
Simulador de Gerência de Processos para Sistemas Operacionais

Guilherme Moreira Bomfim de Freitas



UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Guilherme Moreira Bomfim de Freitas

**Simulador de Gerência de Processos para
Sistemas Operacionais**

Trabalho de Conclusão de Curso apresentado
à Faculdade de Computação da Universidade
Federal de Uberlândia, Minas Gerais, como
requisito exigido parcial à obtenção do grau de
Bacharel em Sistemas de Informação.

Área de concentração: Sistemas de Informação

Orientador: Prof. Dr. Thiago Pirola Ribeiro

Uberlândia - MG

2023

Este trabalho é dedicado à minha família, à minha namorada, aos meus amigos, e especialmente à mim mesmo, pois mesmo havendo vários motivos para desistir continuei determinado a alcançar essa conquista.

Agradecimentos

Agradeço aos meus pais e avós, por servirem como ótimos exemplos de dedicação e persistência para correr atrás de seus sonhos e por sempre me darem apoio para alcançar os meus objetivos.

À minha namorada, por sempre estar disposta a me acolher, especialmente em momentos de desmotivação.

Aos meus irmãos e aos meus amigos, por sempre conseguirem me proporcionar momentos descontraídos.

Ao meu orientador, pelos ótimos direcionamentos e ensinamentos que me passou durante o desenvolvimento do TCC.

E a mim, por não ter desistido.

“Remember when you wanted what you currently have.”
(Autor desconhecido)

Resumo

Com o objetivo de reduzir a abstração dos conceitos na disciplina de Sistemas Operacionais e facilitar o entendimento dos estudantes, este trabalho aborda o desenvolvimento de uma ferramenta Web que simula o funcionamento de um sistema operacional, com ênfase em processador e gerência de processos. As definições e funcionalidades são representadas visualmente em uma interface moderna e de fácil utilização, possibilitando que o usuário crie e configure processos, visualize as listas de processos separadas por status e parametrize as configurações do processador (*Clock* da CPU, *Fatia de Tempo* e *Tempo de Espera de Entrada e Saída*). Os processos podem ser simulados com parametrizações diferentes durante a mesma simulação, como prioridades e tipos de processos (CPU-bound, IO-bound ou CPU e IO-bound). O software foi desenvolvido utilizando a plataforma *Angular*, além de também contar com a biblioteca *Angular Material* para a utilização de componentes atuais e de fácil manutenção.

Palavras-chave: Simulador, Sistemas Operacionais, Gerência de Processador, Angular, Ensino-Aprendizagem.

Lista de ilustrações

Figura 1 – Estrutura de um processo (MAIA; MACHADO, 2007)	15
Figura 2 – Transições entre os estados dos processos (TANENBAUM, 2010)	17
Figura 3 – Interface do simulador SOsim	20
Figura 4 – Interface do simulador SSOG	21
Figura 5 – Interface do simulador TBC-SO/WEB	22
Figura 6 – Interface do simulador WxProc	22
Figura 7 – Exemplo de estrutura de um processo.	24
Figura 8 – Exemplo da estrutura do vetor que contém os processos.	25
Figura 9 – Maior prioridade encontrada: 5.	25
Figura 10 – Lista resultante de processos cuja prioridade é igual a 5.	25
Figura 11 – Primeiro processo da lista resultante da filtragem.	25
Figura 12 – Visualização dos dados armazenados no <i>Local Storage</i>	26
Figura 13 – Pagina inicial do Simulador <i>SimulateOS</i>	28
Figura 14 – Listagem de Processos Prontos.	29
Figura 15 – Listagem de processos na Fila de I/O	29
Figura 16 – Lista de Processos Suspensos	30
Figura 17 – Indicação dos campos de Parametrização do Processador.	30
Figura 18 – Indicação da posição do botão para criar processos.	31
Figura 19 – Modal de Criação de Processo.	31
Figura 20 – Indicação do processo selecionado.	32
Figura 21 – Modal de Edição de Processo.	32
Figura 22 – Indicação da posição do botão Suspend Process.	33
Figura 23 – Indicação do processo "2543311" no final da lista de processos suspensos.	33
Figura 24 – Indicação da posição do botão Resumir	33
Figura 25 – Indicação da posição do botão Finalizar	34

Lista de tabelas

Tabela 1 – Tabela comparativa entre os simuladores analisados	23
---	----

Sumário

1	INTRODUÇÃO	10
1.1	Objetivos	10
1.2	Justificativa	11
1.3	Metodologia	11
1.4	Organização da Monografia	12
2	FUNDAMENTAÇÃO TEÓRICA	13
2.1	Educação e Sistemas Operacionais	13
2.2	Simulador	14
2.3	Didática por meio da Simulação	14
2.4	Componentes do Sistema Operacional	15
2.4.1	Processo	15
2.4.2	Espaço de Endereçamento	15
2.4.3	Contexto de Hardware	16
2.4.4	Contexto de Software	16
2.4.5	Estado dos Processos	16
2.4.6	Tipos de Processos	17
2.4.7	Modos de Escalonamento	17
2.5	Angular	18
2.5.1	TypeScript	18
2.5.2	Angular Material	18
2.6	Trabalhos Relacionados	18
2.6.1	S ² O	19
2.6.2	SOsim	19
2.6.3	SSOG	20
2.6.4	SWSO	20
2.6.5	TBC-SO/WEB	21
2.6.6	WxProc	21

2.6.7	Comparativo entre os Trabalhos Relacionados apresentados	23
3	ESTRUTURAÇÃO DO SOFTWARE	24
3.1	Estrutura de um processo	24
3.2	Vetor de processos	25
3.3	Escalonamento	25
3.4	Armazenamento de dados	26
4	EXPERIMENTOS E ANÁLISE DOS RESULTADOS	27
4.1	<i>SimulateOS</i>	27
4.1.1	Listagens de Processos	28
4.1.2	Campos de parametrização do processador	29
4.1.3	Criação dos processos	30
4.1.4	Editar processo	32
5	CONCLUSÃO	35
	REFERÊNCIAS	37

Introdução

Ao observar o desempenho de alunos da área da computação em algumas disciplinas mais complexas, tomando como exemplo a disciplina de Sistemas Operacionais, pode-se perceber que o nível de reprovação costuma ser elevado. Existem diversos materiais físicos e digitais, como livros, trabalhos científicos, artigos, documentações, entre outros, com o objetivo facilitar o processo de aprendizagem acerca dos conteúdos da disciplina. No entanto, ainda assim são instrumentos de teor apenas teórico e acredita-se que um dos principais motivos para o baixo aproveitamento dos alunos na disciplina seja a falta de assimilação do conteúdo teórico que pode ficar abstrato.

Pensando nisso, surge a ideia de que a utilização de um simulador por parte dos alunos pode despertar a curiosidade acerca dos assuntos abordados na disciplina de Sistemas Operacionais e, além disso, facilitar a absorção do conhecimento, visto que é instaurada uma "metodologia prática" no processo de aprendizado. Atualmente já existem projetos voltados para a simulação de Sistemas Operacionais como o de Maia (2001) e Carvalho et al. (2006) que auxiliam no entendimento das rotinas ilustradas.

1.1 Objetivos

Esse trabalho tem como principal objetivo desenvolver uma plataforma Web que busca facilitar o entendimento do conteúdo ministrado na disciplina de Sistemas Operacionais (com ênfase na gerência de processador) por meio da simulação de eventos e processos. O intuito é possibilitar a visualização e interação do usuário com os elementos abordados em sala de aula que estarão disponíveis no simulador, além de permitir que o usuário parametrize algumas configurações do processador e tenha total controle sobre os testes a serem realizados. Também haverá foco na experiência do usuário, buscando desenvolver uma interface fluida e que seja fácil de usar, tendo em vista que em soluções alternativas não é percebida uma grande preocupação com essa questão.

Em termos de aplicação, o propósito é que sejam desenvolvidas funcionalidades o suficiente para que seja possível realizar uma simulação completa relacionada ao gerenciamento

de processos. Dentre as funcionalidades essenciais destacam-se: criar processo, exibir os processos criados, atualizar dados do processo, manipular o estado do processo, finalizar processo, parametrizar configurações do processador e interromper funcionamento do simulador.

1.2 Justificativa

A disciplina de Sistemas Operacionais exige um certo grau de abstração que, normalmente, não é cobrado no ensino médio. Com isso, quando os alunos são apresentados aos conceitos da disciplina, iniciam-se as dificuldades e, por sua vez, inicia-se o desinteresse pela matéria, aumentando a retenção. Existem diversos softwares que realizam a simulação de conceitos de Sistemas Operacionais, como por exemplo, os desenvolvidos por Rocha et al. (2004), Kioki, Santiago e Soares (2008), Maia (2001), Reis, Júnior e Costa (2009), Ribeiro, Lima e Lobo (2014a), Ribeiro, Lima e Lobo (2014b), Carvalho et al. (2006) e TROPIX (2008), porém após feita uma análise desses softwares, pôde-se perceber que cada um deles trata apenas de conceitos, ou muito básicos ou avançados da disciplina, sem ter em um mesmo software ambos os níveis de complexidade dos conceitos.

1.3 Metodologia

Foi realizada uma pesquisa sobre os diversos simuladores que seguem a mesma premissa sugerida nesse trabalho (simular conceitos de Sistemas Operacionais), porém não foi encontrado um simulador ideal, que seja simples, possua uma interface amigável e também completo para facilitar o aprendizado. Dentre os analisados percebe-se que, ou são tratados conceitos muito básicos ou aprofundados demais para a disciplina e mesmo que possam ser ferramentas interessantes para ajudar alguns alunos, boa parte deles tem uma interface gráfica não muito convidativa.

O projeto foi desenvolvido utilizando-se a metodologia ágil no formato de *sprints*, de forma que a cada duas semanas se iniciou uma nova *sprint*, que por sua vez contou com etapas de refinamento de atividades, planejamento, desenvolvimento e revisão das tarefas. A metodologia ágil foi escolhida para que se garantisse uma entrega contínua de valor, com o objetivo de possibilitar ao orientador um maior acompanhamento da evolução do desenvolvimento do trabalho e, conseqüentemente, obter sugestões pontuais de melhorias.

Para o desenvolvimento da ferramenta foi utilizada a plataforma de desenvolvimento Angular, tendo como objetivo disponibilizar o simulador para acesso online, visando torná-lo acessível para a maioria dos professores e alunos, sem que haja a necessidade de instalação de software e, que possa ser acessado de qualquer lugar ou dispositivo conectado à Internet.

1.4 Organização da Monografia

No capítulo 2 serão apresentados os referenciais teóricos que darão base para entender o que foi proposto, já no capítulo 3 será explicado como foram definidos os comportamentos das funcionalidades desenvolvidas. No capítulo 4 serão apresentados os experimentos realizados no decorrer do desenvolvimento do simulador e alguns recursos que foram planejados, mas não tiveram continuidade. Por fim, no capítulo 5, será abordado como o simulador cumpriu com os objetivos propostos e quais são as recomendações para trabalhos futuros.

Fundamentação Teórica

Nesse capítulo serão apresentados os referenciais teóricos que darão base para o entendimento da proposta do trabalho. Inicialmente são abordadas informações sobre o processo de educação e como isso se dá no aprendizado da disciplina de Sistemas Operacionais, a didática por meio da simulação, conceitos fundamentais sobre os componentes do Sistema Operacional que tiveram foco no desenvolvimento da parte inicial do sistema (processo, estado dos processos e gerenciamento do processador), a plataforma Angular, que será a principal tecnologia utilizada no desenvolvimento do simulador proposto e, por fim, alguns trabalhos que compartilham da mesma premissa sugerida nesse trabalho.

2.1 Educação e Sistemas Operacionais

Para estudantes dos cursos de graduação com enfoque em tecnologia da informação, é essencial ter plena compreensão dos assuntos abordados na disciplina de Sistemas Operacionais, visto que, entender tais tópicos pode auxiliar na tomada de decisões durante o desenvolvimento de um software, além de também contribuir para uma melhor assimilação dos componentes de hardware de um sistema (MAZIERO, 2002).

Apesar de muito necessária, a disciplina de Sistemas Operacionais vem se mostrando um tanto desafiadora para os alunos que tentam cursá-la. Acredita-se que isso é consequência da vasta gama de conteúdos complexos apresentados na disciplina, tais como: gerenciamento de memória, arquiteturas, processadores, processos, *threads*, entre outros (MAZIERO, 2002).

Nos últimos anos, tem sido notável que, no processo de educação no sistema de ensino brasileiro ainda predomina-se o estilo de aula focado no modelo de aula expositiva, ou seja, o professor discorre sobre determinado assunto ou conceito durante um intervalo de tempo, ao passo que a conduta do aluno é quase que inteiramente passiva. Apesar de ser uma forma de ensino que, em alguns casos pode ser eficiente, nota-se uma carência na absorção das informações por parte dos estudantes. No caso da disciplina de Sistemas Operacionais, há uma quantidade considerável de assuntos complexos que, para serem ensinados aos

alunos com uma maior eficácia seguindo esse modelo, muitas vezes precisariam passar por um processo de simplificação, e isso provavelmente resultaria na omissão de alguns fragmentos importantes do conteúdo (MAZIERO, 2002).

Para suprir o deficit de assimilação do conteúdo apresentado no modelo de aula expositiva, surge uma abordagem voltada para um teor mais prático, que envolve simular os ambientes de um Sistema Operacional, concedendo ao aluno total controle e possibilidade de interação com os componentes, fazendo com que ele se torne o protagonista daquele sistema e adote uma conduta ativa perante sua obtenção de conhecimento, enquanto o professor adota uma postura de instrutor/tutor e oferece assistência aos alunos (MAZIERO, 2002).

2.2 Simulador

Um simulador é um aparelho/software capaz de reproduzir e simular o comportamento de algum sistema. Os simuladores reproduzem fenômenos e sensações que na realidade não estão ocorrendo.

Um simulador pretende reproduzir tanto as sensações físicas (velocidade, aceleração, percepção de paisagens) como o comportamento dos equipamentos da máquina que se pretende simular, ou ainda, de um qualquer produto final sem haver a necessidade de se gastar matéria prima, utilizar máquinas, mão-de-obra e gastar tempo (ODEN et al., 2006).

2.3 Didática por meio da Simulação

A simulação é uma técnica de estudo que tem como objetivo facilitar a compreensão de conceitos, que foram apresentados de forma teórica, por meio da reprodução de eventos e processos reais, permitindo que os estudantes trabalhem de forma realista com os conteúdos apresentados na disciplina.

Um dos benefícios mais relevantes de se adotar a simulação no processo didático de uma disciplina é intensificar a acessibilidade ao conteúdo, uma vez que o simulador contará com os principais componentes do tema abordado (MAIA, 2001). Além disso, pelo fato do simulador estar disponível em uma plataforma Web, basta que o estudante acesse um determinado endereço eletrônico, e não tenha a necessidade de realizar qualquer instalação de software. Isso pode ser verificado no simulador implementado por Borges, Martinez e Ribeiro (2021) para auxiliar na disciplina de Estrutura de Dados.

2.4 Componentes do Sistema Operacional

Dentre os diversos componentes de um Sistema Operacional, foram priorizados os Processos e alguns outros itens associados a eles, tais como: Espaço de Endereçamento, Contexto de Hardware, Contexto de Software, Estado dos processos, Tipos de Processos e Modos de Escalonamento.

2.4.1 Processo

Um processo é basicamente um programa em execução. Associado a cada processo está o espaço de endereçamento, uma lista de posições de memória que vai de 0 a algum máximo, onde o processo pode ler e escrever. O espaço de endereçamento contém o programa executável, os dados do programa e sua pilha. Também associado com cada processo há um conjunto de recursos, em geral abrangendo registradores (incluindo o contador de programa e o ponteiro de pilha), uma lista de arquivos abertos, alarmes pendentes, listas de processos relacionados e todas as demais informações necessárias para executar um programa. Um processo é na essência um contêiner que armazena todas as informações necessárias para executar um programa (TANENBAUM, 2010).



Figura 1 – Estrutura de um processo (MAIA; MACHADO, 2007)

2.4.2 Espaço de Endereçamento

Um espaço de endereçamento é o conjunto de endereços que um processo pode usar para endereçar a memória. Cada processo tem seu próprio espaço de endereçamento (onde pode ler e escrever), independente daqueles pertencentes a outros processos (exceto em algumas circunstâncias especiais onde os processos querem compartilhar seus espaços de endereçamento) (TANENBAUM, 2010). Essa área também é responsável por armazenar os dados utilizados pelo processo.

2.4.3 Contexto de Hardware

O Contexto de Hardware é composto pelo conteúdo dos principais registradores especiais, sendo eles o contador de programas, que contém o endereço de memória da próxima instrução a ser buscada; o ponteiro de pilha, que aponta para o topo da pilha atual na memória; e a palavra de estado do programa, que armazena os bits do código de condições.

Quando um processo está em execução, o seu contexto de hardware está armazenado nos registradores do processador. Além disso, é onde o sistema salva as informações do processo quando ele perde a utilização da CPU.

2.4.4 Contexto de Software

O contexto de Software armazena informações de identificação do processo, como o PID e UID, além de também definir os privilégios dele, delimitando suas ações sobre ele mesmo, outros processos e sobre o Sistema Operacional.

Ele também pode estabelecer limites de recursos que o processo pode alocar, como: número máximo de arquivos abertos, número máximo de endereços de memória, limite de tamanho do *buffer* para ações de entrada e saída, etc. Essas características são definidas assim que um processo é criado e algumas delas podem ser alteradas durante a execução.

2.4.5 Estado dos Processos

Apesar de um processo ser uma unidade independente e possuir informações próprias, muitas vezes existe a necessidade de que ele interaja com outros processos e, para que isso aconteça, seu estado precisa ser alterado durante sua execução.

Existem três estados nos quais um processo pode se encontrar (Figura 2): em execução (quando ele está efetivamente sendo executado pela CPU), pronto (quando está pronto para ser executado, porém aguardando sua vez de ser executado) e bloqueado (impossibilitado de ser executado até que seja desbloqueado) (TANENBAUM, 2010). Alguns autores ainda mencionam os estados iniciado e finalizado, porém nesses casos a transição é muito rápida e acaba sendo desconsiderada para fins didáticos.

Além do que foi mencionado, também existe o estado suspenso, que não foi implementado pois o simulador não engloba aspectos relacionados ao gerenciamento de memória. Em contrapartida, usou-se o termo "suspenso" para indicar o estado de um processo que foi pausado manualmente pelo usuário, esse processo só pode sair desse estado caso seja alterado pelo usuário.

1. O processo é bloqueado aguardando uma entrada
2. O escalonador seleciona outro processo para execução
3. O escalonador seleciona esse processo

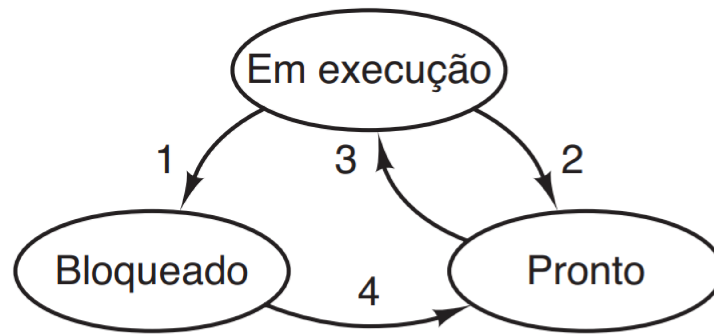


Figura 2 – Transições entre os estados dos processos (TANENBAUM, 2010)

4. A entrada torna-se disponível e o processo é desbloqueado

2.4.6 Tipos de Processos

Pode-se destacar dois principais tipos de processos, que estão relacionados ao local de execução (Entrada e Saída de dados e CPU).

- ❑ **Processos CPU Bound (orientados à CPU):** os processos passam a maior parte do tempo no estado de execução e realizam poucas operações de I/O.
- ❑ **Processos I/O Bound (orientados à Entrada/Saída):** os processos realizam muitas operações de entrada e saída de dados e seu tempo de execução é definido pela duração dessas operações. Comum em aplicações interativas.
- ❑ **Processos CPU e I/O Bound (orientados à CPU e à Entrada/Saída):** no simulador desenvolvido é possível criar processos que podem se comportar das duas formas (CPU Bound e I/O Bound), a orientação do processo é definida aleatoriamente pelo sistema, podendo variar a cada vez que ele entrar na fila de pronto.

2.4.7 Modos de Escalonamento

- ❑ **Preemptivo:** Um processo em execução permanece no processador por um tempo máximo pré-determinado ou até que o escalonador o remova de lá devido a outro processo de maior prioridade. Se ainda estiver executando ao final desse intervalo de tempo, o processo terá sua execução interrompida e o escalonador escolherá outro processo para executar (se houver algum disponível) (TANENBAUM, 2010). Se houver um processo mais prioritário (tempo real) o processo em execução também será interrompido, permitindo a atenção imediata ao mais prioritário (ROCHA et al., 2004).
- ❑ **Não-preemptivo:** o processo que está em estado de execução não pode ser interrompido, ele executa até que seja bloqueado (à espera de E/S ou de outro processo)

ou até que voluntariamente libere a CPU (TANENBAUM, 2010). Esse tipo de escalonamento foi utilizado nos primeiros sistemas multiprogramáveis, onde se tinha um processamento em batch (ROCHA et al., 2004).

2.5 Angular

Angular é uma plataforma de desenvolvimento construída em *TypeScript* e teve seu lançamento em setembro de 2016. Atualmente está na versão 15. Nele são incluídos: uma estrutura baseada em componentes para a construção de aplicativos web expansíveis; uma coleção de bibliotecas bem integradas que cobrem uma ampla variedade de recursos; e um conjunto de ferramentas de desenvolvedor para auxiliar no desenvolvimento, construção, homologação e atualização do código. Do protótipo à implantação global, o Angular oferece a produtividade e a infraestrutura escalável que suporta os maiores aplicativos do Google (ANGULAR, 2021).

Além de oferecer diversos modelos de componentes de interface para aceleração do desenvolvimento, com o Angular também é possível criar seus próprios modelos de componentes e utilizá-los em qualquer parte do sistema.

2.5.1 TypeScript

TypeScript é uma linguagem de programação gratuita e open source desenvolvida e mantida pela Microsoft e teve seu início em outubro de 2012. Atualmente está na versão 4.9. A linguagem é um superconjunto sintático de *JavaScript*, basicamente são adicionadas algumas sintaxes em cima do *JavaScript*, fazendo com que seja possível atribuir tipos às variáveis e propriedades no código (TYPESCRIPT, 2022).

2.5.2 Angular Material

Angular Material é uma biblioteca de componentes da interface do usuário que pode ser utilizada em projetos construídos em *Angular* para acelerar o desenvolvimento de interfaces elegantes e estáveis. Ela oferece componentes reutilizáveis como cartões, Inputs, tabelas, campos para seleção de data e vários outros (MENSAH, 2022).

2.6 Trabalhos Relacionados

Atualmente, durante o processo de aprendizagem, pode-se contar com a ajuda de simuladores para aprimorar o entendimento dos conceitos ministrados na aula teórica. Dos simuladores disponíveis na atualidade focados em Sistemas Operacionais, percebe-se que há uma divisão em dois grandes grupos: Simuladores Abrangentes que, como o próprio

nome já indica, englobam vários tópicos de um Sistema Operacional e Simuladores Específicos que têm como foco abordar, de maneira mais aperfeiçoada, uma menor quantidade de tópicos. Além dos simuladores, ainda existem Sistemas Operacionais para fins acadêmicos que são utilizados pelos alunos para implementação de rotinas criadas por eles próprios.

2.6.1 S²O

O simulador S²O foi desenvolvido por Carvalho et al. (2006) com o intuito de simular o comportamento de um Sistema Operacional, tendo como foco o gerenciamento de memória, gerência de processador e acesso ao disco. O software tem como objetivo auxiliar na fixação de conceitos ministrados nas aulas teóricas acerca de Sistemas Operacionais e tem como diferencial a exibição gráfica e gradual de Gerência dos Processos por diferentes políticas de escalonamento.

Esse simulador possui uma perspectiva pedagógica que aborda a utilização dos conceitos ministrados em diferentes disciplinas como: Algoritmos, Estrutura de Dados, Análise e Programação Orientada a Objetos, Arquitetura e Organização de Computadores, Linguagem de Programação e a própria disciplina de Sistemas Operacionais (CARVALHO et al., 2006).

2.6.2 SOsim

O simulador SOsim¹ pertencente à categoria de simuladores abrangentes voltando-se para auxiliar no aprendizado da disciplina de Sistemas Operacionais. A Figura 3 ilustra algumas telas do sistema.

Esse simulador é focado em uma abordagem visual do conteúdo e engloba alguns dos principais componentes de um Sistema Operacional: gerenciamento de memória virtual, gerenciamento de processos e processadores. Por este ser um simulador categorizado como abrangente, é possível realizar simulações superficiais dos temas listados anteriormente, contudo não há o devido aprofundamento em partes importantes do conteúdo, fazendo com que as interações do aluno sejam limitadas ao algoritmo ou abordagem definidos pelo software.

O software foi elaborado por Luiz Paulo Maia utilizando-se da linguagem Pascal, em um ambiente voltado para a orientação de objetos (MAIA, 2001). Apesar de ser visualmente interessante, o software é executado apenas em sistemas Windows por depender de bibliotecas específicas, o que atualmente acaba sendo um limitante para os estudantes.

¹ Pode ser acessado em: <<http://www.training.com.br/sosim/>>

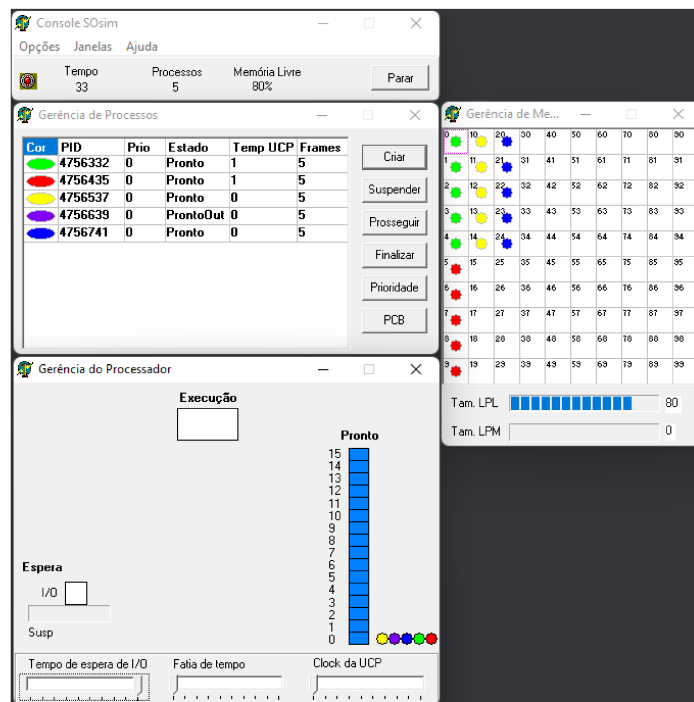


Figura 3 – Interface do simulador SOsim

Fonte: Autoria própria

2.6.3 SSOG

O Simulador de Sistema Operacional Genérico (SSOG) tem uma abordagem didática, contemplando temas como gerência de memória, gerência de processadores e gerência de processos. O simulador também possui um módulo de animação que possibilita a exibição de alguns problemas, por exemplo o problema "O Jantar dos Filósofos", que aborda a alocação de recursos críticos compartilhados por mais de um processo. O software foi elaborado utilizando-se da linguagem Java com paradigma de orientação a objetos (KIOKI; SANTIAGO; SOARES, 2008). As telas do sistema podem ser observadas na Figura 4.

Ainda que o SSOG contemple vários aspectos de um Sistema Operacional, percebe-se que existe um foco maior em problemas relativos aos processos e processadores, enquanto que houve certa omissão quando se trata do assunto de gerenciamento de memória, tratando apenas da questão de alocação de memória e demonstrando a quantidade de memória utilizada pelos processos, além de também exibir a quantidade de memória livre.

2.6.4 SWSO

O Simulador Web de Sistemas Operacionais (SWSO) tem o objetivo de oferecer ao professor uma ferramenta online de apoio às aulas, com foco nos conceitos: Gerência de Disco, Gerência de Memória (Memória Virtual) e Gerência de Processos. Ele é um

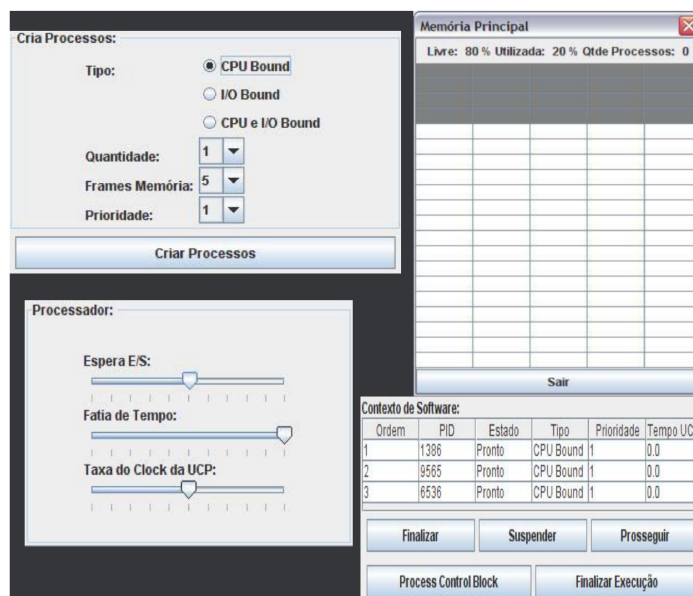


Figura 4 – Interface do simulador SSOG

Modificado de <https://fai-mg.br/portal/download/revista_inicia_2008/pub_dw_artigo_simulador.pdf>

software que permite alterar a propriedade *timeslice* do simulador e a prioridade dos processos, configurar os algoritmos do sistema e possui a praticidade de funcionar na web.

2.6.5 TBC-SO/WEB

O simulador TBC-SO/WEB possui caráter educativo sendo focado em uma abordagem visual do conteúdo, contando com uma interface gráfica para utilização via Web (Figura 5). Sua finalidade é auxiliar no ensino das políticas de escalonamento de processos e de alocação de memória.

Ainda que o simulador possua atributos interessantes, como por exemplo a apresentação do conteúdo teórico de determinado evento simulado, nota-se que há uma limitação quando se trata da quantidade de assuntos de gerenciamento de memória abordados.

O software foi elaborado utilizando-se da linguagem de programação Java, por meio do desenvolvimento de aplicativos embarcados em páginas Web (*applets*) (REIS; JÚNIOR; COSTA, 2009).

2.6.6 WxProc

Rocha et al. (2004) propõem o simulador WxProc (Figura 6) que tem como objetivo auxiliar na fixação de conceitos ministrados nas aulas teóricas da matéria de Sistemas Operacionais, tendo como foco a simulação de escalonamento de processos, proporcionando

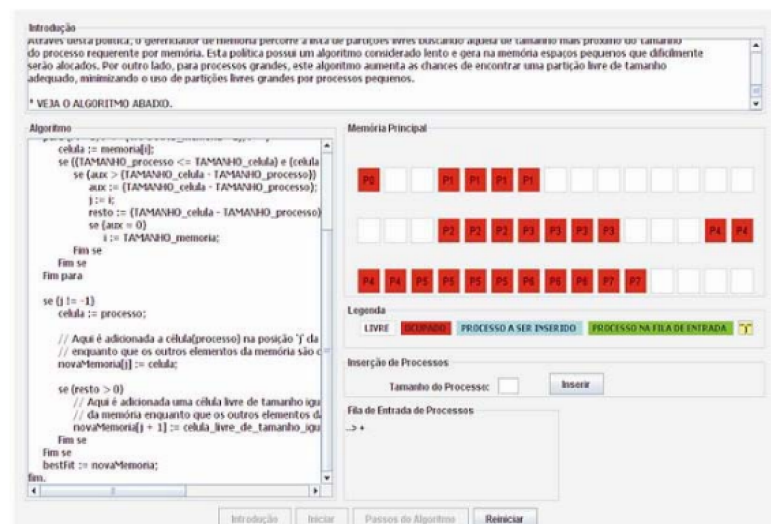


Figura 5 – Interface do simulador TBC-SO/WEB

Modificado de <http://www.niee.ufrgs.br/eventos/SBIE/2009/conteudo/artigos/completos/61463_1.pdf>

ao aluno a visualização do comportamento de um conjunto de processos após a submissão deles a alguma política de escalonamento presente no software. O WxProc simula os algoritmos *First-Come-First-Served*, *Remaining-Job-First*, *Round Robin* e *Shortest-Job-First*.

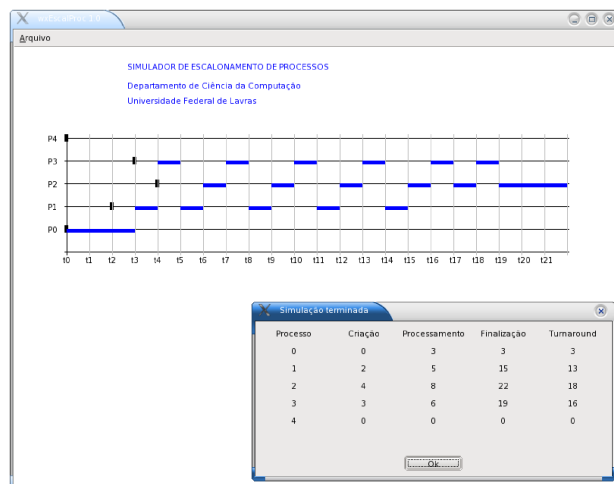


Figura 6 – Interface do simulador WxProc

Fonte: <<https://www.ic.unicamp.br/~rocha/college/src/wxEscalProc.png>>

O software foi elaborado utilizando-se da linguagem de programação C++ e das bibliotecas *Standart Template Library* (SLT) e *WxWindows*.

2.6.7 Comparativo entre os Trabalhos Relacionados apresentados

Os softwares dos trabalhos relacionados listados não apresentam disponibilidade online, com exceção dos simuladores *TBC-SO/WEB* e *SWSO*, podendo dificultar o acesso para alguns estudantes e consequentemente resultar em um maior desinteresse. Nota-se que o simulador *SWSO* atende todos os itens que foram comparados, contudo, a principal motivação para desenvolver o *SimulateOS* foi simular uma visualização simplificada do gerenciamento de processos de um Sistema Operacional, além de proporcionar uma alta manutenibilidade, ou seja, uma fácil manutenção do código e dos componentes desenvolvidos, e também possibilitar uma integração contínua, para que futuramente não haja nenhuma dificuldade em criar novas funcionalidades e componentes.

Tabela 1 – Tabela comparativa entre os simuladores analisados

Software	Gerência de Processador	Gerência de Processos	Disponível online	Gerência de Memória
SimulateOS	X	X	X	
S ² O	X	X		X
SOsim	X	X		X
SSOG	X	X		X
SWSO	X	X	X	X
TBC-SO/WEB		X	X	X
WxProc		X		

Estruturação do software

3.1 Estrutura de um processo

Cada processo é criado seguindo um modelo de estrutura pré-determinado que contém as seguintes propriedades:

- ❑ **Identificador:** É uma sequência de números gerada automaticamente pela aplicação para cada processo criado.
- ❑ **Prioridade:** Indica a prioridade de execução do processo. É definida no momento de criação do processo e pode ser alterada a qualquer momento.
- ❑ **Cor:** Indica a cor do processo. É definida no momento de criação do processo e pode ser alterada a qualquer momento.
- ❑ **Tipo:** Indica o tipo do processo. É definido no momento de criação do processo e pode ser alterado a qualquer momento.
- ❑ **Estado:** Indica em qual estado o processo se encontra.
- ❑ **Tempo de CPU:** Indica quanto tempo o processo já ficou em execução na CPU.
- ❑ **Tempo Criado:** Indica em qual momento da execução o processo foi criado.

```
color: "#aa00ff"  
cpuTime: 4  
id: "4149131"  
priority: 0  
state: "ready"  
timeCreated: 830  
type: "cpuBound"
```

Figura 7 – Exemplo de estrutura de um processo.

3.2 Vetor de processos

Todos os processos são contidos dentro de um único vetor, sendo inseridos na posição correta de acordo com sua prioridade.

```
▼ data: [{id: "2040640", priority: 5, color: "#40c4ff", type: "cpuBound", state: "execution", cpuTime: 4,...},...]
▶ 0: {id: "2040640", priority: 5, color: "#40c4ff", type: "cpuBound", state: "execution", cpuTime: 4,...}
▶ 1: {id: "2040589", priority: 5, color: "#d50000", type: "cpuBound", state: "ready", cpuTime: 6,...}
▶ 2: {id: "2033239", priority: 2, color: "#2979ff", type: "cpuBound", state: "ready", cpuTime: 4,...}
▶ 3: {id: "2033290", priority: 2, color: "#ff5252", type: "cpuBound", state: "ready", cpuTime: 4,...}
▶ 4: {id: "2033189", priority: 2, color: "#ff1744", type: "cpuBound", state: "ready", cpuTime: 6,...}
▶ 5: {id: "2030407", priority: 0, color: "#ff5252", type: "cpuBound", state: "ready", cpuTime: 2,...}
▶ 6: {id: "2030457", priority: 0, color: "#40c4ff", type: "cpuBound", state: "ready", cpuTime: 2,...}
▶ 7: {id: "2030508", priority: 0, color: "#b2ff59", type: "cpuBound", state: "ready", cpuTime: 2,...}
```

Figura 8 – Exemplo da estrutura do vetor que contém os processos.

3.3 Escalonamento

Para definir qual processo será executado, é realizado o seguinte procedimento:

1. primeiro identifica-se qual é a maior prioridade dentre os processos existentes que não estão suspensos;

```
▼ data: [...]
```

▶ 0:	{id: "2040589", priority: 5,
▶ 1:	{id: "2040640", priority: 5,
▶ 2:	{id: "2033239", priority: 2,
▶ 3:	{id: "2033290", priority: 2,
▶ 4:	{id: "2033189", priority: 2,
▶ 5:	{id: "2030407", priority: 0,
▶ 6:	{id: "2030457", priority: 0,
▶ 7:	{id: "2030508", priority: 0,

Figura 9 – Maior prioridade encontrada: 5.

2. depois é realizada uma filtragem no *vetor de processos*, buscando apenas por aqueles cuja prioridade é igual ao valor encontrado;

```
▼ data: [...]
```

▶ 0:	{id: "2040589", priority: 5,
▶ 1:	{id: "2040640", priority: 5

Figura 10 – Lista resultante de processos cuja prioridade é igual a 5.

3. por fim, é selecionado o primeiro processo da lista resultante da filtragem.

```
▶ 0: {id: "2040589", priority: 5
```

Figura 11 – Primeiro processo da lista resultante da filtragem.

Após ser executado, o processo deve ser realocado dentro do *vetor de processos*, de modo que seja posicionado depois de todos os demais processos cuja prioridade é maior ou igual à dele, porém, se só houverem processos com prioridade menor, ele será inserido logo no início do *vetor de processos*. Dessa forma, mantém-se a sequência de fila e a ordenação por prioridade.

3.4 Armazenamento de dados

Para o armazenamento de dados foi utilizado o *Local Storage* do navegador, nele são armazenadas todas as informações dos processos, cores, o tempo de execução do simulador, a fatia de tempo, o clock da CPU e o tempo de espera de I/O.

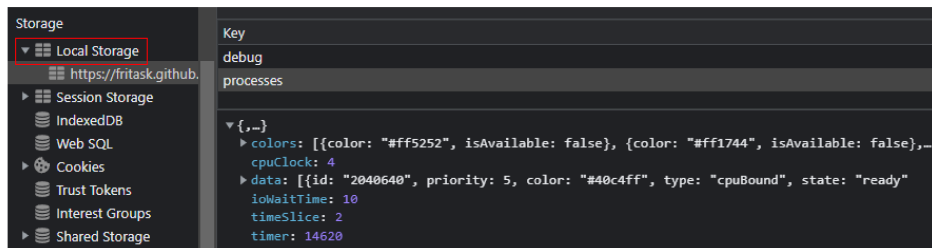


Figura 12 – Visualização dos dados armazenados no *Local Storage*

Experimentos e Análise dos Resultados

Nesse capítulo serão apresentados os experimentos realizados durante o desenvolvimento do simulador e as funcionalidades planejadas que não foram desenvolvidas. Além disso, como análise dos resultados haverá um conjunto de imagens para ilustrar a interface do simulador, assim como o detalhamento das funcionalidades que estão presentes no sistema.

O simulador *SimulateOS* foi desenvolvido utilizando-se o *Framework* de aplicações Web *Angular*, tendo como principal objetivo simplificar o desenvolvimento e manutenção da aplicação através da divisão de cada funcionalidade do sistema em componentes (componentização). Para criar as tabelas de processos, os campos parametrizáveis do processador simulado e outras partes do sistema foram usados componentes da biblioteca *Angular Material*.

4.1 *SimulateOS*

O sistema desenvolvido tem como objetivo simular o funcionamento de componentes de um sistema operacional, com ênfase na gerência de processador. O software foi projetado pensando em proporcionar ao usuário uma experiência de usabilidade simples e com uma interface amigável, visando facilitar o entendimento acerca do que está acontecendo no todo da simulação, além de contar com a facilidade de acesso por estar disponível online. Inicialmente, para testes, o sistema está disponibilizado temporariamente em <<https://fritask.github.io/>>.

A Figura 13 ilustra a página inicial do simulador *SimulateOS*. Na parte superior da tela encontra-se a logo e título do software. Logo abaixo fica o menu principal, no qual estão situados o tempo decorrido desde a abertura do software, a quantidade de processos, os campos de parametrização do processador (tempo de espera de I/O, fatia de tempo e *clock*) e o botão "Parar", que serve para interromper a simulação e remove os processos existentes. No centro da página está situada a seção de gerência de processos possibili-

tando criar processos, acompanhar as listas de processos criados, as informações de cada processo e visualizar qual processo está em execução e qual está em I/O.

De início, foi pensado em dividir o simulador em duas abas: na primeira aba seria possível visualizar os processos e suas propriedades, e a segunda aba seria responsável por conter as informações do processador simulado. Visando alcançar uma maior praticidade, essa divisão de informações foi repensada. Com isso, houve uma reestruturação do planejamento visual para que fosse possível unir todas as informações em uma única página, de modo que fosse mantida a organização das informações e a fluidez da interface.

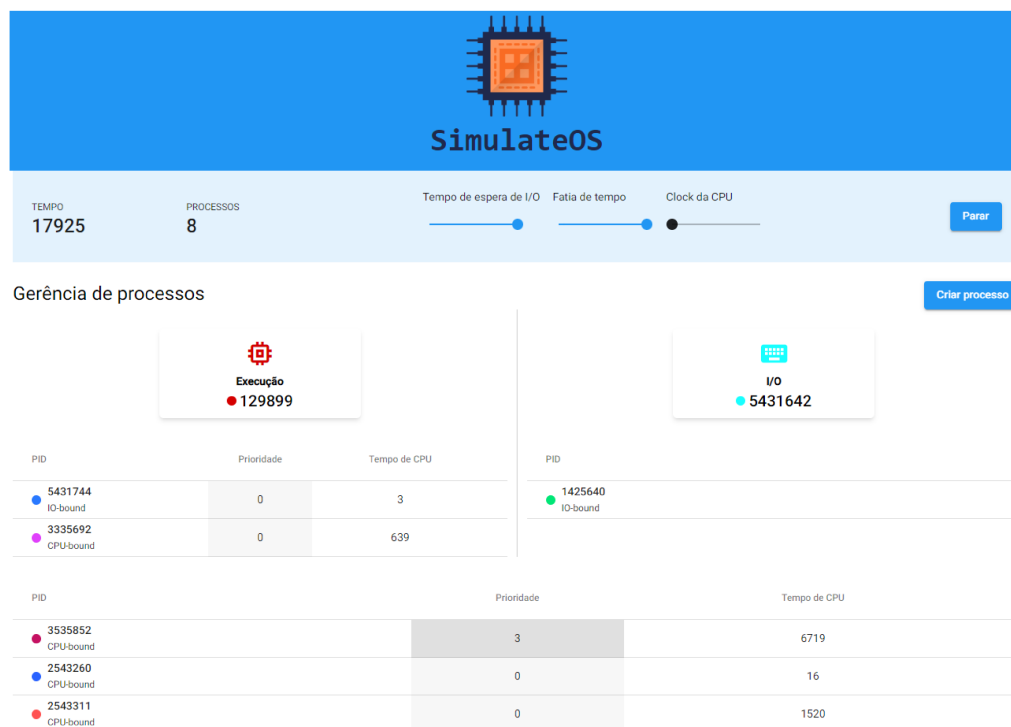
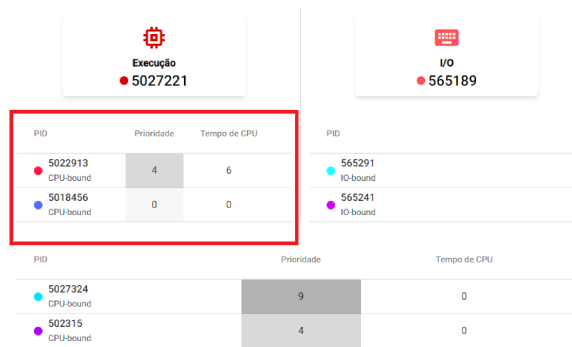


Figura 13 – Pagina inicial do Simulador *SimulateOS*.

4.1.1 Listagens de Processos

O simulador foi proposto para fins didáticos e, com isso, os processos foram separados visualmente em 3 filas/listas: Processos Prontos, Processos na Fila de I/O e Processos Suspensos.

- ❑ **Processos Prontos:** aqui encontram-se os processos que estão com o estado definido como "pronto", ou seja, aguardando na fila para execução (destacado na Figura 14). Essa listagem é ordenada de acordo com a prioridade dos processos, exibindo primeiro os processos com maior prioridade. Quanto maior o valor no campo prioridade, maior será a prioridade do processo. Dentro das prioridades, o processo que volta da execução entrará no final da fila de sua prioridade, trabalhando no esquema de fila circular com prioridades.

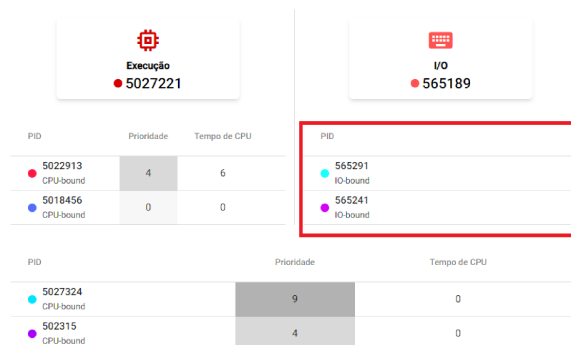


PID	Prioridade	Tempo de CPU
5022913 CPU-bound	4	6
5018456 CPU-bound	0	0

PID	Prioridade	Tempo de CPU
5027324 CPU-bound	9	0
502315 CPU-bound	4	0

Figura 14 – Listagem de Processos Prontos.

- ❑ **Processos na Fila de I/O:** nessa lista encontram-se os processos que estão com o estado definido como "pronto" para realizar a Entrada/Saída (I/O) (destacado na Figura 15). A listagem é ordenada respeitando a ordem de chegada nessa fila, independente da prioridade do processo.



PID	Prioridade	Tempo de CPU
5022913 CPU-bound	4	6
5018456 CPU-bound	0	0

PID	Prioridade	Tempo de CPU
5027324 CPU-bound	9	0
502315 CPU-bound	4	0

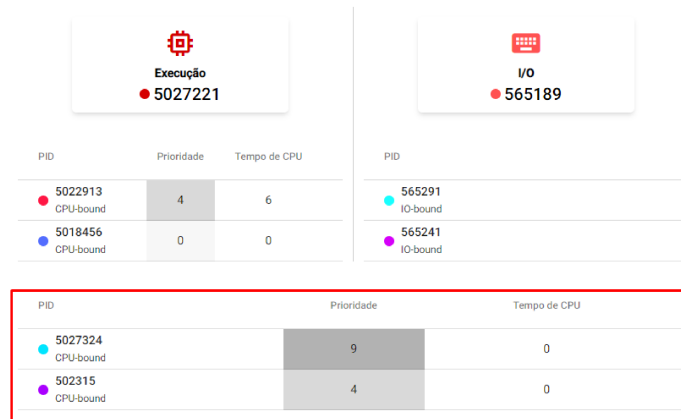
Figura 15 – Listagem de processos na Fila de I/O

- ❑ **Processos Suspensos:** nessa listagem encontram-se os processos que estão com o estado definido como "suspensão" (destacado na Figura 16). A listagem é ordenada de acordo com a prioridade dos processos, respeitando a ordem de chegada nessa fila.

4.1.2 Campos de parametrização do processador

Na Figura 17 (recorte da parte superior do simulador), em destaque, os três campos de parametrização do processador: Tempo de Espera de I/O, Fatia de Tempo e o Clock da CPU.

- ❑ **Tempo de Espera de I/O:** esse campo serve para definir quantos segundos um processo do tipo IO-bound vai permanecer em I/O. Os valores possíveis para esse



PID	Prioridade	Tempo de CPU
5022913 CPU-bound	4	6
5018456 CPU-bound	0	0

PID	Prioridade	Tempo de CPU
565291 I/O-bound		
565241 I/O-bound		

PID	Prioridade	Tempo de CPU
5027324 CPU-bound	9	0
502315 CPU-bound	4	0

Figura 16 – Lista de Processos Suspensos



Figura 17 – Indicação dos campos de Parametrização do Processador.

campo podem variar de 1 até 10 segundos. Esse tempo foi definido para que a simulação não seja afetada por uma parametrização incorreta e acabe "travando".

- ❑ **Fatia de Tempo:** nesse campo é definido quanto tempo (em segundos) um processo CPU Bound ocupará o processador a cada execução. Os processos I/O Bound sempre ocupam o processador por apenas 1 segundo, independente do valor definido para a fatia de tempo. Os valores possíveis para esse campo variam de 2 até 10 segundos. Na literatura esse termo aparece como *Time-Slice* ou *Quantum* e apenas é aplicado à sistemas de Tempo Compartilhado.
- ❑ **Clock da CPU:** esse campo serve para definir a taxa de velocidade do clock da CPU. Por ser uma simulação, não é alterado o clock real do processador físico, com isso, os valores possíveis para esse campo variam de 1 até 10 segundos.

4.1.3 Criação dos processos

Na Figura 18, indicada pela seta vermelha, está a localização do botão "Criar processo".

Ao clicar no botão, é aberto o modal de criação de processo (Figura 19). Nesse modal encontram-se os campos: Prioridade, Tipo de Processo, Cor do Processo e Quantidade

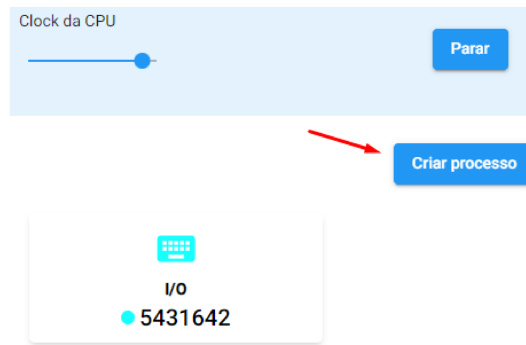


Figura 18 – Indicação da posição do botão para criar processos.

de Processos. Foi definido um limite de 15 processos simultâneos no simulador. Caso um processo seja finalizado, este deixará de ocupar sua posição no conjunto de processos e permitirá que seja criado outro processo em seu lugar.

O modal de criação de processo é uma janela com o título 'Criação de processo'. Ele contém quatro campos de entrada: 'Prioridade' com o valor '0', 'Tipo de processo' com o valor 'CPU-bound' e uma seta para baixo, 'Cor do processo' com o valor '#ff5252' e um círculo vermelho, e 'Quantidade de processos' com o valor '1'. Na parte inferior direita, há dois botões: 'Cancelar' e 'Ok'.

Figura 19 – Modal de Criação de Processo.

Os itens que aparecem para a parametrização da criação do processo são:

- ❑ **Prioridade:** nesse campo é definida a prioridade do processo (podendo variar de 0 a 15), quanto maior o número, maior prioridade de execução terá o processo.
- ❑ **Tipo de processo:** nesse campo é definido qual será o tipo do processo, sendo que as possibilidades de escolhas implementadas foram: CPU-bound, IO-bound ou CPU e IO-bound.
- ❑ **Cor do processo:** nesse campo é definida a Cor do Processo, sendo que cada processo tem uma cor própria para facilitar sua identificação visual durante a execução.
- ❑ **Quantidade de processos:** nesse campo é definida a quantidade de processos a serem criados com base nas informações fornecidas nos outros campos, com exceção da cor. Quando são criados vários processos simultaneamente, a cor é definida somente para um deles e as cores dos demais processos criados são definidas automaticamente pelo sistema. O máximo de processos permitido será 15, porém no

momento da criação, será permitido apenas criar a quantidade de processos que restam para completar o máximo permitido.

4.1.4 Editar processo

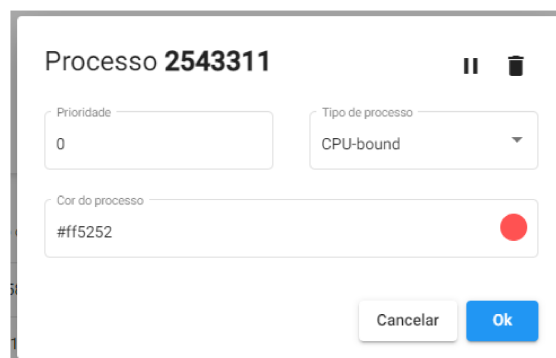
Ao clicar em um processo de qualquer uma das listas, é possível editar suas informações. Para esse exemplo foi selecionado o processo "2543311" na lista de processos prontos (Figura 20).



PID	Prioridade	Tempo de CPU
5431642 IO-bound	1	45
2543311 CPU-bound	0	1520
5431744 IO-bound	0	3

Figura 20 – Indicação do processo selecionado.

Assim que um processo é clicado, abre-se o modal de edição (Figura 21), no qual é possível alterar as informações do processo (Prioridade, Tipo e Cor). Esse modal contém os botões: Suspendar Processo, Resumir Processo e Finalizar Processo, sendo estes:



Processo 2543311

Prioridade: 0

Tipo de processo: CPU-bound

Cor do processo: #ff5252

Cancelar Ok

Figura 21 – Modal de Edição de Processo.

- ❑ **Suspendar Processo:** este item permite a ação para suspender o processo (indicado na Figura 22). Esta opção ficará disponível no modal apenas quando o processo em questão não estiver suspenso.

Ao clicar no botão de *Suspendar Processo*, o estado do processo é alterado para "suspenso" e o processo vai para o final da lista de processos suspensos (Figura 23).

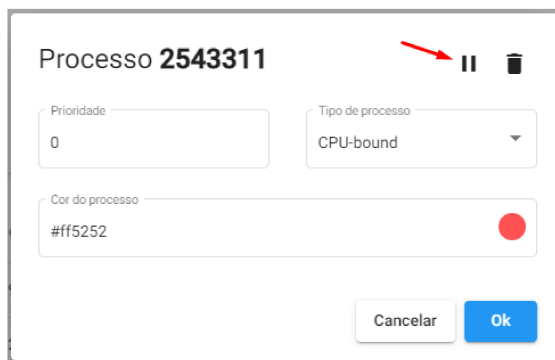


Figura 22 – Indicação da posição do botão Suspende Processo.

PID	Prioridade	Tempo de CPU
3535852 CPU-bound	3	6719
2543311 CPU-bound	0	1520

Figura 23 – Indicação do processo "2543311" no final da lista de processos suspensos.

- ❑ **Resumir Processo:** a ação de resumir (indicado na Figura 24) fica disponível apenas quando o processo estiver suspenso. Ao clicar no botão, o estado do processo é alterado para "pronto" e o processo vai para o final da lista de processos prontos, respeitando a ordem de prioridade.

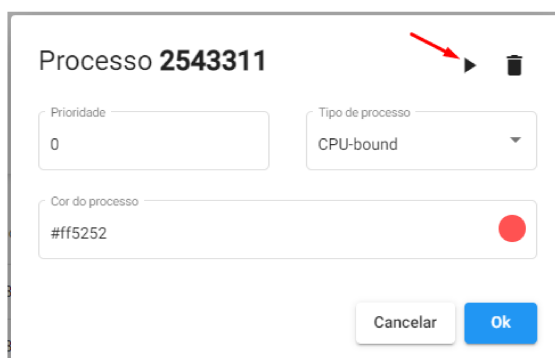


Figura 24 – Indicação da posição do botão Resumir

- ❑ **Finalizar Processo:** a ação de finalizar processo (indicado na Figura 25) faz com que o estado do processo seja alterado para "finalizado", tornando sua cor disponível para uso e removendo-o do fluxo normal dos processos. As informações sobre o processo finalizado são mantidas internamente no simulador, porém após finalizar um processo não é mais possível editá-lo ou reativá-lo.

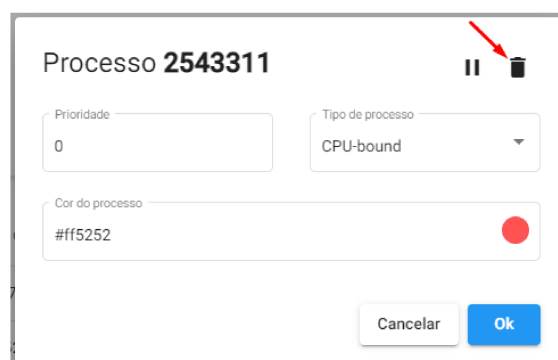


Figura 25 – Indicação da posição do botão Finalizar

Conclusão

Nesse trabalho foi apresentada uma ferramenta que foi desenvolvida para facilitar a assimilação do conteúdo aprendido na disciplina de Sistemas Operacionais por meio da simulação de eventos e processos reais. O foco do software é simular o gerenciamento de um processador e seus processos, permitindo que o usuário parametrize algumas configurações do processador, e crie e gerencie os processos da maneira que desejar.

Também foi levada em conta a fluidez da interface e a facilidade de uso da ferramenta por parte do usuário, pois foi observado que, em soluções semelhantes, a interface e usabilidade não foram pontos que tiveram muita atenção. Além desse fato fazer com que o usuário tenha uma experiência confortável e imersiva durante a simulação, também pode influenciar na receptividade da ferramenta para adoção dela como uma opção viável de apoio ao ensino/aprendizado. A escolha das tecnologias utilizadas e dos métodos de desenvolvimento foi crucial para que se fosse possível alcançar um alto nível de qualidade gráfica e de usabilidade.

O *SimulateOS* foi desenvolvido utilizando-se a tecnologia Angular, visando uma estrutura de aplicação expansível e que facilite a atualização do código. Com isso, é possível que os componentes já existentes do sistema sejam aprimorados e que sejam implementadas novas funcionalidades com facilidade, além de contar com uma vasta gama de componentes já existentes e visualmente agradáveis.

Devido à restrição de tempo imposta para o desenvolvimento e escrita do TCC, foi possível desenvolver apenas os componentes focados em simular um processador e seus processos, deixando de lado as funcionalidades referentes a relatórios. Porém, como mencionado nos objetivos do trabalho, cumpriu-se a meta de desenvolver todas as funcionalidades essenciais para que seja possível realizar uma simulação completa.

Apesar do desenvolvimento ter sido restringido ao gerenciamento do processador, muito foi feito e pensado nesse projeto. Com isso, muitos itens foram preparados e implementados já pensando em futuras implementações de novas funcionalidades para auxiliar na simulação de outros itens estudados em Sistemas Operacionais.

Como trabalhos futuros, faz-se necessária a realização de testes com o simulador por

estudantes durante aulas da disciplina de Sistemas Operacionais, tendo em vista o enriquecimento do trabalho ao medir a efetividade e aceitação do sistema. Além disso, seria pertinente desenvolver funcionalidades que abordem outros componentes de um Sistema Operacional, como Gerência de Memória, Threads, simulação de múltiplos processadores, etc.

Em termos de diagnósticos, poderia ser desenvolvida uma funcionalidade de relatórios que registrasse o comportamento dos processos, como o momento em que o processo entrou em execução, quanto tempo ficou em execução, se houve alteração de prioridade, entre outras informações relevantes.

Este trabalho, mesmo antes de ser finalizado, já foi parcialmente publicado no IX FACOM TechWeek e XVI Workshop de Teses e Dissertações em Ciência da Computação (FREITAS; RIBEIRO, 2022) e pretende-se submeter para um evento nacional ainda em 2023.

Referências

- ANGULAR. **What is Angular?** 2021. <<https://angular.io/guide/what-is-angular>>. Citado na página 18.
- BORGES, L. F.; MARTINEZ, A. C.; RIBEIRO, T. P. O ensino de estrutura de dados auxiliado por uma plataforma didática na web. In: **Anais Estendidos do XVII Simpósio Brasileiro de Sistemas de Informação**. Porto Alegre, RS, Brasil: SBC, 2021. p. 37–40. ISSN 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/sbsi_estendido/article/view/15351>. Citado na página 14.
- CARVALHO, D. S. et al. S2O: Uma Ferramenta de Apoio ao Aprendizado de Sistemas Operacionais. In: **XXVI Congresso da SBC – XIV Workshop sobre Educação em Computação (XIV WEI)**. [S.l.: s.n.], 2006. Citado 3 vezes nas páginas 10, 11 e 19.
- FREITAS, G. M. B.; RIBEIRO, T. P. Simulador de gerenciamento de processos para sistemas operacionais. In: **Anais do IX FACOM TechWeek e XVI Workshop de Teses e Dissertações em Ciência da Computação**. [S.l.: s.n.], 2022. Citado na página 36.
- KIOKI, E. Y.; SANTIAGO, P. P.; SOARES, A. C. Um simulador didático como ferramenta de apoio ao ensino da disciplina de sistemas operacionais. **Revista INICIA**, v. 8, n. 8, p. 41–48, 2008. Citado 2 vezes nas páginas 11 e 20.
- MAIA, L. P. **SOsim: Simulador para o Ensino de Sistemas Operacionais**. Dissertação (Mestrado) — Universidade Federal do Rio de Janeiro – UFRJ, 2001. Citado 4 vezes nas páginas 10, 11, 14 e 19.
- MAIA, L. P.; MACHADO, F. B. **Arquitetura De Sistemas Operacionais**. São Paulo: LTC, 2007. Citado 2 vezes nas páginas 6 e 15.
- MAZIERO, C. A. Reflexões sobre o ensino prático de sistemas operacionais. In: **Workshop de Ensino de Informática da SBC**. [S.l.: s.n.], 2002. Citado 2 vezes nas páginas 13 e 14.
- MENSAH, O. **Creating Beautiful Apps with Angular Material**. 2022. <<https://auth0.com/blog/creating-beautiful-apps-with-angular-material/>>. Citado na página 18.

ODEN, J. T. et al. Revolutionizing engineering science through simulation. **National Science Foundation Blue Ribbon Panel Report**, v. 65, 2006. Citado na página 14.

REIS, F. P.; JÚNIOR, P. A. P.; COSTA, H. A. X. TBC-SO/WEB: Um Software Educacional para o Ensino de Políticas de Escalonamento de Processos e de Alocação de Memória em Sistemas Operacionais. In: **Anais do Simpósio Brasileiro de Informática na Educação**. [S.l.: s.n.], 2009. v. 1. Citado 2 vezes nas páginas 11 e 21.

RIBEIRO, T. P.; LIMA, R. L. B.; LOBO, E. A. Simulador de rotinas do sistema operacional para auxílio às aulas teóricas. In: **Anais do X Simpósio Brasileiro de Sistemas de Informação**. Porto Alegre, RS, Brasil: SBC, 2014. p. 737–748. ISSN 0000-0000. Disponível em: <<https://sol.sbc.org.br/index.php/sbsi/article/view/6160>>. Citado na página 11.

_____. Simuladores de gerência de memória e processador para auxílio às aulas teóricas de sistemas operacionais. In: **Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)**. [S.l.: s.n.], 2014. v. 25, p. 1028. Citado na página 11.

ROCHA, A. R. et al. wxProc: Um Simulador de Políticas de Escalonamento Multiplataforma. **INFOCOMP Journal of Computer Science**, v. 3, n. 1, p. 43–47, 2004. Citado 4 vezes nas páginas 11, 17, 18 e 21.

TANENBAUM, A. **Sistemas Operacionais Modernos**. [S.l.]: Pearson Prentice Hall, 2010. ISBN 9788587918574. Citado 5 vezes nas páginas 6, 15, 16, 17 e 18.

TROPIX. **Projeto TROPIX - Sistema Operacional**. 2008. <<http://www.tropix.nce.ufrj.br>>. Citado na página 11.

TYPESCRIPT. **What is TypeScript?** 2022. <<https://www.typescriptlang.org/>>. Citado na página 18.