Simulador de Sistemas Operacionais PampaSim

Igor G. Dutra[1], **Pedro P. Souza** [1], **Rafael B. Amaral**[1]

[1]Universidade Federal de Pelotas (UFPEL)
Rua Gomes Carneiro, 1 – Centro, Pelotas, RS, CEP 96010-610
{*igdutra, ppsouza, rafael.amaral*}*@inf.ufpel.edu.br*

Resumo. *O ensino de sistemas operacionais enfrenta o desafio da abstração, onde mecanismos internos como o escalonamento de processos são de difícil observação em sistemas reais. O PampaSim surge como uma solução didática baseada em eventos discretos, permitindo que estudantes visualizem e experimentem cenários de execução de processos em tempo real. Construído sob o padrão arquitetural MVVM e desenvolvido utilizando Java e JavaFX, o simulador oferece uma interface intuitiva para a configuração de algoritmos de escalonamento e criação de processos. Este trabalho explora a estrutura modular do sistema, suas funcionalidades principais e sua relevância como recurso aberto e expansível para suporte do estudo de sistemas operacionais.*

1. Introdução

Os sistemas operacionais são pilares da computação, gerenciando recursos de *hardware* e fornecendo abstrações essenciais para usuários e desenvolvedores. Entretanto, a própria natureza dessas abstrações oculta o funcionamento destes mecanismos de gerência de recursos, dificultando a aprendizagem prática. Compreender profundamente esses conceitos dinâmicos é vital para a formação de profissionais capazes de otimizar aplicações e recursos. Diante desse cenário, o projeto PampaSim foi concebido para atuar como um instrumento pedagógico capaz de materializar modelos teóricos em um ecossistema observável. Embora a literatura apresente simuladores consolidados no ensino, como o SOSim [Maia 2001] e o Nachos [Christopher et al. 1993], um dos diferenciais do PampaSim é ser fruto de um projeto[1] de pesquisa em desenvolvimento contínuo. Esse aprimoramento acadêmico ativo permite à ferramenta conciliar rigor arquitetural com uma experiência de uso moderna e interativa. Com foco inicial na gerência de processos, o programa suporta melhorias constantes graças à sua estrutura modular, viabilizando a extensão de funcionalidades por meio de novas classes, respeitando as interfaces do sistema desenvolvido. As principais contribuições deste trabalho são: (i) a especificação de uma arquitetura modular que separa a lógica de simulação da interface visual; (ii) o desenvolvimento de uma ferramenta interativa que viabiliza a observação determinística da transição de estados de processos; e (iii) a disponibilização de um recurso educacional de código aberto preparado para extensão contínua por acadêmicos e pesquisadores.

2. Fundamentação Teórica

Sustentado nos princípios clássicos da gerência de processos em Sistemas Operacionais e na técnica de Simulação por Eventos Discretos (SED) [Robinson 2004, Banks et al. 2014], o simulador abstrai o funcionamento de um *kernel* para visualizar

[1]https://institucional.ufpel.edu.br/projetos/id/u3777

o comportamento de um sistema em execução, tratando os processos como entidades dinâmicas. Seu ciclo de vida é governado por uma máquina de estados finitos. Seguindo a literatura clássica [Silberschatz et al. 2015], adotou-se o modelo de cinco estados: Novo, Pronto, Executando, Bloqueado (Em Espera) e Finalizado. A inclusão do estado "Em Espera" é um diferencial pedagógico crítico, pois reflete os momentos em que a CPU é liberada enquanto um processo aguarda a conclusão de operações de Entrada e Saída (I/O). Isso permite a visualização e aplicação prática de algoritmos de escalonamento de CPU, que implementam as regras lógicas responsáveis por determinar a ordem e o tempo de permanência de cada tarefa no processador, englobando conceitos fundamentais como a preempção e a fatia de tempo (*quantum*) [Tanenbaum and Bos 2015]. O programa é determinístico, o mesmo cenário sempre emite o mesmo resultado. No contexto educacional, essa característica permite que o estudante configure o cenário para um experimento pensado para demonstrar algum aspecto específico do SO, facilitando a compreensão de mecanismos complexos que outrora seriam opacos.

3. Implementação e Estrutura Interna

A implementação baseia-se em uma arquitetura modular e extensível, fundamentada na linguagem Java versão 21 LTS e o *framework* JavaFX para a interface gráfica. O sistema adota o padrão de projeto MVVM (*Model-View-ViewModel*) [Fuksa et al. 2025], para assegurar a separação entre a lógica de simulação e a lógica que governa a representação visual do sistema. Aliado ao uso das propriedades observáveis do JavaFX, o núcleo de simulação (*Model*) opera de forma completamente desacoplada da interface (*View*). Isso permite que novos módulos ou algoritmos de escalonamento sejam acoplados ao núcleo sem o risco de comprometer a estabilidade da visualização.

O núcleo do PampaSim opera via simulação de eventos discretos, sincronizado por um relógio global que coordena o avanço dos *ticks* (unidade de tempo) no cerne da simulação de gerência de processos. O sistema adota um roteador central que orquestra a comunicação por eventos das três entidades fundamentais que compõem o modelo do sistema operacional: o *Process Manager*, responsável por gerenciar o ciclo de vida e a transição de estados dos processos; o *Scheduler*, que encapsula as políticas e heurísticas de escalonamento; e o *Processor*, que simula a execução física das tarefas no hardware.

Essa arquitetura, aliada ao ciclo de etapas estritas do relógio, garante um comportamento determinístico e rastreável. Consequentemente, o simulador consolida-se como um ambiente ideal para o ensino prático, onde experimentos podem ser pausados, inspecionados e reproduzidos com exatidão.

4. Funcionalidades

A base do programa é a simulação dos mecanismos de escalonamento de processos. É permitido incluir módulos agregados para aumentar a profundidade e complexidade das simulações, como por exemplo: módulos de memória e armazenamento secundário. Os recursos estão divididos em quatro grandes pilares:

- Gerência e Escalonamento: parametrização de políticas e carga de trabalho.
- Controle Dinâmico de Execução: acelerar, desacelerar, avançar passo a passo ou pausar a simulação para análise de instantes e estados específicos.

- Análise de Desempenho: estatísticas globais e por processo em tempo real tais como *throughput*, ócio de CPU, tempo de retorno, tempo de espera e gráfico Gantt.
- Gestão de Cenários: salvar e carregar configurações completas de experimentos contendo todos os cadastros que o usuário é capaz de realizar.

Embora a arquitetura baseada em eventos suporte a instanciação de volumes maiores de processos, a interface visual é otimizada didaticamente para cenários controlados, privilegiando a clareza analítica das trocas de contexto sobre simulações massivas.

5. Interface Gráfica

A interface gráfica atua como um tradutor visual das complexas estruturas de dados internas de um sistema operacional. O *dashboard* principal (Figura 1) foi estruturado para representar a máquina de cinco estados do ciclo de vida dos processos, mapeando a área de visualização em filas de escalonamento dinâmicas: Criados, Prontos, Em Espera e Finalizados; e o estado de alocação da CPU. A barra lateral concentra as funções de controle do fluxo da simulação. A barra horizontal no topo da janela expõe botões para cadastro de processos, gerência de cenários, configurações da simulação, e leitura de estatísticas. A visualização em tempo real é central, onde animações e atualizações constantes refletem fielmente as transições dos processos entre as filas de espera e o núcleo de processamento, consolidando a experiência do usuário através da observação direta do comportamento do sistema.

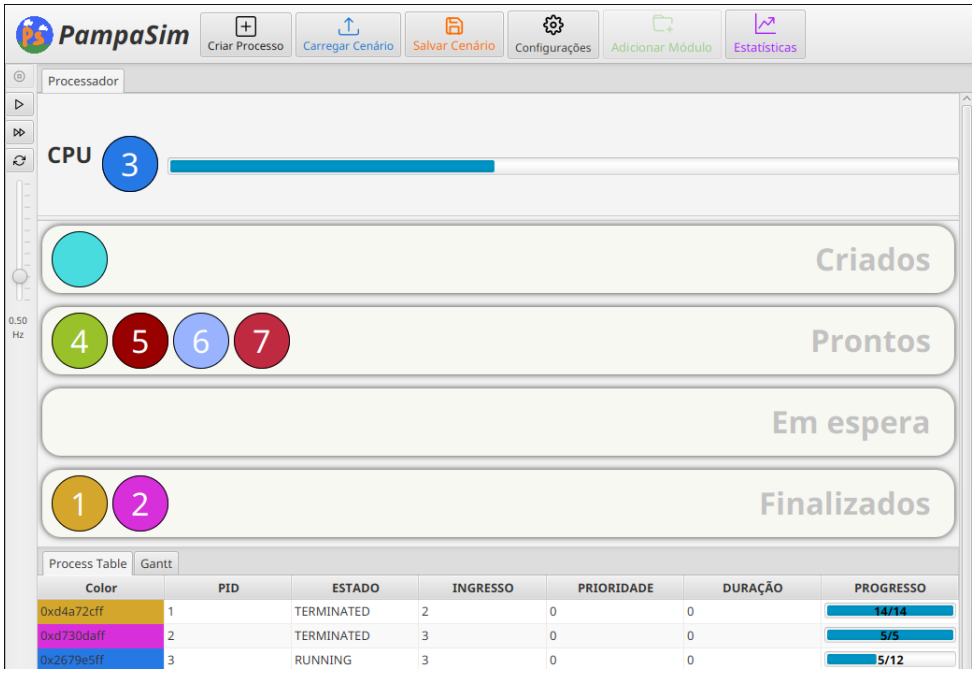

Figura 1. Janela principal do simulador PampaSim com um cenário em execução.

6. Relevância Acadêmica e Aplicações

A maior contribuição do simulador é a capacidade de transformar a teoria complexa dos sistemas operacionais em uma experiência visual e interativa. Seu ambiente permite que os estudantes explorem o funcionamento interno de gerência de recursos sem a necessidade de dominar as técnicas de implementação de um *kernel* real. A ferramenta alinha-se diretamente às conclusões de [Grotzer and Shane Tutwiler 2014]. Segundo os

autores, a compreensão de sistemas complexos é significativamente ampliada quando as relações causais são demonstradas de múltiplas maneiras e suportadas por ferramentas que ampliam a janela de atenção e percepção do aluno. Nesse contexto, as funcionalidades de controle de tempo e rastreabilidade viabilizam a aplicação de metodologias ativas, como o Aprendizado Baseado em Problemas (PBL), onde estudos de casos e narrativas de escalonamento podem ser validadas de forma empírica. Por ser de código aberto e modular, a ferramenta serve como plataforma de pesquisa universitária, permitindo que a comunidade acadêmica contribua com correções e novas funcionalidades, favorecendo a manutenibilidade contínua do simulador.

7. Conclusão

O PampaSim moderniza o ensino de Sistemas Operacionais através de sua arquitetura MVVM e interface visual animada, que esclarecem os estados dos processos. O determinismo do motor de simulação aliado ao sistema de cenários facilita o estudo dos conceitos de preempção e dos algoritmos de escalonamento, dando suporte à experimentação e a descoberta de relações causais. A modularidade da implementação permite que a ferramenta evolua para além da gerência de processos, tornando-a extensível. Como trabalho futuro, prevê-se implementar módulos de gerência de memória e armazenamento secundário, expandindo o ecossistema do simulador. O projeto se beneficia do modelo código aberto, que possibilita o seu aprimoramento pela comunidade de usuários, não somente o núcleo de desenvolvedores. O PampaSim configura um recurso valioso capaz de transformar a teoria abstrata em uma experiência de aprendizado tangível e interativa, que fomenta a formação de desenvolvedores com base sólida na gerência de recursos, preparando-os para os desafios da engenharia de sistemas concorrentes e de alto desempenho, pilares do avanço da computação moderna.

Referências

Banks, J., Carson II, J. S., Nelson, B. L., and Nicol, D. M. (2014). *Discrete-Event System Simulation*. Pearson, 5 edition.

Christopher, W. A., Procter, S. J., and Anderson, T. E. (1993). The nachos instructional operating system. In *Proceedings of the USENIX Winter 1993 Conference*, page 481.

Fuksa, M., Speth, S., and Becker, S. (2025). *MVVM Revisited: Exploring Design Variants of the Model-View-ViewModel Pattern*. Springer Nature Switzerland.

Grotzer, T. A. and Shane Tutwiler, M. (2014). Simplifying causal complexity: How interactions between modes of causal induction and information availability lead to heuristic-driven reasoning. *Mind, Brain, and Education*, 8(3):97–114.

Maia, L. P. (2001). Sosim: Simulador para o ensino de sistemas operacionais. Dissertação de mestrado, Universidade Federal do Rio de Janeiro (UFRJ).

Robinson, S. (2004). *Simulation The practice of model development and use*. Addison-Wesley, 15th edition.

Silberschatz, A., Galvin, P. B., and Gagne, G. (2015). *Fundamentos de Sistemas Operacionais*. LTC, 9 edition.

Tanenbaum, A. S. and Bos, H. (2015). *Sistemas Operacionais Modernos*. Pearson Education do Brasil, 4 edition.