ULBRA
CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS
Recredenciado pela Portaria Ministerial nº 1.162, de 13/10/16, D.O.U. nº 198, de 14/10/2016
AELBRA EDUCAÇÃO SUPERIOR - GRADUAÇÃO E PÓS-GRADUAÇÃO S.A.

wesley.miranda@rede.ulbra.br

MÓDULO DE SIMULAÇÃO DO MECANISMO DE PAGINAÇÃO POR DEMANDA COM SUBSTITUIÇÃO DE PÁGINAS DO OSLIVE: ALGORITMO DE SEGUNDA CHANCE

Palmas – TO
2022

Wesley Miranda Novaes

MÓDULO DE SIMULAÇÃO DO MECANISMO DE PAGINAÇÃO POR DEMANDA COM SUBSTITUIÇÃO DE PÁGINAS DO OSLIVE: ALGORITMO DE SEGUNDA CHANCE

Projeto de Pesquisa elaborado e apresentado como requisito parcial para aprovação na disciplina de Projeto Tecnológico do curso de bacharel em Sistemas de Informação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Profª. M.e Madianita Bogo Marioti.

Palmas – TO

2022

Wesley Miranda Novaes

MÓDULO DE SIMULAÇÃO DO MECANISMO DE PAGINAÇÃO POR DEMANDA COM SUBSTITUIÇÃO DE PÁGINAS DO OSLIVE: ALGORITMO DE SEGUNDA CHANCE

Projeto de Pesquisa elaborado e apresentado como requisito parcial para aprovação na disciplina de Projeto Tecnológico do curso de bacharel em Sistemas de Informação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Profª. M.e. Madianita Bogo Marioti.

Aprovado em: _____/_____/_____

BANCA EXAMINADORA

_____

Profª. M.e Madianita Bogo Marioti

Orientador

Centro Universitário Luterano de Palmas – CEULP

_____

Profª. Esp. Fernanda Pereira Gomes

Centro Universitário Luterano de Palmas – CEULP

_____

Profº. Esp. Fábio Castro Araújo

Centro Universitário Luterano de Palmas – CEULP

Palmas – TO

2022

RESUMO

NOVAES, Wesley Miranda. **MÓDULO DE SIMULAÇÃO DO MECANISMO DE PAGINAÇÃO POR DEMANDA COM SUBSTITUIÇÃO DE PÁGINAS DO OSLIVE: ALGORITMO DE SEGUNDA**. 2022. Trabalho de Conclusão de Curso (Graduação) - Curso Sistemas de Informação, Centro Universitário Luterano de Palmas, Palmas/TO, 2022.

O uso de ferramentas que auxiliam na compreensão do funcionamento de sistemas operacionais é de grande valia no auxílio do ensino e aprendizado da disciplina Sistemas Operacionais. Matéria presente em grande parte dos cursos de computação no Brasil, como recomendado pelo Ministério da Educação. Sistemas operacionais realizam ações em baixo nível e que exigem alto grau de abstração para serem compreendidas, o uso dessas ferramentas como complemento de aprendizado visa facilitar a compreensão dos conceitos por trás dessas ações. Pensando nisso, existe um projeto em desenvolvimento que trabalha este tema e está em uso pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA), o OSLive, plataforma online que oferece recursos para auxiliar os professores e alunos no processo de ensino-aprendizagem da disciplina, com simulações dos conceitos e representações gráficas. As diferenças conceituais entre os algoritmos de substituição de páginas na paginação por demanda como o FIFO e o Segunda Chance se tornam mais compreensíveis através de representações visuais oferecidas pelo OSLive. A ferramenta possibilita ao aluno um aprendizado com maior interação, podendo criar suas simulações e observar o comportamento desses algoritmos. Ao professor, a plataforma se mostra uma ferramenta de auxílio que pode tornar as aulas mais dinâmicas e estimulantes, permitindo criar demonstrações de situações que deseja apresentar aos alunos. Esta plataforma já possui um módulo que oferece a simulação do mecanismo de paginação por demanda com substituição de página utilizando o algoritmo FIFO, o objetivo deste trabalho foi o de acrescentar ao módulo o algoritmo de Segunda Chance, aumentando o escopo de possibilidades de simulação dos conceitos de mecanismos da paginação por demanda e permitindo a comparação entre eles.

Palavras-chave: Sistemas Operacionais, paginação por demanda, substituição de páginas, Segunda Chance, OSLive.

LISTA DE FIGURAS

Figura 1. Visão abstrata de um sistema computacional 9
Figura 2. MMU: Tradução de endereços físico/lógico 12
Figura 3. Mecanismo de paginação simples 13
Figura 4. Mecanismo de paginação por demanda 15
Figura 5. Etapas da paginação por demanda 16
Figura 6. Etapas do mecanismo de substituição de páginas 18
Figura 7. Algoritmo de substituição de páginas FIFO, modelo conceitual 21
Figura 8. Localizando página vítima com algoritmo de Segunda Chance 23
Figura 9. Etapas do desenvolvimento do projeto 25
Figura 10. Arquitetura do software 28
Figura 11. Tela inicial do módulo 30
Figura 12. Simulação paginação por demanda 32
Figura 13. Simulação da paginação por demanda 34
Figura 14. Simulação da substituição de página: FIFO 35
Figura 15. Resultado da simulação da substituição de página: FIFO 36
Figura 16. Tabela de ordem de carga com algoritmo FIFO 37
Figura 17. Tabela de ordem de carga de acordo com algoritmo de substituição de página 38
Figura 18. Estado do bit de referência 39
Figura 19. Simulação da substituição de página: Segunda Chance 40
Figura 20. Resultado da simulação 40

LISTA DE ABREVIATURAS E SIGLAS

CPU - Unidade Central de Processamento

MMU - Unidade de Mapeamento de Memória (Memory Management Unit)

E/S - Entrada e Saída

FIFO - Primeiro a Entrar Primeiro a Sair (First-in-First-out)

RAM - Memória de Acesso Aleatório (Random Access Memory)

HD – Disco rígido (Hard Disk)

SO - Sistema Operacional

SUMÁRIO

1 INTRODUÇÃO **6**

REFERENCIAL TEÓRICO **8**

Sistemas Operacionais 9

Gerência de Memória 12

Paginação 14

Paginação por Demanda 16

Algoritmos de Substituição de Páginas 19

Algoritmo FIFO 22

Algoritmo Segunda Chance 23

METODOLOGIA **25**

Desenho do estudo 26

Ferramentas utilizadas 27

RESULTADOS E DISCUSSÃO **29**

4.1 Arquitetura do Software 29

4.2 Versão Atual do Módulo de Paginação por Demanda 31

4.2.1 Interface da aplicação 31

4.2.2 Demonstração da simulação da Paginação por Demanda 33

4.2.3 Demonstração da Simulação da Paginação por Demanda com Substituição de página FIFO 35

4.3.1 Representação do Algoritmo Segunda Chance 38

4.3.2 Demonstração da Simulação da Paginação por Demanda com Substituição de página Segunda Chance 40

CONSIDERAÇÕES FINAIS **42**

REFERÊNCIAS **44**

1 INTRODUÇÃO

Com o avanço da tecnologia, novas formas de aprimorar os métodos de ensino são possibilitadas, como o uso de ferramentas computacionais que auxiliam o processo de aprendizagem do aluno. A evolução dos computadores para aparelhos mais rápidos e com maior poder de processamento permitem a elaboração de sistemas mais complexos e, ao mesmo tempo, o aperfeiçoamento dos *softwares* e recursos mais amigáveis ao usuário. Com isso, é possível ter softwares com interfaces gráficas mais intuitivas, maior nível de interação e simulações que representam conceitos abstratos considerados de difícil entendimento, como os vistos nas disciplinas que estudam Sistemas Operacionais.

Segundo Tanenbaum e Bos (2016), os computadores são equipados com uma camada de software, chamada sistema operacional (SO), cuja função é fornecer aos programas do usuário um modelo de computador melhor, mais simples e mais limpo de gerenciar todos os recursos. Eximir o usuário do controle direto do *hardware,* como dispositivos de entrada e saída, o disco físico e a memória, é uma função do SO que permite que o usuário possa se dedicar a sua tarefa sem que precise se preocupar, por exemplo, em gerenciar o espaço de memória onde seus arquivos serão gravados. É através do Sistema Operacional que programas realizam acesso aos periféricos, dessa forma, o SO controla qual programa acessa qual recurso permitindo uma distribuição equilibrada e eficiente dos recursos. (OLIVEIRA, CARISSIMI E TOSCANI 2010).

A disciplina de Sistemas Operacionais estuda os conceitos de gerência desses recursos, organizadas em tópicos que contemplam as quatro áreas de gerência de um Sistema Operacional: gerenciamento de arquivos, gerência dos dispositivos de entrada e saída, gerência de processos e a gerência do uso da memória. A conveniência que o SO traz ao ocultar processos executados em baixo nível também aumenta a exigência por abstração do aluno, que precisa entender o funcionamento desses conceitos sem poder observá-los em funcionamento.

No contexto acadêmico, a simulação com representações gráficas de conceitos abstratos da computação como ferramenta de aprendizagem aumentam o escopo de interação e compreensão do aluno, como, por exemplo, de áreas ligadas ao processamento de baixo nível, onde há interação direta entre o sistema operacional e a máquina. Ela permite ao aluno criar situações, testá-las, analisar os resultados, ajudando-o na construção de seus conhecimentos. A simulação como ferramenta de auxílio de aprendizagem possui um grande potencial educacional, superando programas tradicionais, como tutoriais e exercício-e-prática

(MAIA, 2003). Atualmente, existem no mercado ferramentas que oferecem essa simulação, porém, em sua maioria são sistemas complexos, de difícil manuseio e com representações visuais que tendem a ser pouco intuitivas, o que dificulta a compreensão dos conceitos.

Nesse contexto, no Centro Universitário Luterano de Palmas (CEULP/ULBRA), está sendo desenvolvida a plataforma *web* OSLive como ferramenta de aprendizagem, que visa auxiliar os alunos no entendimento dos conceitos abordados na disciplina Sistemas Operacionais de uma forma mais interativa e visual. O OSLive possui módulos que oferecem a simulação da gerência dos recursos da máquina, que é feita pelo SO, disponibilizando, também, exercícios que ajudam no entendimento dos conceitos estudados. Um dos módulos da plataforma, disponíveis até o momento, é o da paginação por demanda com substituição de páginas FIFO **(First *in, first out)***, que oferece ao aluno a possibilidade de simular o funcionamento do gerenciamento da memória feita pelo SO.

Quando um programa é executado, um processo é gerado, e para o seu funcionamento correto ele precisará ser copiado para a memória RAM (ou memória principal), porém, a memória principal é um componente físico com tamanho limitado, o que impossibilita o seu uso irrestrito. Nesse contexto, o gerenciamento de memória provém mecanismos que permitem o uso eficiente e justo da memória. Na gerência de memória, a paginação é o mecanismo que cria uma abstração do uso da memória, utilizando uma memória auxiliar (o HD, por exemplo) como um espaço de memória virtual, onde os processos são divididos em blocos de tamanho fixos denominado páginas e armazenado nessa memória de endereçamento lógico. A memória principal, nesse caso a memória que é física, também é organizada em blocos de tamanhos fixos de igual tamanho as páginas, recebem o nome de *Frames*. Na paginação, todas as páginas são carregadas para a memória quando referenciada, com isso, pode ocorrer o carregamento de páginas que não são necessárias para o uso do programa naquele momento. A possível solução para situações como essa foi proposta no conceito da paginação por demanda.

Na paginação por demanda, as páginas dos processos são transferidas da memória lógica para a principal apenas quando são referenciadas. Dessa forma, são levadas para a memória principal apenas as páginas realmente necessárias à execução do programa (MACHADO E MAIA, 2014). Com isso, é possível a execução de programas que superam em tamanho o espaço disponível na memória principal sem precisar que todas as páginas de um processo estejam na memória principal ao mesmo tempo.

O limite físico da memória RAM impõe restrições acerca da quantidade de páginas que podem ser referenciadas. Assim, quando todos os espaços da memória estiverem

indisponíveis, o SO deverá escolher qual página deverá ser removida da memória para que uma nova passa ser adicionada e é nesse momento que o algoritmo de substituição é acionado.

O algoritmo de substituição de páginas FIFO *(First in, first out***)** usa o conceito de fila para determinar qual página deverá ser removida, onde a primeira página que foi referenciada na memória será a página que estará a mais tempo na memória e, portanto, será removida e cederá o lugar para a nova página. As páginas são organizadas em fila ao sofrerem a carga, no momento da substituição, a página no início da fila é removida (H. ARPACI-DUSSEAU e C. ARPACI-DUSSEAU, 2014).

O algoritmo FIFO, que é o algoritmo de substituição de páginas disponível na versão atual do OSLive, não é único algoritmo de substituição de páginas que pode ser implementado na paginação por demanda, existem outros, como o algoritmo de Segunda Chance, que é uma derivação do FIFO. Nesse algoritmo, as páginas são organizadas por ordem de carga, assim como no FIFO, porém, um *bit* de referência é vinculado à página para determinar que ela foi referenciada no último ciclo de acesso do processador, o algoritmo percorre a todas as páginas da fila de forma circular, procurando a página com menor *timestamp* (tempo de carga) e com o bit de referência igual a 0. Caso a página possua o bit igual a 1, ela recebe uma segunda chance e passa para o final da fila. Assim, semelhante ao FIFO, o algoritmo de Segunda Chance irá substituir a página mais antiga, porém, somente se ela não tiver sido acessada recentemente, que receberá uma nova chance, considerando que ela pode então ser acessada novamente em breve.

Como complemento ao módulo de simulação de paginação por demanda com substituição de páginas do OSLive, este projeto propõe a inclusão do algoritmo de substituição de páginas Segunda Chance ao módulo de simulação da paginação por demanda. Assim, permitirá ao aluno criar simulações de paginação por demanda utilizando mais de um tipo de algoritmo de substituição de páginas e observar como cada um se comporta.

.

2 REFERENCIAL TEÓRICO

Esta seção abordará de maneira geral os Sistemas Operacionais e suas quatro áreas de gerência: gerência de arquivos, gerência de entrada e saída, gerência de memória e gerência de processos. Com o foco principal na gerência de memória, serão apresentados os conceitos relacionados ao mecanismo de gerência de memória e paginação por demanda com algoritmos de substituição de páginas.

2.1 Sistemas Operacionais

Um sistema operacional é um programa que gerencia o *hardware* de um computador, fornece uma base para os programas e aplicativos e age como um intermediário entre o usuário do computador e o hardware (SILBERSCHATZ, GALVIN E GAGNE, 2015). Desta maneira, um SO proporciona ao usuário uma maneira mais amigável de utilizar o dispositivo sem que seja necessário ao usuário saber como acessar diretamente os recursos de hardware para poder executar suas tarefas, programas etc.. A figura 1 apresenta uma simplificação da hierarquia do funcionamento de um computador, na primeira camada está o hardware, fazendo intermédio entre ele e os programas dos quais o usuário interage está o sistema operacional.

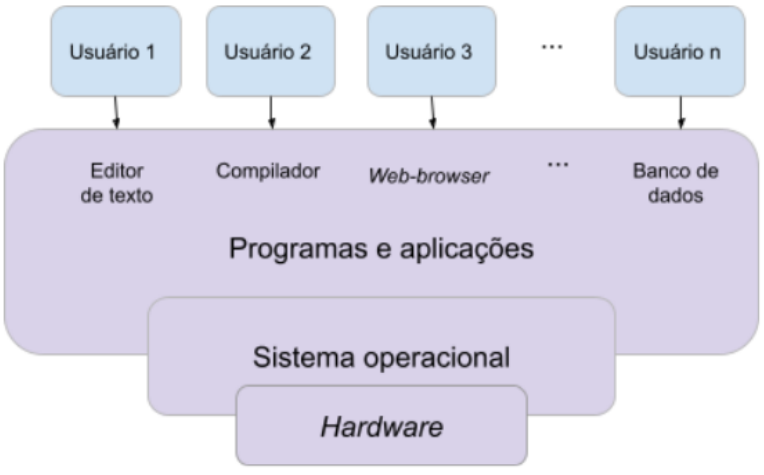

Figura 1. Visão abstrata de um sistema computacional
Fonte: Adaptada de (SILBERSCHATZ, GALVIN E GAGNE, 2015)

Segundo Stuart (2011), além das interfaces de interação com o usuário, o Sistema Operacional é encarregado das seguintes áreas de gerência:

- Gerência de arquivos: para Cunha et al. (2017), o sistema de arquivos é uma padronização e conjunto de estruturas para meios de armazenamento que permite que arquivos sejam gravados e lidos de uma forma segura;

- Gerência de processos: a principal responsabilidade do componente de gerenciamento de processo é a criação e término de processos. Segundo Silberschatz et all. 2012, um processo é uma unidade de trabalho, sendo que um sistema é o conjunto dessas unidades, que fazem parte do próprio sistema operacional ou são oriundas de códigos executados pelo usuário, o gerenciamento dos processos definirá como os processos podem ser executados concomitantemente pelo mesmo processador. Assim, essa área de gerência também inclui o escalonador de processos, que escolhe qual processo ou *thread* deverá ser executado na CPU a cada momento (TANENBAUM E BOS, 2016);

- Gerência de dispositivos de Entrada e saída (E/S): o SO faz o intermédio entre os programas em execução e os dispositivos de E/S conectados ao computador, assim, os programas não têm acesso direto ao *hardware*. Esse acesso é realizado por meio dos serviços implementados em rotinas disponibilizadas pelo SOs, que permitem ler e escrever dados nos dispositivos (CUNHA, et al. 2017);

- Gerência de memória: segundo SILBERSCHATZ, et al. (2015), o único local de armazenamento que o processador acessa através de endereço direto é a memória principal (RAM), para que ele possa buscar partes, ou informações que o programa atual necessita para a sua execução. Considerando que espaço físico da memória é limitado, o gerenciamento de memória é necessário para que a maior quantidade de informação possível possa ser carregada e disponibilizada na memória, assim permitindo que vários processos possam ao mesmo tempo fazer uso da memória.

A seção a seguir abordará de forma mais detalhada conceitos relacionados a área de gerência de memória, a fim de aprofundar nos conceitos que fundamentaram a construção do módulo de simulação de paginação por demanda com o algoritmo de substituição de página segunda chance que é o objetivo deste trabalho.

2.2 Gerência de Memória

Nas últimas décadas, sistemas, programas, aplicativos e outros recursos computacionais estão ficando cada vez maiores e mais robustos, aumentando a exigência imposta ao *hardware* dos dispositivos. Em contrapartida, as memórias de acesso rápido, como a memória RAM, ainda possuem um alto custo de produção que inviabiliza sua produção em tamanhos extensos como tem acontecido com memórias não voláteis usadas para armazenamento como o HD (*Hard Disk*). Uma das maiores preocupações dos projetistas é desenvolver SOs que não ocupem muito espaço de memória e, ao mesmo tempo, otimizem a utilização dos recursos computacionais (Machado e Maia, 2014).

A solução para conciliar a alta demanda de recursos advindas dos *softwares* com a limitação física da memória RAM é oferecida pelo subsistema de gerência de memória dos SOs. A gerência de memória é a parte do SO responsável por garantir que cada programa tenha uma área de memória para o seu código e os seus dados, garantindo a segurança e integridade para a sua execução (CUNHA et al. 2017). Através do gerenciamento de memória é possível estabelecer mecanismos que permitem a execução não só de programas que ultrapassam em tamanho o espaço disponível da memória principal como a execução de múltiplos programas ao mesmo tempo.

Na multiprogramação, vários programas são executados ao mesmo tempo, para isso, vários processos precisam ser carregados para a memória principal. A gerência de memória provém mecanismos que permitem o uso compartilhado da memória por esses processos e previne acesso direto dos mesmos à memória principal, evitando que processos sobrescrevam espaços de memória já utilizados por outros processos. Em um ambiente de multiprogramação, o sistema operacional deve proteger as áreas de memória ocupadas por cada processo, além da área onde reside o próprio sistema operacional (Machado e Maia, 2014).

Para evitar os problemas que o acesso direto à memória pode causar, foi criada uma abstração para a memória: os espaços de endereçamentos físicos e lógicos. Segundo Oliveira et al. 2010, endereços lógicos são os únicos no qual um processo enxerga e manipula, já os endereços físicos representam uma posição real de memória física. Assim, quando um programa referencia um processo, que pode ou não estar na memória física, ele o faz através de seu endereço lógico e fica a cargo do gerenciador de memória interpretar a localização dele na memória física ou realizar a carga. Neste contexto, a memória lógica é formada por todos os endereços lógicos de um processo e cada processo tem a memória lógica independente. A

memória física é o conjunto de endereços físicos disponibilizado pela memória RAM (Figueredo e Marioti, 2022).

Para realizar a carga do processo na memória, o SO precisa que esse endereço seja traduzido para o endereço físico da memória principal, o que é função da Unidade de Gerência de Memória (MMU). Segundo Silberschatz et al (2015), na paginação por demanda endereços físicos e lógicos podem ser diferentes entre si, assim, se faz necessário realizar o mapeamento dos endereços através de tabela relacionais. Os computadores atuais contam com o componente de *Hardware* chamado Unidade de Gerência de Memória MMU (*Memory Management Unit*), de forma que, sempre que um endereço lógico é referenciado, a MMU realiza a tradução desse endereço para o endereço real da memória principal. A figura 2 ilustra o processo onde o *CPU* passa a consultar a *MMU* quando precisa realizar uma leitura ou escrita na memória física.

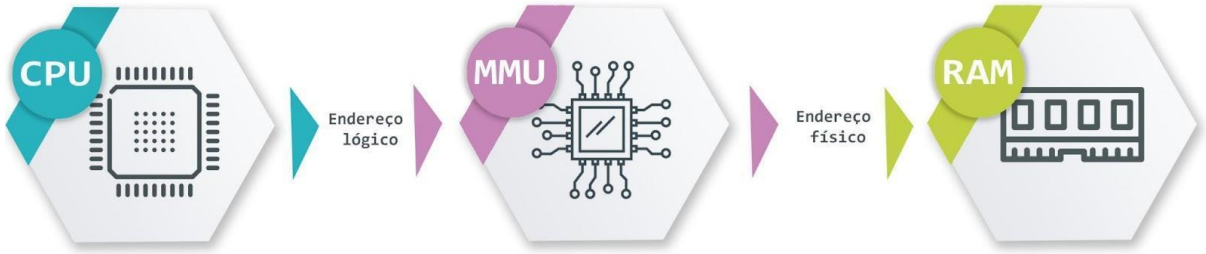

Figura 2. MMU: Tradução de endereços físico/lógico
Fonte: Elaborada pelo autor

A maioria dos sistemas operacionais atuais oferece o recurso de memória virtual, este mecanismo utiliza a memória secundária, o HD (*hard disk),* para armazenar partes do processo que não estão em uso naquele momento pelo processador. Desta forma, é possível executar programas carregados parcialmente na memória física, possibilitando um melhor uso do espaço disponível na memória *RAM* ao permitir que mais programas possam fazer uso da memória simultaneamente, aumentando o fator de multiprocessamento.

O mecanismo de paginação por demanda, que é baseado no mecanismo de Paginação, faz uso da memória virtual. Nele, as pequenas partes em que o processo foi dividido são denominadas Páginas, que são carregadas para a memória conforme a necessidade de seu uso (demanda). Sistemas operacionais como *Windows* e *Linux* utilizam esse princípio no gerenciamento da memória.

2.3 Paginação

Segundo Oliveira et al (2010), na paginação, a memória lógica e a memória física são divididas em blocos de tamanho único e fixo, o bloco lógico recebe o nome de página e o bloco físico recebe o nome de quadro. Geralmente, as páginas lógicas e os quadros possuem o mesmo tamanho, que é definido pela MMU.

Quando um programa é executado, um processo é criado e seu espaço de endereçamento é alocado na memória lógica. Esse espaço é dividido em partes denominadas páginas, no momento da execução do programa, o conteúdo referente a cada página é carregado para os quadros na memória física. Na memória lógica encontram-se os endereçamentos de cada parte do processo, quando uma parte é referenciada e a carga para a memória principal é feita, os dados dessa parte do processo referente ao endereço que são armazenados na memória principal.

Na paginação, o conteúdo referente a todas as páginas é copiado para a memória principal de maneira não contígua, ou seja, quando um processo está para ser executado, suas páginas são carregadas em quaisquer quadros de memória disponíveis (SILBERSCHATZ, GALVIN E GAGNE, 2015). Desta forma, há um melhor aproveitamento do espaço disponível na memória principal, evitando a obrigatoriedade do alocamento sequencial.

Ao realizar a carga do conteúdo relacionado a uma página lógica para a memória física, é registrado na tabela de páginas o número que representa a página e o correspondente ao quadro na memória física. A tabela de páginas auxilia a unidade de gerenciamento de memória (MMU) a mapear o endereçamento das páginas em relação aos seus respectivos quadros.

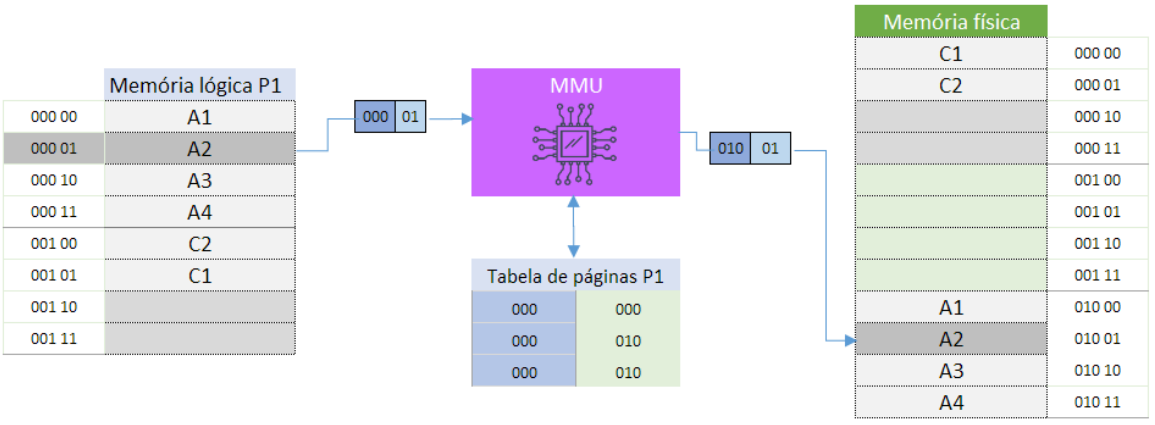

Figura 3. Mecanismo de paginação simples
Fonte: Adaptado de Figueiredo (2021)

A figura 3 apresenta um cenário em que o processo P1 tem o tamanho de 6 *bytes*, armazenados em duas páginas de 4 *bytes* de tamanho, sendo que na segunda página fica uma sobra de 2 *bytes*. O endereço de uma página é composto pela página mais o deslocamento dentro da página, que é a posição do *byte* dentro da página. A memória física possui 12 bytes de tamanho divididos em 3 quadros de 4 *bytes*. Na imagem pode-se observar a relação entre os endereços lógicos e físicos do byte A2, sendo que, quando a CPU solicita um endereço, a MMU recebe o endereço lógico de A2, sendo ele 000 01, e realiza a consulta a tabela de páginas do processo, identificando o seu endereço físico, que é o 010 01. No quadro (página física) 000 existe uma sobra de 2 bytes que não pode ser usada por outro processo, o que é denominado fragmentação interna.

Na paginação, para que o processo seja executado, todas as páginas deverão antes ser carregadas na memória física, mesmo que os dados não sejam utilizados pelo processo. No exemplo da figura 3, a memória física possui espaço suficiente para comportar as páginas dos processos, porém, em sistemas multiprogramáveis, é comum que processos grandes possam exigir mais espaço da memória do que está disponível, assim, pode ocorrer de ao tentar realizar a carga da página, não haja espaço suficiente na memória. Uma forma de amenizar esse problema é utilizar a paginação por demanda, que associa os conceitos de paginação ao da memória virtual, criando um mecanismo de gerência de memória que permite que não seja necessário realizar a carga de todas as páginas para a memória principal no momento da execução do programa.

2.4 Paginação por Demanda

Na paginação por demanda, que é baseada na paginação simples, tanto a memória lógica quanto a memória física são divididas em blocos de tamanhos fixos, utilizando a nomenclatura de páginas para os blocos da memória lógica e de quadros para os da memória física. Assim como na Paginação, ao carregar páginas da memória lógica para os quadros na memória física, o processo é feito de forma não contígua, ou seja, as páginas são carregadas de acordo com a disponibilidade de quadros na memória evitando a necessidade de serem armazenadas de forma sequencial. O armazenamento não contíguo pode gerar endereços lógicos e físicos que são diferentes entre si, para evitar que um processo seja referenciado em um endereçamento incorreto, é utilizada a tabela de páginas, que auxilia o mapeamento de endereços das páginas feitos pela MMU, associando os dois endereços.

A principal diferença entre os mecanismos é que, na paginação por demanda, as páginas são carregadas somente quando são necessárias durante a execução do programa (SILBERSCHATZ, GALVIN E GAGNE, 2015). Sendo assim, os conteúdos das demais páginas ficam armazenados no disco e somente são carregados para a memória principal quando são referenciados. Para tanto, a tabela de páginas mantém, além da identificação das páginas lógicas e dos quadros físicos, um *bit* de referência associado a cada página, válido (v) ou inválido (i), que permite identificar se aquela página está carregada na memória física quando estiver demarcado com V ou que a página possa não estar carregada ou ainda fora da área de endereçamento do processo quando o *bit* for i.

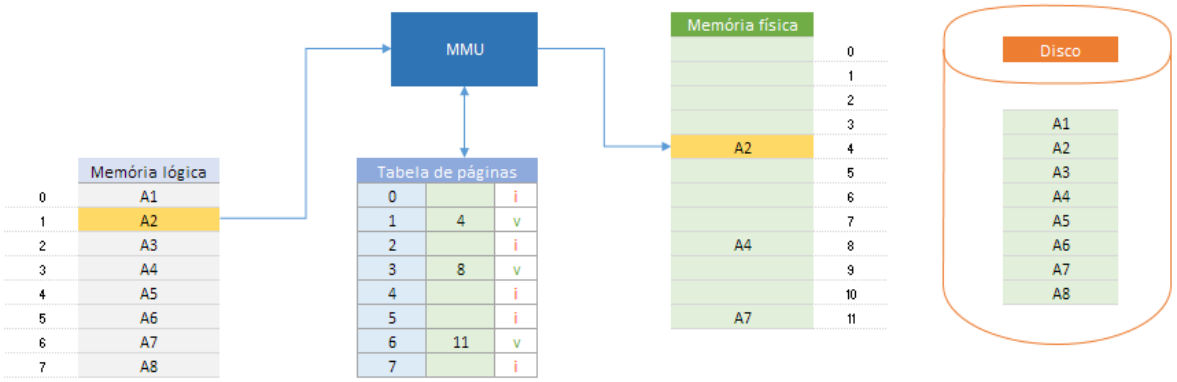

Figura 4. Mecanismo de paginação por demanda

A figura 4 apresenta o modelo do mecanismo de paginação por demanda onde a memória lógica possui 8 páginas e a memória física contém 12 quadros. No cenário apresentado na imagem 4, as páginas A2, A4 e A7 estão carregadas na memória física, sendo possível observar na tabela de páginas o endereço equivalente de cada uma das três páginas dos seus quadros: a página A2 está no quadro 4 da memória física, enquanto as páginas A4 e A7 encontram-se nos quadros 8 e 11, respectivamente. Na tabela de páginas, também é possível observar o *bit* de referência válido (v), indicando que as páginas estão carregadas na memória física.

Destacada em amarelo, a imagem 4 apresenta um fluxo onde a página foi referenciada, a MMU realiza a consulta à tabela de páginas e identifica através do *bit* de referência que ela está carregada na memória física, assim, ela realiza a consulta na memória física considerando o endereço mapeado na tabela de páginas. Os demais espaços da tabela de páginas estão marcados com o *bit* inválido (i), indicando que as páginas não foram carregadas para a

memória física, de forma que, em uma possível demanda futura, as páginas deverão ser carregadas da memória virtual, representada pela tabela Disco na figura 4, para a memória principal.

Segundo Oliveira et al. (2010), na paginação por demanda quando um processo é referenciado e está marcado na tabela de páginas com o *bit* válido (v), o acesso a ele se dá como na paginação simples. Porém, se o *bit* indicar inválido (i), a unidade de gerência de memória (MMU) irá gerar uma interrupção, informando ao sistema operacional que a página precisa ser carregada da memória secundária (disco) para a memória principal.

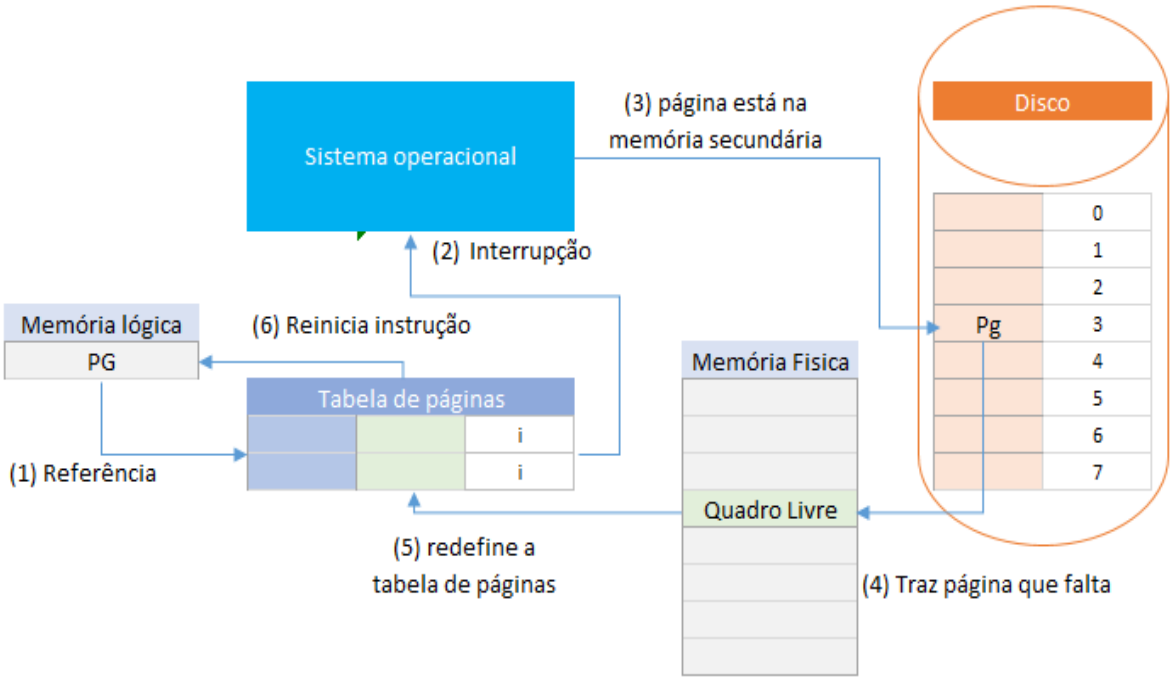

Figura 5. Etapas da paginação por demanda
Fonte: Adaptada de Silberschatz, Galvin e Gagne (2015)

A figura 5 apresenta as etapas da paginação por demanda quando uma página é referenciada e está marcada com o *bit* de inválida (i) na tabela de páginas (1), uma interrupção é gerada e o sistema operacional é acionado (2) para fazer a localização da página na memória secundária (3), o SO copia o conteúdo da página solicitada do disco para a memória física (4), a tabela de páginas é atualizada com uma nova entrada de endereço e o *bit* de referência passa a ser válido (v) (5) e, por fim, a instrução é reiniciada.

Ao executar múltiplos programas, é possível acontecer que todos os quadros da memória física estejam ocupados quando for realizada a tentativa de carga. Para contornar a situação, foram criados os algoritmos de substituição de páginas.

2.5 Algoritmos de Substituição de Páginas

A substituição de páginas é um recurso usado pelo SO, que determina, através de um conjunto de regras pré-estabelecidas, qual página será removida da memória principal para a memória virtual e liberará espaço para adicionar a nova página em seu lugar, causando o menor impacto possível ao usuário durante a execução dos programas. A página que atende aos critérios para a remoção da memória física para a memória virtual é chamada de página vítima. No momento em que ocorre a falta de páginas, fica a cargo do sistema operacional escolher a página vítima e realizar a troca pela nova página referenciada.

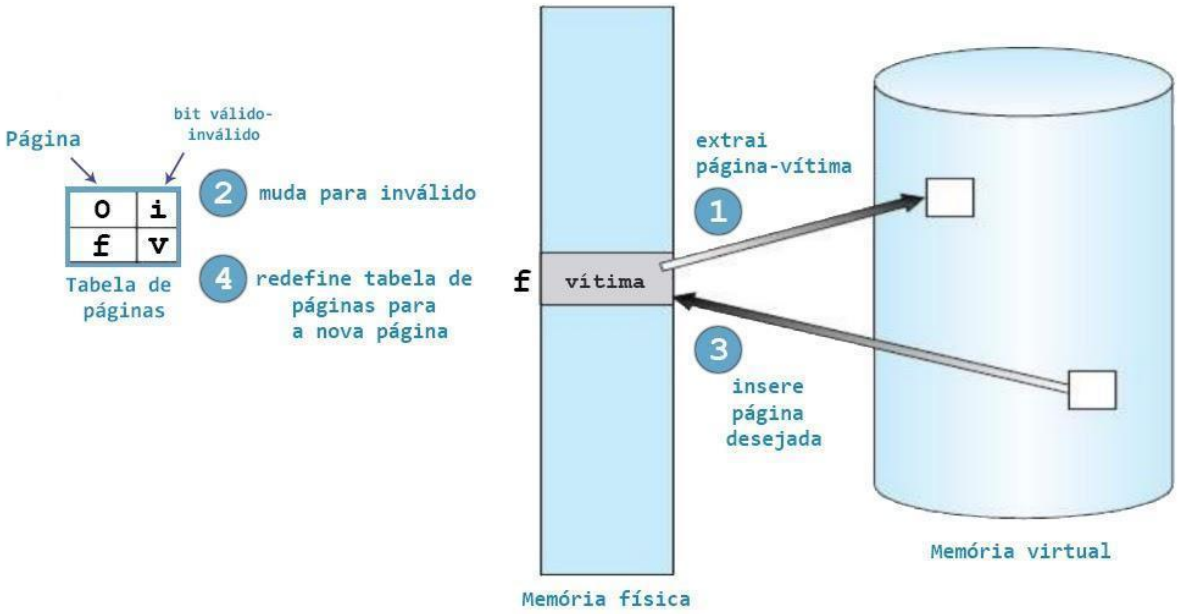

Figura 6. Etapas do mecanismo de substituição de páginas
Fonte: Adaptada de Silberschatz, Galvin e Gagne (2015)

A figura 6 apresenta as etapas da substituição de páginas, onde a página vítima é extraída na primeira ação (1) e seu conteúdo é armazenado novamente no disco, em seguida a tabela de páginas é atualizada e o *bit* de verificação (ou validade) passa a ter o valor de inválido (2). Na terceira etapa, a página solicitada é localizada no disco e alocada na memória física (3) e, por fim, a tabela de páginas é atualizada e o *bit* de verificação é alterado para válido. Durante o processo de substituição de página, o processador fica livre para a execução de um outro processo enquanto a página é movida da memória virtual para a física.

A transferência de páginas da memória para o disco e vice-versa pode se mostrar um processo oneroso, no exemplo da figura 6, o processador referenciou uma página, e para que ela fosse acessada foram feitas duas cópias de conteúdos de páginas. Quando se leva em consideração que vários processos podem estar sendo executados simultaneamente, essa duplicidade pode aumentar ou criar um atraso dos processos em tempo de execução e seria interessante que fosse feita a cópia de páginas que sofreram efetivamente uma alteração. Além disso, existem situações em que é preferível que determinadas páginas não sejam escolhidas como vítimas.

Uma das maneiras encontradas para resolver esses problemas foi adicionar *bit* auxiliares que são usados para indicar a situação das páginas. Os *bits* auxiliares mais utilizados são:

- *bit* de sujeira ou modificação: esse *bit* recebe um valor válido se a página for modificada. Com isso, é possível definir se a página precisará ser reescrita na memória virtual caso seja escolhida como vítima;
- *bit* de referência: recebe um valor válido sempre que a página for lida ou alterada, a cada determinado período de tempo as páginas são verificadas, se não houver alteração o seu valor é atualizado na tabela de página e passa a ser zero. O *bit* de referência ajuda a entender quais são as páginas que estão sempre em uso e, provavelmente, serão usadas em breve, no próximo ciclo de verificação essas páginas devem ter prioridade em relação às páginas não utilizadas quando o algoritmo de substituição estiver a procura de uma página vítima;
- *bit* de tranca: é um *bit* de segurança que impede que os algoritmos de substituição escolham como vítima a página que tiverem esse *bit* com valor válido. Por exemplo, é ativado em páginas em uso por processos, quando ele é encerrado seu valor passa a ser zero novamente.

Os algoritmos de substituição utilizam os *bits* auxiliares de forma individual ou combinada para a melhor escolha de páginas vítimas.

Na literatura é possível encontrar diversos algoritmos de substituição de páginas que utilizam diferentes critérios para definir as regras para a escolha da página vítima:

- FIFO: é um anagrama para *First In, First Out* (primeiro a entrar, primeiro a sair), é o algoritmo mais simples de se implementar, nele as páginas são organizadas em uma fila definida pelo tempo de carga, da primeira página carregada para a memória até a mais recente, a página mais antiga é selecionada para ser a vítima, considerando que a página que está a mais tempo na memória provavelmente não será referenciada em breve;
- ÓTIMO: considera que a página com o período mais longo de tempo sem ser acessada deverá ser removida para dar lugar a nova página. Em teoria é o melhor mecanismo a ser utilizado por possibilitar a menor ocorrência de erros ao mover as páginas, evitando a interrupção por falta de página por mais tempo. Na prática, o algoritmo ótimo de substituição de páginas é difícil de implementar porque requer o conhecimento antecipado da sequência de referência (SILBERSCHATZ, GALVIN E GAGNE, 2015), entretanto, esse algoritmo é comumente utilizado na comparação de desempenho de outros algoritmos.
- LRU: substituição de páginas usadas menos recentemente (*Least Recently Used*) é possivelmente o algoritmo implementável que mais se aproxima do ÓTIMO. O algoritmo LRU considera que páginas que foram recentemente referenciadas com frequência tendem a continuar sendo referenciadas frequentemente. Segundo Tanenbaum e Bos (2016), apesar de ser teoricamente realizável, para que o algoritmo seja implementado seria necessário criar uma lista encadeada de todas as páginas da memória, organizando-a de forma que as páginas mais referenciadas fossem movidas para o início da lista mas para que isso aconteça, seria necessário realizar uma atualização na lista sempre que uma referência a memória fosse feita, tornando o mecanismo oneroso em termos de tempo de execução;
- Histórico de *bit* de referência: este algoritmo se assemelha ao LRU ao considerar a página com menos acesso frequente como a candidata ideal para página vítima. Para tanto é guardado um histórico de *bit* de referência, dessa forma é possível identificar a página que está a mais tempo sem ser referenciada;
- Segunda Chance: é uma melhoria do algoritmo FIFO que verifica o *bit* de referência antes de realizar a substituição da página. Percorrendo todas as páginas de forma circular em uma fila ordenada por tempo de carga na memória, o algoritmo permite

que páginas que foram acessadas recentemente recebam uma segunda chance ao ser avaliada para ser a página vítima.

A versão atual do OSLive oferece o algoritmo FIFO como opção de algoritmo de substituição no módulo de simulação de paginação por demanda. Este trabalho tem como objetivo acrescentar ao módulo o algoritmo de Segunda Chance como alternativa de algoritmo de substituição, portanto, a seção seguinte abordará de forma mais detalhada os conceitos de ambos algoritmos.

2.6 Algoritmo FIFO

O algoritmo de substituição de página mais simples de se implementar é o FIFO, primeiro a entrar, primeiro a sair (*first in, first out* - FIFO). O sistema operacional mantém uma lista de todas as páginas atualmente na memória, com a chegada mais recente no fim e a mais antiga na frente. Em uma falta de página, a página da frente é removida e a nova página acrescentada ao fim da lista (TANENBAUM E BOS, 2016). A premissa é a de que uma página que foi carregada a mais tempo não será referenciada em breve, ela é removida e pode dar espaço a uma página mais recente.

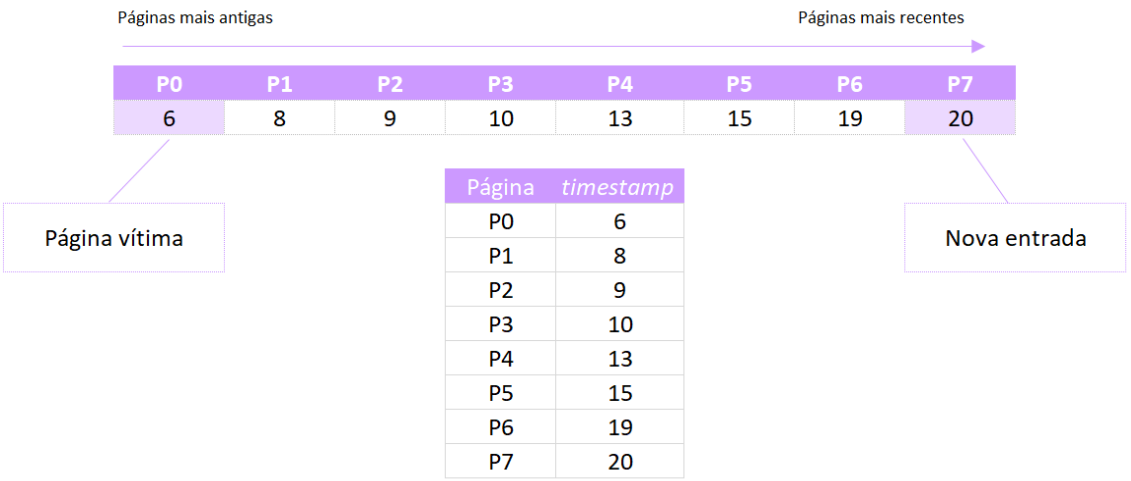

Figura 7. Algoritmo de substituição de páginas FIFO, modelo conceitual

A figura 7 apresenta o modelo da fila estabelecida pelo algoritmo FIFO, nela os processos de P0 à P7 estão organizados por ordem do momento no qual houve a carga (*Timestamp*), a página P7 é a que sofreu a carga mais recente, por isso está no final da fila, e a primeira página carregada foi a P0 que possui o menor *timestamp* indicado, por isso está no

início da fila. Em uma falta de páginas, a página escolhida como vítima seria a P0, por ser a página carregada a mais tempo.

O algoritmo FIFO considera somente o tempo em que houve a carga para definir a prioridade de escolha de uma página vítima, considerando que uma página que foi carregada a menos tempo tem mais chance de ser referenciada em breve. Entretanto, uma página que foi carregada a mais tempo em relação às outras páginas pode estar sendo consultada constantemente, por exemplo, assim, não seria interessante que ela fosse escolhida como vítima, mesmo estando no início da fila. Outros algoritmos de substituição de páginas, como Segunda Chance, utilizam o FIFO como base, mas somam a ela outros critérios que ajudam a melhorar seu desempenho.

2.7 Algoritmo Segunda Chance

Segundo Tanenbaum e Bos (2016), o algoritmo de segunda chance implementa uma versão melhorada do algoritmo FIFO, pois busca evitar que páginas que são constantemente referenciadas sejam substituídas. Neste modelo, quando o algoritmo de substituição de páginas inicia o processo de escolha da página vítima, antes de selecionar a primeira página carregada, ou seja, aquela que está a mais tempo na memória, verifica o *bit* de referência. Se o *bit* de referência for igual a 0, indicando que a página não foi referenciada recentemente, ela é substituída, se for igual a 1, indicando que ela foi acionada recentemente, é dada uma segunda chance à página, seu *bit* é alterado para 0 e o algoritmo segue para a próxima página.

Uma das maneiras de implementar o algoritmo de segunda chance é criar uma fila circular, de forma que o algoritmo percorre todas as páginas a partir da primeira página FIFO, até encontrar a primeira ocorrência de uma página com o *bit* de referência com valor igual a zero. Conforme ele avança, vai zerando os *bits* de referência. Uma vez que uma página vítima seja encontrada, ela é substituída e a nova página é inserida na fila circular nessa posição (SILBERSCHATZ, GALVIN E GAGNE, 2015).

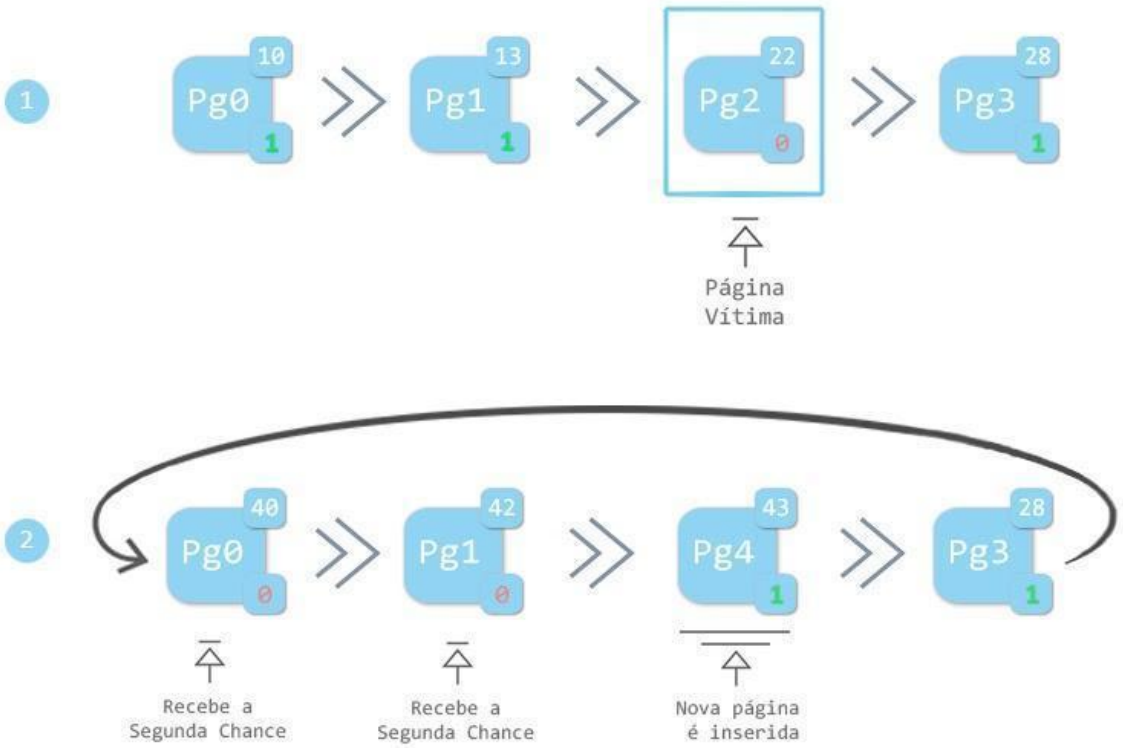

Figura 8. Localizando página vítima com algoritmo de Segunda Chance

Fonte: Adaptada de Silberschatz, Galvin e Gagne (2015)

A figura 8 apresenta uma lista contendo quatro páginas - Pg0, Pg1, Pg2 e Pg3 - mostrando, para cada uma, o nome página, o tempo de carga da página (*timestamp)* no canto superior direito, e o *bit* de referência no canto inferior direito. A seta indica a movimentação circular que o algoritmo realiza ao percorrer a fila, avançado para a próxima página até encontrar a página vítima, que é a primeira com o *bit* de referência igual a 0 (zero), indicando que não foi referenciada recentemente.

O exemplo da figura 8 apresenta a fila de páginas carregadas na memória antes da escolha da página vítima (1), organizada de acordo com seu tempo de carga, isso é, a página Pg0, por exemplo, é a primeira página carregada, indicada por seu *timestamp* 10, menor que os das páginas seguintes. A página Pg0 e a página Pg1 possuem o *bit* de referência 1, indicando que foram referenciadas recentemente e receberão uma segunda chance. Assim, o algoritmo percorrerá a fila até chegar a Pg2, que é a página mais antiga com o *bit* de referência com valor 0.

O algoritmo, ao passar pelas páginas Pg0 e Pg1, altera o status do seu *bit* de referência e seu *timestamp.* Já ao passar na página Pg2 e verificar que o *bit* de referência é 0, indicando

que é a página mais antiga e que não foi acessada na última referência feita a memória, escolhe-a como página vítima. A nova página, que precisou ser carregada na memória, Pg4, é copiada no lugar de Pg2, com o *bit* de referência igual a 1 e o *timestamp* com o tempo da sua carga.

3 METODOLOGIA

Esta seção apresenta e descreve as ferramentas que foram utilizadas no desenvolvimento e implementação do módulo proposto neste trabalho, além do desenho do estudo, ilustrando as etapas desenvolvidas no projeto.

3.1 DESENHO DO ESTUDO

A figura 9 ilustra as etapas de desenvolvimento do módulo de simulação da paginação por demanda com algoritmo de substituição de página Segunda Chance.

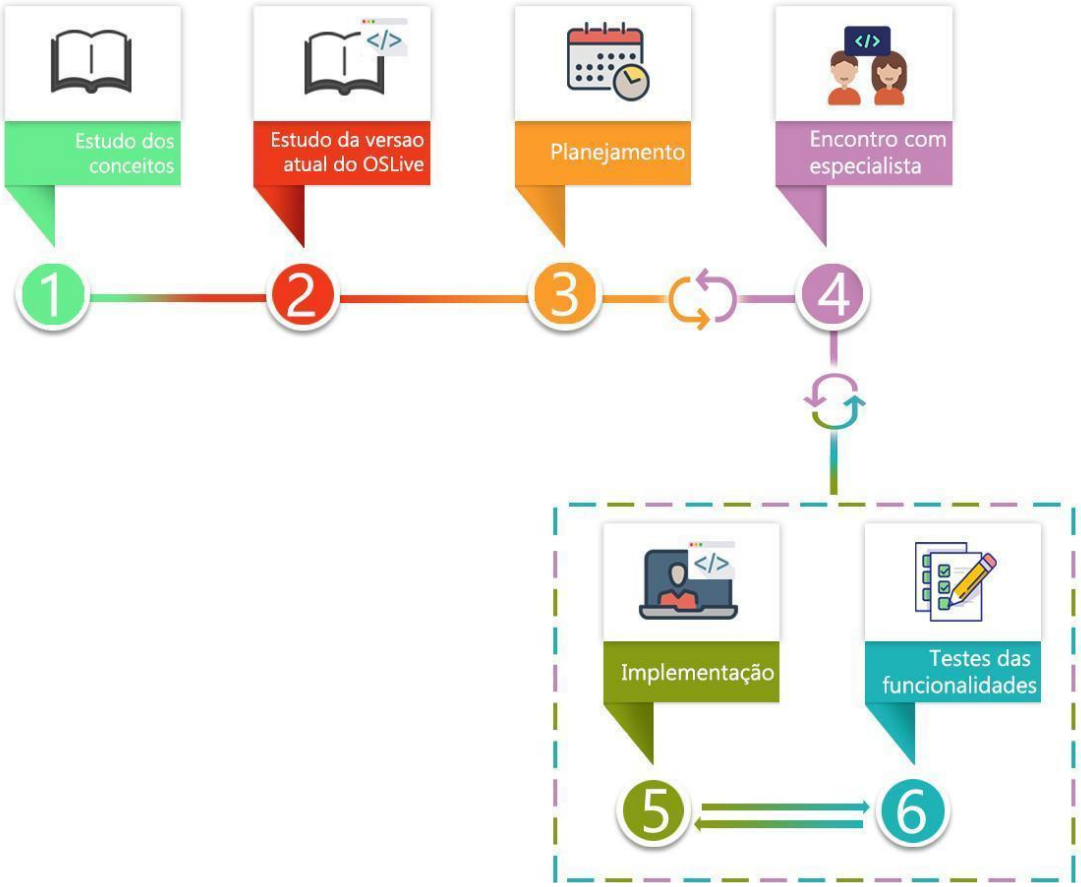

Figura 9. Etapas do desenvolvimento do projeto
Fonte: Elaborada pelo autor

A etapa inicial (1) consistiu na realização da pesquisa em material didático, artigos e literatura para construir o conhecimento necessário e elaborar o referencial teórico. Na etapa seguinte (2), foi feito o estudo da versão atual do módulo de simulação de paginação por demanda com algoritmo de substituição de página FIFO, a fim de entender os recursos oferecidos e adquirir familiaridade com a arquitetura das funcionalidades. Por fim, foi feito o estudo das linguagens de programação e tecnologias utilizadas na implementação da versão atual do OSLive.

Na terceira etapa, planejamento (3), foram reunidas e estudadas as representações gráficas encontradas nos livros didáticos, em materiais disponíveis na Internet, pesquisadas na fase de estudo do algoritmo de substituição de páginas Segunda Chance, além do material didático utilizado nos cursos do Departamento de Computação do CEULP. A partir disso, foi definida a melhor forma de criar uma representação gráfica didática para demonstrar o funcionamento do algoritmo em tempo de execução. A etapa de planejamento aconteceu em paralelo à etapa seguinte, encontro com especialista (4), em que foram apresentadas as propostas de abordagem do conceito, possíveis formas de implementação do algoritmo e discutido o seu valor didático juntamente a professora da disciplina de Sistemas Operacionais do CEULP.

A fase encontro com especialista (4) também norteou, do início ao fim, o ciclo que foi realizado nas duas últimas fases, que ocorreram concomitantemente: implementação (5), codificação dos serviços, classes e funções que compõem a funcionalidade da simulação do algoritmo de substituição de páginas Segunda Chance e testes das funcionalidades (6), que contempla o conjunto de testes de funcionalidades planejadas para certificar que os objetivos foram alcançados. Os testes, que foram acompanhados pela especialista da área, validaram as funcionalidades conforme eram implementadas.

3.2 Ferramentas utilizadas

Para o desenvolvimento do trabalho, foram utilizadas as tecnologias que compõem a arquitetura do OSLive, são elas:

- Angular: é uma plataforma de desenvolvimento construída em TypeScript, um *framework* baseado em componentes feito para aplicações Web. (Angular, 2021). O

Angular foi utilizado neste projeto no intermédio entre as regras de negócio e a interface do usuário.

- Notify.Js: é um *plugin* em linguagem jQuery que provê notificações que são simples porém completamente personalizáveis (Notify.Js, 2021). O Notify foi utilizado para adicionar notificações aos eventos da simulação, permitindo ao aluno acompanhar as mudanças que ocorrem na interface decorrente das suas interações.
- JavaScript: é uma linguagem leve, interpretada e baseada em objetos com funções de primeira classe, mais conhecida como a linguagem de script para páginas Web (Mozilla, 2021). JavaScript foi utilizada nesse projeto na criação das regras e modelo de negócio do sistema.
- Bootstrap: é um *framework opensource* criado em 2010 pela equipe da rede social Twitter, desenvolvida em para ser um guia de estilos para interfaces de aplicações *web* (Bootstrap, 2021). O bootstrap foi utilizado na composição da interface gráfica da aplicação, escolhida por possibilitar uma aparência mais objetiva e amigável ao usuário no uso do módulo da simulação.

4 RESULTADOS E DISCUSSÃO

Nesta seção é apresentada a inclusão do algoritmo de Segunda Chance no módulo de paginação por demanda com substituição de página do OSLive e as alterações aplicadas à interface do módulo, para possibilitar a simulação do algoritmo. Espera-se que o acréscimo do algoritmo Segunda Chance ao módulo de simulação, proporcione ao aluno desenvolver uma melhor compreensão dos conceitos discutidos em aula através do auxílio que a ferramenta proporciona.

A plataforma está disponível *online* e pode ser acessada via navegador, dispensando a necessidade de instalação local e permitindo a isenção do usuário quanto às preocupações em relação aos requisitos mínimos da máquina, exigindo apenas uma conexão com a internet.

As seções seguintes abordarão a arquitetura do software, apresentando a estrutura adotada e as tecnologias utilizada em cada camada da aplicação; a versão atual da interface de interação com o usuário com exemplos de simulação da paginação por demanda utilizando os recursos atuais; apresentação da nova versão da aplicação contendo a adição do algoritmo de Segunda Chance, as mudanças realizadas nessa nova versão do módulo e a descrição de simulação de exemplo.

4.1 Arquitetura do Software

Para ilustrar a organização da aplicação quanto a sua estrutura, a Figura 10 representa o padrão de arquitetura utilizado na implementação da aplicação.

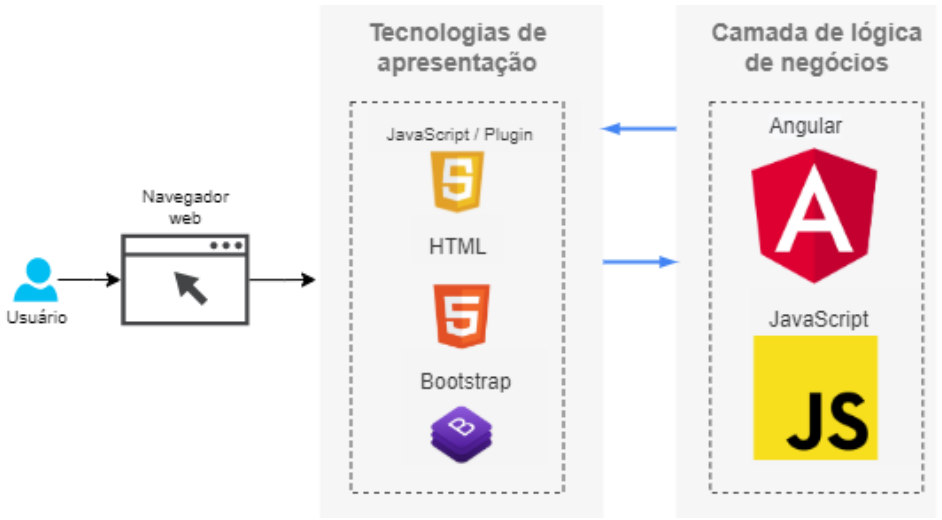

Figura 10. Arquitetura do software

A arquitetura é dividida em duas camadas, a primeira Tecnologia de apresentação, apresentam os resultados processados na camada de lógica de negócios e servem ao navegador *web* toda informação para a apresentação da interface e funcionalidades da aplicação. É através do navegador que o aluno interage com a aplicação, inserindo nos campos de formulários as informações necessárias para a geração da simulação de acordo com os cenários escolhidos. Essa camada foi construída utilizando as tecnologias: HTML na construção dos elementos da página como formulários, botões e *layout*; Bootstrap para a personalização da identidade visual da aplicação; e JavaScript para criação das janelas de notificações interativas que ocorrem durante a simulação.

A camada de lógica de negócios é uma camada que possui funções e métodos advindos das linguagens de programação utilizadas na implementação do módulo, criadas com o intuito de interpretar e servir a entrada da camada Tecnologias de apresentação. Nela estão as funções, classes e demais mecanismos que interpretam, processam e geram as informações necessárias que são entregues a camada Tecnologias de apresentação para a geração das simulações. A camada utiliza o *framwork* Angular e a linguagem JavaScript para compor os métodos que permitem gerenciar esse fluxo.

4.2 Versão Atual do Módulo de Paginação por Demanda

4.2.1 Interface da aplicação

A apresentação visual da interface do módulo de paginação por demanda com substituição de páginas segue as escolhas de *design* já utilizadas nos outros módulos do OSLive. Os campos de interação e as ações que resultam em elementos gráficos dispostos em tela são padronizados e buscam proporcionar uma interação intuitiva e amigável ao usuário.

A figura a seguir apresenta a tela inicial da interface disponibilizada na versão atual do OSLive. Na imagem, pode-se observar a tela inicial da simulação antes de qualquer interação com o usuário.

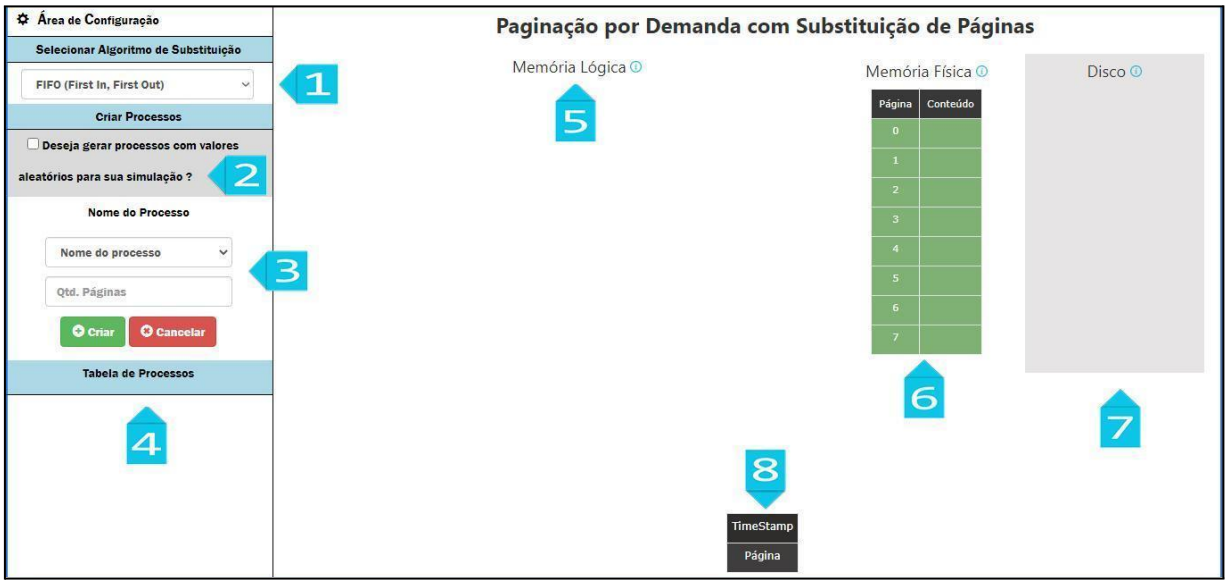

Figura 11. Tela inicial do módulo

Na imagem da tela inicial, figura 11, foram adicionadas marcações numerando cada componente da interface, que são descritas a seguir:

1) Menu de seleção do algoritmo que será utilizado como mecanismo de substituição de páginas quando a memória estiver cheia. Atualmente, o módulo possui o algoritmo FIFO como única opção de escolha, mas na nova versão foi inserida a opção Segunda Chance;

2) Permite ao usuário escolher, caso não queira criar uma simulação específica, que a aplicação crie de forma randomizada e automática, quatro processos. Cada processo criado pode conter de uma a quatro páginas cada, aleatoriamente;
3) Caso não queira criar processos com valores aleatórios, o usuário poderá escolher através do formulário o nome, com as opções fixas A, B, C ou D, e a quantidade de páginas de cada processo, limitadas ao intervalo de 1 a 4;
4) Nessa área é apresentada uma tabela contendo a lista dos processos criados, apresentando o nome e quantidade de páginas de cada um deles. Nela, para cada processo da lista, é disponibilizado um botão que permite a exclusão do processo;
5) Essa área exibe a representação do processo na memória lógica dos processos, um de cada vez, de acordo com a seleção do usuário. Após a criação dos processos, apresenta as páginas lógicas e a tabela de páginas do processo selecionado. A tabela de páginas informa que páginas do processo estão ou não carregadas na memória física;
6) Representação da memória física, contendo 8 *bytes* de espaço disponíveis. Nela, a primeira coluna mostra o número que representa o endereço do espaço de memória física, enquanto a segunda coluna apresenta o conteúdo armazenado, quando está em uso, ou fica vazia, quando o espaço está disponível para uso.
7) Representa o disco utilizado como memória virtual. Essa área ressalta visualmente o conceito da paginação pode demanda, quando exibe que todas as páginas lógicas foram criadas e armazenadas na memória virtual (disco), mas que somente serão carregadas para a memória principal sob demanda. Quando um processo é excluído, ele também é retirado dessa área;
8) Nessa área é representada a execução do algoritmo de substituição de páginas usado para escolher qual página deverá ser substituída quando a memória principal (Item 7) estiver cheia. A tabela apresenta a ordem de carga das páginas, sendo que a linha *timestamp* representa o momento de carga da página, enquanto a linha Página mostra o nome da página carregada, dessa forma é exibida uma fila ordenada pelo tempo de carga de cada página, da primeira página carregada até a página mais recente. Essa área exibirá informações diferentes de acordo com o algoritmo de substituição de páginas selecionado pelo usuário, e a Figura 11 apresenta a área ao selecionar o algoritmo FIFO, disponível na versão atual. Na seção 4.2.3 são apresentadas as modificações desenvolvidas para o uso do algoritmo Segunda Chance acrescentada na nova versão.

A proposta dessa seção é apresentar a interface do módulo de paginação por demanda com substituição de páginas e os elementos interativos que a compõem, nas próximas seções serão apresentadas simulações realizadas e será possível observar o comportamento da interface em decorrência das situações simuladas.

4.2.2 Demonstração da simulação da Paginação por Demanda

Para exemplificar o funcionamento do módulo de simulação de paginação por demanda com substituição de páginas, foram gerados 4 processos, são eles: A com 4 páginas, B e C com 3 páginas cada e o processo D com 4 páginas. Após a criação dos processos a interface, com a opção Segunda Chance selecionado no seletor de algoritmo de substituição, deverá ser exibida conforme mostrado na imagem a seguir.

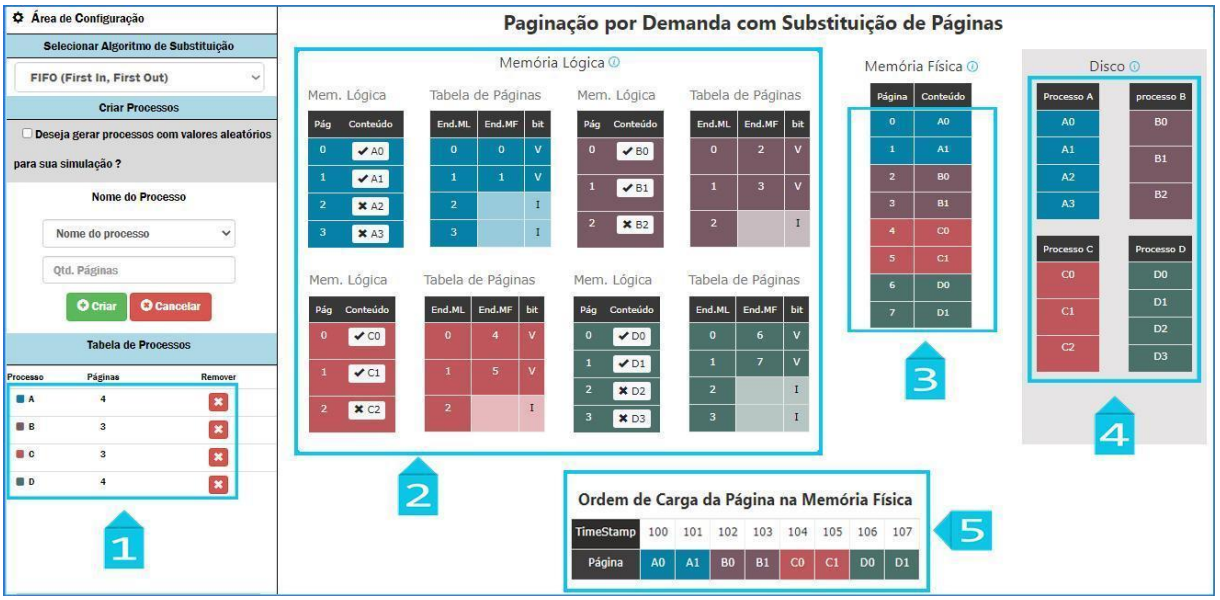

Figura 12. Simulação paginação por demanda

Na área 1, os processos que foram criados são exibidos juntamente com informações como a cor do processo e sua quantidade de páginas, cada processo também recebe um botão que permite sua exclusão caso o usuário deseje. Caso o usuário exclua o processo, suas páginas carregadas na memória física serão removidas (área 3) e seu processo removido do disco (área 4).

Na área 2, são exibidas as representações dos processos A, B, C e D através de duas tabelas, na tabela "Mem. Lógica" é mostrada a quantidade de páginas do processo e o nome atribuído a cada página. O campo onde é exibido o nome da página é interativo e tem sua

aparência alterada de acordo com o status da página, isso é, se a página está carregada na memória principal, um ✔ é mostrado ao lado esquerdo do nome dela, caso contrário, é exibido um **X** indicando que a página não está carregada.

O botão também permite, ao ser clicado, alterar o *status* da página do que ele está associado. Isso quer dizer que o usuário ao clicar no botão, a página que já estiver carregada na memória será retirada, caso contrário ela será carregada em um espaço disponível.

Além da informação visual dos ícones no nome da página, para cada tabela "Mem. Lógica" de um processo criado, há uma “Tabela de Páginas”, nela são exibidas informações mais detalhadas em relação a carga das páginas do processo. Por exemplo, seguindo as informações da Tabela de Páginas sobre o processo A, pode-se observar que a página 0 e 1 estão carregadas na memória e ambas recebem o valor V na coluna *bit* para indicar isso, enquanto na coluna end.MF, consta a posição em que aquela página está carregada na memória (item 3).

Por fim, na área 5 é mostrada a tabela da lista FIFO, pode-se observar as páginas que foram inseridas na memória e para cada uma delas foi atribuído um *timestamp* simbolizando o momento da carga desta página, iniciando em 100 o *timestamp* continua de forma incremental sempre que uma nova página for incluída na memória e adicionada à fila. Quando uma página é removida da memória, ela também é removida da fila.

Conceitualmente a paginação por demanda trabalha com o carregamento apenas de páginas necessárias para a execução do processo, seguindo essa premissa, duas abordagens podem ser empregadas quando um programa é executado: carregar as páginas iniciais do programa ou não carregar nenhuma página e aguardar serem referenciadas para realizar a carga. O módulo de simulação de paginação por demanda do OSLive trabalha com a primeira abordagem, realizando a carga de duas páginas do processo criado para a memória física.

Figura 13. Simulação da paginação por demanda

Na figura 13 pode-se observar os passos para realizar a simulação da paginação por demanda. Na área 1, o usuário ao clicar sobre o campo que apresenta a página B2 enviará a solicitação de inserção da página, com isso a Tabela de Páginas (área 2) será atualizada, adicionando o endereço da página na memória física e alterando o valor da coluna *bit* de I para V, indicando que a página foi carregada na memória, nesse caso no primeiro espaço disponível mostrado na área 3. Na área 4 pode-se observar como as tabelas ficarão após a ação do usuário.

4.2.3 Demonstração da Simulação da Paginação por Demanda com Substituição de página FIFO

Nesta seção é apresentada uma demonstração da paginação por demanda utilizando o algoritmo FIFO. O algoritmo segue a abordagem que considera que páginas que estão a mais tempo carregadas na memória deverão ser escolhidas como página vítimas, cedendo seu lugar a nova página. A fim de demonstrar o seu funcionamento, foi executada uma simulação para a qual foram criados os processos A, C e D, com 4 páginas cada, e o processo B com 3 páginas, que são apresentados na figura 14. Após o término da criação dos processos, a interface passou a apresentar informações como mostrada na Figura 14.

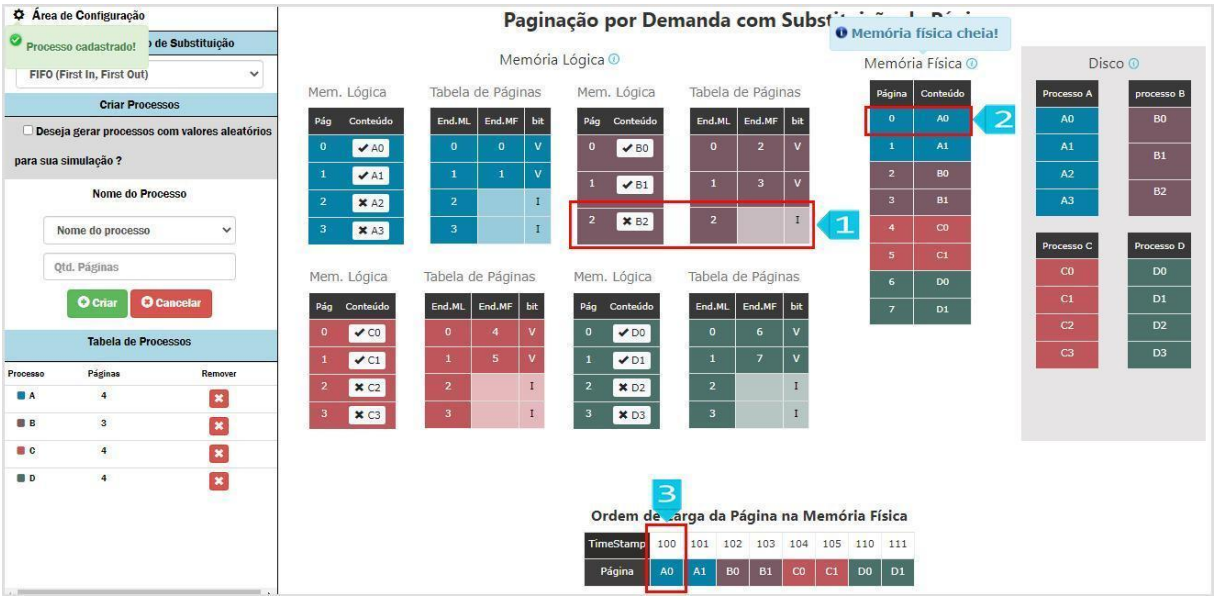

Figura 14. Simulação da substituição de página FIFO

Na figura 14, pode-se observar que a memória física está completamente ocupada, assim, uma notificação é exibida logo acima da tabela Memória Física para informar isso ao usuário. Com memória totalmente ocupada, novas cargas de páginas só serão possíveis após a remoção de uma página já carregada. Considere que o usuário gostaria de simular a carga da página B2, ao clicar no botão do nome da página (1), o algoritmo FIFO irá localizar a página com menor *timestamp* (que representa o tempo de carga), nessa simulação é a página A0 (3), e fará sua remoção tanto da memória física quanto da lista FIFO (3).

Após a remoção da página A0 da memória (2) e da lista FIFO (3) a página B2 é inserida na memória em seu lugar e a lista FIFO é atualizada, recebendo B2 ao seu final com o *timestamp* com valor 112. A situação é apresentada na Figura 15, na qual é possível observar as notificações geradas que informam ao usuário as alterações realizadas de acordo com o funcionamento do algoritmo.

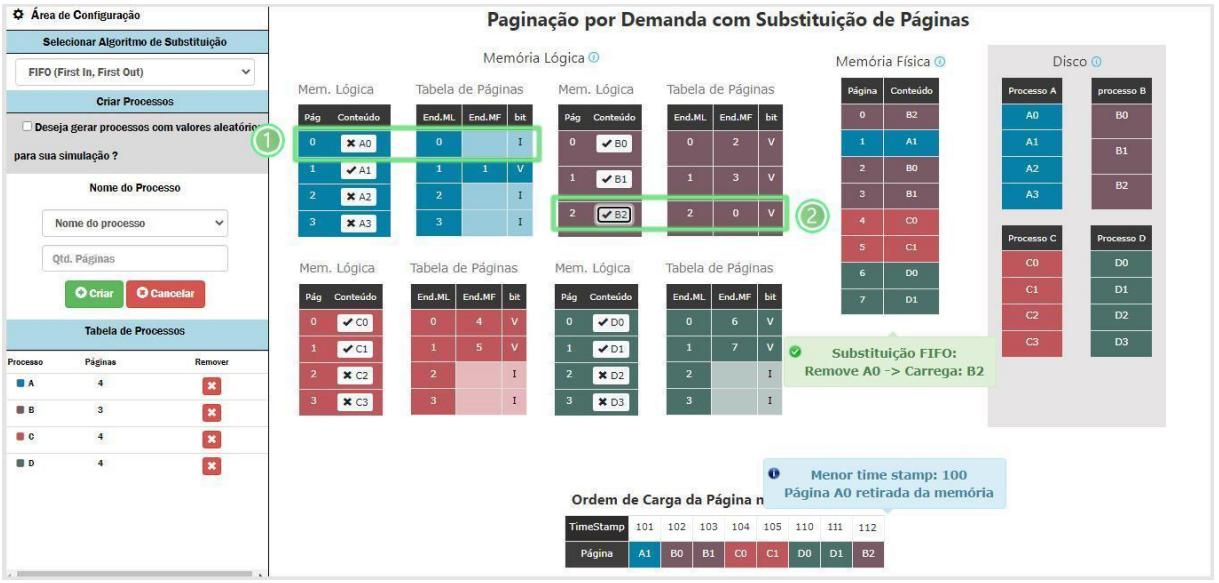

Figura 15. Resultado da simulação da substituição de página: FIFO

Na figura 15, também é possível notar que os processos A e B, sofreram alterações em suas tabelas de páginas, para a página A0 do processo A (Figura 15-1), o endereço da coluna End MF foi retirado e o seu *bit* verificador foi alterado de V para I, para B (Figura 15-2) a página B2 recebe o endereço 0 na coluna End MF, referente a sua posição na memória física e seu *bit* verificador recebe o valor V.

4.3 Nova Versão do Módulo de Simulação da Paginação por Demanda

A nova versão do módulo de paginação por demanda do OSLive mantém as funcionalidades atuais do módulo, apresentadas nas seções anteriores, e conta com o acréscimo do algoritmo de substituição de páginas Segunda Chance. Também, foram adicionadas à nova versão do módulo telas com informações sobre o algoritmo e/ou sobre a aplicação, que são acessadas através de botões presentes em várias áreas da interface.

Nas duas seções seguintes, serão apresentadas as alterações implementadas no módulo, destacando as mudanças ou inclusões feitas, que visam representar o funcionamento do algoritmo Segunda Chance da forma mais didática possível e, por fim, a simulação completa do funcionamento do mecanismo de substituição de páginas utilizando o algoritmo Segunda Chance.

4.3.1 Representação do Algoritmo Segunda Chance

Essa seção mostra em detalhes a representação escolhida para o algoritmo de substituição de páginas Segunda Chance, trançando um paralelo com a representação do algoritmo FIFO presente na versão atual do módulo.

A figura 16 (1) mostra a tabela em ordem de carga, utilizada na representação do algoritmo FIFO. Nela, são apresentadas duas linhas contendo as informações sobre a página carregada: *Timestamp* e Página (nome da página).

O algoritmo Segunda Chance faz o uso de *bit* de referência, que indica se uma página foi acessada após a última execução do algoritmo de substituição, uma página acessada recebe o valor 1 caso contrário ela permanece 0. Assim, permite que uma página não seja removida caso tenha sido acessada recentemente. Para implementar essa funcionalidade no módulo de simulação de paginação por demanda, a tabela que apresenta a ordem de carga das páginas recebeu duas linhas a mais, são elas: *bit* Ref, e Acessar PG, figura 16 (2).

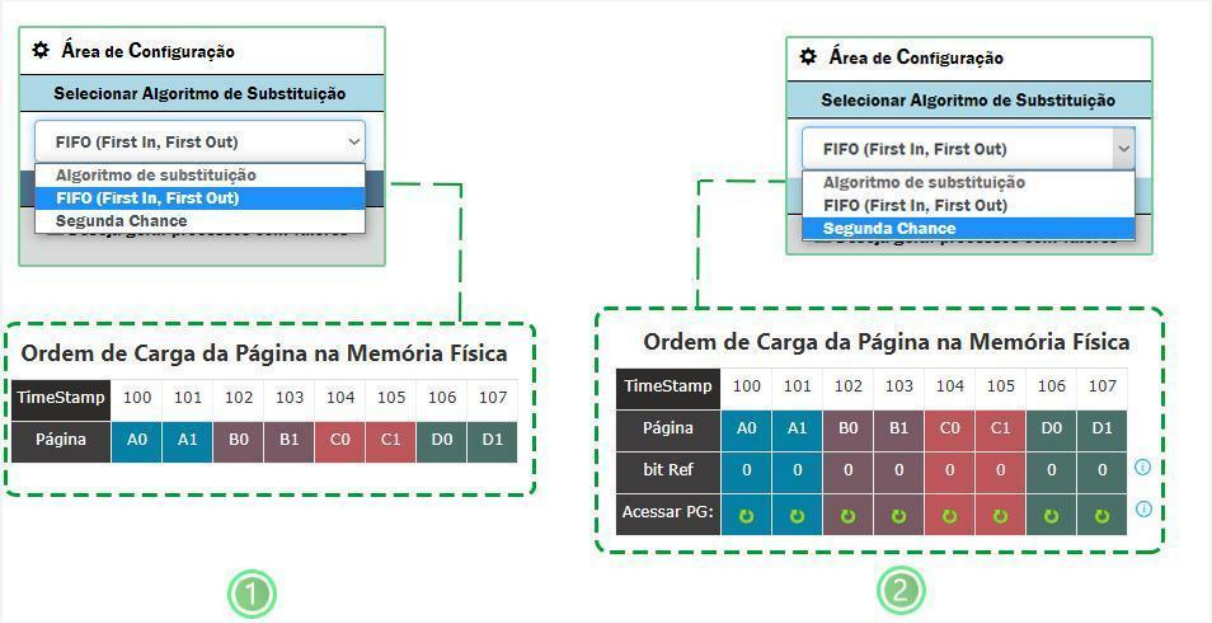

Figura 16. Tabela de ordem de carga de acordo com algoritmo de substituição de página

Na nova tabela de ordem de carga, disponível quando o usuário seleciona o algoritmo de Segunda Chance, é disponibilizada a funcionalidade na linha Acessar PG que permite simular um acesso à página ao clicar no botão referente a página, botão representado por uma seta circular na cor verde mostrado na imagem (figura 16 - 2), o usuário pode alternar o valor

da coluna "*bit* Ref" entre 0 e 1 conforme desejar, assim o usuário poderá criar situações e observar como o algoritmo de Segunda Chance irá lidar com ela.

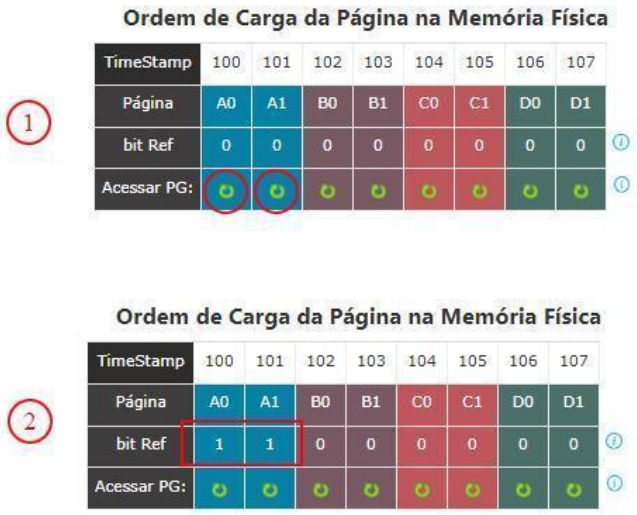

Figura 17. Estado do bit de referência

A Figura 17 apresenta o uso da funcionalidade que representa o acesso às páginas. Na área 1, as páginas A0 e A1 possuem ambas o bit de ref com o valor 0, assim como quando foram inseridas na memória. O usuário pode acionar o botão disponível na linha Acessar PG e o valor será alterado para 1, assim como mostrado na área 2. Dessa forma, o usuário consegue simular situações nas quais ele define qual página irá receber a segunda chance quando o algoritmo for acionado.

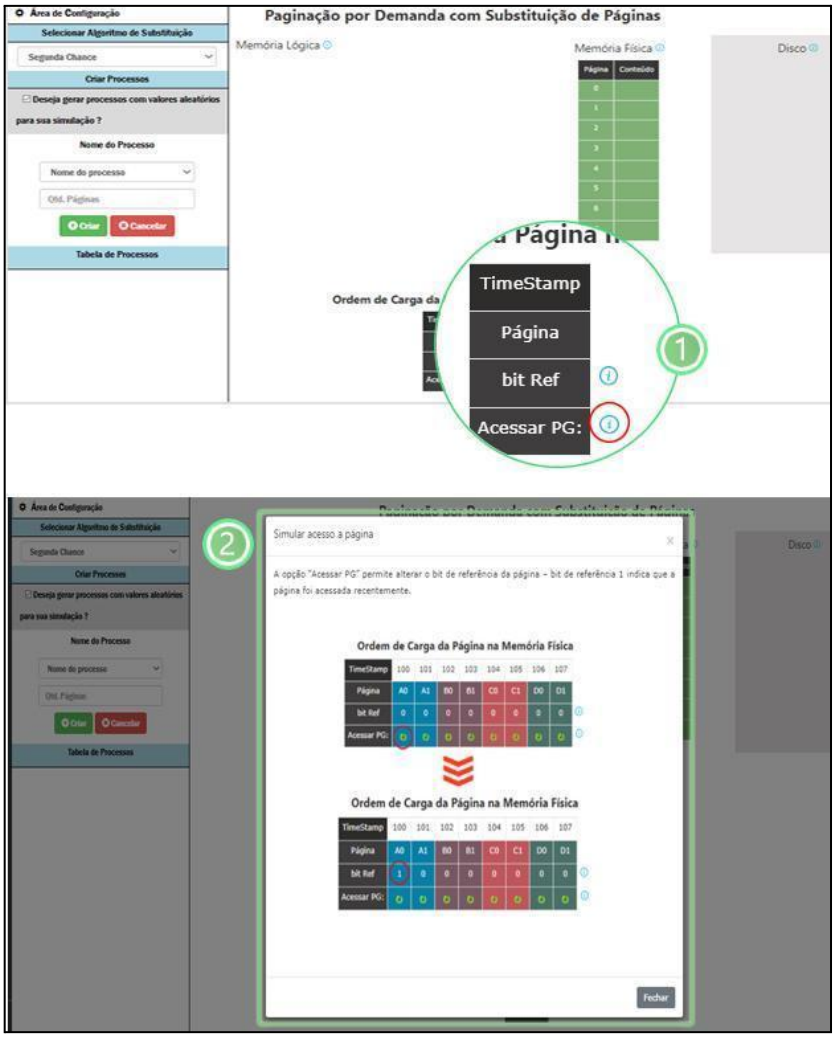

Figura 18. Modal de informações

Outra funcionalidade implementada na nova versão do módulo são as telas de informações adicionais (figura 18). O usuário pode selecionar um dos botões que aparecem ao lado dos elementos da interface, representado por um círculo com a letra i em seu interior (1), que será acionado um *modal* com informações sobre o conceito ou funcionalidade que o botão está atrelado. Na Figura 18 é mostrado o acionamento do *modal* após a seleção do botão ao lado da linha Acessar PG (1), em seguida o *modal* é carregado na tela (2).

4.3.2 Demonstração da Simulação da Paginação por Demanda com Substituição de página Segunda Chance

Essa seção apresenta a simulação completa do mecanismo de substituição de páginas utilizando o algoritmo Segunda chance na paginação por demanda. Para a escolha da página

vítima na substituição, o algoritmo Segunda Chance selecionará a página de menor *timestamp* com o *bit* de referência igual a 0.

Para simular o funcionamento da Paginação com Demanda com algoritmo Segunda Chance, foram criados os processos A, B e C, Figura 19 (1). As páginas A0 e A1 tiveram o valor do seu bit de referência alterado para 1 (3).

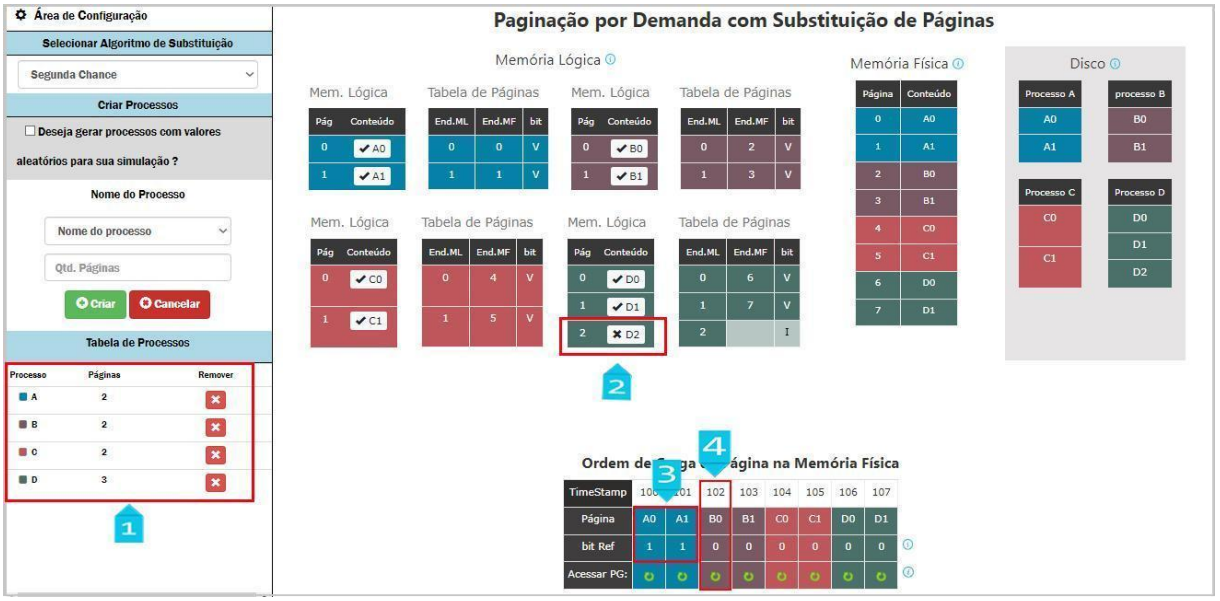

Figura 19. Simulação da substituição de página: Segunda Chance

Na sequência, a página D2 (2) foi escolhida para carga acionando o algoritmo de substituição, pois a memória está cheia e uma página deverá ser removida para que a memória possa comportar a nova página. A Figura 20 mostra o resultado da simulação, apresentando as mudanças ocorridas após o término do processo de carga de D2, apresentando os elementos de interatividade que foram exibidos durante a simulação.

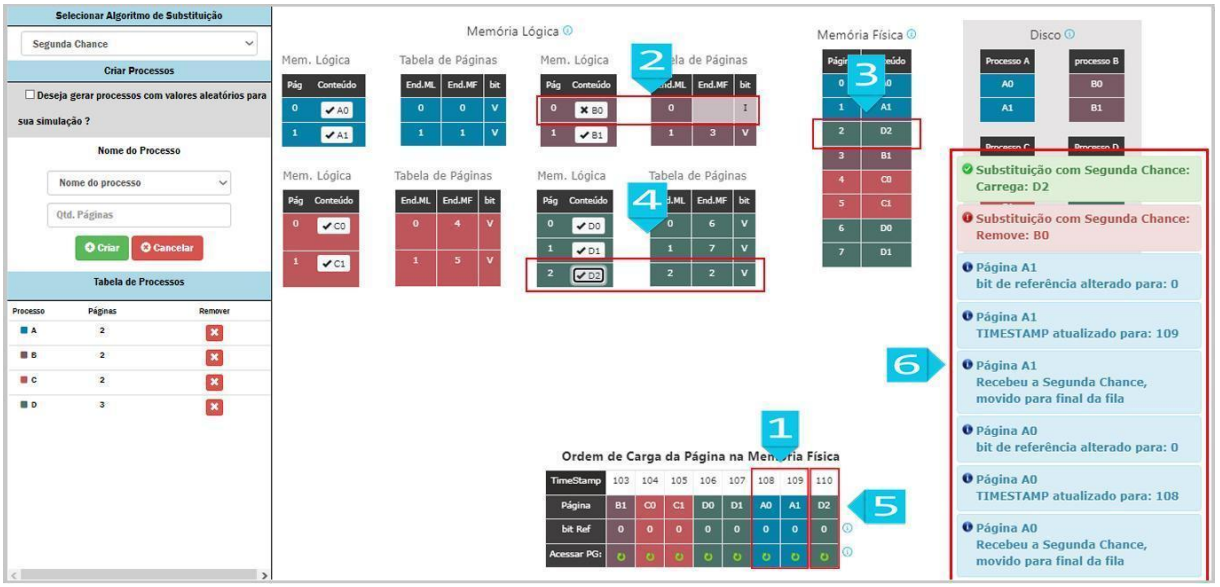

Figura 20. Resultado da simulação

Após a solicitação de carga da página D2 (Figura 20), as páginas A0 e A1, por terem *bits* de referência 1, receberam a segunda chance (1) e tiveram seu *timestamp* atualizado, respectivamente, para 108 e 109, e foram movidas para o final da fila da ordem de carga com o *bit* de referência agora zerado. A página B0 foi escolhida como página vítima, por ter o menor *timestamp* e estar com o *bit* de referência igual a 0 e foi removida da memória física. A tabela de páginas de B0 foi atualizada, tendo sido retirado o valor do campo End.MF, pois sua associação com a memória física não existe mais, e mudado seu bit de validade de V para I (2), indicando que não está carregada na memória física.

Com a remoção da página B0, foi liberado um espaço na memória física para inclusão da página D2 que foi inserida em seu lugar (3). A tabela de páginas de D2 foi atualizada (4), sendo que o endereço da memória física 2 foi inserido na coluna End.MF e seu bit de validade recebeu o valor V. Por fim, a página B0 foi adicionada ao final da fila de ordem de carga com o *timestamp* 110 (5). Vale ressaltar que todo o processo de substituição de página foi notificado no canto inferior direito da tela (6). Para cada página que recebe uma segunda chance e quando há substituição da página, a aplicação exibe informações que aparecem em *pop-ups* e ficam temporariamente na tela.

Conforme os exemplos apresentados, adicionar o algoritmo de substituição de páginas Segunda Chance ao módulo de simulação da paginação por demanda do OSLive possibilita ao aluno simular o mecanismo do algoritmo e comparar com a abordagem do outro algoritmo presente na versão atual da aplicação, FIFO. Assim, o aluno pode praticar de forma visual e interativa os conceitos estudados em sala de aula e reforçar o seu aprendizado.

5 CONSIDERAÇÕES FINAIS

Este trabalho teve como objetivo incluir o algoritmo de substituição de página Segunda Chance ao módulo de simulação de paginação por demanda com substituição de página do OSLive, plataforma utilizada como auxílio ao aprendizado na disciplina de Sistemas Operacionais. A adição do algoritmo Segunda Chance permite ao aluno acompanhar de forma interativa outro tipo de abordagem da substituição de página por demanda, por meio da simulação do uso do algoritmo, auxiliando-o a melhor compreender conceitos que são apresentados na disciplina e que não podiam ser simulados utilizando o algoritmo FIFO presente na versão atual.

O desenvolvimento desse projeto passou pela supervisão e orientação da professora da disciplina de Sistemas Operacionais dos cursos do departamento de computação do CEULP/ULBRA, que contribuiu indicando formas que permitiram o desenvolvimento da simulação de forma a alcançar o resultado esperado. Também se acredita que o uso da aplicação pela professora em ambiente de sala de aula, com retorno positivo dos alunos que utilizaram o módulo compensaria a falta de uma pesquisa sistematizada com documentação.

Para os trabalhos futuros sugere-se: inclusão de outros algoritmos de substituição de páginas existentes, exportar ou salvar simulações e aperfeiçoar a interface e animações utilizadas a fim de tornar o sistema ainda mais intuitivo.

Por fim, espera-se que o presente trabalho permita uma melhoria no módulo de simulação da paginação por demanda com substituição de páginas do OSLive e auxilie no ensino e aprendizagem dos conceitos estudados na disciplina de Sistemas Operacionais.

6 REFERÊNCIAS

Angular.io, **WHAT IS ANGULAR**, 2021. Disponível em: https://angular.io/guide/what-is-angular. Acesso em: 24, Nov. 2021.

BOOTSTRAP, **About bootstrap**, 2021. Disponível em: https://getbootstrap.com/docs/4.1/about/overview. Acessado em 24/11/2021

CUNHA, Guilherme B. PREUSS, Evandro. MACEDO, Ricardo T. Macedo. **Sistemas Operacionais.** 1. ed. – Santa Maria, RS : UFSM, NTE, 2017.

FIGUEIREDO, R. V. **OSLIVE: Desenvolvimento Do Módulo De Simulação Da Paginação Por Demanda Com Substituição De Páginas**, Projeto de pesquisa – Ciência da Computação, Ciências Exatas e da Terra, CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS, Palmas, 2021.

Figueiredo, R. V.; Marioti, M. B.. **OSLIVE: Módulo Web de Simulação da Paginação por Demanda**. In: ENCOINFO - Congresso de Computação e Tecnologias da Informação, 23., 2021, Palmas - TO. **Anais** [...]. Palmas - TO: CEULP/ULBRA, 2021. p. 147 - 157. ISSN e-ISSN: 2447-0767 versão online. Disponível em: https://ulbra-to.br/encoinfo/edicoes/2021/artigos/oslive-modulo-web-de-simulacao-da-paginacao-por-demanda/. Acesso em: 25 jun. 2022

OLIVEIRA, Rômulo S.; CARISSIMI, Alexandre S.; TOSCANI, SIMÃO S. **Sistemas Operacionais-Vol. 11 - Série Livros Didáticos Informática UFRGS**. Bookman Editora, 2010.

MAIA, L.P., PACHECO, A.C. ***A Simulator Supporting Lectures on Operating Systems*.** Apresentado no 33rd ASEE/IEEE Frontiers in Education Conference, Nov. 2003, Boulder, CO, EUA.

MACHADO, Francis B; MAIA, Luiz P. **Arquitetura de sistemas operacionais. 5. ed. - [Reimpr.]**. - Rio de Janeiro : LTC, 2014.

MOZILLA FOUNDATION, **JAVASCRIPT**. 2021. Disponível em: https://developer.mozilla.org/ptBR/docs/Web/JavaScript. Acesso em: 24 Nov. 2021

NOTIFY.JS. **notify.js**, 2021. Disponível em: https://notifyjs.jpillora.com. Acesso em: 24 Nov. 2021.

OLIVEIRA, Rômulo S.; CARISSIMI, Alexandre S.; TOSCANI, Simão S. **Sistemas Operacionais-Vol. 11 - Série Livros Didáticos Informática UFRGS**. Bookman Editora, 2010.

H. ARPACI-DUSSEAU e C. ARPACI-DUSSEAU. **Operating Systems: Three Easy Pieces,** Arpaci-Dusseau Books, Inc. 2014.

SILBERSCHATZ, Abraham; GALVIN, Peter Baer; GAGNE, Greg. **Fundamentos de Sistemas Operacionais**. 9. ed. Rio de Janeiro: LTC, 2015.

STUART, Brian L. **Princípios de sistemas operacionais:** projetos e aplicações**.** São Paulo: Cengage Learning, 2011.

TANENBAUM, Andrew S.; BOS, Helbert. **Sistemas Operacionais Modernos**. 4ª. ed. São Paulo: Pearson, 2016. Tradução de: Daniel Vieira e Jorge Ritter.