
Simulador de Rotinas de Memória de Sistemas Operacionais

Diego Henrique Correia Resende



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

Monte Carmelo - MG
2024

Diego Henrique Correia Resende

**Simulador de Rotinas de Memória de 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. Thiago Pirola Ribeiro

Monte Carmelo - MG

2024

Ofereço este trabalho primeiramente a Deus, pela força e sabedoria concedidas ao longo desta jornada. Dedico este esforço aos estudantes e professores da área de tecnologia, que compartilham a paixão por tecnologia e inovação. Que esta ferramenta de simulação de operações de memória seja uma ferramenta útil para o aprendizado dos alunos e inspiradora para novos simuladores. Aos meus familiares, amigos e ao professor que me orientou, que caminharam ao meu lado durante todo o percurso, ofereço este trabalho com gratidão e reconhecimento.

Agradecimentos

Agradeço a Deus pela força e inspiração ao longo desta jornada. Expresso minha gratidão ao professor que me orientou, cuja orientação, paciência e comprometimento foram essenciais para a realização deste trabalho. Aos professores, pelo conhecimento compartilhado e pelo exemplo de dedicação, que contribuíram imensamente para minha formação. Agradeço profundamente meus familiares, pelo suporte incondicional, e aos amigos, por estarem ao meu lado nos momentos desafiadores. A cada um de vocês, deixo aqui meu sincero reconhecimento e agradecimento.

“Somos o que repetidamente fazemos. A excelência, portanto, não é um ato, mas um hábito.”
(Aristóteles)

Resumo

Este projeto apresenta o aprimoramento de uma plataforma interativa voltada ao estudo da administração da memória em sistemas operacionais. O objetivo é apoiar estudantes e professores na compreensão prática e teórica da área, utilizando uma plataforma online desenvolvida com TypeScript e Angular. A realização da implementação de algoritmos proporciona uma vivência prática e visual, facilitando a assimilação dos conceitos. Os resultados indicam que a ferramenta promove interatividade e retenção do conteúdo, sendo eficiente no aprendizado e aprimoramento das técnicas de gestão de memória.

Palavras-chave: Gestão de Memória, Simulador, Alocação, Processos, Sistemas Operacionais.

Lista de ilustrações

Figura 1 – Painel do simulador SOsim.	20
Figura 2 – Painel principal do SimulateOS.	20
Figura 3 – SimulateOS reestruturado por Sousa (2024).	21
Figura 4 – Interface criada no figma.	23
Figura 5 – Interface Implementada para Alocação Contígua.	24
Figura 6 – Interface Implementada para Alocação com Paginação.	25
Figura 7 – Cabeçalho que possui os dados e campos de ações da memória.	25
Figura 8 – Módulo para selecionar o método de alocação de memória desejado.	26
Figura 9 – Relação completa de métodos de alocação disponíveis no sistema.	26
Figura 10 – Módulo para inicialização de processos com alocação contígua.	27
Figura 11 – Módulo para inicialização de processos com alocação com paginação.	27
Figura 12 – A janela demarca, com destaque em vermelho, as alterações para inicialização de processos utilizando alocação com paginação.	28
Figura 13 – A janela demarca, com destaque em vermelho, as alterações para inicialização de processos utilizando alocação contígua.	28
Figura 14 – Interface inicial com os blocos livres.	29
Figura 15 – Interface inicial com processos criados.	30
Figura 16 – Interface inicial com processos encerrados.	30
Figura 17 – Interface inicial com os blocos livres e a tabela de páginas vazia.	33
Figura 18 – Interface de execução das técnicas de substituição de páginas.	33
Figura 19 – Interface após a remoção de processos, exibindo os blocos e páginas liberadas.	34
Figura 20 – Painel de modificação de processos exibindo detalhes e controles.	34

Lista de siglas

BestFit Melhor Escolha - *Best Fit*

CPU Unidade Central de Processamento - *Central Processing Unit*

FIFO Primeiro a Entrar, Primeiro a Sair - *First In, First Out*

FirstFit Primeira Escolha - *First Fit*

LRU Menos Recentemente Utilizado - *Least Recently Used*

NRU Não Recentemente Utilizado - *Not Recently Used*

PID Identificador de Processo - *Process Identifier*

UFU Universidade Federal de Uberlândia

WorstFit Pior Escolha - *Worst Fit*

Sumário

1	INTRODUÇÃO	10
1.1	Objetivos	10
1.2	Justificativa	11
1.3	Hipótese	11
1.4	Metodologia	12
1.5	Divisão da Monografia	12
2	FUNDAMENTAÇÃO TEÓRICA	13
2.1	Ensino e Sistemas Operacionais	13
2.2	Ferramenta de Simulação	14
2.3	Elementos de um Ambiente Operacional	14
2.3.1	Processo	15
2.3.2	Alocação de memória	15
2.4	Angular	17
2.5	TypeScript	18
2.6	Angular Material	18
2.7	Figma	18
2.8	Trabalhos Relacionados	18
2.8.1	S^2O	19
2.8.2	SWSO	19
2.8.3	SOsim	19
2.8.4	SimulateOS	19
3	EXPERIMENTOS E ANÁLISE DOS RESULTADOS	22
3.1	Simulador	22
3.2	Memória	22
3.3	Planejamento e Estruturação do Simulador	23
3.3.1	Planejamento das Funcionalidades	23

3.3.2	Prototipagem no Figma	23
3.3.3	Implementação da Interface	23
3.4	Processo	25
3.4.1	Inicialização do Processo	26
3.4.2	Alterações	26
3.5	Desenvolvimento das abordagens de alocação contígua.	27
3.6	Desenvolvimento das abordagens de Log e Sincronização dos Sistemas	31
3.7	Desenvolvimento das abordagens de Paginação	31
4	CONCLUSÃO	35
4.1	Principais Contribuições	35
4.2	Trabalhos Futuros	36
4.3	Contribuições em Produção Bibliográfica	37
REFERÊNCIAS		38

Introdução

Analizando o rendimento escolar de alunos no setor de computação, especialmente em matérias que apresentam desafios técnicos, como Sistemas Operacionais, observa-se uma elevada proporção de reprovações (FREITAS, 2023). Existe uma variedade abrangente de recursos educativos, como livros, estudos acadêmicos, artigos e guias, disponíveis tanto em formato impresso quanto digital, que visam aprimorar a compreensão dos tópicos abordados. No entanto, como esses materiais tendem a ser majoritariamente teóricos, podem dificultar a compreensão e a motivação dos alunos, pois o conteúdo pode soar excessivamente abstrato (FREITAS, 2023).

Nesse cenário, é recomendável o uso de simuladores como uma estratégia educacional para melhorar o ensino no estudo sobre Sistemas Operacionais. Essa abordagem tem o potencial de aumentar o interesse dos estudantes pelos temas abordados, tornando a compreensão do conteúdo mais acessível através de uma vivência de aprendizado prática e interativa. Iniciativas anteriores, como os projetos dos autores Carvalho et al. (2006) e Maia (2001), já tiveram como objetivo tornar mais fácil a compreensão das rotinas de Sistemas Operacionais.

1.1 Objetivos

Desenvolver e aprimorar uma plataforma online interativa para o aprendizado teórico e prático sobre gerência de memória e técnicas de alocação. Esse objetivo principal se desdobra nos seguintes objetivos específicos:

- Proporcionar uma interface intuitiva que auxilie os usuários no entendimento de princípios fundamentais de gestão de memória.
- Representar, através de simulações visuais e em tempo imediato, as operações de alocação, liberação e controle da memória.

- Demonstrar as diversas abordagens de organização de processos dentro da memória, como alocação contígua e paginação, destacando suas diferenças e aplicações.
- Tornar o entendimento comparativo das diversas formas de gerenciar memória mais acessível por através de opções interativas.
- Permitir a interrupção e reinício das simulações, incentivando a exploração de cenários variados e ajustes de configuração.

1.2 Justificativa

O estudo dos fundamentos relacionados, possui um nível de complexidade no qual pode dificultar a compreensão dos alunos ao terem o primeiro contato com esses conceitos. Isso pode gerar obstáculos no aprendizado, resultando em falta de interesse e uma taxa de reprovação mais elevada (FREITAS, 2023). Diversos programas, como os criados pelos autores TROPIX (2008), Kioki e Santiago (2008), Reis, Júnior e Costa (2009), Carvalho et al. (2006), Rocha et al. (2004), Ribeiro, Lima e Lobo (2014), Maia (2001). No entanto, constatou-se que esses programas tendem a focar em temas isolados, apresentando-os de forma muito básica ou complexa, sem oferecer uma estratégia que una diferentes níveis de dificuldade em uma única ferramenta.

Diante dessa análise, propôs-se o aprimoramento do software desenvolvido por Sousa (2024), que originalmente aborda apenas a alocação de processos. O aprimoramento consiste em estender suas capacidades para incluir a administração dos recursos de memória nos softwares de base, além de realizar a integração entre os dois aspectos: alocação de processos no processador e gestão de memória. Essa integração busca superar as limitações do software original, oferecendo uma solução abrangente que combina conceitos essenciais e avançados, com um enfoque especial nas operações de memória também na atividade de processos, facilitando o aprendizado dos alunos sobre essa área fundamental da disciplina.

1.3 Hipótese

O aprimoramento de um ambiente de simulação online, desenvolvido usando Angular, que integre a alocação de processos também gestão de memória em sistemas operacionais, considerando foco em simplicidade, contribuirá para superar as limitações dos simuladores existentes como os criados pelos autores Carvalho et al. (2006) e Maia (2001). Essa solução possibilitará o aprendizado eficiente através de uma interface visualmente atrativa, acessível em qualquer dispositivo com acesso à internet, e que permita a exploração de conceitos fundamentais e avançados de maneira integrada.

1.4 Metodologia

O objetivo deste trabalho foi o aprimoramento de um ambiente de simulação online existente, que originalmente abordava apenas a alocação de processos, feito por Sousa (2024). A proposta consistiu em estender os recursos do sistema para incluir a gestão de memória, permitindo uma estratégia mais completa dos princípios básicos de Sistemas Operacionais.

Optou-se pelo aprimoramento, em vez da implantação de um novo simulador, devido à estrutura já consolidada da ferramenta original, que permitiu agregar novas funcionalidades com maior eficiência.

O aprimoramento foi conduzido em reuniões quinzenais, que incluíram etapas de planejamento, execução e revisão. Cada ciclo teve como foco a definição de novas funcionalidades, o ajuste das existentes e a validação contínua das mudanças implementadas. Essas etapas foram acompanhadas pelo orientador, garantindo alinhamento com os propósitos do projeto e permitindo refinamentos pontuais.

Além disso, priorizou-se a acessibilidade da ferramenta, mantendo-a como um simulador online, acessível de qualquer dispositivo com internet. Essa abordagem visa atender docentes e discentes, eliminando a exigência de instalações locais e promovendo maior praticidade no aproveitamento da ferramenta em cenários acadêmicos.

1.5 Divisão da Monografia

A monografia está organizada em quatro capítulos, além desta introdução. No Capítulo 2, são expostos os fundamentos teóricos que sustentam o estudo, tratando de conceitos essenciais de ambientes computacionais, ferramentas de simulação, e os elementos técnicos que suportam o aprimoramento do simulador. No Capítulo 3, são detalhadas as etapas de planejamento e estruturação do simulador, incluindo a definição das funcionalidades, a prototipagem das interfaces e a implantação de métodos para alocação de memória. O Capítulo 4 aborda as conclusões alcançadas e as contribuições do trabalho, destacando o impacto dos recursos implementados na aquisição de conhecimento em ambientes computacionais. Além disso, são apresentadas sugestões para trabalhos futuros que possam expandir as funcionalidades desenvolvidas.

Fundamentação Teórica

Este capítulo apresenta os fundamentos teóricos essenciais para a compreensão do projeto proposto. Inicialmente, é analisado o contexto educacional no ensino superior, com foco na compreensão de sistemas operacionais. Destaca-se a importância da simulação como um recurso didático que facilita o entendimento prático de conceitos abstratos, como aqueles ligados à administração da memória.

Em seguida, são explorados os principais conceitos teóricos que embasam o projeto, com destaque para a administração da memória em ambientes computacionais. Tópicos como alocação de memória contígua e alocação com paginação são apresentados como elementos centrais para a funcionalidade do simulador, auxiliando os estudantes a entenderem as diferenças dessas abordagens.

A escolha do Angular como tecnologia principal para o aprimoramento do simulador é justificada pela sua capacidade de criar interfaces dinâmicas e acessíveis, características essenciais para um sistema que busca atender tanto os docentes quanto os discentes de forma intuitiva e prática.

Por fim, este capítulo revisa estudos e projetos relacionados que compartilham objetivos semelhantes. Identificam-se suas contribuições e limitações, estabelecendo de maneira semelhante a qual este trabalho se posiciona no campo, ao propor um simulador que integra funcionalidades avançadas de administração de memória com acessibilidade e para o auxílio pedagógico.

2.1 Ensino e Sistemas Operacionais

Voltado para discentes de graduação em Tecnologia de tecnologia e áreas relacionadas, é fundamental ter uma compreensão abrangente dos temas tratados no campo de sistemas operacionais. Esse conhecimento é essencial para ajudar no processo decisório durante a aplicação de software, também favorece um entendimento mais aprofundado dos elementos físicos de uma plataforma computacional (REIS; COSTA, 2009).

O aprendizado em sistemas operacionais pode ser desafiadora para os discentes devido à complexidade dos conteúdos apresentados, como gestão de memória, processos, threads, processadores, arquiteturas, entre outros. Essa ampla gama de temas complexos pode tornar a compreensão da disciplina um desafio para os discentes (REIS; COSTA, 2009). No âmbito do ensino relacionado a sistemas operacionais, entende-se que a metodologia predominante adotada pelas instituições é a exposição didática, no qual o professor apresenta os conteúdos de forma passiva aos estudantes. Embora essa metodologia possa ser eficaz em certas situações, os alunos encontram dificuldades para compreender as informações, especialmente em disciplinas como esse campo, que demandam a assimilação de conceitos especializados e especializados. Adaptar esses conteúdos para o formato expositivo pode levar à exclusão de partes essenciais do tema, como exemplos práticos, abordagens específicas de algoritmos ou explicações mais detalhadas de conceitos fundamentais, o que compromete a compreensão integral e a aplicação dos temas abordados (REIS; COSTA, 2009).

2.2 Ferramenta de Simulação

Ferramenta de simulação é um dispositivo ou programa que é capaz de imitar as características operacionais em um ambiente computacional. Ele cria condições e efeitos que, embora não estejam acontecendo de fato, parecem reais. A finalidade desta ferramenta é emular tanto as experiências físicas (tais como velocidade e aceleração, assim como a visão do ambiente) quanto as operações de equipamentos do sistema que está sendo simulado. Além disso, pode replicar o resultado de um protótipo funcional, dispensando o uso de recursos materiais, empregar maquinário, economizando tempo e força de trabalho (ODEN et al., 2006).

Em relação aos maiores benefícios do uso de simulações na educação é facilitar o acesso ao conteúdo, visto que a ferramenta reúne os elementos essenciais do assunto abordado (SILVA; COSTA, 2020).

2.3 Elementos de um Ambiente Operacional

Entre os diversos elementos que compõem um Ambiente Operacional, este estudo focará principalmente na gestão da memória, abordando aspectos como: alocação de memória contígua, paginação. Esses conceitos são fundamentais para compreender como a memória é gerenciada em ambientes operacionais.

2.3.1 Processo

Cada processo em execução possui um espaço de endereço próprio, que é composto por diversos locais de memória designados exclusivamente a ele. Esse espaço inclui o código executável do programa e os dados associados.

O gerenciamento desse espaço de memória é fundamental para garantir que diferentes processos não interfiram entre si, promovendo a segurança e a estabilidade do sistema. Ademais, o espaço de endereço define os limites dentro dos quais o processo tem a capacidade de realizar operações de leitura e escrita, sendo gerenciado pela estrutura do sistema através de técnicas como alocação contígua e paginação (TANENBAUM, 2010).

2.3.2 Alocação de memória

A alocação de memória é um aspecto crítico na gestão de sistemas operacionais, pois afeta diretamente a eficiência e a capacidade do sistema. As estratégias de alocação buscam organizar o espaço de memória de maneira eficaz, mas frequentemente enfrentam problemas de fragmentação.

A fragmentação externa ocorre quando pequenos blocos de memória ficam dispersos e inutilizáveis, enquanto a fragmentação interna surge quando a memória alocada não é completamente utilizada pelos processos. Esses desafios são especialmente críticos em sistemas que operam continuamente por longos períodos, exigindo técnicas de gerenciamento como paginação (TANENBAUM, 2010).

Os métodos tradicionais de alocação, que utilizam listas contíguas para gerenciar espaços livres, apresentam limitações em sistemas modernos. Essa abordagem pode resultar em tempos de alocação variáveis, dado que o tamanho da lista influencia o tempo necessário para encontrar um espaço adequado. Para superar essas limitações, foram implementados neste trabalho dois tipos principais de algoritmos baseados em Tanenbaum (2010):

- ❑ Algoritmos de Alocação Contígua
- ❑ Algoritmos de Alocação com Paginação

2.3.2.1 Algoritmos de Alocação Contígua

A alocação contígua é uma das formas clássicas de gerenciar o uso da gestão de recursos em plataformas operacionais, utilizando listas encadeadas para monitorar os espaços livres e os segmentos alocados. Cada segmento pode conter um processo podendo ser um espaço vazio entre processos. A lista é organizada por endereço, facilitando a localização de áreas disponíveis para novos processos ou para aqueles que precisam ser carregados na memória.

Essa organização permite ao sistema operacional aplicar diferentes estratégias de alocação para decidir onde posicionar um novo processo. O sistema verifica quanta memória

precisa ser alocada e escolhe um dos blocos contíguos disponíveis para satisfazer a solicitação. A alocação contígua garante ao processo uma área contínua de memória, o que minimiza a complexidade de acesso, mas pode levar à fragmentação externa, notadamente em sistemas com muitas operações de alocação e liberação.

- ❑ *FirstFit*: Nesta estratégia, o gerenciador de memória examina a lista de segmentos até encontrar um espaço livre que seja grande o suficiente. O espaço livre é então dividido em duas partes, uma atribuída ao processo e outra para a memória não utilizada, exceto no caso estatisticamente improvável de um encaixe exato. FirstFit é um algoritmo rápido, pois ele procura fazer a menor busca possível.
- ❑ *BestFit*: Nesta estratégia, o BestFit faz uma busca em toda a lista, do início ao fim, e escolhe o menor espaço livre que seja adequado. Em vez de escolher um espaço livre grande demais que talvez seja necessário mais tarde, o BestFit tenta encontrar um que seja de um tamanho próximo do tamanho real necessário, para atender da melhor maneira possível a solicitação com os segmentos disponíveis.
- ❑ *WorstFit*: Nesta estratégia, é escolhido o maior espaço livre na memória, de maneira que o novo segmento livre gerado seja grande o bastante para ser útil. No entanto, simulações demonstraram que o worstfit não é uma boa ideia.

2.3.2.2 Técnica de Paginação

Dentro do escopo dos sistemas operacionais, técnica de paginação ocorre quando o sistema precisa liberar espaço na memória para carregar novas páginas durante uma falta de página. Nesse processo, a plataforma do sistema deve escolher qual página deverá ser retirada da memória e essa escolha impacta diretamente a eficiência do sistema. Se a página removida foi alterada enquanto estava na memória, ela precisa ser gravada no disco para manter a versão atualizada. Caso contrário, se a página não foi modificada, como no caso de uma página de código, ela é simplesmente substituída pela nova, sem necessidade de reescrita.

Escolher uma página que não é intensamente usada para remoção pode melhorar o desempenho, pois evita o custo de trazê-la de volta logo em seguida. Por isso, os algoritmos para alocação de páginas procuram otimizar essa seleção para minimizar as trocas desnecessárias e os custos adicionais. A alocação de páginas também é um problema em outras áreas, como servidores web, onde dados frequentemente acessados precisam ser mantidos para rápido acesso, e decisões de alocação são necessárias quando o espaço está cheio.

Existem várias abordagens para esses algoritmos, cada uma com características e níveis de eficiência específicos. Considerando ambientes de memória virtual, essa alocação pode ocorrer limitando cada processo a um conjunto fixo de páginas ou possibilitando a reutilização de páginas por diferentes processos.

- ❑ *LRU*: Uma boa aproximação para o algoritmo ótimo é baseada na observação de que as páginas que foram usadas intensamente nas últimas instruções provavelmente o serão em seguida de novo. De maneira contrária, páginas nas quais não foram usadas recentemente provavelmente seguirão sem ser utilizadas por um longo tempo. Essa ideia sugere um algoritmo realizável: quando ocorre uma falta de página, jogue fora aquela que não tem sido usada há mais tempo. Essa estratégia é chamada de paginação Menos Recentemente Utilizado (LRU).
- ❑ *NRU*: O algoritmo Não Recentemente Utilizado (NRU) remove uma página ao acaso de sua classe de ordem mais baixa que não esteja vazia. Implícito nesse algoritmo é melhor remover uma página modificada, mas não referenciada, a pelo menos um tique do relógio (em geral em torno de 20 milissegundos) do que uma página não modificada que está sendo intensamente usada. A principal atração do NRU é que ele é fácil de compreender, moderadamente eficiente de implementar e proporciona um desempenho que, embora não ótimo, pode ser adequado.
- ❑ *FIFO*: O Primeiro a Entrar, Primeiro a Sair (FIFO) é simples e de baixo custo. Ele opera como um supermercado que possui espaço limitado para exibir produtos: quando um item novo e popular precisa ser estocado, o supermercado precisa remover o produto que está nas prateleiras há mais tempo, presumindo que a demanda por ele diminuiu. No âmbito dos sistemas operacionais, o FIFO mantém um agrupamento de páginas na memória organizada pela ordem de chegada. A página mais antiga, que está na frente dessa lista, é removida para dar espaço à nova página. Contudo, no contexto específico deste estudo, o FIFO será ajustado para que as páginas permaneçam na memória até que a operação do processo ao qual pertencem termine. Essa modificação evita que páginas sejam removidas enquanto o processo permanece em atividade, garantindo que todos os detalhes necessários permaneçam carregadas até o término do processo. Esse ajuste proporciona uma gestão de memória mais estável.

2.4 Angular

Angular¹ é um framework para desenvolvimento que utiliza TypeScript como base. Esta plataforma incorpora um esquema de componentes que facilita a implementação de softwares para a web expansíveis, ademais uma quantidade de bibliotecas integradas para suportar a variedade de funcionalidades. Também disponibiliza um conjunto completo de ferramentas destinadas a desenvolvedores para o auxílio em diversas etapas, desde o desenvolvimento até a construção. Angular é capaz de acompanhar a jornada de implementação de uma aplicação, da fase de prototipagem até o lançamento em escala global,

¹ Pode ser obtido em: <<https://angular.dev/>>

oferecendo uma eficiência escalonável, que estão por trás dos mais importantes aplicativos da empresa Google (ANGULAR, 2021).

2.5 TypeScript

TypeScript² trata-se de uma tecnologia de programação de código aberto e sem custo, criada e sustentada pela Microsoft, com seu lançamento inicial em 2012. Encontra-se agora na versão 4.9. Esta linguagem funciona como um superconjunto de JavaScript, adicionando sintaxe adicional à base do JavaScript, o que possibilita a atribuição de tipos, enriquecendo assim a linguagem com funcionalidades especificação estática de tipos (TYPESCRIPT, 2022).

2.6 Angular Material

Angular Material³ oferece uma variedade de componentes destinado à construção de interfaces de usuário projetado para integrar em projetos desenvolvidos com Angular, visando agilizar a criação de interfaces atrativas e robustas. Esta biblioteca disponibiliza uma variedade de componentes reutilizáveis, incluindo cartões, campos de entrada, tabelas, seletores de data, entre outros, facilitando significativamente a abordagem de desenvolvimento (MENSAH, 2022).

2.7 Figma

O Figma⁴ trata-se de um espaço para design colaborativa baseada na web, que permite a criação e edição de gráficos vetoriais e protótipos. Fundada em 2012 por Dylan Field e Evan Wallace, a empresa lançou seu produto ao público em 2016 (FIGMA, 2016).

2.8 Trabalhos Relacionados

Hoje em dia, no contexto educacional, os simuladores emergem como ferramentas valiosas para reforçar o aprendizado dos conteúdos abordados em aulas teóricas. Observa-se que os simuladores voltados para o estudo de plataformas operacionais dividem-se em duas categorias principais: os Simuladores Abrangentes, no qual englobam diversos tópicos relacionados a Sistemas Operacionais, e os Simuladores Específicos, que se concentram em explorar um número limitado de tópicos de forma mais detalhada. Adicionalmente, existem também Sistemas Operacionais desenvolvidos especificamente para propósitos

² Pode ser obtido em: <<https://www.typescriptlang.org/>>

³ Pode ser obtido em: <<https://material.angular.io/>>

⁴ Pode acessar em <<https://figma.com>>

educacionais, os quais permitem que os alunos ponham em prática a criação de seus próprios fluxos de trabalho.

2.8.1 S^2O

O S^2O é uma ferramenta para simulação feita pelo Carvalho et al. (2006), concebido para replicar as operações de uma plataforma operacional, focando no controle da memória, no escalonamento da CPU, abrangendo a leitura e escrita no disco. Esse sistema visa ampliar o aprendizado teórico de Sistemas Operacionais, diferenciando-se por sua habilidade em apresentar visualmente a gestão de processos, incluindo a aplicação prática de diversas estratégias de escalonamento (CARVALHO et al., 2006).

2.8.2 SWSO

O SWSO foi criado para dar suporte ao processo de ensino e aprendizagem no estudo de sistemas operacionais. Seu principal propósito é oferecer uma simulação interativa dos principais aspectos de uma plataforma operacional, abrangendo a administração de armazenamento, memória e processamento. Adicionalmente, este software é acessível a indivíduos com conhecimentos básicos em Java, permitindo-lhes baixar e ajustar o código do mesmo a fim de incluir novos algoritmos de maneira fácil.

2.8.3 SOsim

O SOsim, classificado como um simulador abrangente, foi criado com a finalidade de tornar mais fácil o aprendizado em materiais de ensino em Sistemas Operacionais. Na Figura 1 são ilustradas algumas interfaces do simulador, evidenciando sua abordagem visual na educação. Este simulador abrange conceitos essenciais de Sistemas Operacionais, incluindo a administração de armazenamento virtual, a administração de processos, controle do processador e uma vivência educacional fundamentada na simulação de conceitos fundamentais. No entanto, por possuir um escopo abrangente, ele oferece apenas uma visão geral dos tópicos, sem adentrar profundamente em cada aspecto, limitando assim as interações dos alunos a algoritmos ou métodos específicos predefinidos pelo software. Criado por Maia (2001), foi programado em Pascal orientado a objetos. Apesar de seu apelo visual, seu uso fica restrito ao Windows devido à dependência de bibliotecas específicas, o que representa uma limitação para a acessibilidade dos discentes atualmente.

2.8.4 SimulateOS

O SimulateOS, especialmente na parte de gestão de processos, foi dado início por Freitas (2023) com o propósito de simplificar o aprendizado dos discentes no estudo de

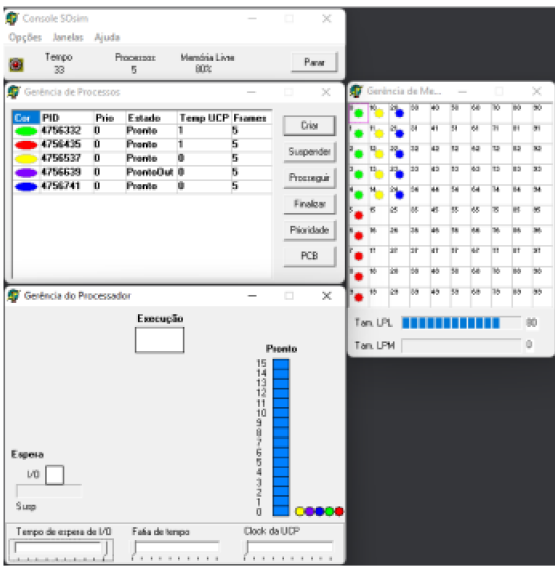


Figura 1 – Painel do simulador SOsim.

Fonte: Adaptado de Freitas (2023)

sistemas operacionais (Figura 2). As funcionalidades e definições do simulador são apresentadas dentro de uma interface visual moderna de fácil uso, possibilitando ao usuário criar e configurar processos, visualizar listas de processos organizadas de acordo com o status e ajustar as parâmetros do processador, tais como o clock de CPU, "o período de latência para operações de entrada e saída, assim como a fatia de tempo. A criação do simulador utilizou Angular e inclui bibliotecas que permitem o uso de componentes modernos e com manutenção simplificada.

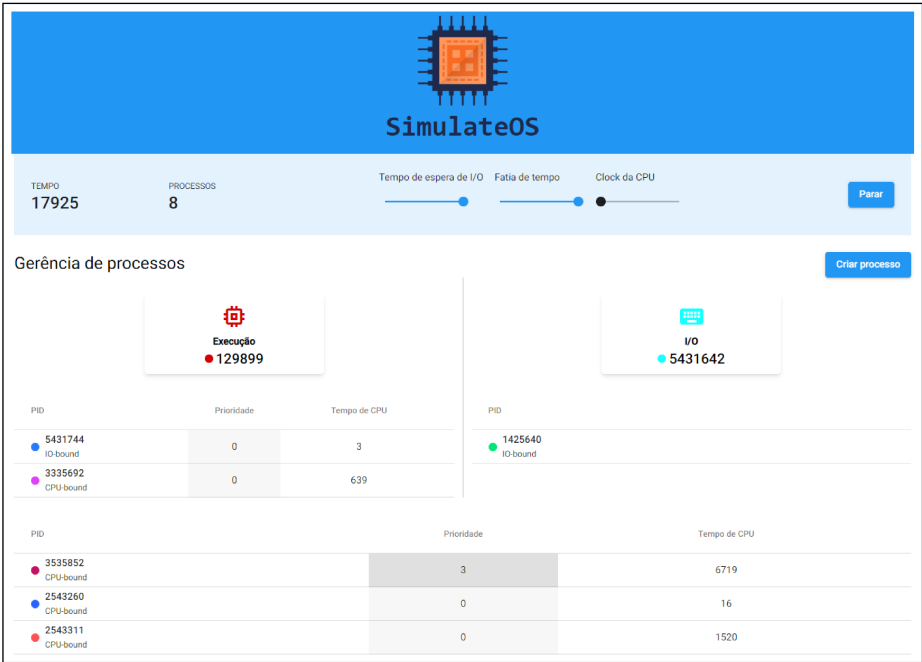


Figura 2 – Painel principal do SimulateOS.

Fonte: Figura Adaptada: Freitas (2023)

Após o desenvolvimento inicial realizado por Freitas (2023), o SimulateOS, focado na parte de gerenciamento de processos, foi aprimorado para tornar o compreensão dos discentes no contexto de sistemas computacionais ainda mais acessível. As funcionalidades e definições foram reorganizadas através de interface prática e simplificado para uso, permitindo ao criar, configurar processos e visualizar listas de processos.

O trabalho, realizado por Sousa (2024), trouxe avanços adicionais ao SimulateOS, incluindo a adoção de novos algoritmos voltado ao escalonamento de processos, o redesenho completo da estrutura da aplicação e a inclusão de gráficos que mostram a realização dos fluxos concluídos. A Figura 3 apresenta as evoluções feitas na tela principal:

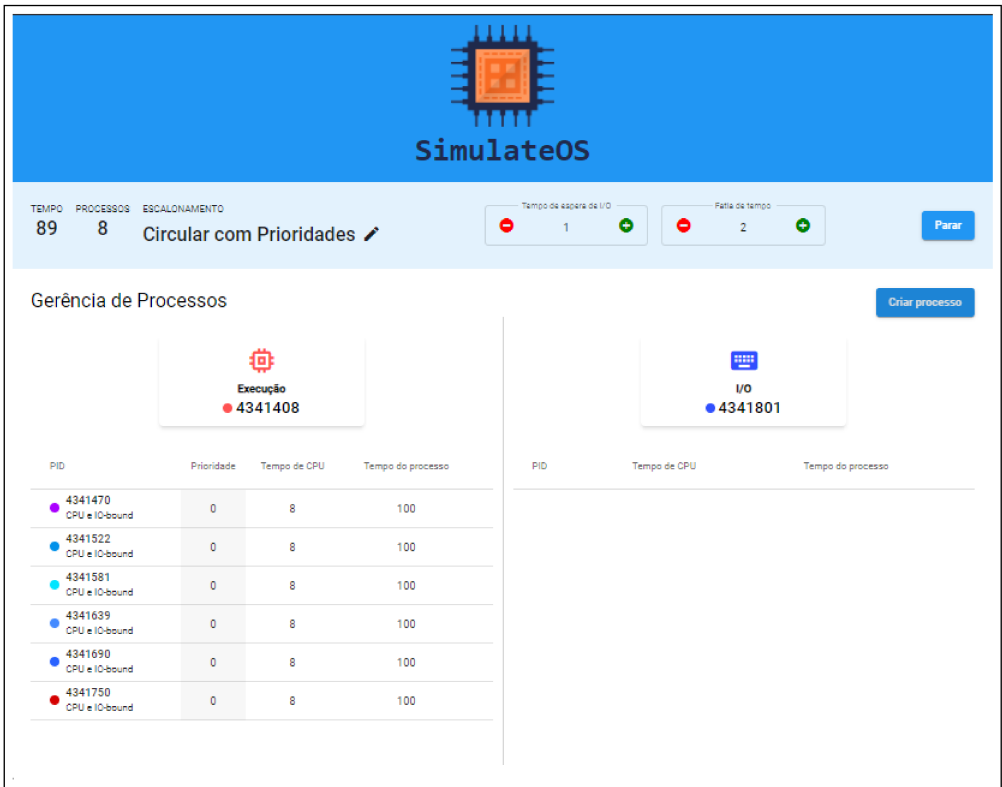


Figura 3 – SimulateOS reestruturado por Sousa (2024).

Fonte: Captura feita pelo autor.

Experimentos e Análise dos Resultados

No presente capítulo, será exibido os testes efetuados na aplicação, com a finalidade de evidenciar suas funcionalidades. Visando simplificar a compreensão, cada funcionalidade testada será acompanhada por uma descrição sucinta e representações visuais do sistema que ilustram a funcionalidade abordada.

É fundamental enfatizar que, por se tratar deste projeto é uma continuidade e aprimoramento de um trabalho anterior feito Freitas (2023) e Sousa (2024), diversos temas e ilustrações mostrarão similaridades. No entanto, o sistema feito anteriormente foi focado na gestão de processos, enquanto este se concentra no controle de memória.

3.1 Simulador

O simulador passou por melhorias visando reproduzir a gestão de memória dentro de ambiente computacional. Ele permite adicionar, editar e remover processos, bem como exibir uma tabela que mostra o índice inicial dos espaços livres e o número de blocos contíguos disponíveis. A ferramenta está disponível diretamente na página do SimulateOS¹ para uso. Além do mais, o código da aplicação foi disponibilizado em um repositório público, visando facilitar estudos futuros e a continuidade da pesquisa ².

3.2 Memória

Conforme descrito na Seção 3.1, a memória neste sistema corresponde a um módulo fictício encarregado de gerenciar os processos, possibilitando que se altere algumas configurações, como selecionar a categoria de alocação e parar o simulador, assim finalizando cada um dos processos alocados na memória.

¹ Endereço do simulador: <<https://www.facom.ufu.br/~thiago/simulateOS/>>

² Endereço para o repositório: <<https://github.com/DiegoHCR-code/TCC>>

3.3 Planejamento e Estruturação do Simulador

A concepção do SimulateOS foi planejada e estruturada em etapas organizadas, com a finalidade de assegurar clareza e eficiência tanto no design quanto na definição de suas funcionalidades. Uma das ferramentas utilizadas foi o Figma, que desempenhou um papel essencial na prototipagem de interfaces. Por meio dele, foram elaborados esboços e *mockups* detalhados das telas, permitindo simular e visualizar o usuário utilizando o sistema.

3.3.1 Planejamento das Funcionalidades

Inicialmente no Capítulo 2, foram listadas todas as capacidades essenciais para o SimulateOS, como simulação de processos, algoritmos de alocação contígua e técnicas de paginação. Esta fase incluiu a definição das necessidades e o estudo de quais métodos de gerência da memória seriam implementados.

3.3.2 Prototipagem no Figma

No Figma, foram criados protótipos de exemplo das telas do SimulateOS, abordando a tela inicial e a visualização de processos. Cada elemento da interface foi projetado para proporcionar uma experiência intuitiva ao usuário. O Figma foi utilizado para criar e refinar o design das telas antes de iniciar a implementação. Na Figura 4 está o protótipo realizado antes de iniciar a implementação da aplicação e das funcionalidades:

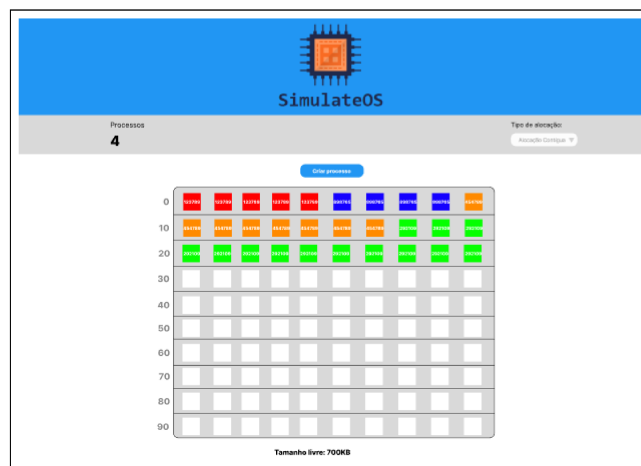


Figura 4 – Interface criada no figma.

Fonte: Captura feita pelo autor.

3.3.3 Implementação da Interface

Com base no protótipo feito no Figma, foi realizada a implementação da interface. A interface foi desenvolvida com foco em modularidade e reutilização de componentes,

permitindo que a interface respondesse às interações do usuário de maneira interativa. O protótipo foi utilizado como base como referência para a formação do sistema. Entre o processo, a interface foi continuamente aprimorada para torná-la mais atrativa e agradável, atendendo às necessidades dos usuários.

O sistema oferece duas interfaces distintas para a gestão da memória: uma para a alocação contígua e outra para a paginação. Na interface de alocação contígua (Figura 5), uma tabela é exibida com o título de Sequência, blocos contíguos que são os blocos livres contíguos e qual o bloco inicial que começa a sequência. Essa tela exibe o layout de memória dividido em 120 blocos, embora nem todos os blocos estejam visíveis na captura, devido ao limite de espaço para visualização.

Na segunda interface, dedicada ao gerenciamento por paginação (Figura 6), pode-se observar uma tabela de páginas facilitando a compreensão do processo de paginação pelo usuário. Essas duas telas foram criadas para proporcionar uma visão completa das diferentes abordagens, possibilitando ao usuário experimentar e analisar o comportamento do sistema nas diversas configurações.

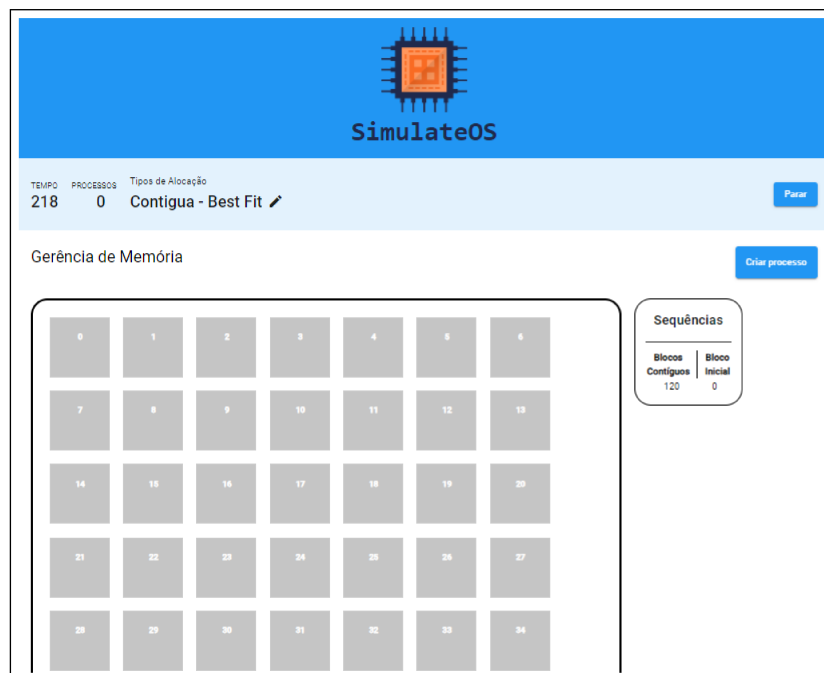


Figura 5 – Interface Implementada para Alocação Contígua.

Fonte: Captura feita pelo autor.

No cabeçalho, existe uma barra que contém informações sobre a memória (Figura 7):

- ❑ **Tempo:** O período desde que foi iniciada a simulação. Este tempo indica o número de intervalos de tempo simuladas que já transcorreram.
- ❑ **Processos:** Indicador de processos pendentes.

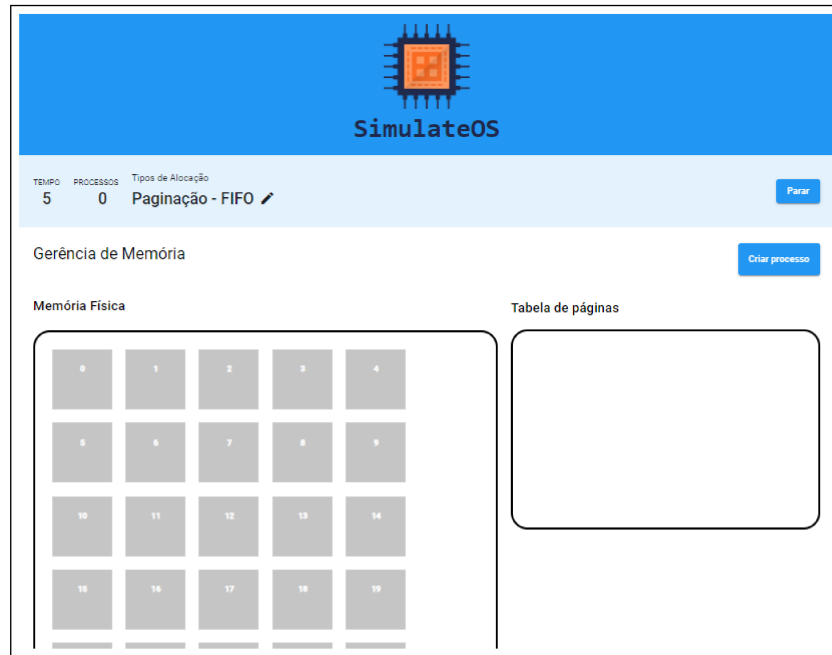


Figura 6 – Interface Implementada para Alocação com Paginação.

Fonte: Captura feita pelo autor.

- ❑ **Botão Parar:** Uma opção que, quando ativada, reinicializa o tempo e deleta os processos em execução.



Figura 7 – Cabeçalho que possui os dados e campos de ações da memória.

Fonte: Captura feita pelo autor.

Ao pressionar no símbolo (lápiz) a fim de selecionar a categoria de alocação, uma janela é exibida (Figura 8) informando que a mudança no tipo de alocação resultará na remoção e reinicialização de todos os processos, semelhante ao efeito do botão “Parar”. Essa janela inclui um menu de seleção incluindo todas as modalidades de alocação disponíveis na aplicação (Figura 9). Após escolher o novo tipo de alocação e confirmar clicando em “Ok”, o painel é fechado, o tipo de alocação é atualizado na barra referente ao processador e todo o andamento do aplicativo é reinicializado. Se o usuário optar por não prosseguir com a alteração, ou tenha selecionado a opção acidentalmente, ele pode clicar em “Cancelar”, o que encerrará a janela sem efetuar mudanças no sistema.

3.4 Processo

O processo, elemento central de nossa aplicação, representa um componente que simboliza uma operação em andamento dentro deste “ambiente computacional” simulado.

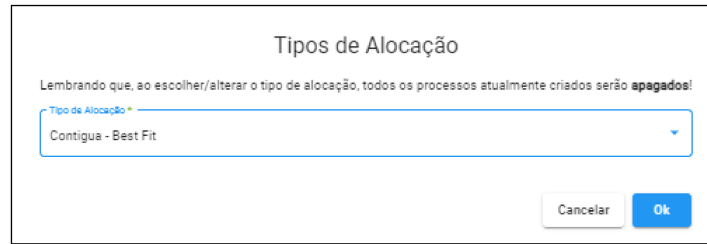


Figura 8 – Módulo para selecionar o método de alocação de memória desejado.

Fonte: Captura feita pelo autor.

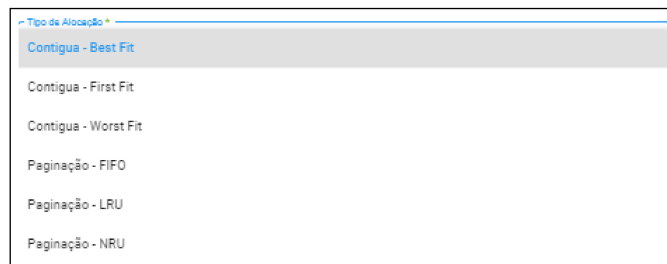


Figura 9 – Relação completa de métodos de alocação disponíveis no sistema.

Fonte: Captura feita pelo autor.

É disponível adicionar, modificar, pausar e remover processos, permitindo uma gestão completa das tarefas dentro do simulador.

3.4.1 Inicialização do Processo

Para iniciar um processo, basta acionar o botão "Criar Processo", que abrirá um modal contendo um formulário. Este formulário possui dois tipos, conforme a categoria de alocação escolhida, alocações contíguas (Figura 10) ou alocação com paginação (Figura 11). A criação é suspensa quando é excedida a quantidade máxima de processos, ou a quantidade máxima de blocos no simulador, sendo liberado novamente ao momento no qual um processo for finalizado ou excluído.

3.4.2 Alterações

Parte da estruturação inicial realizada por Sousa (2024) foi mantida, com ajustes feitos para integrar o processador ao gerenciador de memória. Foram adicionados dois novos campos no módulo de inicialização de processos, possibilitando a configuração de parâmetros específicos para a administração de memória. Esses campos extras possibilitam ao usuário ajustar o uso da memória de forma mais detalhada, dando a possibilidade no qual o simulador reproduza com maior precisão o comportamento real dos processos no que se refere à gestão da memória. Além do mais, a interface foi adaptada para acomodar as novas opções, mantendo a organização, e proporcionando uma vivência educacional

The dialog box is titled "Criação de processo". In the top right corner, it says "Máx Processos Permitido: 15". The main area contains several input fields: "Cor do processo" with the value "#ff5252" and a red circular icon; "Tipo de processo" with a dropdown menu showing "CPU-bound"; "Prioridade" with the value "0"; "Quantidade de processos" with the value "1"; "Unidade de tempo *" with the value "1"; and "Quantidade de blocos de memória" with the value "5". At the bottom left, it says "Máximo de blocos permitido: 120". At the bottom right, there are two buttons: "Cancelar" and "Ok".

Figura 10 – Módulo para inicialização de processos com alocação contígua.

Fonte: Captura feita pelo autor.

The dialog box is titled "Criação de processo". In the top right corner, it says "Máx Processos Permitido: 15". The main area contains several input fields: "Cor do processo" with the value "#ff5252" and a red circular icon; "Tipo de processo" with a dropdown menu showing "CPU-bound"; "Prioridade" with the value "0"; "Quantidade de processos" with the value "1"; "Unidade de tempo *" with the value "1"; and "Quantidade de páginas" with the value "1". At the bottom left, it says "Máximo de páginas permitido: 12". At the bottom right, there are two buttons: "Cancelar" and "Ok".

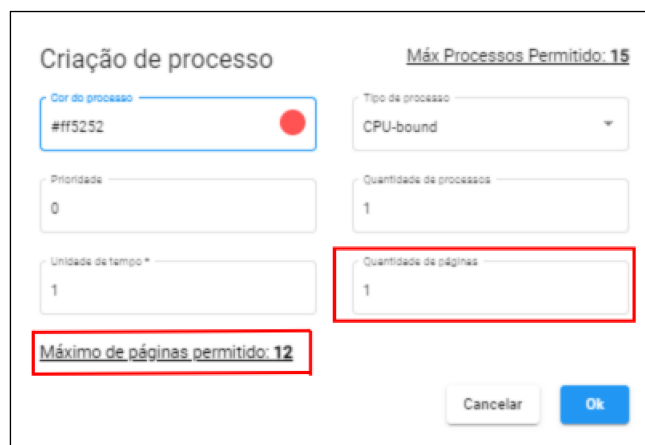
Figura 11 – Módulo para inicialização de processos com alocação com paginação.

Fonte: Captura feita pelo autor.

mais rica para o usuário. Na Figura 12 e Figura 13 está as alterações (demarcadas em vermelho) realizadas no módulo de inicialização:

3.5 Desenvolvimento das abordagens de alocação contígua.

Através dos fundamentos teóricos discutidos na subseção 2.3.2.1, foi desenvolvida a implementação das abordagens de alocação contígua no SimulateOS. A finalidade desta implementação é oferecer aos discentes de Sistemas Operacionais uma vivência prática



Criação de processo

Máx Processos Permitido: 15

Cor do processo: #ff5252

Tipo de processo: CPU-bound

Prioridade: 0

Quantidade de processos: 1

Unidade de tempo: 1

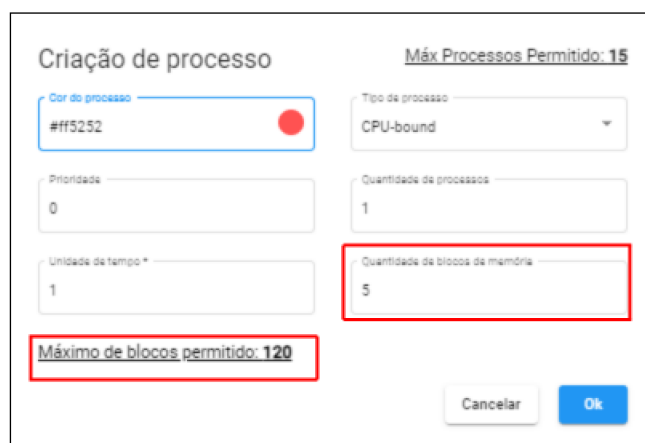
Quantidade de páginas: 1

Máximo de páginas permitido: 12

Cancelar Ok

Figura 12 – A janela demarca, com destaque em vermelho, as alterações para inicialização de processos utilizando alocação com paginação.

Fonte: Captura feita pelo autor.



Criação de processo

Máx Processos Permitido: 15

Cor do processo: #ff5252

Tipo de processo: CPU-bound

Prioridade: 0

Quantidade de processos: 1

Unidade de tempo: 1

Quantidade de blocos de memória: 5

Máximo de blocos permitido: 120

Cancelar Ok

Figura 13 – A janela demarca, com destaque em vermelho, as alterações para inicialização de processos utilizando alocação contígua.

Fonte: Captura feita pelo autor.

e interativa, que permita visualizar como diferentes estratégias de alocação de memória afetam o desempenho e a organização da memória.

Os algoritmos FirstFit, BestFit e WorstFit foram incorporados ao sistema, utilizando uma tabela que exibe os blocos contíguos e o bloco inicial, para monitorar os segmentos de memória ocupados e os espaços livres. A sequência de telas apresentada nas imagens a seguir exemplifica o funcionamento da alocação contígua no contexto de uma mesma simulação, na qual empregou-se o algoritmo Best Fit para ilustrar o comportamento do sistema durante cada etapa.

A interface inicial do SimulateOS (Figura 14) exibe a memória contendo todos os segmentos de memória disponíveis, pronta para a inserção de novos processos. Em seguida, ao criar processos, os blocos de memória ocupados são atualizados de forma instantânea, como mostrado na Figura 15, permitindo ao usuário observar a distribuição dos processos.

Após a remoção de alguns processos, o sistema reorganiza a interface automatica-

mente, exibindo os blocos liberados, como ilustrado na Figura 16. Essa sequência de imagens permite ao usuário acompanhar o efeito das operações de alocação e desalocação, observando o comportamento da memória durante um ciclo de simulação contínua. Essa estrutura modular e a visualização contínua proporcionam uma análise prática e clara sobre métodos de alocação contígua, permitindo ao utilizador observar as vantagens e limitações de cada abordagem de forma instantânea.

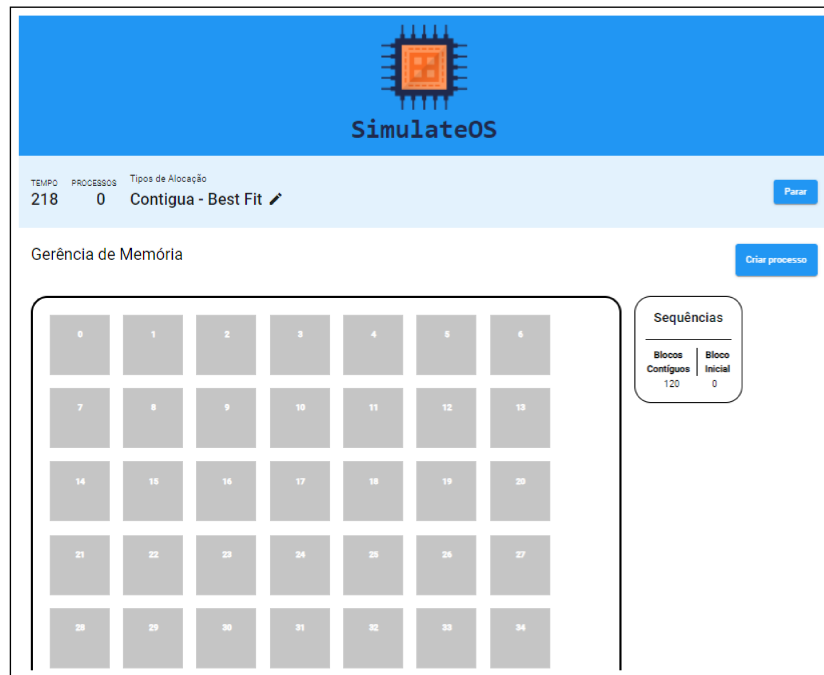


Figura 14 – Interface inicial com os blocos livres.

Fonte: Captura feita pelo autor.

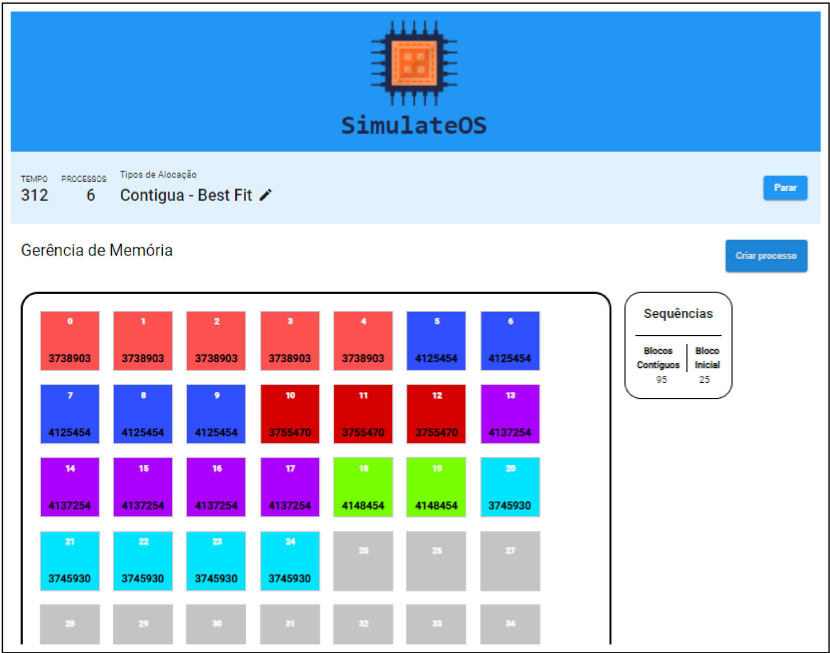


Figura 15 – Interface inicial com processos criados.
Fonte: Captura feita pelo autor.

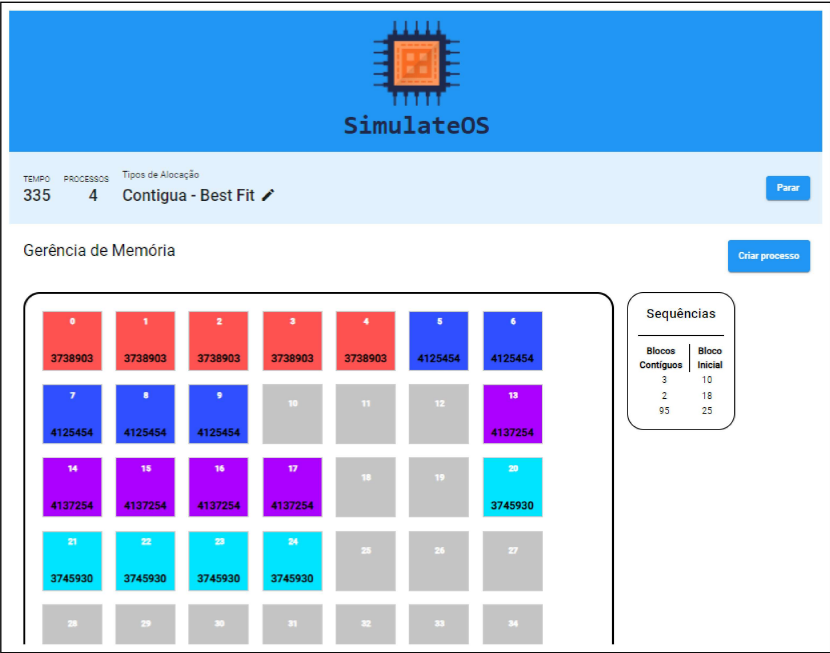


Figura 16 – Interface inicial com processos encerrados.
Fonte: Captura feita pelo autor.

3.6 Desenvolvimento das abordagens de Log e Sincronização dos Sistemas

Para garantir o controle e a integridade das operações no SimulateOS, foram desenvolvidos métodos de log e sincronização que auxiliam tanto no monitoramento das ações realizadas quanto na integração entre o gerenciador de memória também o sistema de processador desenvolvido por Sousa (2024).

Os métodos de log, inicialmente implementados por Sousa (2024) utilizando o Local Storage — um mecanismo de armazenamento local baseado em navegador que permite guardar dados no formato chave-valor de maneira persistente — não apenas registram as operações realizadas no sistema relacionado ao processador, mas também armazenam informações detalhadas sobre cada processo. Esses logs incluem dados essenciais como o Identificador de Processo (PID), prioridade, cor, tipo, estado atual, tempo de Unidade Central de Processamento (CPU) acumulado, momento de criação, tempo necessário para finalização, tempo em execução, tipo atual, blocos de memória requeridos e blocos alocados. Essa estrutura de Log detalhada permite acompanhar as etapas do ciclo de cada processo, facilitando a análise e depuração do sistema.

No entanto, essa versão inicial do Log não comunicava diretamente o gerenciador de memória com o sistema de processador, no qual limitava a sincronização entre os dois módulos. Para resolver essa limitação, desenvolvemos uma camada de sincronização que permite a integração entre o gerenciador de memória e o processador. Agora, ao inicializar um processo no gerenciador de memória, o sistema também registra automaticamente esse processo no módulo do processador contendo dados essenciais, garantindo consistência e sincronização entre os dois componentes.

Essa abordagem assegura que todas as operações de alocação e inicialização de processos sejam refletidas em ambos os sistemas, evitando conflitos e permitindo que as informações dos processos sejam atualizadas e mantidas em sincronia. Adicionalmente, preservou-se a estrutura modular do Log e da sincronização, facilitando a manutenção e expansão dessas funcionalidades. Com esses mecanismos, o SimulateOS proporciona um ambiente de simulação mais robusto e confiável, possibilitando os estudantes a interagir com um sistema sincronizado que representa de maneira fiel, a operação de uma plataforma computacional real.

3.7 Desenvolvimento das abordagens de Paginação

Após os fundamentos teóricos discutidos na subseção 2.3.2.2, foi desenvolvida a implementação dos algoritmos de substituição de páginas no SimulateOS. Essa implementação tem como finalidade proporcionar aos discentes uma experiência interativa que lhes permita observar, na prática, de que forma distintos algoritmos de substituição de páginas

funcionam e como afetam o uso e o desempenho da memória em um sistema operacional.

No sistema, os algoritmos de substituição de páginas LRU, NRU e FIFO foram implementados para oferecer uma variedade de estratégias que representam abordagens distintas para a escolha de páginas a serem substituídas durante uma falta de página. Para ilustrar o funcionamento desses algoritmos, a interface inicial do SimulateOS (Figura 17) exibe todos os blocos livres na memória e uma tabela de páginas vazia, pronta para que novas alocações sejam realizadas conforme processos são criados. A partir deste estado inicial, o usuário pode começar a criar processos e observar como as páginas e blocos são alocados conforme a necessidade.

Uma funcionalidade importante do sistema é ao clicar em um processo, seja na memória física ou na tabela de páginas, um painel dedicado à edição do processo, chamada Edit Process, é exibida (Figura 20). Nesta janela, o usuário pode visualizar detalhes importantes do processo, dentre eles a categoria do processo, cor, ID e páginas associadas, bem como os blocos de memória alocados que fazem referência a esse processo. Adicionalmente, a janela permite ao usuário pausar ou encerrar o processo diretamente.

À medida que novos processos são criados e os blocos livres são utilizados, os algoritmos de substituição entram em ação para liberar espaço na memória quando ocorre uma falta de página. A Figura 18 demonstra o sistema em operação, com as páginas sendo substituídas e os blocos de memória atualizados instantaneamente, conforme a lógica de cada algoritmo selecionado. Essa visualização permite ao usuário acompanhar o ciclo de substituição, com os dados das páginas e blocos sendo movidos, alocados e desalocados conforme as decisões de cada método.

Após a remoção de alguns processos, o sistema exibe os blocos e páginas que foram liberados, como ilustrado na Figura 19. Esse estado final da memória permite ao usuário observar a fragmentação e a organização dos blocos de memória após várias operações de substituição, o que auxilia na análise das vantagens e desvantagens de cada algoritmo em variados contextos de aplicação.

Essa sequência de visualizações, juntamente com a funcionalidade de alteração de processos, possibilita ao usuário um entendimento prático de como funciona a substituição de páginas, mostrando os efeitos de cada estratégia de maneira objetiva e instrutiva, e permitindo que eles observem as características de cada abordagem instantaneamente.

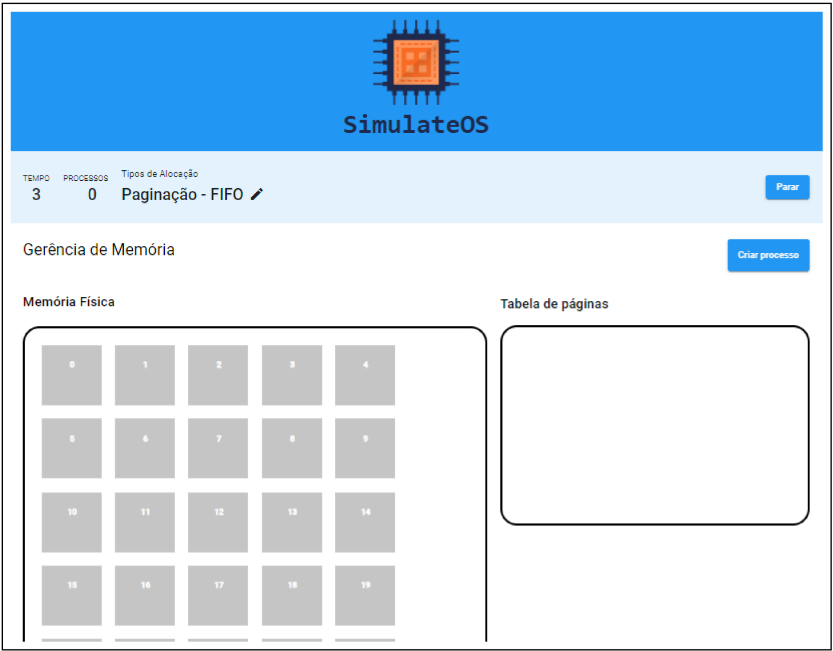


Figura 17 – Interface inicial com os blocos livres e a tabela de páginas vazia.
Fonte: Captura feita pelo autor.

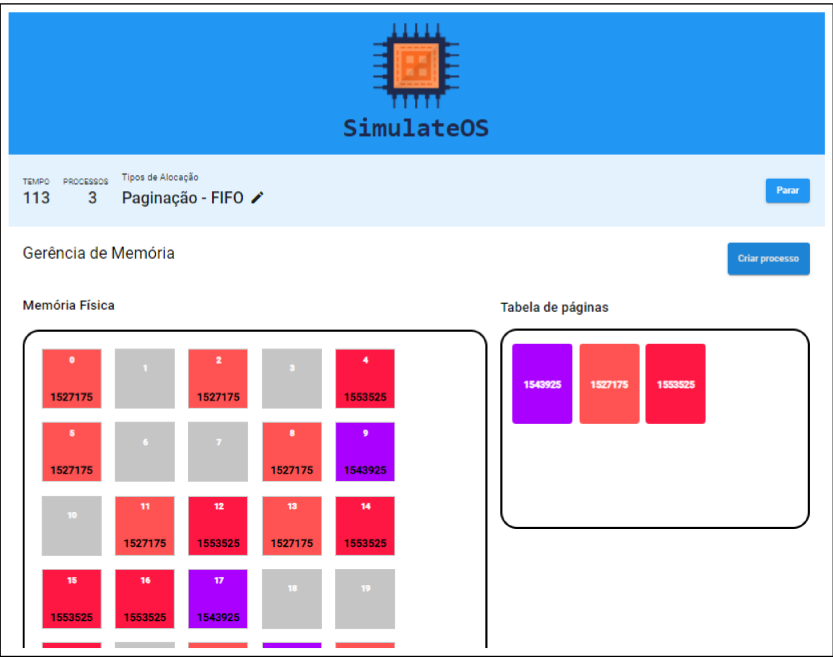


Figura 18 – Interface de execução das técnicas de substituição de páginas.
Fonte: Captura feita pelo autor.

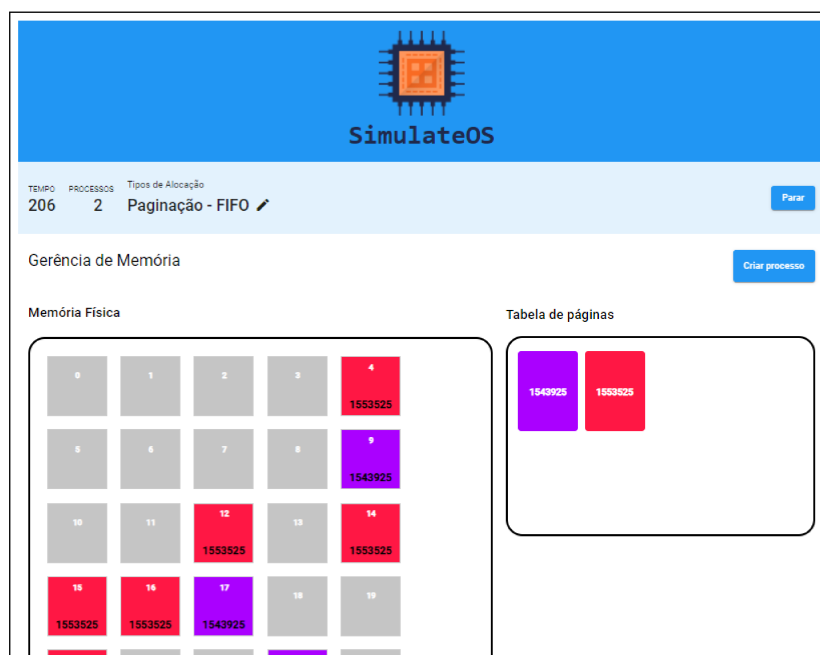


Figura 19 – Interface após a remoção de processos, exibindo os blocos e páginas liberadas.

Fonte: Captura feita pelo autor.

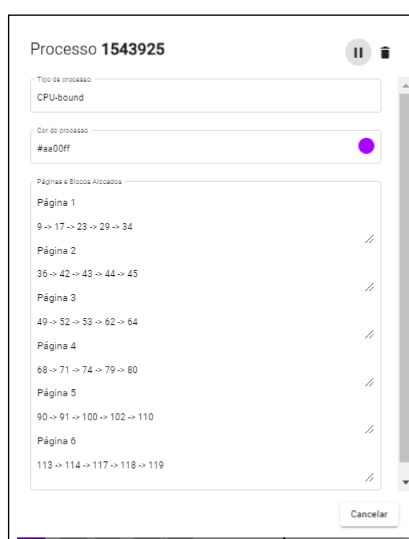


Figura 20 – Pannel de modificação de processos exibindo detalhes e controles.

Fonte: Captura feita pelo autor.

Conclusão

O trabalho desenvolvido no SimulateOS teve como propósito principal aprimorar uma plataforma web interativa para o aprendizado teórico e prático sobre gestão da memória e técnicas de alocação dos processos em plataformas operacionais. Durante o aprimoramento, foram implementadas funcionalidades que simulam operações de alocação e liberação de memória, apresentando de maneira visual e interativa os conceitos essenciais da gestão da memória.

A plataforma foi projetada com uma interface que organiza de forma clara os conceitos apresentados, permitindo a representação de métodos como alocação contígua e paginação. Também foi possível incluir funcionalidades para interrupção e reinício das simulações, possibilitando a exploração de diferentes cenários.

Embora não tenham sido realizados testes formais com usuários, o SimulateOS apresenta características que indicam o alcance dos objetivos propostos, como oferecer uma ferramenta funcional que complementa o estudo de Sistemas Operacionais por via de simulações práticas. Como trabalhos futuros, sugere-se a execução de testes feitos com usuários para validar e aprimorar a usabilidade e a interface, bem como a inclusão de novos algoritmos e cenários mais complexos, ampliando o potencial educativo da ferramenta.

A pesquisa e o aprimoramento do SimulateOS permitiram alcançar os objetivos estabelecidos, com contribuições significativas na aplicação dos algoritmos de alocação contígua e paginação, além do desenvolvimento de métodos de Log e sincronização. Esses elementos possibilitam uma aplicação prática de simulação, oferecendo uma base sólida para o estudo dos diferentes conceitos de gestão de memória abordados neste trabalho.

4.1 Principais Contribuições

Este trabalho trouxe aprimoramentos significativos ao SimulateOS, consolidando-o como uma ferramenta com potencial para apoiar a aprendizagem de Sistemas Operacionais. Com base no estudo de princípios teóricos, com foco especial na gerência de memória

e processos, foram implementadas melhorias que tornam o simulador mais funcional e alinhado às necessidades educacionais.

Um dos maiores aportes deste projeto foi a implementação da gerência de memória, que permite ao simulador oferecer uma visão prática sobre como a memória é organizada e utilizada. Essa adição possibilita a experimentação de diferentes cenários de alocação de memória, destacando as implicações de cada método no que diz respeito a performance e eficiência.

Adicionalmente, o aprimoramento do sistema de Log trouxe melhorias no rastreamento das operações realizadas dentro do simulador. O Log foi ajustado para capturar informações detalhadas sobre cada processo, incluindo sua criação, alocação de memória e mudanças de estado. Essa funcionalidade permite revisar o histórico das operações realizadas, promovendo um melhor entendimento das etapas de execução de cada processo.

Essas melhorias, somadas à integração entre o gerenciamento de memória e o sistema feito por Sousa (2024), consolidam o potencial do SimulateOS como uma ferramenta de aprendizado interativa. Através de simulações práticas e cenários diversos, o simulador possibilita a exibição e manipulação de aspectos cruciais de um sistema operacional, auxiliando no entendimento e aplicação de conceitos teóricos.

4.2 Trabalhos Futuros

Para o aprimoramento da ferramenta, alguns pontos de melhoria foram identificados e podem ser desenvolvidos em trabalhos futuros:

- ❑ **Expansão dos algoritmos implementados:** A inclusão de novos algoritmos de alocação e substituição pode tornar o simulador ainda mais abrangente, abordando variações mais avançadas e atuais desses métodos.
- ❑ **Melhorias na experiência do usuário:** A criação de uma interface mais responsiva e a inclusão de tutoriais interativos poderiam auxiliar os discentes a navegar pelo simulador com mais facilidade e entender cada funcionalidade.
- ❑ **Integração com plataformas de ensino:** A adaptação do simulador para integração com plataformas de ensino pode ampliar o alcance e permitir que professores acompanhem o desempenho dos alunos diretamente.
- ❑ **Exportação e Compartilhamento de Cenários de Simulação:** Permitir que os usuários salvem, exportem e compartilhem seus cenários de simulação, possibilitando a colaboração e o estudo de casos variados entre discentes.

4.3 Contribuições em Produção Bibliográfica

Ao longo da execução deste projeto, os achados intermediários foram submetidos/apresentados no XVII Workshop de Teses e Dissertações em Ciência da Computação (WTDCC) e no XVIII WTDCC de 2024, sendo este um evento acadêmico promovido pela Instituto de Computação da Universidade Federal de Uberlândia (UFU). Foi enviado também ao IV Seminário de Pesquisa, Inovação e Pós-Graduação da UFU, promovido pela Pró-Reitoria de Pesquisa e Pós-Graduação (PROPP), da Universidade Federal de Uberlândia (UFU). Além do estudo elaborado para o evento, o conceito foi exposto aos participantes "através de uma apresentação visual incluindo descrições e ilustrações explicativas, facilitando o compartilhamento do projeto com outros estudantes.

Em um futuro próximo, planeja-se submeter um trabalho completo a um evento nacional na área, visando ampliar sua visibilidade e promover discussões sobre seu impacto educacional.

Referências

- ANGULAR. **O que é o Angular?** 2021. Acesso pelo link: <<https://angular.io/guide/what-is-angular>>. Consulta realizada em: 5 out. 2024. Citado na página 18.
- CARVALHO, D. S. et al. S²o: Uma ferramenta de apoio ao aprendizado de sistemas operacionais. In: **Anais do XXVI Congresso da SBC – XIV Workshop sobre Educação em Computação (XIV WEI)**. Sociedade Brasileira de Computação (SBC), 2006. Consulta realizada em: 5 out. 2024. Disponível em: <<http://www.natalnet.br/sbc2006/pdf/arq0107.pdf>>. Citado 3 vezes nas páginas 10, 11 e 19.
- FIGMA. 2016. <<https://figma.com>>. Acesso em: 5 out. 2024. Citado na página 18.
- FREITAS, G. M. B. de. **Simulador de Gerência de Processos para Sistemas Operacionais**. Trabalho de Conclusão de Curso (Graduação) — Universidade Federal de Uberlândia, 2023. Graduação em Sistemas de Informação. Consulta realizada em: 10 nov. 2024. Disponível em: <<https://repositorio.ufu.br/handle/123456789/37373>>. Citado 6 vezes nas páginas 10, 11, 19, 20, 21 e 22.
- KIOKI, E. Y.; SANTIAGO, P. P. S. A. C. Um simulador educacional como ferramenta de apoio ao ensino da disciplina de sistemas operacionais. **Revista INICIA**, v. 8, n. 8, p. 41–48, 2008. Citado na página 11.
- MAIA, L. P. **SOsim: Simulador para o Ensino de Sistemas Operacionais**. Dissertação (Dissertação (Mestrado)) — Universidade Federal do Rio de Janeiro, 2001. Citado 3 vezes nas páginas 10, 11 e 19.
- MENSAH, O. **Criando Aplicativos Bonitos com Angular Material**. 2022. Acesso pelo link: <<https://auth0.com/blog/creating-beautiful-apps-with-angular-material/>>. Consulta realizada em: 5 out. 2024. Citado na página 18.
- ODEN, J. T. et al. **Revolutionizing Engineering Science Through Simulation**. Arlington, VA, USA, 2006. Citado na página 14.
- REIS, F. P.; COSTA, H. A. X. Tbc-so/web: Software educativo para aprendizagem de políticas de escalonamento de processos e de alocação de memória em sistemas operacionais. **Revista de Educação em Computação**, 2009. Disponível em: <http://professores.dcc.ufla.br/~heitor/Projetos/TBC_SO_WEB/artigo_TBC_SO_WEB.pdf>. Citado 2 vezes nas páginas 13 e 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**. Brasil: Sociedade Brasileira de Computação (SBC), 2009. Citado na página 11.

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: Sociedade Brasileira de Computação (SBC), 2014. p. 737–748. Disponível em: <<https://sol.sbc.org.br/index.php/sbsi/article/view/6160>>. 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 na página 11.

SILVA, J.; COSTA, M. Revolutionizing physics education through simulation. **Revista Brasileira de Ensino de Física**, 2020. Disponível em: <<https://periodicos.ufsc.br/index.php/fisica/article/view/2175-7941.2020v37n1p242>>. Citado na página 14.

SOUSA, A. M. de. **Simulador de Gerenciador de Processos para Sistemas Operacionais**. Trabalho de Conclusão de Curso (Graduação) — Universidade Federal de Uberlândia, 2024. Graduação em Sistemas de Informação. Consulta realizada em: 8 out. 2024. Disponível em: <<https://repositorio.ufu.br/handle/123456789/41683>>. Citado 8 vezes nas páginas 6, 11, 12, 21, 22, 26, 31 e 36.

TANENBAUM, A. S. **Sistemas Operacionais Modernos**. São Paulo, Brasil: Pearson Prentice Hall, 2010. ISBN 978-85-87918-57-4. Citado na página 15.

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

TYPESCRIPT. 2022. <<https://www.typescriptlang.org/>>. Acesso em: 05 out. 2024. Citado na página 18.