de páginas. Este, semelhante aos algoritmos de escalonamento de disco e processo, utilizou o padrão de projeto Strategy. A interface definida para implementar o algoritmo de substituição é a "IPageReplacementAlgorithm", que pode ser vista no **Algoritmo 5**.

Algoritmo 5 - Interface IProcessesScheduler.

```
public interface IPageReplacementAlgorithm {
   int findPageToReplace(RealMemory memory, Process
        process, PageTable pageTable);
}
```

Pode-se observar que o algoritmo de substituição recebe um objeto do tipo "RealMemory´´, que basicamente serve para encapsular um *array* de frames. Em posse dessa representação da memória, cada um deles utiliza uma política diferente para encontrar o frame que deve ser substituído.

No entanto, o algoritmo de substituição pode se comportar de duas maneiras diferentes. Caso o processo que está solicitando *Page In* já tenha alocado a quantidade máxima permitida de frames, o algoritmo de substituição vai escolher um dos frames do próprio processo para deixar a memória. Caso contrário, poderá escolher uma página de qualquer processo.

4.3 Simulador

Nesta seção serão apresentadas as principais funcionalidades do SWSO.

4.3.1 Configurar Parâmetros da máquina virtual

Para dar início a simulação, o usuário tem que criar uma máquina virtual. A tela de criação da máquina virtual pode ser vista na Figura 13. Através desta tela o usuário deve informar um nome para a máquina que está sendo criada, o tamanho da memória, o tamanho das páginas e o tamanho dos setores.

O nome da máquina virtual é muito importante para o restante da simulação, o sistema utiliza o nome como identificador único daquela simulação. Assim, não é possível que existam duas simulações com o mesmo nome sendo executadas ao mesmo tempo.

Como se trata de um projeto web, no qual o número de acessos pode ser muito grande, cada simulação é adicionada em uma estrutura de dados, permanecendo lá somente enquanto a seção do usuário que a criou está ativa.

Para o tamanho da memória existem quatro opções: 8(oito), 16(dezesseis), 32 (trinta e dois) e 64(sessenta e quatro) *Kilobytes*. Foram escolhidos valores potência de dois e numa faixa que viabilizasse a exibição numa página web.

As opções de tamanho da página são: 256(duzentos e cin-

Figura 13: Criar Simulação

quenta e seis), 512(quinhentos e doze) e 1024(mil e vinte e quatro) bytes. Deste modo, a quantidade de páginas que cabe na memória pode variar de 8(oito) à 256(duzentos e cinquenta e seis), dependendo da combinação entre os parâmetros tamanho da página e tamanho da memória.

As opções para tamanho dos setores foram 64(sessenta e quatro), 128(cento e vinte e oito) e 256(duzentos e cinquenta e seis) bytes. Escolhidas de modo que uma página seja no mínimo do mesmo tamanho do setor, e no máximo 16 vezes maior.

4.3.2 Gerência de Arquivos e Disco

Após definir os parâmetros da máquina virtual, a tela da simulação torna-se disponível para o usuário. Esta tela foi dividida em abas, o usuário pode navegar livremente entre elas. Na ordem de aparição as abas são: Disco, Processo, Memória e Máquina Virtual. A Figura 14 mostra a aba de gerência de disco.

A carga dos arquivos é feita por meio do upload de um arquivo, o arquivo a ser carregado deve estar dividido em instruções hexadecimais de 4 dígitos, seguidas pelo caractere ";"(ponto e vírgula). Para que o upload do arquivo possa ser feito, o simulador obriga o usuário a informar uma cor para representar o arquivo durante a simulação.

São cinco cores disponíveis por padrão no simulador: azul, laranja, rosa, vermelho, verde. Além disso, também é possível selecionar a opção personalizar, neste caso, uma paleta de cores aparece para o usuário montar a própria cor. A tela do sistema utilizada para fazer o upload pode ser vista na Figura 15.

Após selecionar o arquivo e a cor, o usuário envia as informações para o servidor. A partir deste momento, o simulador

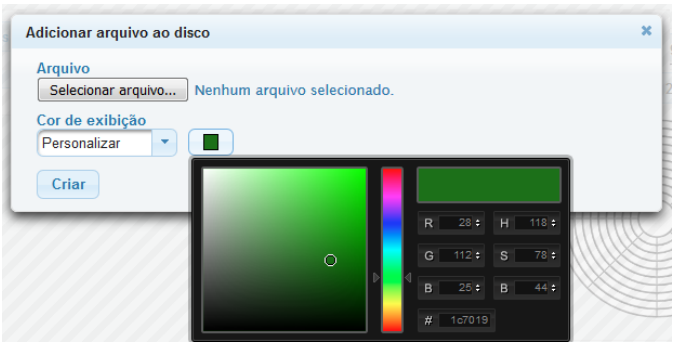

Figura 15: Criar arquivo

chama um método da classe que representa núcleo do sistema operacional. Esta chamada simula o que seria uma chamada de sistema solicitando uma gravação no disco. Será, então, gerada uma lista de requisições de acesso ao disco.

Visando otimizar o tempo de acesso ao disco, o sistema operacional por meio de um algoritmo de escalonamento do disco ordena as requisições. O algoritmo usado para estabelecer esta ordem é escolhido pelo usuário, por meio da opção de configurações da aba Disco.

As porções do disco ocupadas pelo arquivo carregado poderão ser vistas na grade "Disco Rígido"da aba Disco. A grade mostra uma ilustração do disco com a divisão prato, trilha e setor(Figura 14).

O movimento realizado pelo cabeçote de leitura/gravação é ilustrado abaixo da grade do "Disco rígido", sendo portanto uma das saídas mais importantes desta primeira aba. Este movimento também é exibido em outros momentos da simulação.

Um destes momentos é quando é acionada a opção "Simular Busca ao Disco", ela foi criada para melhorar a dinâmica das aulas de escalonamento de disco. Ao selecionar esta opção, uma *pop-up* modal é exibida (Figura 16) para que o usuário insira uma lista de números separados por virgulas, estes números representarão cilindros requisitados.

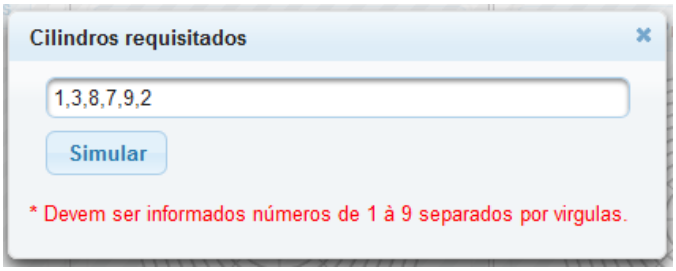

Figura 16: Simular Busca ao Disco

As requisições são ordenadas seguindo o algoritmo de escalonamento de disco selecionado, e a movimentação do cabeçote após a ordenação é exibida graficamente no painel de movimentação do cabeçote (Figura 17).

4.3.3 Gerência Processo

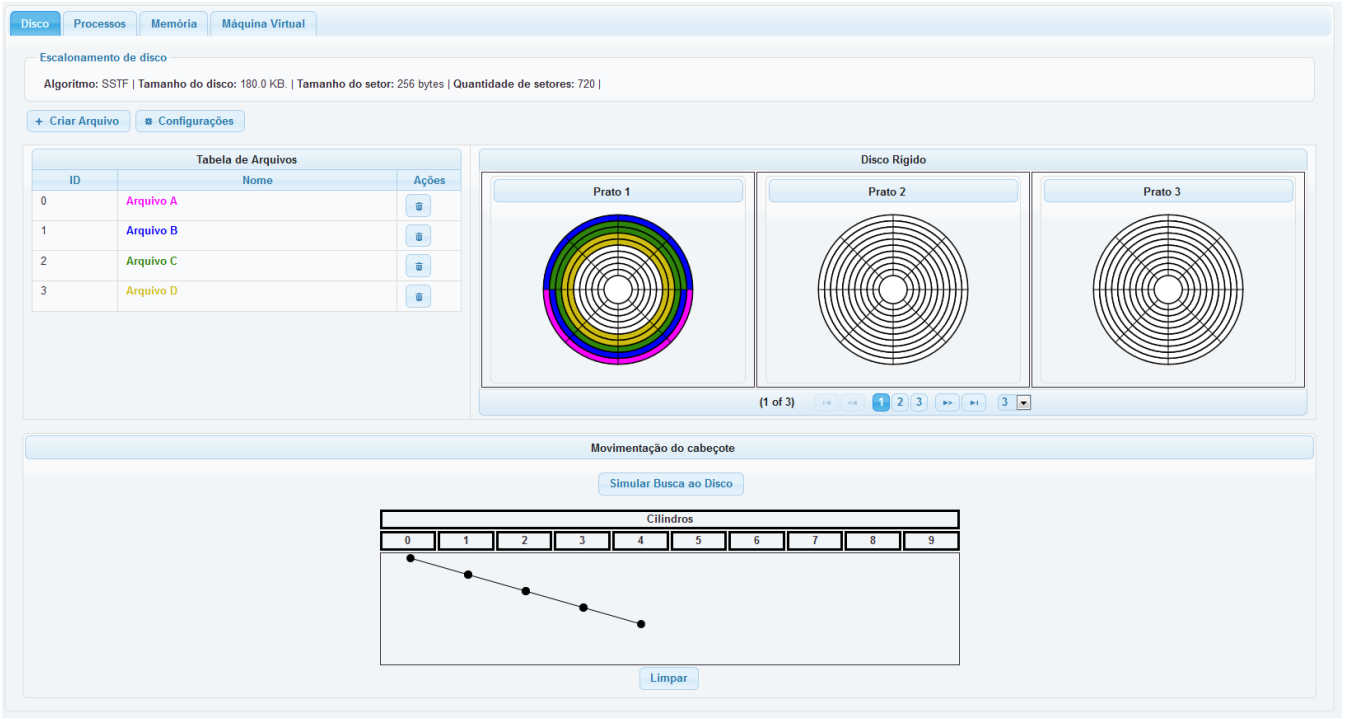

Figura 14: SWSO - Aba de gerência de disco

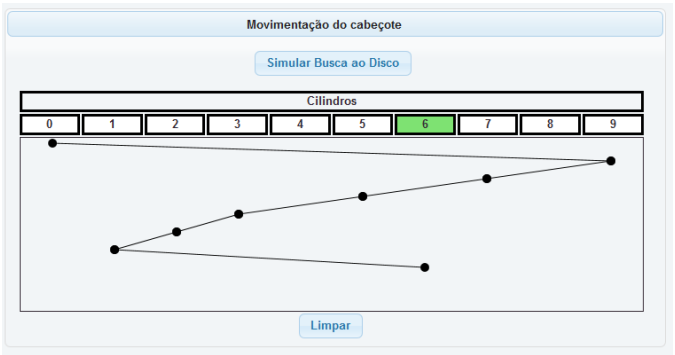

Figura 17: Painel de movimentação

A simulação de gerência de processo é feita pela aba "Processos"(Figura 18). Nela é possível criar um processo a partir de um programa, todos os arquivos carregados na aba de disco são visualizados como opções de programas nesta aba.

No momento da criação dos processos é possível definir a prioridade do processo, esta opção só fica disponível caso o algoritmo de simulação selecionado utilize prioridade. Entretanto, uma vez que é possível trocar o algoritmo de escalonamento no meio da simulação, a prioridade é sempre definida com o valor padrão 5 (cinco), este valor foi definido por marcar o meio entre a maior e menor prioridade possível neste simulador, 0 (zero) e 10 (dez) respectivamente.

Após a criação do primeiro processos, o usuário pode iniciar a simulação da gerência de processo, para isso ele tem a opção de simular a passagem do tempo, informando a quantidade de unidades de tempo que deseja adiantar. Este é um dos diferenciais deste trabalho, uma vez que dá a opção de ver a execução passo a passo.

Entretanto a adição de um processo na simulação pode ocorrer a qualquer momento, sendo que a quantidade de unidades de tempo, incrementadas a partir do inicio da simulação, determina o tempo de chegada dos processos. Assim, se o usuário deseja, por exemplo, simular que determinado processo foi criado no tempo 0(zero) e um outro no tempo 5(cinco), ele deverá criar o primeiro, incrementar cinco unidades de tempo e só então criar o segundo.

Depois de iniciar a simulação de processo, uma linha do tempo dos processos é exibida, mostrando quais processos ocuparam o processador durante a passagem do tempo. Esta linha permite ao professor focar na passagem do conhecimento, economizando o tempo que gastaria para fazer desenhos no quadro.

Além da linha do tempo, mais duas grades com informações relevantes à gerência de processos são exibidas, visando trazer o maior número de informações possíveis para a simulação.

A primeira grade mostra o nome, estado e quantidade de instruções, ou seja, as informações do processo. Já a segunda mostra as estatísticas de execução dos processo: tempo de chegada, de espera, de processador e turnaround.

Semelhante ao que ocorre na aba "Disco", nesta também existe uma opção de configuração, através dela o usuário consegue alterar o algoritmo de escalonamento. Caso o algo-

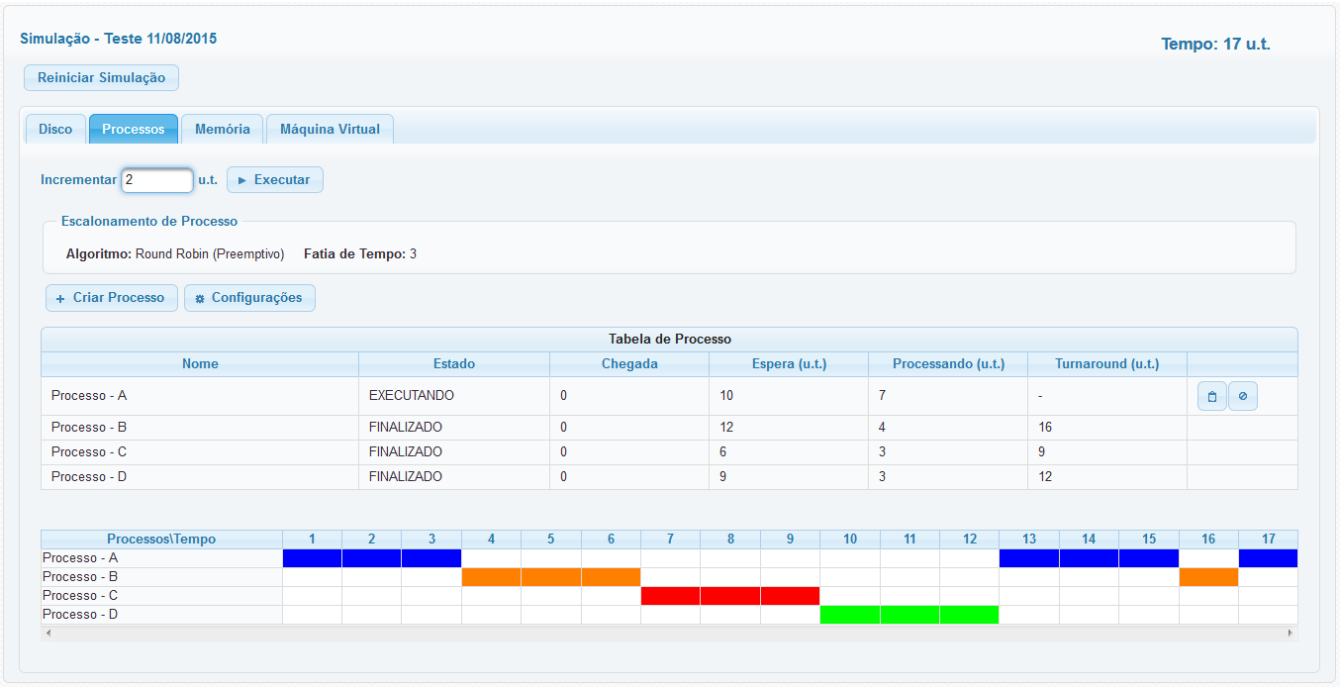

Figura 18: SWSO - Aba de Gerência de Processos

ritmo selecionado seja preemptivo, é possível também definir o timeslice da simulação.

4.3.4 Simular Substituição de Páginas da Memória

A aba "Memória"(Figura 19) é responsável pela simulação da gerência de memória. Nela é possível observar o comportamento da memória ao longo do tempo, vendo a quantidade de páginas que um processo tem na memória principal, a quantidade total e também a tabela de página dos processos.

Diferente das abas "Processo"e "Disco"não há muitas opções de ação nesta aba, uma vez que a simulação é integrada, as situações de memória a serem simuladas dependem da execução dos processos. Assim, para que, por exemplo, haja uma situação de falta de página, é necessário executar as instruções de um processo até que alcance a página que não está na memória principal.

Apesar de ter poucas opções de ação, a aba de memória é a que tem mais opções de configuração, nesta aba é possível configurar a política de busca, a política de alocação, o algoritmo de substituição de páginas e a quantidade de *frames* que um processo pode ter na memória principal.

Há também nesta aba uma grade que representa a memória principal, é por meio desta grade que o usuário consegue associar o processo e a posição da memória que ele ocupa. A cor associada ao processo é utilizada no background desta grade, sempre associada a posição que ele ocupa. Sendo, portanto, um elemento visual importante também para a esta etapa da simulação.

5. RESULTADOS

O estudo de caso deste simulador iniciou-se com testes do software, os testes iniciais visaram garantir que o simulador está atendendo a todos os requisitos elicitados, estes testes foram sendo executados a medida que o software estava sendo desenvolvido.

No entanto, ao fim do desenvolvimento foram criados testes para validar os algoritmos de escalonamento de disco, escalonamento de processo e substituição de páginas em diversas situações. Uma vez que eles foram desenvolvidos de forma mais independente (como já explicado na seção de implementação), além de serem o ponto principal da simulação, foi necessário testá-los com maior atenção.

Em seguida, após os testes funcionais, foi feito o teste de validação. Este teste consistiu em apresentar a ferramenta para os alunos da disciplina arquitetura de computadores e software básico, do Curso de Análise de Sistemas do Instituto Federal da Bahia. Esta etapa foi muito importante pois conseguimos identificar alguns pontos de melhoria na interface visual do simulador.

Além disso, como a apresentação ocorreu durante uma das aula da disciplina, conseguimos observar o comportamento da turma diante da nova ferramenta de ensino. Com isso, conseguimos avaliar que o simulador é capaz de ajudar no dinamismo das aulas, após a inserção da ferramenta, os alunos passaram a interagir mais com a preleção.

Outro aspecto positivo observado foi a economia de tempo, o trabalho permitiu que as novas situações e hipóteses criadas em sala de aula fossem verificadas no mesmo momento, e, o mais importante com demonstração visual, o que sem a ferramenta não seria possível, pois o tempo para fazer os desenhos e tabelas seria muito grande, e provavelmente ex-

Figura 19: SWSO - Aba de Gerência de Memória

cederia o tempo das aulas.

Também, foi feito uma levantamento quantitativo, para avaliar o nível de aprendizagem, com e sem o uso do simulador. Para isto, foi feita uma breve explicação de alguns temas relacionados ao ensino de sistemas operacionais.

Ao fim da explicação foi aplicada uma avaliação com perguntas relacionadas a alguns conceitos de sistemas operacionais. O questionário foi respondido por 20 alunos e os resultados foram comparados com o desempenho das turmas dos semestres anteriores, nos mesmos quesitos. O gráfico da Figura 20 mostra o resultado desta comparação.

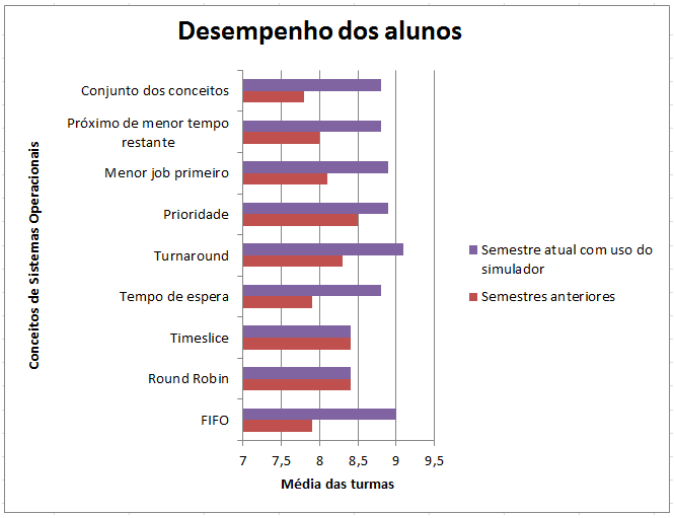

Figura 20: Gráfico de desempenho dos alunos

A análise dos resultados mostrou que no conjunto dos conceitos, a nota média dos alunos subiu de 7,9 para 8,8 comparado aos anos anteriores. Houve, portanto, uma melhora de aproximadamente 12,8% na avaliação geral. Para a obtenção desta nota foram consideradas todas as perguntas específicas de um determinado conceito e as que englobavam vários destes.

Os resultados foram quantificados também por conceito, a maior evolução percentual foi no entendimento do escalonamento FIFO, para este houve uma melhora de 13,9% na comparação das médias. A nota subiu de 7,9 para 9,0.

Outros conceitos que as notas aumentaram foram: tempo de espera (melhora de 11,3%), *turnaround* (melhora de 9,6%), algoritmos por prioridade (melhora de 4,7%), menor *job* primeiro e menor *job* primeiro preemptivo(aproximadamente 10% para ambos).

Já no entendimento dos conceitos relacionados ao algoritmo Round Robin e *timeslice* não houve alteração na média das notas.

Por último, os alunos responderam a uma pesquisa de opinião baseada na escala *Likert*, com o objetivo de conhecer o nível de concordância com afirmações relacionadas a eficácia do simulador como ferramenta de ensino. Nesta pesquisa foram feitas estas cinco afirmações:

1. O uso do simulador SWSO facilitou a visualização e entendimento dos conceitos de Escalonamento de Processos.
2. O simulador ajuda a entender o funcionamento dos algoritmos de escalonamento.
3. A interface visual do simulador é suficientemente boa para explicar os conceitos de sistemas operacionais.
4. O uso do simulador estimula a maior participação do aluno nas aulas.
5. Os alunos utilizarão o simulador fora da sala de aula para auxiliar no estudo da disciplina e nas atividades de casa.

Os níveis de concordância/discordância possíveis foram: concordo plenamente, concordo parcialmente, não possuo opi-

nião, discordo parcialmente e discordo plenamente. Na Figura 21 pode ser vista a distribuição percentual de alunos por nível de concordância com cada uma das afirmações.

	Concordo plenamente	Concordo parcialmente	Não possuo opinião
Afirmação 1	100%	0%	0%
Afirmação 2	50%	25%	25%
Afirmação 3	25%	50%	25%
Afirmação 4	75%	25%	0
Afirmação 5	50%	50%	25%

Figura 21: Quadro de distribuição do percentual de alunos por nível de concordância

O quadro suprimiu o percentual de alunos que discordaram plenamente ou totalmente, pois para nenhuma das cinco afirmações houve resposta discordante.

As respostas obtidas mostraram um alto nível de concordância, embora tenha sido notado que em algumas afirmações não houve 100% de concordância plena, isto mostra que ainda há pontos que podem ser melhorados. Alguns destes pontos estarão presentes na seção destinada aos trabalhos futuros.

Contudo, a avaliação foi muito positiva, no geral 85% dos alunos concordaram parcialmente ou totalmente com as afirmações e outros 15% não opinaram. No universo dos que concordaram parcialmente, sugiram algumas sugestões de melhorias, que foram interpretadas como uma intenção de utilizar a ferramenta e contribuir para torná-la melhor. Dentre os comentários o que melhor exemplifica esta situação foi:

> "O simulador facilitou bastante na compreensão e visualização dos conceitos ensinados. Contudo, construir um fluxograma dinâmico (montado de acordo com as ações do usuário) irá aumentar ainda mais a compreensão do que está acontecendo no sistema."

6. CONCLUSÃO E TRABALHOS FUTUROS

Foi possível avaliar que este trabalho cumpriu o seu principal objetivo, que era apresentar uma ferramenta de software web para a simulação do funcionamento de sistemas operacionais, e por meio desta ferramenta fazer a integração entre os conceitos de sistemas operacionais(gerência de memória e escalonamento de processos e disco).

Com o uso da ferramenta, já é possível simular o funcionamento de mais de dez diferentes algoritmos, entre escalonamento de processo, escalonamento de disco e substituição de páginas na memória. Além disso, ficou muito facilitada a inclusão de novos algoritmos, abrindo assim a possibilidade para o desenvolvimento de novos trabalhos.

A utilização deste trabalho em sala de aula, possibilitou perceber que a ferramenta conseguiu melhorar a dinâmica das aulas, aumentando o estimulo para os alunos manterem-se concentrados e consequentemente aumentando a capacidade de assimilação do conteúdo.

A ferramenta que ficará disponível na rede mundial de computadores, tem potencial para ser utilizada nas diversas instituições brasileiras, sendo portanto uma importante contribuição para o ensino de sistemas operacionais.

Apesar do objetivo principal ter sido alcançado, com o uso na sala de aula, foram percebidas algumas possíveis melhorias. Além disso, outras funções dos sistemas operacionais não foram simuladas, ou foram de forma muito simples. Estas melhorias e novas implementações poderão ser realizadas em trabalhos futuros, sendo as principais delas:

1. Permitir a simulação da estrutura de diretórios de pastas.
2. Implementar algumas funções do sistema de arquivo.
3. Implementação de outros algoritmos de escalonamento disco, escalonamento de processos ou substituição de páginas.
4. Como já existe uma busca e suporte para decodificação da instrução, isso pode ser continuado, tornado o simulador capaz de executar as instruções contidas nos arquivos. Funcionando como máquina virtual.

7. REFERÊNCIAS

[1] R. de Freitas Góis. Analise de requisitos, 2012. Acessado em 02/08/2015.

[2] Deitel, Deitel, and Choffnes. *Sistemas Operacionais*. Pearson Education Brasil, 2005.

[3] E. Y. Kioki, P. P. Santiago, and A. C. Soares. Um simulador didático como ferramenta de apoio ao ensino da disciplina de sistemas operacionais, 2008.

[4] F. B. Machado and L. P. Maia. Um framework construtivista no aprendizado de sistemas operacionais - uma proposta pedagógica com o uso do simulador sosim, 2001.

[5] L. P. Maia. Sosim: Simulador para o ensino de sistemas operacionais. Master's thesis, Universidade Federal do Rio de Janeiro, Março 2001.

[6] L. P. Maia. *Fundamentos de Sistemas Operacionais*. LTC, 2011.

[7] C. Maziero. Reflexões sobre o ensino prático de sistemas operacionais, 2002.

[8] R. S. Pressman. *Engenharia de Software*. McGraw-Hill, 2006.

[9] A. Silberschatz, P. B. Galvin, and G. Gagne. *Operating System Concepts with Java*. John Wiley & Sons, 2010.

[10] M. A. Silva. A importância do levantamento de requisitos no sucesso dos projetos de software. Acessado em 10/02/2014.

[11] I. Sommerville. *Engenharia de Software*. Addison-Wesley, 2007.

[12] A. S. Tanenbaum. *Sistemas Operacionais Modernos*. Pearson, 2010.