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
ASSOCIAÇÃO EDUCACIONAL LUTERANA DO BRASIL

Alisson Melo dos Santos

IMPLEMENTAÇÃO DE SOFTWARE PARA SIMULAÇÃO DO MECANISMO DE PAGINAÇÃO

Palmas – TO

2017

Alisson Melo dos Santos

IMPLEMENTAÇÃO DE SOFTWARE PARA SIMULAÇÃO DO MECANISMO DE PAGINAÇÃO

Trabalho de Conclusão de Curso (TCC) II elaborado e apresentado como requisito parcial para obtenção do título de bacharel em Ciência da Computação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. M.e Madianita Bogo Marioti.

Palmas – TO

2017

Alisson Melo dos Santos

IMPLEMENTAÇÃO DE SOFTWARE PARA SIMULAÇÃO DO MECANISMO DE PAGINAÇÃO

Trabalho de Conclusão de Curso (TCC) II elaborado e apresentado como requisito parcial para obtenção do título de bacharel em Ciência da Computação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. M.e Madianita Bogo Marioti.

Aprovado em: _____/_____/_____

BANCA EXAMINADORA

_____

Prof. M.e Madianita Bogo Marioti

Orientador

Centro Universitário Luterano de Palmas – CEULP

_____

Prof. M.e Fabiano Fagundes

Centro Universitário Luterano de Palmas – CEULP

_____

Prof. M.e Fernando Luiz de Oliveira

Centro Universitário Luterano de Palmas – CEULP

Palmas – TO

2017

AGRADECIMENTOS

A Deus e aos meus familiares que, com muito carinho e apoio, não mediram esforços para que chegasse até esta etapa de minha vida.

A minha orientadora, que carinhosamente chamo de mãe (Madianita), por seus ensinamentos, paciência e confiança ao longo do desenvolvimento deste trabalho. Ganhei uma pessoa maravilhosa na minha vida, que torceu e motivou durante todo o trabalho e em alguns momentos foi quem mais acreditou (as vezes mais do que eu mesmo) que eu conseguiria finalizar este trabalho. Eu posso dizer que o resultado alcançado aqui, não teria sido o mesmo sem você e que foi muito bom trabalhar ao seu lado neste trabalho.

Aos professores Cintia, Conceição, Cristina, Diêmy, Edeilson, Fabiano, Fábio, Fernando, Ilson, Jackson, Madianita, Natanael, Parcilene, Patrícia, Ricardo e Valquiria que foram tão importantes na minha vida acadêmica e que me forneceram todo o conhecimento que hoje tenho e, que sem os seus ensinamentos certamente não teria alcançado as conquistas que consegui em minha vida acadêmica.

Aos meus amigos: Caio, Murillo (Paçoca), Robson, Dennis, Eugênio, Ranyelson, Vinicius (Juvenal), Alexandre e Renato.

Enfim, agradeço a todos aqueles que de alguma forma estiveram e estão próximos de mim e, que de algum modo, me ajudaram durante estes 5 anos de batalha me motivando a não desistir e sempre continuar em frente. Muito obrigado!

RESUMO

SANTOS, Alisson Melo dos. **IMPLEMENTAÇÃO DE SOFTWARE PARA SIMULAÇÃO DO MECANISMO PAGINAÇÃO**. 2017. 58 f. Trabalho de Conclusão de Curso (Graduação) – Curso de Ciência da Computação, Centro Universitário Luterano de Palmas, Palmas/TO, 2017.

A disciplina de Sistemas Operacionais é uma disciplina importante para os cursos de computação. Esta disciplina possibilita ao aluno compreender as técnicas que são utilizadas pelo Sistema Operacional a fim de gerenciar os recursos necessários para um bom funcionamento do sistema computacional, por exemplo, a gerência de memória. A existência de ferramentas educacionais disponibilizadas na internet permite auxílio aos alunos no processo de aprendizagem devido à complexidade presente na disciplina. Isto porque o Sistema Operacional funciona em baixo nível e, por não ser possível a visualização do funcionamento dos mecanismos de gerenciamento, acaba tornando complicada a aprendizagem para alguns alunos. Este trabalho teve como finalidade apresentar ferramentas de simulação de gerenciamento de memória disponíveis na internet realizando um comparativo entre elas e desenvolver um *software* para simulação do mecanismo de paginação, de modo a auxiliar os alunos no processo de aprendizagem.

Palavras-chave: Sistemas Operacionais, Gerência de Memória, Simulador de Paginação.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

SO – Sistemas Operacionais

E/S – Entrada e Saída

CPU – Unidade Central de Processamento

TI – Tecnologia da Informação

MMU – Unidade de Gerenciamento de Memória (*Memory Management Unit*)

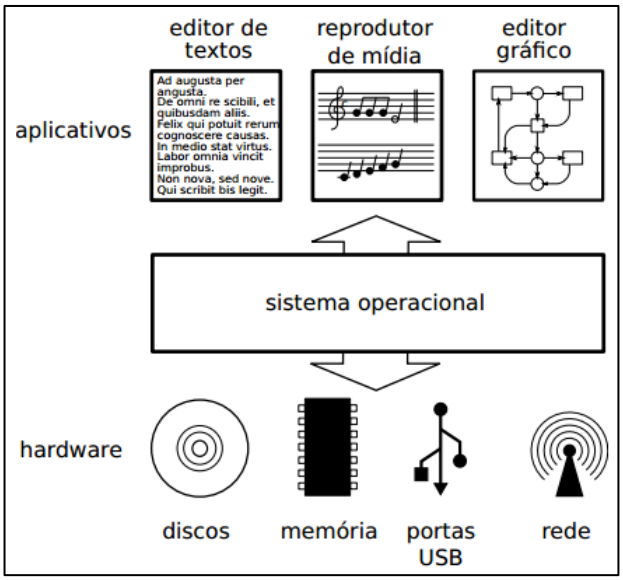
RAM – Memória de Acesso Aleatório (*Randon Access Memory*)

SUMÁRIO

1 INTRODUÇÃO

Os Sistemas Operacionais (SO) possibilitam que o usuário utilize recursos que os computadores oferecem. Segundo Silberschatz, Galvin e Gagne (2013), um SO é responsável por fazer a intermediação entre as aplicações de usuário e o *hardware*, a fim de que ambas as partes possam funcionar corretamente. A Figura 1 demonstra como é um sistema de computação típico.

Figura 1. Representação da intermediação do SO



Fonte: MAZIERO, Carlos Alberto, 2011

A intermediação realizada pelo SO é primordial já que muitos usuários não conseguiriam utilizar um computador sem um SO, isto porque o nível de complexidade para o acesso aos dispositivos de *hardware* é alto.

Para a compreensão do funcionamento dos SOs, nos cursos da área de computação, é oferecida a disciplina de Sistemas Operacionais. Segundo as diretrizes curriculares nacionais para os cursos da área da Computação atribuídas pelo MEC (2012), esta é uma disciplina indispensável. Ainda sobre a importância dessa disciplina, Silberschatz, Galvin e Gagne (2013, p. Prefácio), afirmam que "os sistemas operacionais são uma parte essencial de qualquer sistema de computação. Da mesma forma, um curso sobre sistemas operacionais é parte essencial de qualquer processo educacional em ciência da computação".

A disciplina aborda as quatro áreas do SO: gerência de processos, gerência de memória, gerência de entrada e saída (E/S) e gerência de arquivos. No gerenciamento de memória, foco deste trabalho, o SO controla quais endereços da memória RAM (*Randon Access Memory*) estão sendo utilizados, aloca memória quando solicitado por um processo e libera a memória quando o processo finaliza sua execução, entre outras ações.

Nos conceitos relacionados aos SOs são apresentados vários mecanismos de gerência de memória. Um SO utiliza o que é determinado nesses mecanismos para implementar a sua política de gerenciamento de memória. As políticas de escalonamento dos SOs mais utilizadas, Linux e Windows, se baseiam no mecanismo de paginação e segmentação, o foco deste trabalho é o mecanismo de paginação.

O algoritmo do mecanismo de paginação determina como o SO deve prover recursos necessários para que os processos compartilhem a memória de modo seguro. Mas, por este funcionamento ocorrer em baixo nível, não é possível visualizar as ações que são realizadas e, a não visualização implica diretamente na aprendizagem do aluno. Assim, devido as ações da gerência de memória não serem visíveis, este conceito se torna complexo de se compreender.

Para ajudar os alunos a compreenderem a gerência de memória, pode-se utilizar ferramentas de simulação que demonstre o funcionamento do mecanismo de paginação. Assim, é possível visualizar e interagir com o funcionamento destes mecanismos, o que auxilia na aprendizagem do aluno. De acordo com Gadelha et al. (2010, p. 2), a utilização de ferramentas de simulação para auxílio na educação é interessante, porque a interação visual permite ao aluno compreender melhor o conteúdo por ser mais interessante e por complementar o método tradicional de ensino.

Atualmente, podem ser encontradas algumas ferramentas de simulação de gerência de memória na internet. As ferramentas encontradas foram estudadas a fim de se fazer um levantamento dos seus aspectos técnicos e uma verificação de algumas características destas, também, foi feito um paralelo entre elas. Os estudos das características das ferramentas foram levados em consideração para elaboração do projeto.

Assim, para alcançar o objetivo deste trabalho, buscou-se responder ao seguinte problema de pesquisa: como simular o mecanismo de paginação de forma a auxiliar no aprendizado deste conteúdo na disciplina de SO?

A partir do problema apresentado, é possível considerar a seguinte hipótese de que, utilizando as tecnologias *AngularJS*, *Bootstrap* e *HTML* é possível desenvolver uma ferramenta virtual capaz de simular o funcionamento do gerenciamento de memória, de maneira a facilitar a compreensão deste conteúdo. Desse modo, este trabalho propõe como objetivo geral, o

desenvolvimento de um *software* para simulação do funcionamento do mecanismo de gerência de memória, paginação, que é subdivido nos seguintes objetivos específicos:

- apresentar ferramentas de simulação de gerenciamento de memória disponíveis na internet;
- apresentar um comparativo das características das ferramentas estudadas;
- criar protótipos das telas com base nos conceitos e ferramentas estudados;
- implementar interface para interação com usuário;
- implementar a simulação do mecanismo de paginação.

O mecanismo de paginação é conceito básico e importante de SO e, por isso, é abordado na disciplina. É importante ressaltar, também, que este conceito está presente nos principais livros da área, por exemplo, os livros de Silberschatz, Galvin, Gagne (2013) e Tanenbaum (2009).

O funcionamento do mecanismo de paginação não tem como ser visualizado, assim, resulta na dificuldade de aprendizagem do aluno. Visualizar e ter interação com funcionamento desse algoritmo, através de ações realizadas pelo aluno de alocação e liberação de memória, auxilia-o a compreender o funcionamento desse mecanismo.

Na internet, são disponibilizadas algumas ferramentas de simulação do mecanismo de paginação. Todavia, as ferramentas encontradas apresentam características que dificultam o uso nas disciplinas como, por exemplo, não seguir o modelo dos livros didáticos; apresentar complexidade para utilização; não possuir funções que auxiliem na aprendizagem etc.

Neste contexto, é interessante implementar um *software* que ofereça a simulação do mecanismo de paginação e não apresente os problemas encontrados nas demais ferramentas. Por exemplo, é importante que este atenda aos requisitos de fidelidade ao conteúdo didático, fácil utilização e fácil entendimento. Com isto, o *software* poderá ser utilizado como apoio ao processo de ensino e aprendizagem.

Este trabalho apresenta, no capitulo 2, o Referencial Teórico, que é dividido entre Sistemas Operacionais, Gerência de Memória e as Ferramentas Relacionadas com a apresentação do paralelo entre as ferramentas. No capítulo 3 é apresentada a Metodologia, que é dividida em Desenho de Estudo e Materiais. No capítulo 4 é apresentado os Resultados e Discussão, que é dividido em Arquitetura do *Software*, Interface, Simulação do Mecanismo de Paginação, Codificação e Demonstração de Como a Simulação pode Auxiliar na Compreensão do Funcionamento do Mecanismo de Paginação. No capítulo 5 são apresentadas as Considerações Finais deste trabalho. E, por fim, são apresentadas as Referências utilizadas para o desenvolvimento deste trabalho.

2 REFERENCIAL TEÓRICO

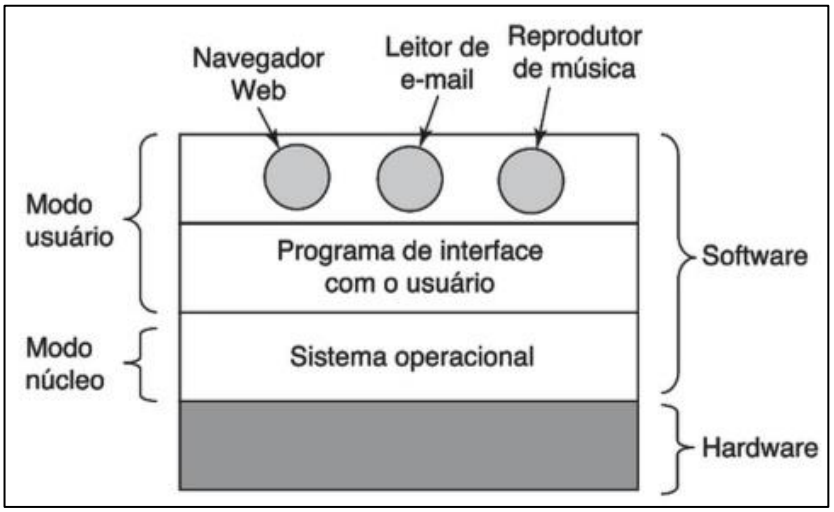
Nesta seção é apresentada uma visão geral de SOs e suas quatro áreas: gerência de processos; gerência de memória; entrada e saída; e arquivos. Também, são apresentadas ferramentas de simulação de gerência de memória disponibilizadas na Internet e um paralelo entre elas.

2.1 SISTEMA OPERACIONAL

Um Sistema Operacional, segundo Silberschatz, Galvin e Gagne. (2013, p. 3), "fornece uma base para os programas aplicativos e atua como intermediário entre o usuário e o hardware do computador". É responsabilidade do SO realizar o gerenciamento dos recursos oferecidos pelo *hardware*, a fim de fornecer um funcionamento adequado para os *softwares*.

Os recursos oferecidos pelo *hardware*, conforme o que foi dito por Tanenbaum (2009, p. 1), são um ou mais processadores, memória principal, discos, teclado, mouse, monitor, interfaces de rede e outros dispositivos de entrada e saída, que constituem um sistema computacional. Na Figura 2 é apresentada uma visão geral dos componentes que compõem um computador, detalhando a hierarquia entre os *softwares* e a interação entre o SO e o *hardware*.

Figura 2. Níveis de um sistema computacional



Fonte: Tanenbaum, Andrew S, 2009

Pode-se observar na imagem que o sistema computacional é composto por níveis. No nível inferior está presente o *hardware*, que é composto por *chips*, placas, interfaces de rede e outros dispositivos de entrada e saída. E, vizinho ao *hardware*, está o *software*, que pode ser classificado em modo núcleo e modo usuário. O SO opera no modo núcleo, que é a camada do *software* que tem acesso total a todos os componentes do *hardware* e é capaz de executar

qualquer instrução que a máquina possa executar. O modo usuário é composto pelo restante dos *softwares* presentes na máquina, porém, eles não possuem acesso livre ao *hardware*. A execução dos *softwares* no modo usuário é controlada pelo SO a fim de que a execução dos aplicativos ocorra de modo transparente para o usuário.

Além da intermediação que o SO faz entre o *hardware* e os demais *softwares* para execução, Tanenbaum (2009, p.3) afirma que os SOs são responsáveis, também, pela interface gráfica que é gerada para o usuário. A interface é um elemento de suma importância, já que uma pessoa leiga seria incapaz de utilizar um computador que apresentasse apenas informações de baixo nível.

Para o gerenciamento de recursos e para que o sistema computacional funcione corretamente, o SO é dividido em quatro áreas:

- **Gerência de Processos:** um processo é um programa em execução e que pode assumir mais de um estado no decorrer do seu ciclo de vida. E, como os sistemas trabalham de modo concorrente e com mais de um processo, é necessário que o SO faça o gerenciamento de processos, que coordena a utilização dos recursos pelos processos. Por exemplo, tempo de uso da CPU, alocação de memória e uso de dispositivos de E/S;
- **Gerência de Memória:** o SO é responsável por alocar e liberar memória a um processo, de forma segura. No uso da memória, deve garantir que um processo não acesse uma área da memória que já está sendo utilizada e, assim, resulte em um funcionamento insatisfatório dos programas;
- **Gerência de Entrada / Saída (E/S):** o SO gerencia os dispositivos que estão conectados ao computador, assim, realizando um controle das operações de E/S quando solicitadas por um processo;
- **Gerência de Arquivos:** gerencia o armazenamento de dados do SO e dos usuários através de coleções de arquivos e diretórios. De todos os aspectos de um SO a gerência de arquivos é a mais visível para o usuário, diferente das citadas anteriormente em que praticamente todas suas ações ocorrem em baixo nível.

O objetivo desse trabalho é a simulação do mecanismo de paginação, que é um conceito de gerência de memória. Assim, a próxima seção aborda esse gerenciamento de forma mais aprofundada.

2.2 GERÊNCIA DE MEMÓRIA

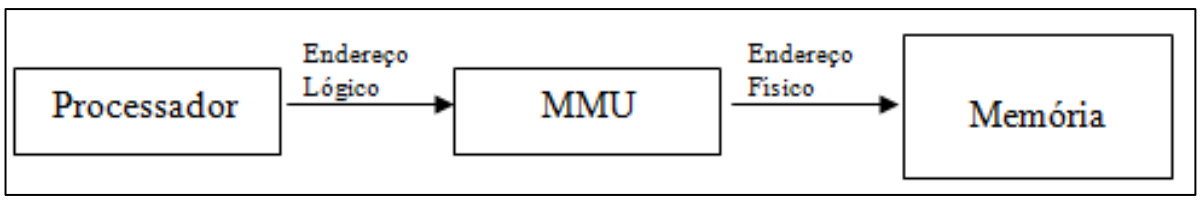
Atualmente, os SOs têm suporte para trabalhar com a multiprogramação, ou seja, são capazes de executar um ou mais processos ao mesmo tempo. E, conforme Oliveira, Carissimi e Toscani (2010, p. 153), na multiprogramação, vários processos utilizam a CPU através do gerenciamento de processos do SO. Para utilização da CPU é necessário realizar o chaveamento que, segundo Toscani (2017), "é a troca do processo em execução (O contexto do processo que estava em execução deve ser guardado e o contexto do próximo a executar deve ser restaurado)". Para que o chaveamento dos processos seja rápido, eles devem estar na memória física prontos para executar, já que o acesso é mais rápido do que o acesso ao disco.

Durante a execução de um processo pela CPU, o local na memória no qual este foi alocado é acessado constantemente para busca das instruções que devem ser realizadas e também para o acesso dos dados. Para que os locais na memória sejam acessados corretamente, é necessário que haja uma abstração para a memória a fim de evitar que o sistema seja danificado ou paralisado. A abstração é a divisão em espaços de endereçamento lógico e físico.

A memória lógica é a que os processos têm acesso, a qual eles são capazes de endereçar, sendo que cada processo possui a sua própria memória lógica independentemente dos outros processos. Os ponteiros da linguagem C são um exemplo de endereçamento lógico. A memória física é a que funciona em nível de máquina, referenciando os circuitos e componentes eletrônicos, sendo que o espaço de endereçamento físico é formado por todos os endereços aceitáveis pelos circuitos. Assim, os endereços físicos correspondem a uma posição real na memória.

A Unidade de gerência de memória (*Memory Management Unit*, MMU) conforme Oliveira et al. (2010, p. 154), "é que vai mapear os endereços lógicos gerados pelos processos nos correspondentes endereços físicos que serão enviados para a memória". Assim, a MMU é o componente de *hardware* responsável por prover os mecanismos de gerência de memória para o SO. A Figura 3 apresenta o diagrama do funcionamento da MMU entre o processador e a memória.

Figura 3. Diagrama MMU



Fonte: Oliveira, Carissimi e Toscani, 2010

Um processo que está sendo executado na CPU e que necessita acessar a memória deve traduzir o endereço lógico para o endereço físico a fim de saber em qual local está alocado na memória. A MMU é responsável pela tradução do endereço. Após a tradução do endereço lógico, o processo pode acessar seu endereço físico na memória.

Para que os processos possam ser alocados na memória e funcionem corretamente sem apresentar erros, o SO utiliza políticas de gerência de memória baseadas em mecanismos conceituais. Existem vários mecanismos para alocação de memória, como partições simples, partições variáveis, paginação, segmentação e paginação com segmentação.

Como o foco desse trabalho é o mecanismo de paginação, ele é descrito de forma mais detalhada na seção a seguir.

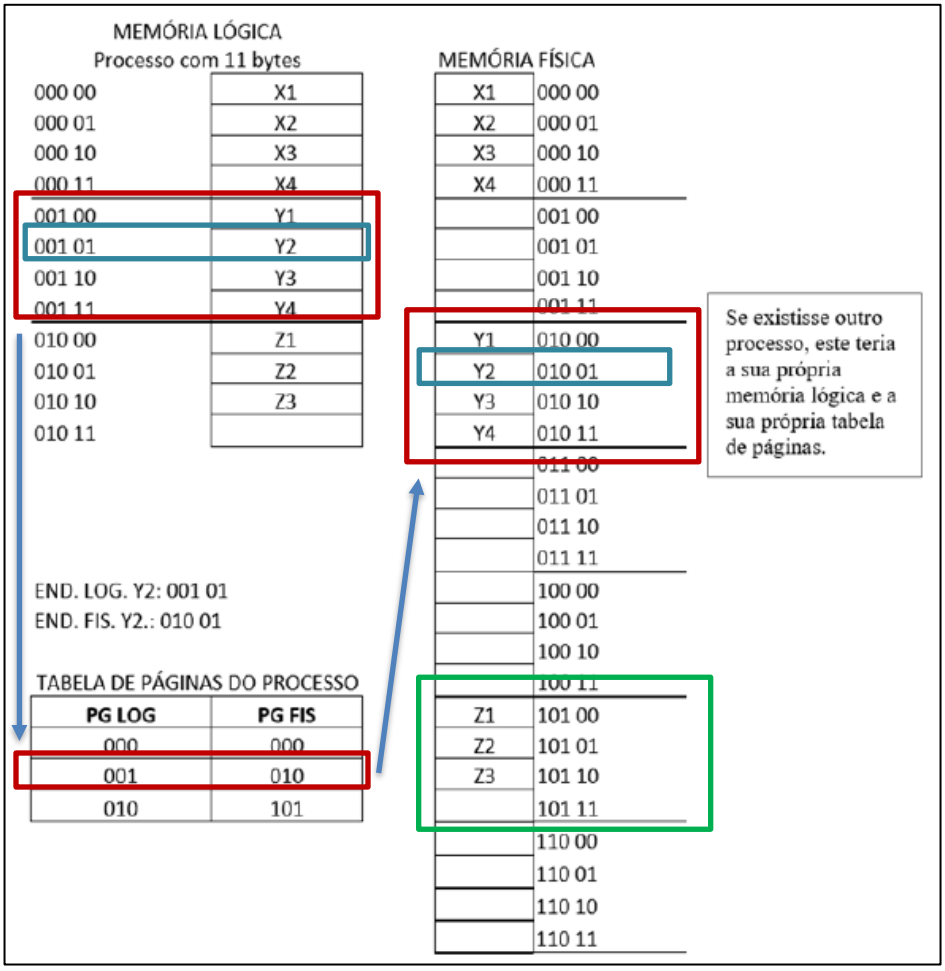
2.2.1 Paginação

O mecanismo de paginação, segundo Silberschatz, Galvin e Gagne (2013, p. 172), é uma solução utilizada para evitar a fragmentação externa, que são espaços pequenos não contíguos de memória livre que surgem durante a alocação e liberação de memória. Esses espaços as vezes são pequenos de mais para alocar um processo, assim, ocasionando no desperdício de memória.

Na paginação, a memória física é dividida em blocos de tamanho fixo, que são denominados de quadros, sendo que o tamanho dos quadros é definido pela MMU e que os processos ocupam apenas os espaços internos destes. A memória lógica é dividida em blocos, denominados páginas lógicas, que possuem o mesmo tamanho dos quadros da memória física.

A Figura 4 apresenta um exemplo do mecanismo de paginação.

Figura 4. Representação da Paginação Entre Memória Lógica e Memória Física



Fonte: Oliveira, 2001

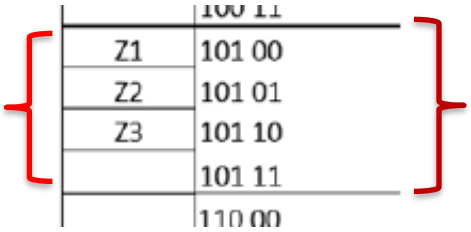
Como pode ser observado na imagem, é apresentada a tabela de páginas do processo, a memória lógica e a memória física. Quando um processo é carregado, é montada uma tabela com as páginas do processo, esse é o mapeamento dos endereços lógicos para os endereços físicos. Na tabela é destacado o número da página da memória lógica e a página da memória física. A memória lógica é composta pelo número da página e o deslocamento de cada byte dentro da página e cada processo possui a sua memória lógica com quantas páginas forem necessárias. A memória física é composta pelo número da página e o deslocamento de cada byte da página. Nela ficam todos os processos e as páginas de um processo não precisam ficar em sequência.

Por exemplo, o byte Y2, com a página destacada em vermelho, está na página lógica 001 no deslocamento 01. A página física na tabela de páginas que corresponde a página lógica 001 é a 010. Assim, pode-se identificar que o byte Y2 foi alocado na página física 010 no deslocamento 01.

A paginação evita a fragmentação externa, porém, não tem como evitar a fragmentação interna. A fragmentação interna é a memória que não é utilizada dentro de um quadro alocado por um processo. Devido à paginação trabalhar com o conceito de página alocando um bloco de tamanho *n* para um processo, pode ser que uma página não seja preenchida por completo, deixando espaços que não são utilizados. Na Figura 5 ocorre este caso de fragmentação interna na página 101 da memória física, destacada em verde na Figura 4, em que é disponibilizado espaço para quatro bytes, mas só é utilizado espaço para três por Z1, Z2 e Z3.

Figura 5. Página 101 da Figura 4

Os quatro espaços dentro da página 101 representa os quatro bytes que cada página disponibiliza



Página 101

2.3 FERRAMENTAS RELACIONADAS

Esta seção apresenta ferramentas disponibilizadas na internet que oferecem a simulação de mecanismos de gerência de memória. Para cada ferramenta, serão apresentadas as seguintes informações: visão geral, apresentação do funcionamento e considerações sobre a ferramenta. Depois, é apresentado um paralelo dos aspectos técnicos, referentes a utilização das ferramentas.

2.3.1 Simulador de Gerenciamento de Memória

A ferramenta Simulador de Gerenciamento de Memória (SGM) é uma ferramenta para simulação dos mecanismos de paginação, segmentação e alocação contígua. As Figuras 6 e 7 apresentam como é feita a simulação da paginação. A Figura 6 apresenta a tela antes de realizar a simulação e a Figura 7 apresenta a tela depois de realizar a simulação.

Figura 6. Tela inicial da simulação de paginação

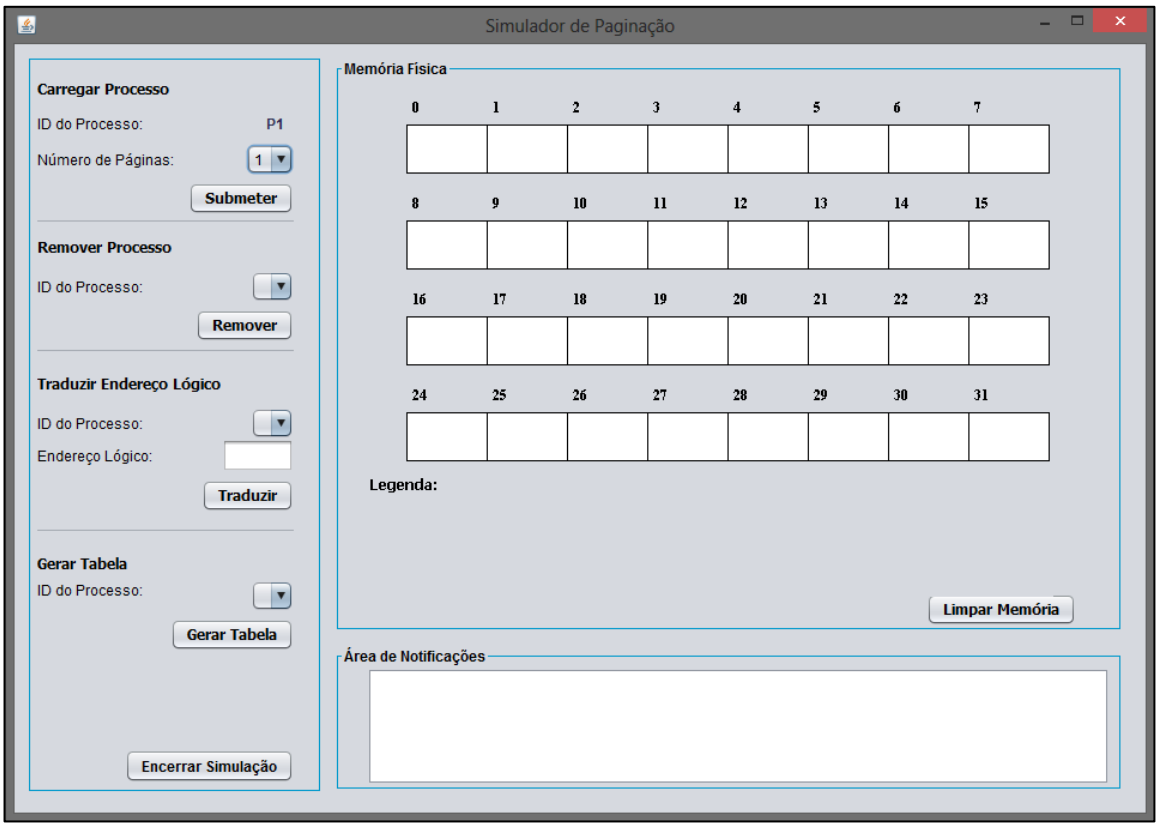
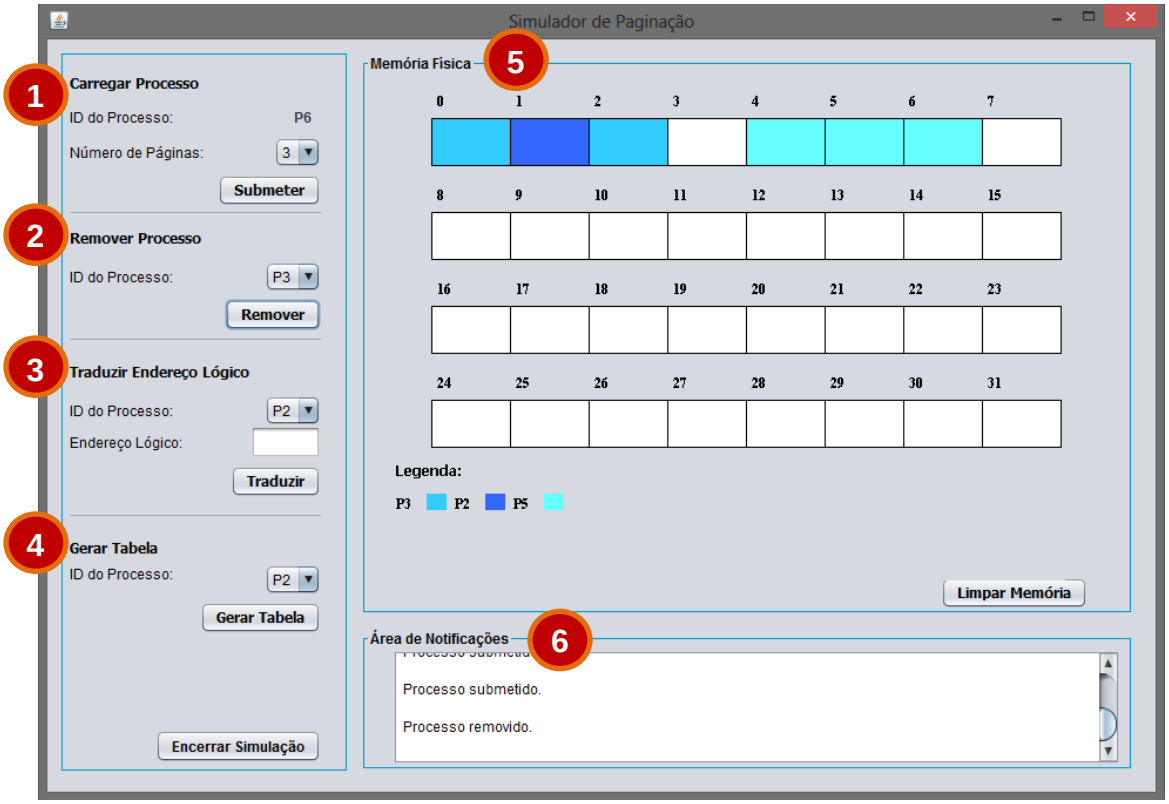


Figura 7. Tela após simulação da paginação



A Figura 6 é o modo como o usuário visualiza a tela de simulação antes de inserir qualquer dado. As ações disponíveis são: carregar processo, remover processo, traduzir endereço lógico, gerar tabela, limpar memória e encerrar simulação. Já a Figura 7, é como o usuário visualiza a tela de simulação após executar ações de alocar e remover processo da memória e, também está numerado os aspectos que a ferramenta apresenta:

1. carregar processo;
2. remover processo;
3. traduzir endereço lógico;
4. gerar tabela com informações do processo;
5. representação gráfica da memória física;
6. área de notificações.

Na Figura 7 o usuário executou as seguintes ações:

- alocou o processo P1 com 1 página;
- alocou o processo P2 com 1 página;
- removeu o processo P1;
- alocou o processo P3 com 2 páginas;
- alocou o processo P4 com 1 página;
- alocou o processo P5 com 3 páginas;
- removeu o processo P4.

Quando um processo é inserido, ele é acrescentado graficamente na memória física, que é representada por blocos e cores para identificar cada processo, também possui uma legenda abaixo da memória para identificar as cores dos processos. Abaixo da memória possui uma área de notificações, que sempre apresenta uma mensagem quando uma ação é executada.

2.3.1.1 Considerações Sobre a Ferramenta

- Visão geral: A ferramenta SGM é uma ferramenta que foi desenvolvida pela Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto da Universidade de São Paulo (FFCLRP). A ferramenta foi desenvolvida em Java e possui versão somente para desktop. Possui como principais funcionalidades a simulação dos mecanismos de paginação, segmentação e alocação contigua. A ferramenta é disponibilizada gratuitamente na internet pela FFCLRP junto com documentação para uso e está disponível em: http://dcm.ffclrp.usp.br/~farias/disciplinas/so/.

- Prós:
    - documentação para instalação e utilização;
    - fácil utilização;
    - interface adequada ao contexto.
- Contra:
    - difícil instalação;
    - ausência de informações básicas sobre os processos;
    - não apresenta os endereços lógicos e nem a tabela de páginas.

2.3.2 JOSim

A ferramenta JOSim é uma ferramenta para simulação do mecanismo de paginação. As Figuras a seguir apresentam a interface da ferramenta, nelas é apresentada a simulação da paginação. A Figura 8 apresenta a tela antes de realizar a simulação e a Figura 9 apresenta a tela depois de realizar a simulação.

Figura 8. Tela inicial da simulação da paginação JOSim

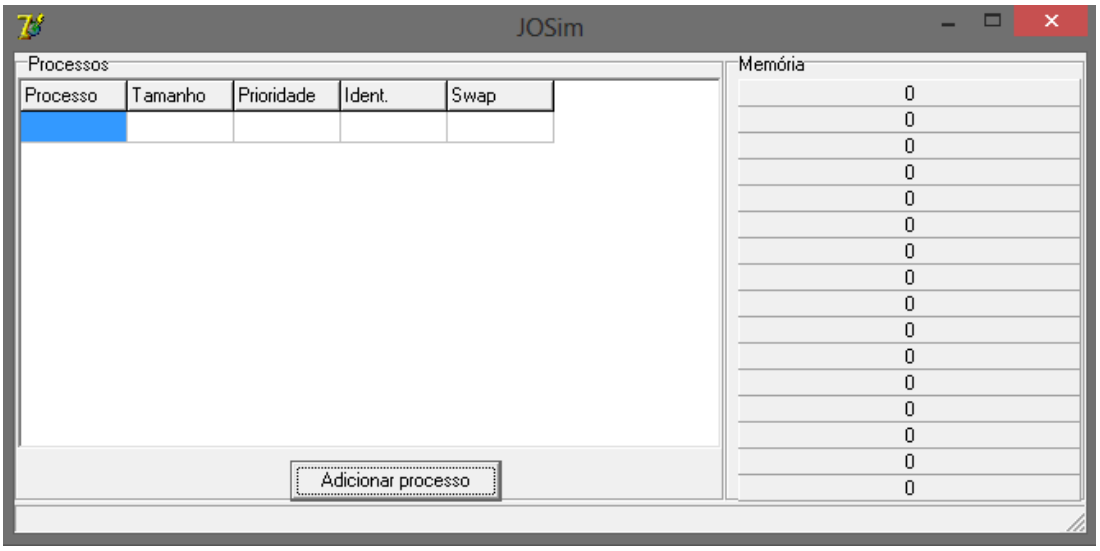
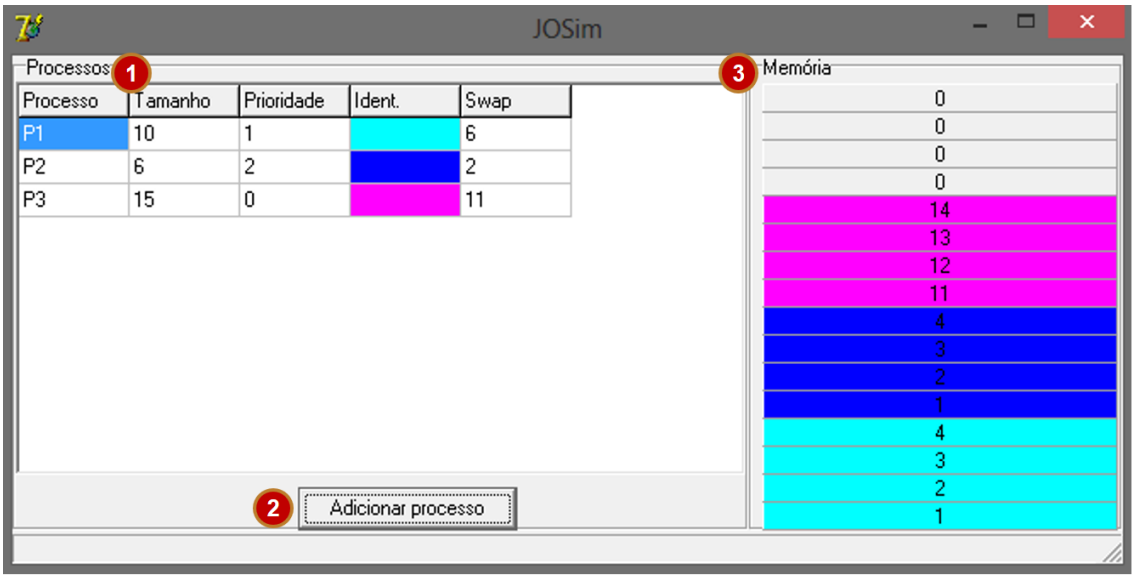


Figura 9. Tela após ações simulação da paginação JOSim



A Figura 8 é o modo como o usuário visualiza a tela de simulação antes de inserir qualquer dado. A única ação que a ferramenta possui é de adicionar processo, em que o usuário digita um nome, tamanho e prioridade para o processo. Já a Figura 9 é como o usuário visualiza a tela de simulação após executar ações de alocar um processo na memória e, também está numerado os aspectos que a ferramenta apresenta:

1. lista de processos;
2. adicionar processo;
3. representação gráfica da memória.

Na Figura 9 o usuário executou as seguintes ações:

- adicionou um processo com nome P1, tamanho 10 e prioridade 1;
- adicionou um processo com nome P2, tamanho 6 e prioridade 2;
- adicionou um processo com nome P3, tamanho 15 e prioridade 0.

O funcionamento da ferramenta é simples e, quando um processo é inserido, ele é acrescentado em uma tabela com os outros processos. Também, após ser inserido o processo, ele é acrescentado graficamente na memória com uma cor que identifica cada processo.

2.3.2.1 Considerações Sobre a Ferramenta

- Visão geral: A ferramenta JOSim é uma ferramenta que foi desenvolvida por Jonis Maurin Ceará, quando foi aluno na disciplina de SO I. Não foi identificado em qual linguagem a ferramenta foi desenvolvida e possui versão somente para desktop. Possui como principal

funcionalidade a simulação do mecanismo de paginação. É disponibilizada gratuitamente na internet no website do autor, estando disponível em: https://www.jonis.com.br/2008/05/20/simulador-de-paginacao/.

- Prós:
  - facilidade para instalação;
  - fácil utilização.
- Contra:
  - não possui documentação;
  - interface com baixa fidelidade ao material didático;
  - falta de funcionalidade para remoção de um processo;
  - permite adicionar somente quatro processos.
  - não apresenta os endereços lógicos e nem a tabela de páginas.

2.3.3 MetroOS

A ferramenta MetroOS é uma ferramenta para simulação do mecanismo de paginação. As Figuras a seguir apresentam como o usuário vê a simulação com esta ferramenta, nelas é apresentada a simulação da paginação. A Figura 10 apresenta a tela antes de realizar a simulação e a Figura 11 apresenta a tela depois de realizar a simulação.

Figura 10. Tela do inicial do mecanismo da paginação MetroOS

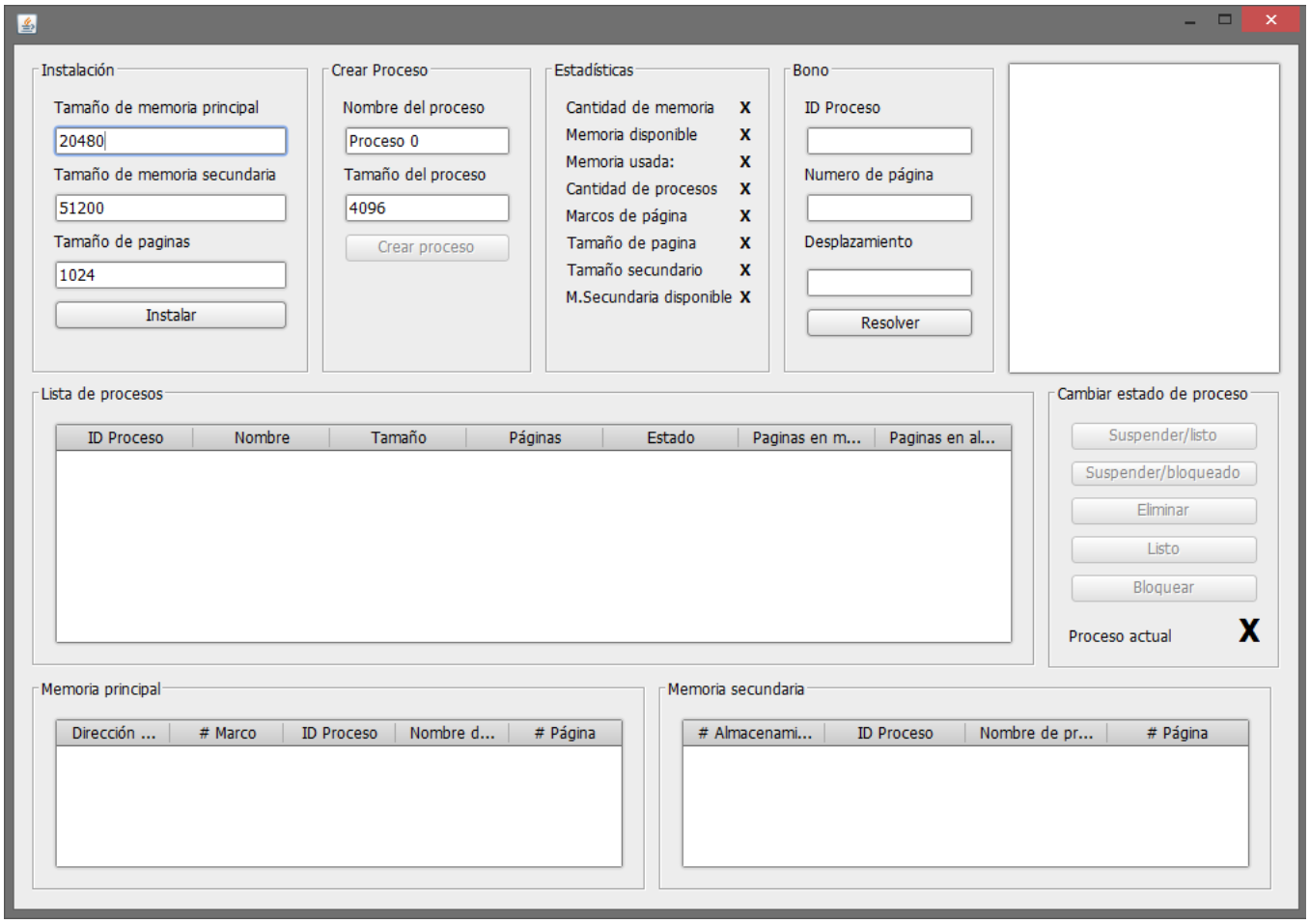
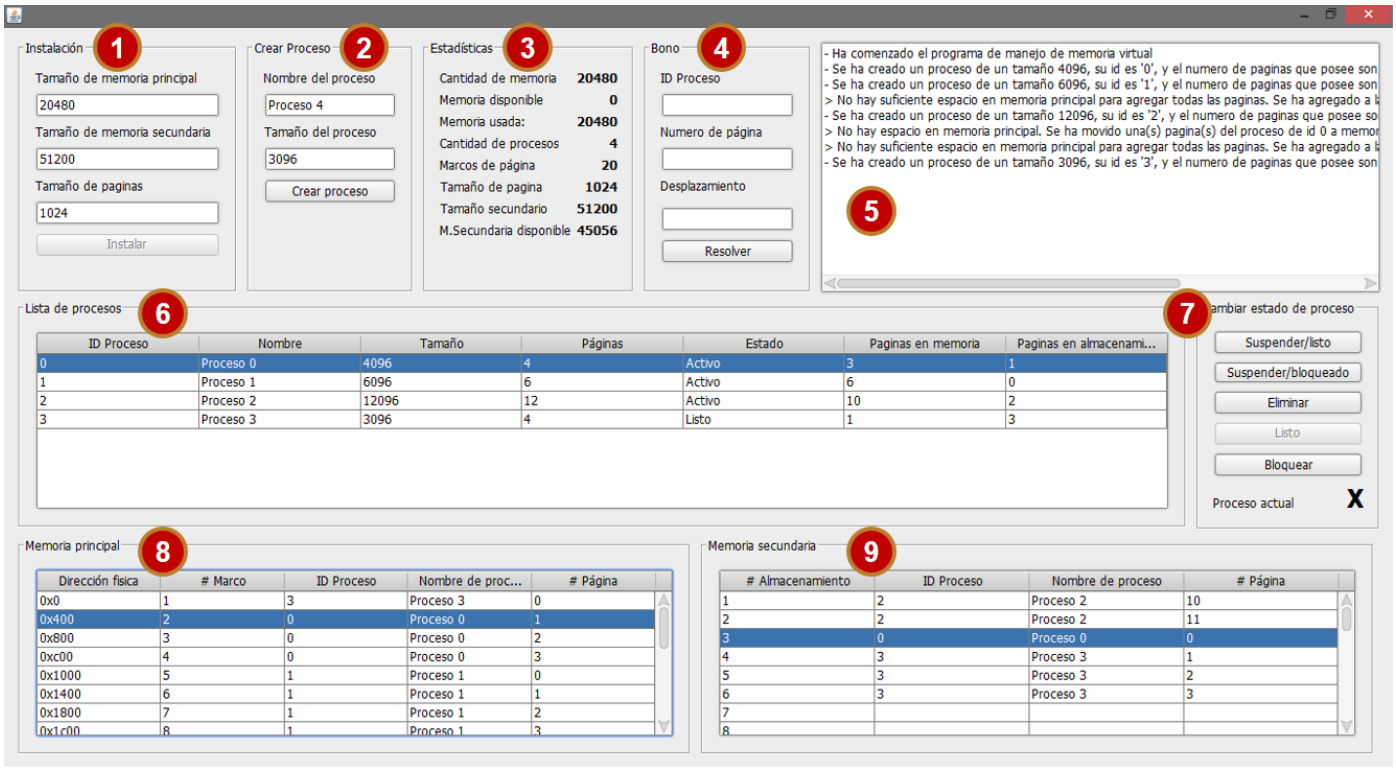


Figura 11. Tela após simulação do mecanismo da paginação MetroOS



A Figura 10 é o modo como o usuário visualiza a tela de simulação antes de inserir qualquer dado. As ações principais são: configurar a memória, criar processo, excluir processo, bloquear e excluir processo. Já a Figura 11 é como o usuário visualiza a tela de simulação após executar ações de alocar e remover processos da memória e, também está numerado os aspectos que a ferramenta apresenta:

1. instalar memória física;
2. criar processo;
3. estatísticas da memória;
4. apresentar informações extra do processo indicado;
5. área de notificações;
6. lista de processos;
7. ações para realizar com os processos;
8. representação gráfica da memória física primária;
9. representação gráfica da memória física secundária.

Na Figura 11 o usuário executou as seguintes ações:

- instalou memória física, com memória primária de tamanho 20480, memória secundaria de tamanho 51200 e páginas com o tamanho de 1024;
- inseriu Processo 0 de tamanho 4096 e, que foi dividido em 4 páginas;
- inseriu Processo 1 de tamanho 6096 e, que foi dividido em 6 páginas;
- inseriu Processo 2 de tamanho 12096 e, que foi dividido em 12 páginas;
- inseriu Processo 3 de tamanho 3096 e, que foi dividido em 4 páginas.

A configuração da memória é a primeira ação que o usuário deve realizar e, somente após configurar a memória indicando o tamanho da memória primária, a memória secundária e o tamanho das páginas é que o usuário pode inserir os processos. Ao lado de inserir um processo são apresentadas estatísticas da memória, em que são apresentados o tamanho da memória, a memória disponível, a memória usada etc.

A cada ação realizada com um processo, é informada uma mensagem no canto superior à direita. E os processos que são alocados são apresentados em uma tabela em que é apresentado o id do processo, nome, tamanho, quantidade de páginas, estado do processo, quantidade de páginas na memória primária e quantidade de páginas na memória secundária. Sempre que um processo é alocado, ele é acrescentado à tabela da memória primária e, caso a memória primária esteja sem espaço o processo pode ser alocado na memória secundária. Por exemplo, o Processo

3 possui 1 página na memória primária e 3 páginas na memória secundária. Também, tem a possibilidade de identificar o endereço físico em binário e hexadecimal. Para realizar isto, tem o painel de vínculo "*Bono*" que fica à esquerda da área de notificações, em que deve ser informado o id do processo, o número da página e o deslocamento.

2.3.3.1 Considerações Sobre a Ferramenta

- Visão geral: a ferramenta MetroOS, é uma ferramenta que foi desenvolvida por Ricardo Rodríguez. A ferramenta foi desenvolvida em Java e possui versão somente para desktop. Possui como única funcionalidade, a simulação do mecanismo de paginação. Esta ferramenta é disponibilizada gratuitamente na internet no github do autor, estando disponível em: https://github.com/Ricardo96r/Simulador-de-memoria-virtual. A ferramenta não dispõe de documentação para instalação e manual para utilização. Possui interface adequada ao contexto, mas, seu uso sem um especialista pode torna-se difícil.
- Prós:
    - fácil instalação;
    - apresenta todas informações sobre um processo;
    - interface adequada ao contexto.
- Contra:
    - difícil utilização;
    - não possui documentação;
    - ausência de elementos gráficos para melhorar a visualização;
    - não apresenta os endereços lógicos e nem a tabela de páginas;
    - baixa fidelidade ao material didático.

2.3.4 Paralelo

Nesta seção será apresentado um paralelo que compara os aspectos técnicos das ferramentas citadas anteriormente. Foram selecionados aspectos técnicos citados por Santos (sem data), Schulze e Paula (2015) e a Prof. M.e Madianita Bogo Marioti. A definição dos aspectos técnicos é descrita na Metodologia na seção "Critérios Para Paralelo Entre as Ferramentas". A comparação dos aspectos técnicos foi realizada com auxílio da especialista da área, a partir da utilização das ferramentas, considerando os critérios apresentados.

Para a avaliação dos aspectos técnicos das ferramentas serão utilizadas escalas de 0 a 5, em que 0 define mais baixa e 5 mais alta. A Tabela 1 apresenta o paralelo entre as três ferramentas.

Tabela 1. Paralelo entre as ferramentas de simulação

Aspectos Técnicos	SGM	JOSim	MetroOS
Orientações da ferramenta para utilização	3	0	0
Documentação para instalação	3	0	0
Conteúdo abordado de forma clara	2	0	2
Contexto adequado ao design e atraente	3	1	2
Complexidade de instalação e desinstalação	4	0	0
Interação aluno X ferramenta	3	1	3
Complexidade de compreensão sem instrutor	2	2	4
Clareza na exposição das informações	3	2	4
Fidelidade ao material didático	2	0	3
Complexidade de utilização	2	0	5
Plataforma	Desktop	Desktop	Desktop

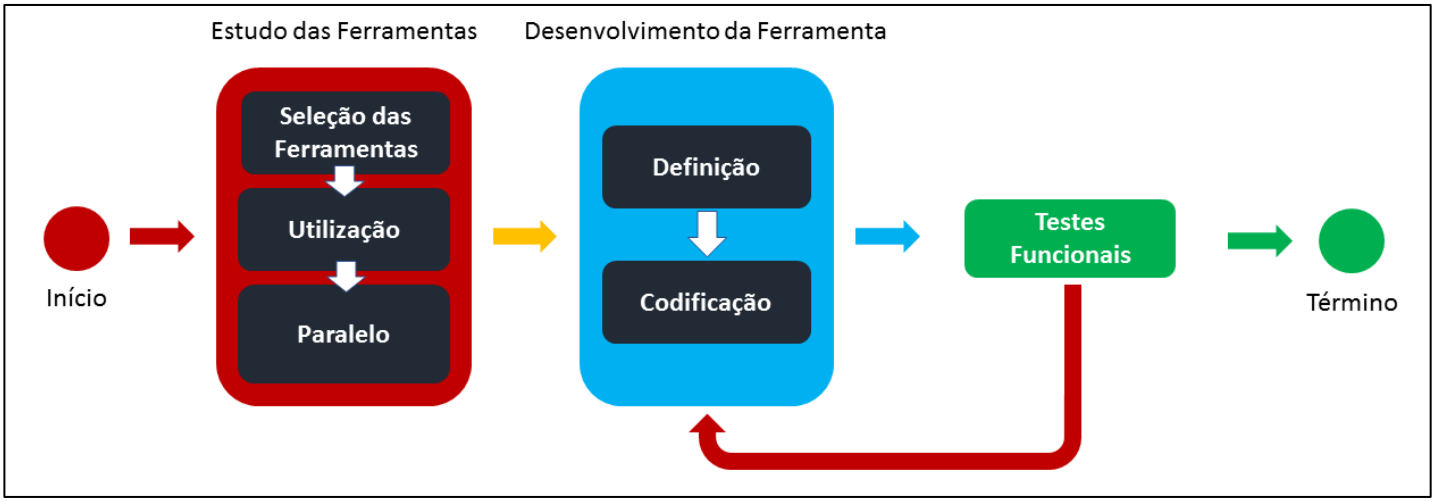
3 METODOLOGIA

Esta seção apresenta o desenho de estudo que foi definido para este trabalho e materiais que foram utilizados para o desenvolvimento do ambiente.

3.1 DESENHO DE ESTUDO

A Figura 12 apresenta as etapas que foram realizadas para o desenvolvimento do trabalho.

Figura 12. Etapas para o desenvolvimento do trabalho



O desenvolvimento do trabalho foi dividido em três etapas:

- **Estudo das Ferramentas:** esta etapa divide-se em três fases:
  - Seleção das Ferramentas: foram pesquisadas ferramentas disponíveis na internet para apresentar para a especialista e, assim escolher três que seriam utilizados no trabalho;
  - Utilização: as três ferramentas selecionadas foram utilizadas a fim de fazer um levantamento dos seus aspectos técnicos para realizar um paralelo entre elas. Os aspectos técnicos são apresentados na seção 3.1.1.;
  - Paralelo: nesta fase, foi realizado um paralelo entre as ferramentas, comparando os aspectos técnicos delas.
- **Implementação do Ambiente:** a etapa de implementação foi dividida em duas fases:
  - Definição: nesta fase foi definida a ferramenta, bem como interface, funcionalidades e funcionamento.
  - Codificação: com a ferramenta definida, foi realizada a codificação que utilizou os materiais definidos na seção 3.2.

- **Testes Funcionais:** após a implementação da ferramenta, foram realizados testes das funcionalidades que a ferramenta possui. Quando identificado problemas no funcionamento, era feita uma revisão das funcionalidades, a fim de identificar o que está incorreto e, assim, retornar para a fase de implementação. O projeto ficou neste ciclo até que alcançou o resultado desejado.

3.1.1 Critérios para Paralelo entre as Ferramentas

Os critérios definidos para a realização do paralelo entre as ferramentas estudadas, baseados nos aspectos técnicos, foram:

- **orientações da ferramenta para utilização:** avalia se a ferramenta oferece instruções para seu uso;
- **documentação para instalação:** avalia se a ferramenta oferece documentação para instalação;
- **conteúdo abordado de forma clara:** avalia se o conteúdo é apresentado de modo a facilitar o entendimento;
- **contexto adequado ao design e atraente:** avalia a interface da ferramenta, animação e posicionamento dos elementos;
- **complexidade de instalação e desinstalação:** avalia se a instalação e desinstalação da ferramenta é simples;
- **interação aluno X ferramenta:** avalia a interação da ferramenta com o aluno, a respeito de ações que o aluno pode realizar durante a simulação;
- **complexidade de compreensão sem instrutor:** avalia a complexidade de compreensão sem que seja necessário o acompanhamento de um especialista;
- **clareza na exposição das informações:** avalia as informações que a fermenta apresenta durante a simulação;
- **fidelidade ao material didático:** avalia se o modo como é apresenta a simulação é fiel ao material presente nos livros de Silberschatz e Tanembaum que são utilizados pela especialista;
- **complexidade de utilização:** avalia se a ferramenta é de fácil uso;
- **plataforma:** indica se a plataforma é *Web* ou *Desktop*.

3.2 MATERIAIS

Para o desenvolvimento da ferramenta web foram utilizados os seguintes materiais:

- **HTML (*Hypertext Markup Language***): linguagem de marcação utilizada para estruturação do conteúdo de páginas web, utilizado para definir a estrutura da ferramenta a ser implementada;
- **Bootstrap:** framework que utiliza HTML, CSS (*Cascading Style Sheets*), e JS (*Java Script*) para desenvolvimento de projetos responsivo e focado para dispositivos móveis na web. Será utilizado para construir a interface da ferramenta;
- **AngularJS:** framework que realiza estruturação de aplicações web, através do HTML, fornecendo funções para a criação de funcionalidades necessárias para a aplicação. Será utilizado para implementar o mecanismo de paginação.

4 RESULTADOS E DISCUSSÃO

Esta seção apresenta a ferramenta *web* que foi desenvolvida para auxiliar no processo de aprendizagem da disciplina de SO. A ferramenta é capaz de realizar, de modo visual, a simulação do mecanismo de paginação através da interação entre a aplicação e o usuário.

Santos (2017) apresenta sete categorias de ferramentas educacionais, sendo elas: tutoriais, programação, aplicativos, exercícios e práticas, multimídia e Internet, simulação e modelagem e jogos. A ferramenta desenvolvida neste trabalho pertence a categoria de simulação, que oferece ao aluno uma experiência difícil de ser reproduzida em aula, visto que o SO não possibilita ao aluno visualizar certas ações realizadas, como o gerenciamento de memória.

Esta ferramenta foi desenvolvida com o objetivo de complementar os conteúdos vistos em sala de aula e encontrados nos livros e ficará disponível na internet para acesso *online*, de forma que qualquer pessoa possa acessar. O objetivo da ferramenta é auxiliar o aluno a compreender o funcionamento do mecanismo de paginação, assim, apresentar visualmente as ações que ocorrem durante a simulação. Também, a interação do aluno com a ferramenta tem o objetivo de fazer com que o aluno aprenda através de acertos e erros.

O desenvolvimento teve como base o estudo realizado sobre as ferramentas disponibilizadas na Internet, buscando aproveitar os pontos positivos e melhorar os pontos negativos; as observações e o material didático da professora especialista da área; e os conteúdos dos livros didáticos.

Os aspectos técnicos subjetivos, que são os que não necessitam de que a ferramenta seja submetida a um público para avaliação, já foram avaliados. Sendo, então, orientação da ferramenta para utilização, a utilização da ferramenta é simples e tudo ocorre em tempo real, visualmente e com notificações, assim, fazendo com que o usuário tenha noção da utilização da ferramenta; documentação para instalação, não necessita, já que a ferramenta é acessada pelo navegador; complexidade de instalação e desinstalação, não possui, já que a ferramenta é acessada pelo navegador; fidelidade ao material didático, a ferramenta segue fielmente o material didático que é utilizado pela especialista, com acréscimo apenas das cores para melhor visualização; e, por fim, a plataforma da ferramenta é web.

A seguir, é apresentada a arquitetura do software implementado; a interface da ferramenta; a simulação do mecanismo de paginação; e os principais trechos de códigos.

4.1 ARQUITETURA DO SOFTWARE

Para ajudar a compreender o funcionamento da ferramenta, foi desenvolvida a arquitetura do *software* definindo através de camadas a organização e comunicação entre os elementos do *software*. A Figura 13 apresenta a arquitetura do *software* desenvolvido neste trabalho.

Figura 13. Arquitetura do software

Conforme ilustrado pela Figura 13, o Ambiente é a parte visível para o usuário, em que há a interação do usuário com o ambiente e que apresenta graficamente a simulação do mecanismo. A Camada de Negócio é responsável por toda parte lógica da aplicação, que consiste em controlar o fluxo da informação, assim, sendo intermediária entre o Ambiente e a parte responsável pela simulação da Paginação. E, por fim, a Paginação representa a simulação do mecanismo de gerência de memória.

Primeiro, foi implementado o ambiente. Para o desenvolvimento desta parte foi utilizado o HTML e Bootstrap. Depois, foi implementada a camada de negócio e, por fim, o mecanismo de paginação. A camada de negócios e o mecanismo de paginação foram implementados utilizando o AngularJS.

4.2 AMBIENTE

A ferramenta foi projetada para oferecer um ambiente amigável e responsivo. Um ambiente amigável influencia diretamente no bem-estar do usuário ao manusear a ferramenta.

Por isso, a ferramenta apresenta uma quantidade mínima necessária de conteúdo nas telas para evitar poluição visual e os locais de entrada de dados são de fácil localização. Vale ressaltar que, os locais de entrada de dados possuem tratamento a fim de evitar que sejam informados dados incorretos que possam interromper a simulação. Também, apresenta mensagens ao usuário a fim de notificar o sucesso ou erro das ações realizadas. Este ambiente amigável, ajuda no uso do *software* pelo aluno.

Com a interface responsiva do ambiente, independentemente das dimensões da tela do dispositivo em que a ferramenta estiver sendo acessada, a interface se adaptará para melhor visualização. Desta forma, a simulação pode ser realizada em qualquer aparelho.

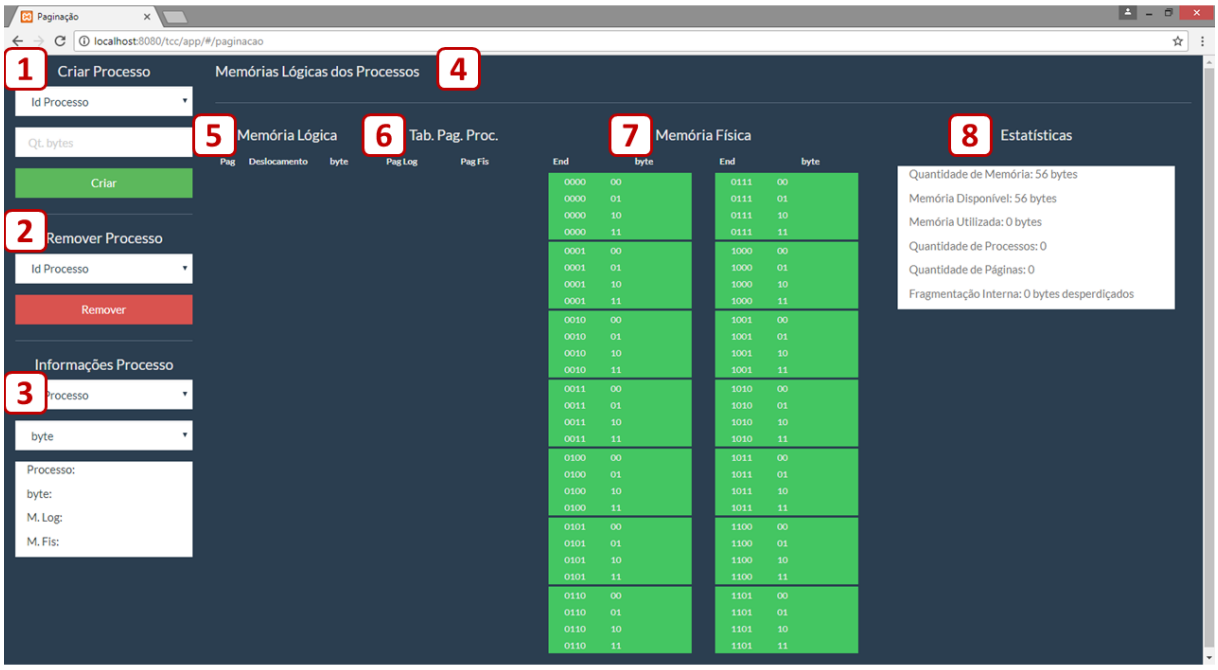
Quanto a visualização da representação gráfica, os livros trabalham apenas com conceitos abstratos, já que o SO não apresenta graficamente o funcionamento do mecanismo de paginação. Para melhorar essa visualização a fim de diferenciar como é apresentado nos livros, mas sem perder o foco do material didático, foram utilizadas cores para identificação dos processos, que foi uma estratégia utilizada visando ajudar o aluno a compreender melhor e mais rápido os conceitos do funcionamento do mecanismo de paginação.

As ações que o usuário pode realizar na simulação do mecanismo de paginação são: criar um processo, remover um processo, ver informações de endereços lógicos e físicos a partir da seleção de um byte da memória lógica um processo e visualizar a representação gráfica da memória lógica de um processo. Assim, o usuário pode compreender melhor como funciona cada ação que é realizada pelo SO. As seções a seguir apresentam a interface da ferramenta para computadores e para celulares.

4.2.1 Ambiente no Computador

Na utilização da ferramenta em um navegador de um computador, o usuário visualizará a tela conforme apresentado na Figura 14.

Figura 14. Tela mecanismo de paginação computador



A Figura 14 apresenta a tela de simulação do mecanismo de paginação antes do usuário realizar qualquer ação na ferramenta:

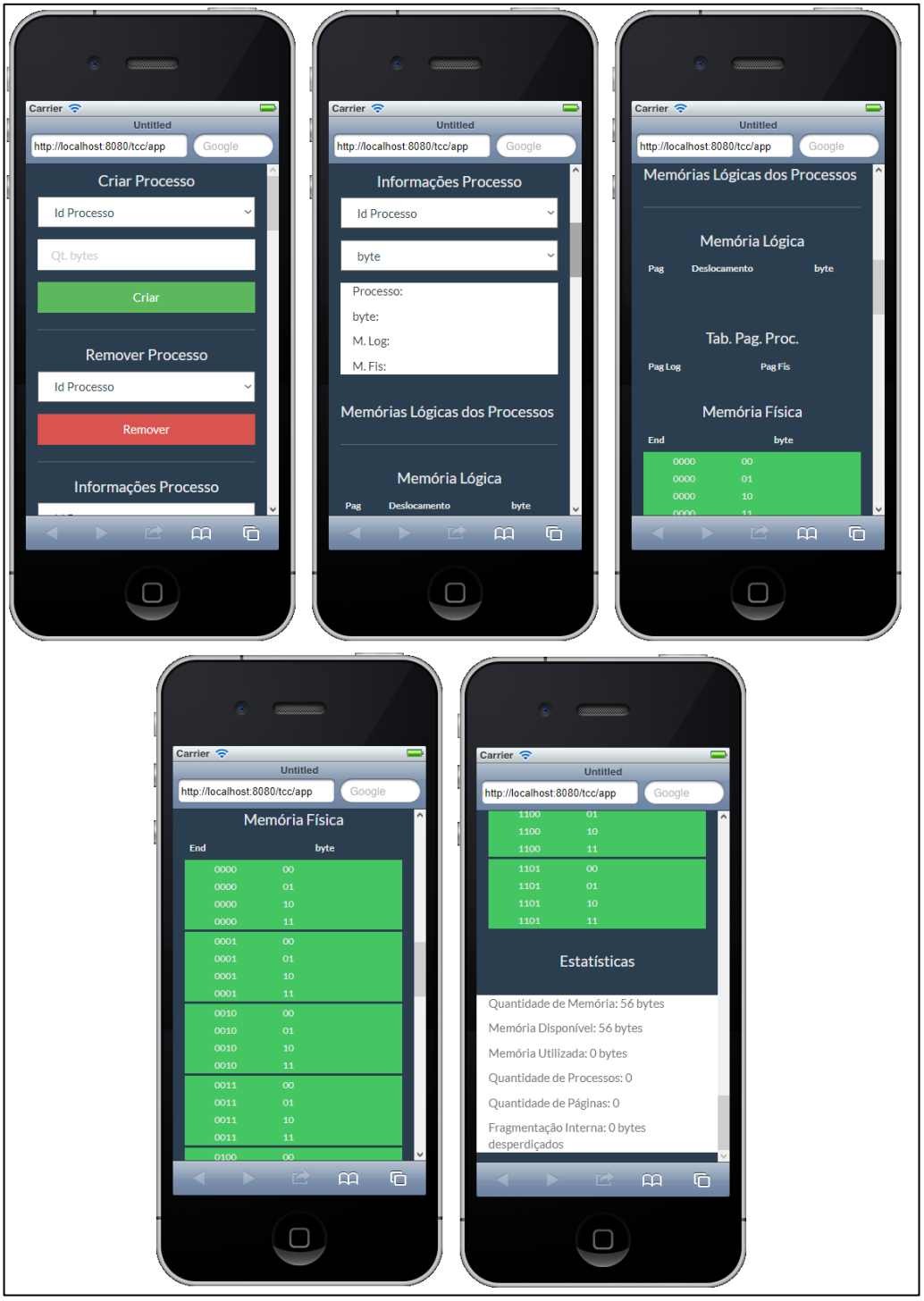
- a área de número 1 apresenta onde o usuário irá criar um novo processo com uma identificação e seu tamanho correspondente em bytes, sendo que o usuário pode criar no máximo quatorze processos cada um com até 4 bytes, em que a soma de bytes total dos processos seja de 56 bytes, que é o tamanho da memória física do simulador;
- a área de número 2 é onde o usuário pode remover um processo, selecionando-o através de um *combobox* que listará todos os processos criados e que ainda estão em execução;
- a área de número 3 possibilitará ao usuário obter informação da área de memória de um determinado processo indicando através de um *combobox* o processo desejado e o byte desejado;
- a área de número 4 apresenta a memória lógica que é gerenciada pelo SO, nesta área serão listados todos os processos ativos. O usuário pode selecionar um processo para visualizar sua memória lógica, sendo que, cada processo será identificado por uma cor, o que facilita a localização na memória física;
- a área de número 5 apresenta a memória lógica do processo, que foi selecionada na área de número 4 ou acabou de ser criada;
- a área de número 6 apresenta a tabela de páginas do processo selecionado, que é utilizada para localizar onde cada página da memória lógica foi alocada na memória física;

- a área de número 7 é a representação gráfica da memória física, onde cada página é alocada durante a execução de um processo, as páginas na cor verde indicam que aquela página está livre para uso. Quando um processo é criado, as páginas físicas alocadas por ele são representadas da cor que o identifica na área 4 e na coluna byte é inserida a identificação do byte que está armazenado em um endereço específico. Quando não é apresentada a identificação do byte em uma página alocada pelo processo, significa que a página possui fragmentação interna, ou seja, ela está reservada para o processo, mas não pode ser utilizada por outro, isso é explicado na seção 2.2.1 na Figura 5;
- a área de número 8 apresenta as informações gerais do simulador, como, quantidade total de memória, memória disponível, memória utilizada, quantidade de processos em execução, quantidade total de páginas e fragmentação interna.

4.2.2 Ambiente no Celular

Utilizando a ferramenta em um navegador de um celular, o usuário visualizará a ferramenta conforme apresentado na Figura 15.

Figura 15. Tela mecanismo de paginação celular



A Figura 15 apresenta a tela de simulação do mecanismo de paginação antes do usuário realizar qualquer ação na ferramenta, também, é apresentada a interface utilizando um smartphone e como os elementos são organizados para melhor visualização da simulação.

Apesar da tela de um dispositivo móvel possuir dimensões menores do que uma tela de um computador, o usuário consegue visualizar o processo de gerenciamento de memória. A

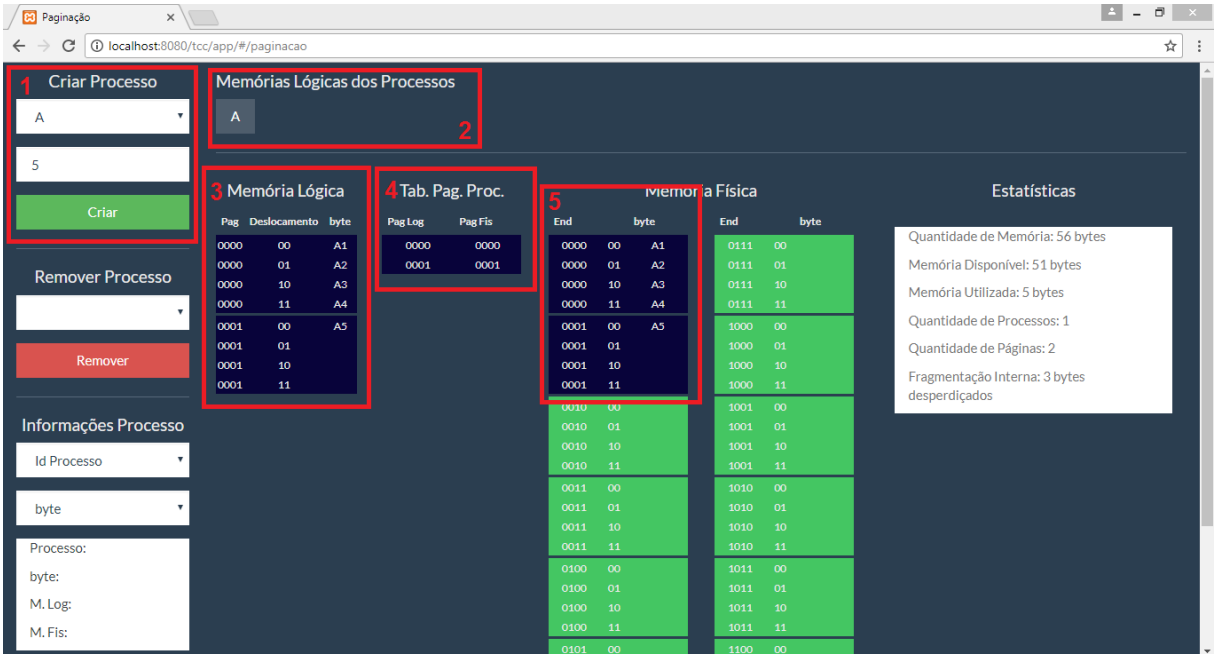
ferramenta funciona do mesmo modo, tanto em dispositivo móvel, como em computador, a única diferença é modo como os componentes da ferramenta são organizados para se ajustar às dimensões da tela do dispositivo móvel.

4.3 SIMULAÇÃO DO MECANISMO DE PAGINAÇÃO

A seguir é apresentado o funcionamento da simulação do mecanismo de paginação, considerando a utilização da ferramenta através de um computador. Para a simulação, serão utilizadas as seguintes ações: criar processo **A** com 5 bytes, criar processo **B** com 12 bytes, criar processo **C** com 15 bytes, criar processo **D** com 7 bytes, criar processo **E** com 8 bytes, remove processo **D** e visualizar memória lógica e física do byte **E3** do processo **E**.

A Figura 16 apresenta a tela da ferramenta ao criar o processo **A**.

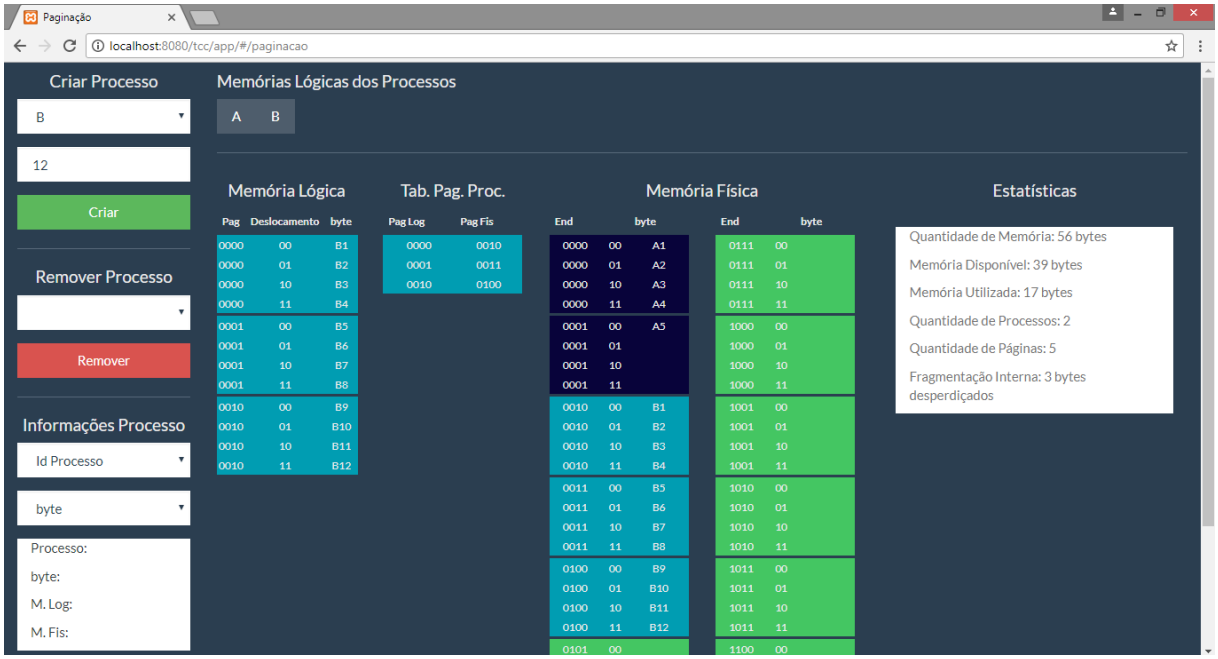
Figura 16. Criar processo A



Conforme apresentado na Figura 16, na área indicada pelo número 1 o usuário preencheu os campos e criou o processo **A** com 5 bytes. Em seguida, a ferramenta apresenta o processo **A** que acabou de ser criado na área indicada pelo número 2. Na área indicada pelo número 3 a ferramenta apresenta a memória lógica gerada para o processo a partir da quantidade de bytes informada pelo usuário. Na área indicada pelo número 4 a ferramenta apresenta a tabela de páginas do processo. Por fim, na área de número indicada pelo número 5 a ferramenta aloca as páginas lógicas do processo nas primeiras páginas físicas disponíveis. Este é o fluxo que a ferramenta segue sempre ao criar um novo processo.

A Figura 17 apresenta a tela da ferramenta ao criar o processo **B**.

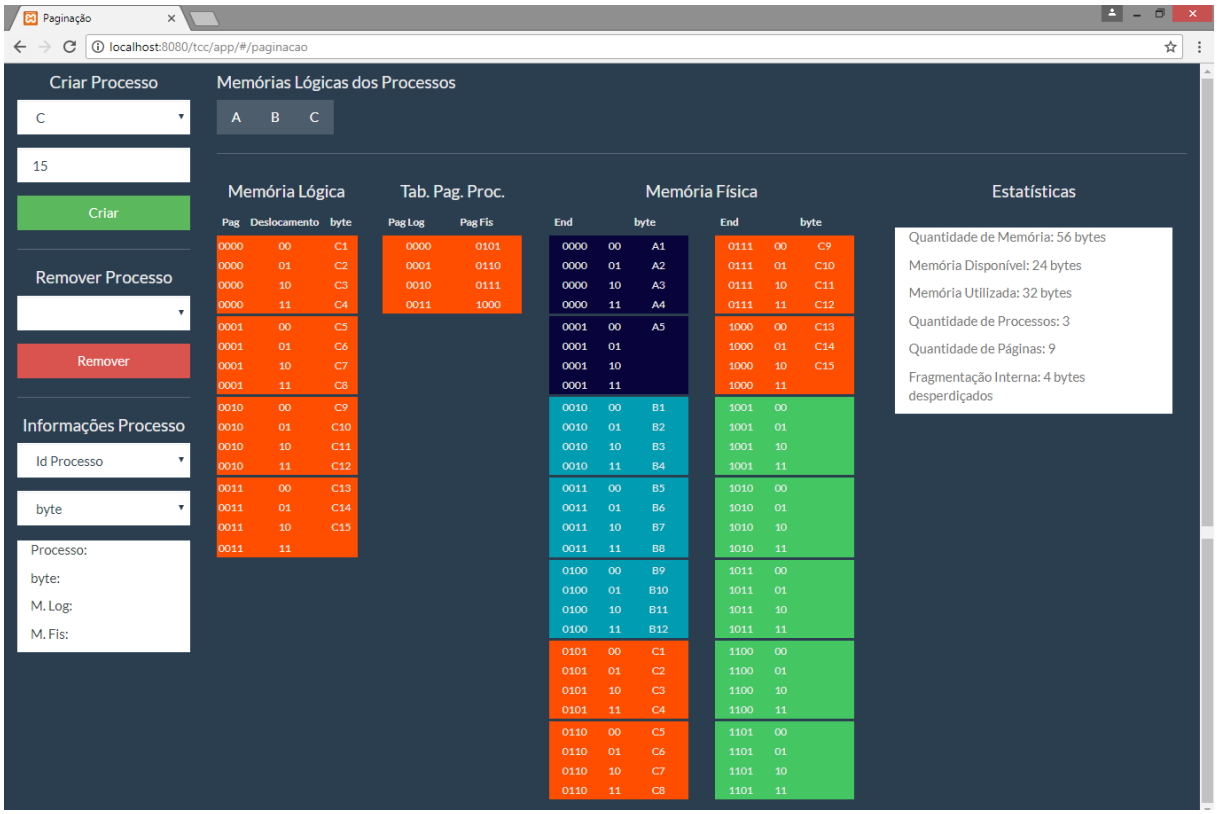
Figura 17. Criar processo B



Conforme apresentado na Figura 17, o usuário criou o processo **B** com 12 bytes. Na memória lógica do processo **B** foram geradas três páginas lógicas que foram alocadas nas três primeiras páginas livres da memória física.

A Figura 18 apresenta a tela da ferramenta ao criar o processo **C**.

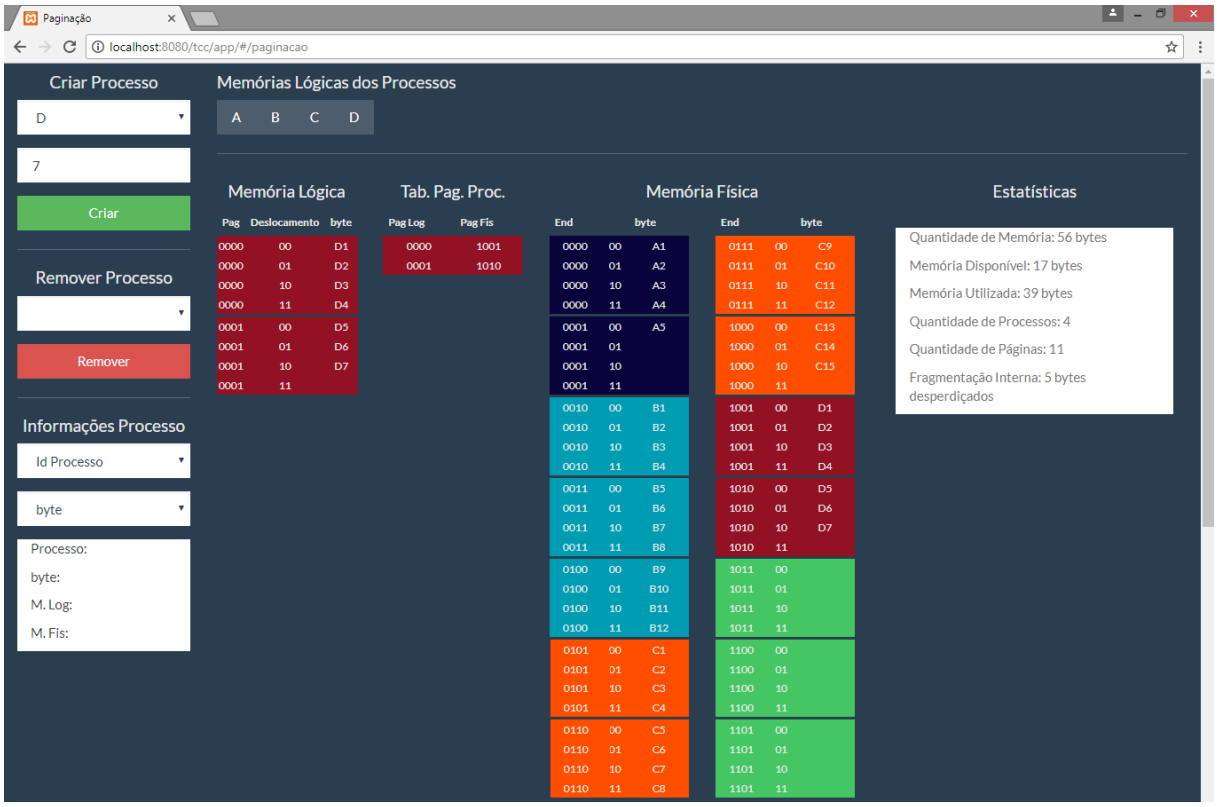
Figura 18. Criar processo C



Conforme apresentado na Figura 18, o usuário criou o processo **C** com 15 bytes. Na memória lógica do processo **C** foram geradas quatro páginas lógicas que foram alocadas nas quatro primeiras páginas livres da memória física.

A Figura 19 apresenta a tela da ferramenta ao criar o processo **D**.

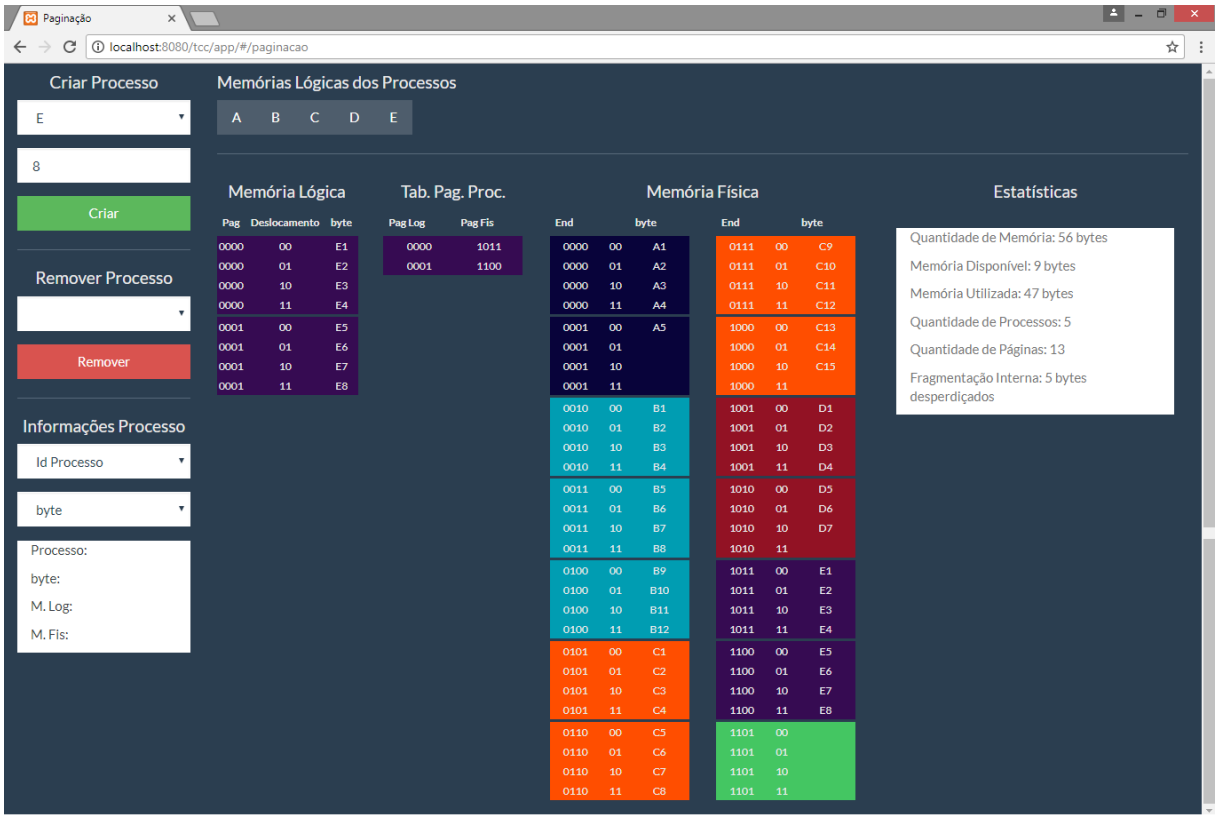
Figura 19. Criar processo D



Conforme apresentado na Figura 19 o usuário criou o processo **D** com 7 bytes. Na memória lógica do processo **D** foram geradas duas páginas lógicas que foram alocadas nas duas primeiras páginas livres da memória física.

A Figura 20 apresenta a tela da ferramenta ao criar o processo **E**.

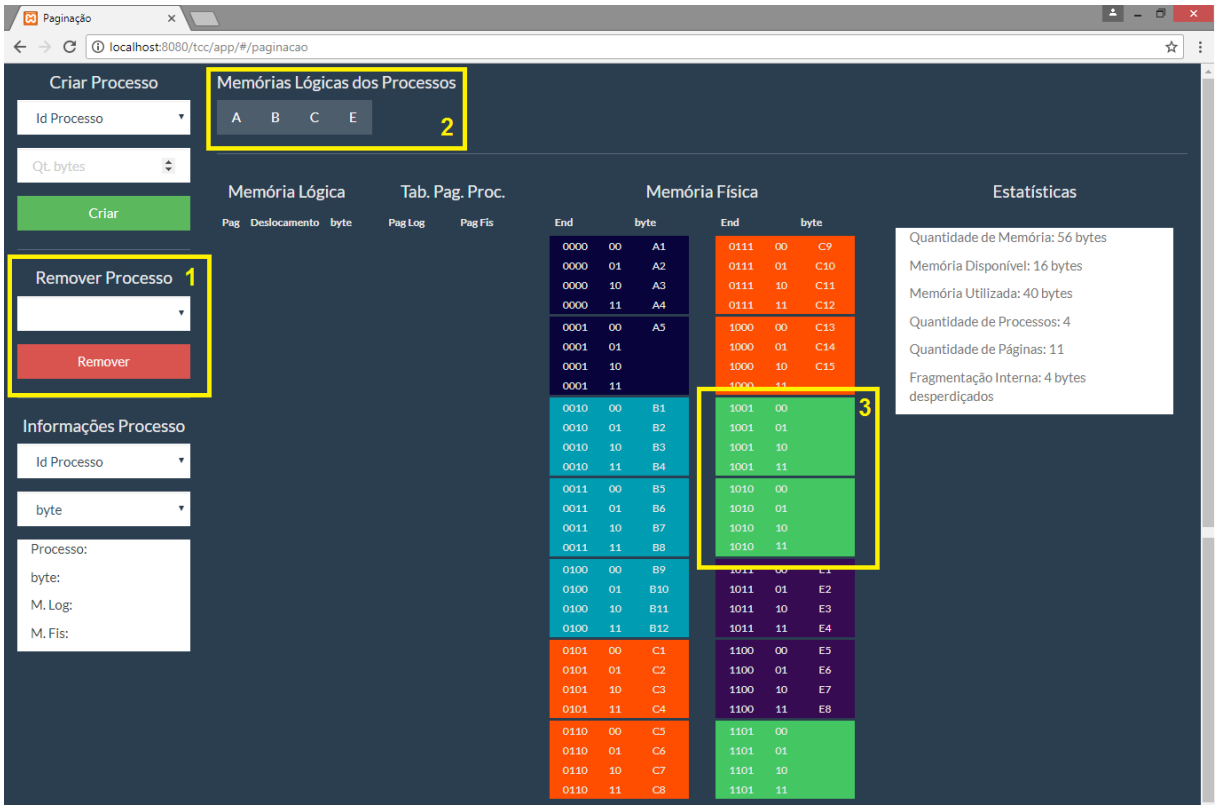
Figura 20. Criar processo E



Conforme apresentado na Figura 20, o usuário criou o processo **E** com 8 bytes. Na memória lógica do processo **E** foram geradas duas páginas lógicas que foram alocadas nas duas primeiras páginas livres da memória física.

A Figura 21 apresenta a tela da ferramenta ao remover o processo **D**.

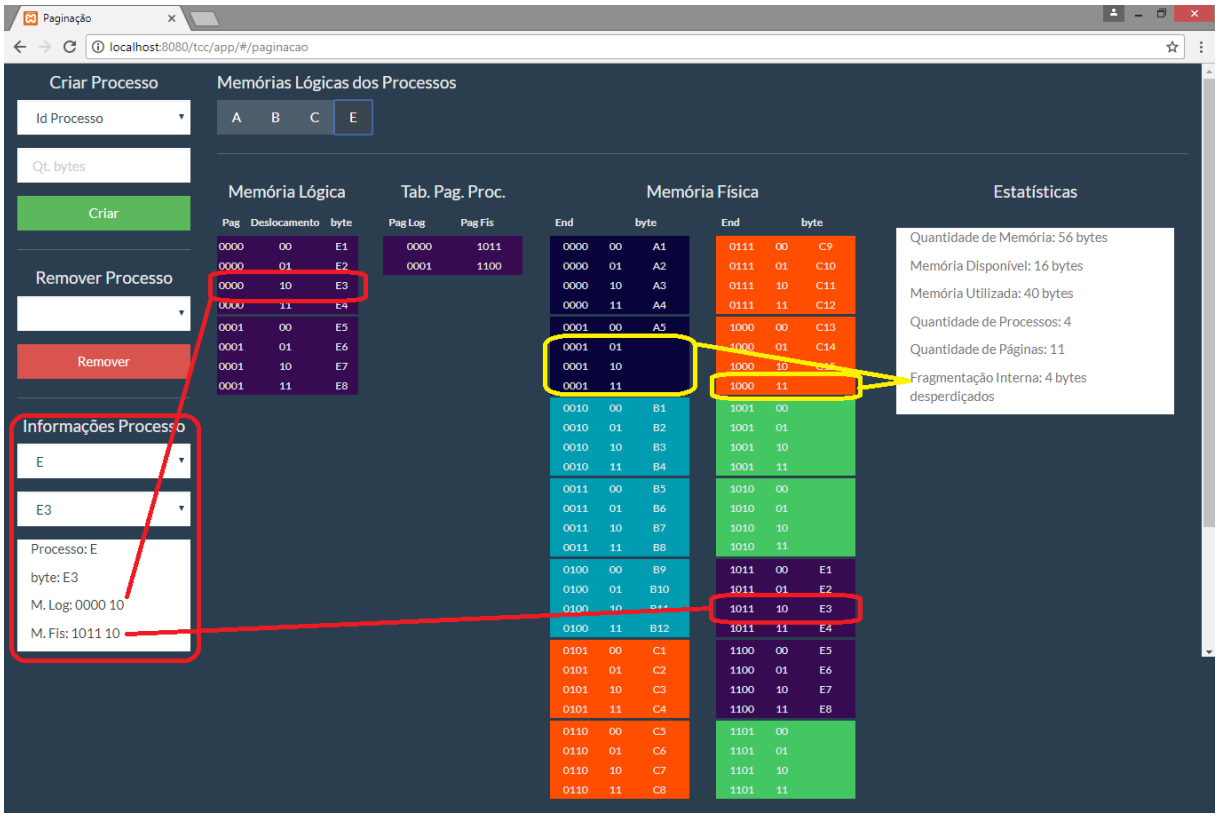
Figura 21. Remoção processo D



Conforme apresentado na Figura 21, na área indicada pelo número 1 o usuário selecionou o processo **D** e em seguida o removeu. Com a remoção do processo **D**, a ferramenta atualizou a lista de memórias lógicas ativas, que está destacada na área indicada pelo número 2, e removeu o processo **D** da memória física, assim, liberando as duas páginas que estavam ocupadas que estão destacadas na área indicada pelo número 3.

A Figura 22 apresenta a tela da ferramenta quando o usuário solicita informações do byte **E3** do processo **E.** Nesta imagem, estão destacadas as fragmentações internas que a situação gerou na memória física.

Figura 22. Localização byte E3 e fragmentações



Conforme apresentado na Figura 22, na área de informações do processo, destacada em vermelho, o usuário selecionou o processo **E** e, em seguida, selecionou o byte **E3**. A ferramenta apresenta as informações sem que seja necessário o usuário acionar um botão para buscar as informações. Na Figura está destacado o relacionamento do endereço da memória lógica com a localização do byte **E3** na página lógica e do endereço da memória física com a localização do byte **E3** na memória física.

Ainda na Figura 22, as áreas destacadas em amarelo indicam que as páginas possuem fragmentação interna, já que os espaços que não possuem uma identificação são sobras dos processos que estão alocados na memória física, que não podem ser alocadas por outros processos. Também, as áreas destacadas apontam para o campo de estatísticas que apresenta a quantidade de bytes que é desperdiçada na memória física por fragmentação interna.

4.4 CODIFICAÇÃO

Esta seção apresenta alguns trechos de códigos da implementação da simulação. Todos os trechos apresentados foram implementados com o AngularJS.

A Figura 23 apresenta a função **criar** responsável por criar um processo, que é chamada sempre que o usuário ativa o evento para criar processo. Nela, são definidos valores, feitas verificações e chamadas as funções responsáveis por alocar o processo criado.

Figura 23. Função para criar processo

```
324     //CRIAR PROCESSO ********************************************************************************
325     $scope.criar = function(processo){
326         processo.qtBytesC = 0;
327         processo.codLocalizador = codLocalizador;
328         processo.ativo = 0; // 0 = ATIVO, 1 = INATIVO
329
330         processoTemp = processo; // PROCESSOTEMP ARMAZENA TEMPORARIAMENTE OS DADOS DO PROCESSO GLOBALMENTE
331
332         verificaBytes(); //VERIFICAR QUANTIDADE DE BYTES DO PROCESSO
333
334         var contPag = 0;
335         //FOR PARA VERIFICAR QUANTIDADE DE PÁGINAS LIVRES NA MEMÓRIA FÍSICA
336         for(var i = 0; i < listaProcessos.length; i++){
337             if(listaProcessos[i].ativo == 0){
338                 contPag = contPag + listaProcessos[i].qtPag;
339             }
340         }
341
342         //IF PARA VERIFICAR SE A QUANTIDADE DE PAG DO PROCESSO NÃO VAI EXCEDER A QT DE PAG FISICA LIVRE
343         if(processoTemp.qtPag + contPag <= 14){
344             codLocalizador++;
345
346             processoTemp.cor = definirCor(); //DEFINIR COR QUE ESTEJA LIVRE PARA O PROCESSO
347             identificarByte(); //MÉTODO PARA IDENTIFICAR CADA BYTE DE UM PROCESSO
348             var p = angular.copy(processoTemp); //CÓPIA DO OBJETO DE PROC TEMPORÁRIO PARA VAR AUXILIAR
349             listaProcessos.push(p); //PROCESSO CRIADO É INSERIDO NO ARRAY DE PROCESSOS ATIVOS
350             apresentarProcAtivRem(); //CHAMADA DE FUNÇÃO PARA INSERIR PROCESSO CRIADO NA LISTA DE MEMÓRIAS LÓGICAS DOS PROCESSOS
351
352             Notification.success('Processo "' + listaProcessos[listaProcessos.length-1].id +
353                 '" com ' + listaProcessos[listaProcessos.length-1].bytes + ' bytes criado com sucesso! ');
354
355             atualizarEstatística(); //MÉTODO PARA ATUALIZAR ESTATÍSTICAS
356         }
357         else{
358             Notification.warning('Não foi possível criar o processo :( insira um processo menor!');
359         }
360     }
```

Na Figura 23, na linha 332 é chamada a função **verificaBytes** que é utilizada para verificar a quantidade de bytes que o processo possui, a fim de saber quantas páginas o processo possuirá. Da linha 336 até a 340, são verificadas quantas páginas livres a memória física possui, isso para realizar verificação na linha 343. Na linha 343 é verificado se a quantidade de páginas do processo que está para ser criado cabem na memória física, isto para evitar que seja criado um processo que não caiba no espaço que está livre no momento. Na linha 344, é realizado um incremento na variável global responsável por dar um código de identificação para cada processo. Na linha 347, é chamada a função **identificaByte** que identifica cada byte de um processo, por exemplo, se um processo **A** possui 3 bytes, eles serão identificados por **A1**, **A2** e **A3**. Na linha 350, é chamada a função **apresentarProcAtivRem** para inserir o processo no

array de processos ativos para ser apresentado na área de memórias lógicas dos processos. Após o processo ser criado, na linha 355, é chamada a função **atualizarEstatistica** para atualizar e apresentar as estatísticas do simulador.

A Figura 24 apresenta a função **identificarByte**, que é responsável por identificar com uma nomenclatura cada byte de um processo antes de alocá-lo na memória lógica e física.

Figura 24. Função identifica byte

```
385    //IDENTIFICAR CADA BYTE DO PROCESSO
386    identificarByte = function(){
387        var cont = 1;
388        var aux = new Object();
389        var contPag = 1;
390        var x = 0;
391
392        for(var i = 0; i < processoTemp.qtBytesC; i++){
393            //ARMAZENAMENTO DE VALORES PARA ASSOCIAR A CADA BYTE
394            if(cont <= processoTemp.bytes){
395                aux.id = processoTemp.id;
396                aux.byte = processoTemp.id + cont; //NOMEIA O BYTE
397                aux.codLocalizador = processoTemp.codLocalizador;
398                aux.qtPag = processoTemp.qtPag;
399                aux.estaAlocado = 0;
400                aux.ativo = 0;
401                aux.cor = processoTemp.cor;
402                cont++;
403            }
404            else{
405                aux.byte = " "; //IDENTIFICAÇÃO PARA FRAGMENTAÇÃO INTERNA
406                aux.codLocalizador = processoTemp.codLocalizador;
407                cont++;
408            }
409
410            //IDENTIFICA A QUAL PÁGINA PERTENCE DETERMINADO BYTE
411            if(x + 4 == i){
412                contPag++;
413                x = x + 4;
414            }
415
416            aux.codPag = contPag;
417            var p = angular.copy(aux);
418            listaBytes.push(p);
419        }
420        gerarTabLogica(processoTemp.codLocalizador); //GERAR TABELA LÓGICA DO PROCESSO
421    }
```

Na Figura 24, da linha 387 a 390 são declaradas variáveis que serão utilizadas para a identificação dos bytes. O **for** na linha 392 verifica todos os bytes do processo. Na linha 394 é verificado se a variável **cont** é menor ou igual a quantidade de bytes (quantidade que o usuário indicou); se for menor, é realizado um carregamento de dados do **processoTemp** para o objeto **aux**; na linha 396 o byte é identificado pelo nome do processo e o valor atual do **cont**, por exemplo, o processo **A** possui 4 bytes, então, os bytes serão identificados como **A1**, **A2**, **A3** e **A4** respectivamente; na linha 394 se o valor de **cont** for maior que o valor da quantidade de

bytes indicados pelo usuário, então este byte será identificado com um nome vazio " " a fim de representar a fragmentação interna, por exemplo, o processo **B** possui 3 bytes, então, os bytes serão identificados **B1**, **B2**, **B3** e " " (as aspas não são apresentadas para o usuário, é apenas um modo para indicar que é um nome vazio). Logo após e ainda dentro do **for**, da linha 411 a 414 é identificado a qual página cada byte pertence. Em seguida, os dados do objeto **aux** são copiados para um *array* que armazena em lista todos os bytes de todos os processos e por fim, na linha 420 é chamada a função **gerarTabLogica** para gerar a tabela de páginas lógicas do processo e apresentar graficamente para o usuário.

A Figura 25 apresenta a função **gerarTabLogica**, que é responsável por gerar as tabelas lógicas de um processo e apresentar graficamente para o usuário.

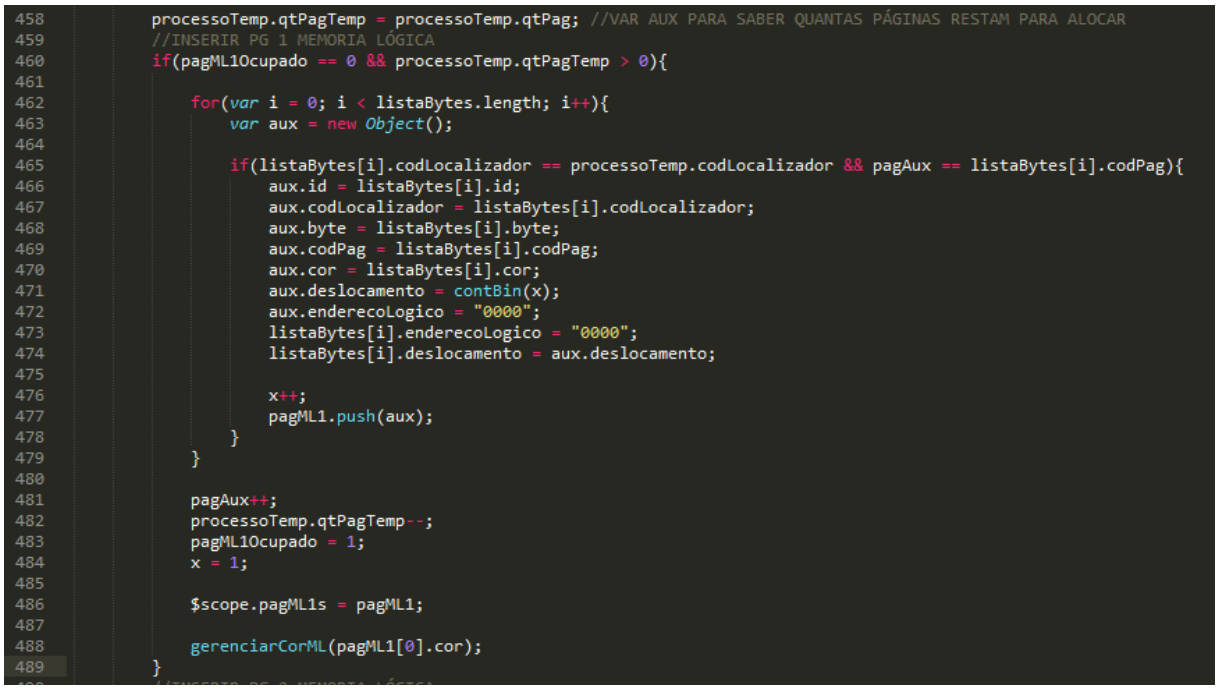
Figura 25. Função gerar tabela lógica parte 1

```
423     //GERAR TABELA LÓGICA************************************************************
424     gerarTabLogica = function(cod){
425         var x = 1; //VAR AUX PARA IDENTIFICAR DESLOCAMENTO DO BYTE
426         var pagAux = 1; //PARA SABER QUAIS PÁGINAS JÁ FORAM ALOCADAS
427 
428         //LIMPAR VARIÁVEIS RESPONSÁVEIS PELO ARMAZENAMENTO DOS BYTES DA ML PARA
429         pagML1.length = 0;
430         pagML1Ocupado = 0;
```

Na Figura 25, é apresentado o código inicial da função responsável por gerar as tabelas lógicas do processo que foi criado. A função recebe um valor que é o código de localização do processo, a fim de localizar os bytes deste processo que estão armazenados no *array* de bytes global. Na linha 429 é alterado o tamanho do *array* responsável pela página lógica 1 para 0, a fim de indicar que este *array* foi limpo e está pronto para armazenar a nova página para apresentar graficamente. Na linha 430 é indicado com 0 que a página lógica 1 está vazia. O código das linhas 429 e 430 é repetido para todas as 14 páginas lógicas antes de iniciar a alocação na memória lógica.

A Figura 26 apresenta outra parte da função **gerarTablogica**.

Figura 26. Função gerar tabela lógica parte 2



```
458         processoTemp.qtPagTemp = processoTemp.qtPag; //VAR AUX PARA SABER QUANTAS PÁGINAS RESTAM PARA ALOCAR
459         //INSERIR PG 1 MEMORIA LÓGICA
460         if(pagML1Ocupado == 0 && processoTemp.qtPagTemp > 0){
461
462             for(var i = 0; i < listaBytes.length; i++){
463                 var aux = new Object();
464
465                 if(listaBytes[i].codLocalizador == processoTemp.codLocalizador && pagAux == listaBytes[i].codPag){
466                     aux.id = listaBytes[i].id;
467                     aux.codLocalizador = listaBytes[i].codLocalizador;
468                     aux.byte = listaBytes[i].byte;
469                     aux.codPag = listaBytes[i].codPag;
470                     aux.cor = listaBytes[i].cor;
471                     aux.deslocamento = contBin(x);
472                     aux.enderecoLogico = "0000";
473                     listaBytes[i].enderecoLogico = "0000";
474                     listaBytes[i].deslocamento = aux.deslocamento;
475
476                     x++;
477                     pagML1.push(aux);
478                 }
479             }
480
481             pagAux++;
482             processoTemp.qtPagTemp--;
483             pagML1Ocupado = 1;
484             x = 1;
485
486             $scope.pagML1s = pagML1;
487
488             gerenciarCorML(pagML1[0].cor);
489         }
```

Na Figura 26, ainda é apresentado um trecho de código da função de gerar tabela lógica. Na linha 458 é criado um atributo **qtPagTemp** no objeto **processoTemp** para armazenar a quantidade total de páginas que o processo possui, isto é necessário para realizar tratamento para a aplicação não inserir uma quantidade de páginas inválida. Na linha 460 é feita uma verificação, se a página lógica 1 está vazia e se a quantidade de páginas que ainda restam para ser alocadas é maior que 0. Se verdadeiro, então, é percorrido o array **listaBytes** e em seguida, na linha 465, é verificado o código localizador e a página do byte. Se verdadeiro, então é armazenado os dados necessários para apresentação na memória gráfica e manipulações futuras em outras funções. Na linha 471 é chamada a função **contBin** que identificará o deslocamento (00, 01, 10 ou 11) do byte dentro da página. Na linha 472 é identificado o endereço lógico da memória em binário, este valor varia de uma página para outra. Na linha 486, os dados da página que acabou de ser criada são enviados para apresentar graficamente para o usuário. Por fim, na linha 488 é chamada a função **gerenciarCorML** para alterar a cor fisicamente das páginas lógicas que são apresentadas para o usuário de acordo com a cor que foi atribuída ao processo. É utilizada a mesma lógica para criar as outras 13 páginas lógicas e alocar as páginas físicas.

A Figura 27 apresenta a função **gerarTabPagProc**, que é responsável por gerar a tabela de páginas do processo após um processo ser alocado na memória física.

Figura 27. Função para gerar tabela de páginas do processo

```
1448    //GERAR TABELA DE PÁGINAS DO PROCESSO
1449    gerarTabPagProc = function(loc){
1450        var auxTab = [];
1451        var cor;
1452
1453        for(var i = 0; i < listaBytes.length; i = i + 4){
1454            if(listaBytes[i].codLocalizador == loc){
1455                var aux = new Object();
1456
1457                aux.id = listaBytes[i].id;
1458                aux.codLocalizador = listaBytes[i].codLocalizador;
1459                aux.codPag = listaBytes[i].codPag;
1460                aux.enderecoLogico = listaBytes[i].enderecoLogico;
1461                aux.enderecoFisico = listaBytes[i].enderecoFisico;
1462                aux.cor = listaBytes[i].cor;
1463                auxTab.push(aux);
1464                cor = listaBytes[i].cor;
1465            }
1466        }
1467        $scope.tabPagProcs = auxTab;
1468        gerenciarCorTPP(cor);
1469    }
```

Na Figura 27, a função recebe o valor da localização do processo. Na linha 1453 o *array* **listaBytes** é percorrido. Na linha 1454 é verificado se o byte pertence ao código de localização do processo que foi recebido. Se verdade, então, as informações são copiadas para um objeto auxiliar. Na linha 1467 os dados da tabela de páginas do processo são enviados para serem apresentados para o usuário. Por fim, na linha 1468 é chamada a função **gerenciarCorTPP** que é responsável por realizar o gerenciamento da cor da tabela de páginas do processo.

A Figura 28 apresenta a primeira parte da função **remover**, que é responsável por remover um processo selecionado pelo usuário.

Figura 28. Função para remover processo parte 1

```
1471    //REMOVER PROCESSO **********************************
1472    $scope.remover = function(id){
1473
1474        for(var i = 0; i < listaBytes.length; i++){
1475            if (listaBytes[i].id == id) {
1476                listaBytes[i].ativo = 1;
1477            }
1478        }
1479
1480        for(var i = 0; i < listaProcessos.length; i++){
1481            if (listaProcessos[i].id == id) {
1482                listaProcessos[i].ativo = 1;
1483            }
1484        }
```

Na Figura 28, a função recebe o **id** do processo selecionado para ser removido da simulação. Na linha 1474, é percorrido o *array* **listaBytes** e em seguida na linha 1475 é feita a verificação para identificar quais bytes pertencem ao processo que será removido. Na linha 1476 os bytes que pertencem ao processo que será removido, seu atributo de ativo receberá 1, para identificar que todo o conjunto de bytes de um determinado processo não estão mais ativos na simulação. Na linha 1480, é realizado o mesmo procedimento no *array* **listaProcessos**, que é o *array* que armazena o identificador de cada processo para apresentar na área de memória lógica dos processos para o usuário os processos que estão ativos na simulação. Nestes dois casos, não é possível excluir por completo as informações dos processos, pois nos futuros acessos nestes *arrays*, a aplicação apresentaria erro na leitura o que ocasionaria mal funcionamento da ferramenta.

A Figura 29 apresenta a segunda parte da função **remover**.

Figura 29. Função para remover processo parte 2

```
1486        //ATUALIZAR AS PAGINAS FÍSICAS
1487        for(var i = 0; i < pag1.length; i++){
1488            if(pag1[i].id == id){
1489                pag1[i].ativo = 1;
1490            }
1491        }
```

O trecho de código da Figura 29 é responsável por acessar o *array* da página física 1 e alterar para 1 o atributo ativo de cada posição a fim de indicar que aquela página não está mais sento utilizada por um processo e, está livre para outra página ser alocada nela. A mesma lógica é utilizada para todas as páginas físicas. Após todas as páginas físicas serem verificadas, é feita uma atualização da lista de processos ativos e da memória física que são apresentadas para o usuário.

Os trechos de códigos apresentados anteriormente não são todo o código da ferramenta, foram selecionadas somente as partes essenciais para o funcionamento da simulação durante a alocação e remoção de processos. Existem, ainda, trechos responsáveis pelo gerenciamento das cores das páginas e da estruturação do HTML para a apresentação das páginas lógicas e físicas que não serão apresentados aqui.

4.5 DEMONSTRAÇÃO DA TEORIA E DA FERRAMENTA

A ferramenta desenvolvida pode ser utilizada pelo professor(a) em aula para explicar os conceitos da gerência de memória e pelos alunos para reforçar o que foi visto em sala.

No que se refere ao gerenciamento de memória, segundo a professora especialista da área, os pontos que alguns alunos apresentam mais dificuldades para compreender são: representação gráfica da memória lógica, associação das páginas lógicas às páginas físicas e visualização da alocação da memória física.

No que se refere a memória lógica, a dificuldade é visualizar as memórias dos processos ativos, pois, para isso é necessário fazer o esboço da memória lógica de cada processo. Os alunos além de compreender este esboço, em determinados contextos, quando a representação gráfica não é apresentada, tem que imaginá-lo ou desenhá-lo. Por exemplo, a Figura 30 apresenta o modo como a memória lógica é vista em aula, na maioria das vezes o esboço é feito no quadro.

Figura 30. Representação da memória lógica utilizada

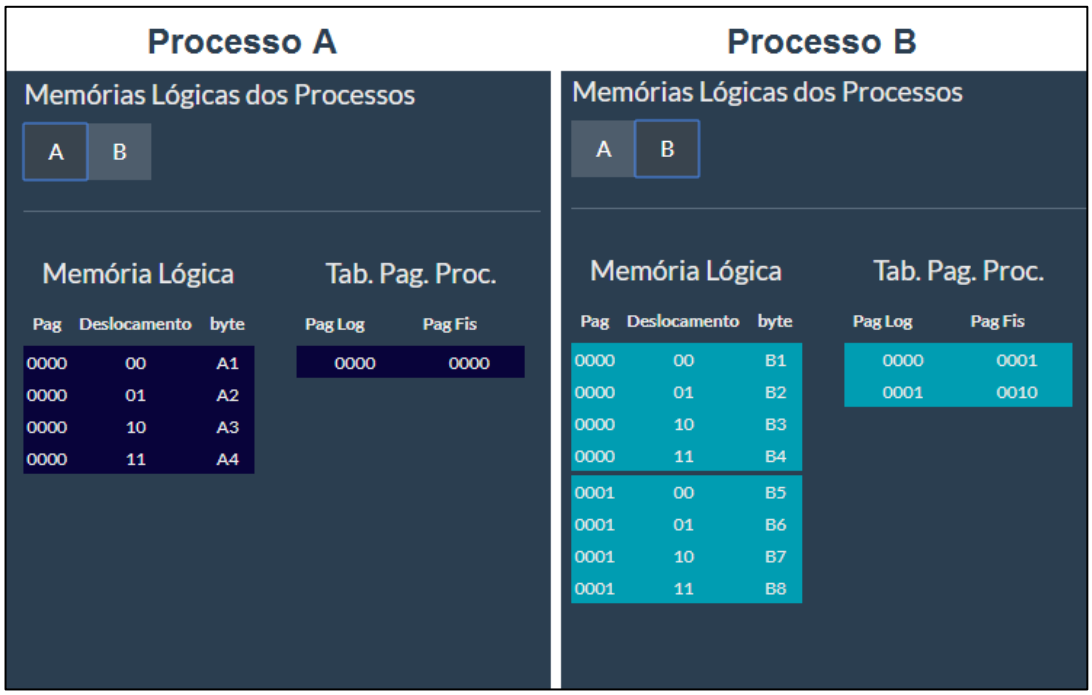
MEMÓRIA LÓGICA
Processo com 11 bytes

000 00	X1
000 01	X2
000 10	X3
000 11	X4
001 00	Y1
001 01	Y2
001 10	Y3
001 11	Y4
010 00	Z1
010 01	Z2
010 10	Z3
010 11	

Na Figura 30 é apresentado o modo como a memória lógica é vista em aula e, para visualizar mais de uma memória lógica, é necessário fazer o esboço da memória ou imaginá-la. Já na ferramenta, durante a simulação, é possível visualizar a qualquer momento qualquer memória lógica que estiver ativa, a seguir é apresentado um exemplo.

A Figura 31 mostra como a ferramenta durante a simulação apresenta as memórias lógicas, assim, possibilitando que o usuário visualize a memória lógica desejada.

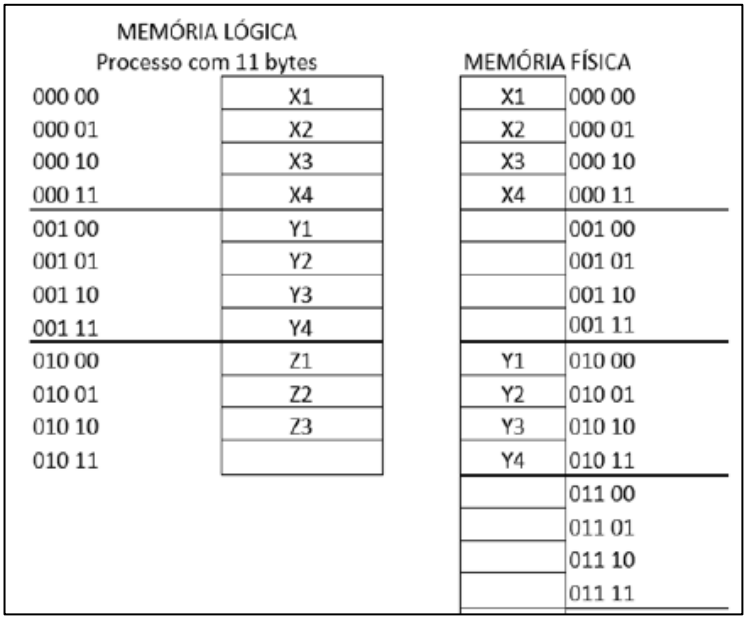
Figura 31. Representação da memória lógica feita pela ferramenta



A Figura 31 apresenta como a ferramenta facilita a visualização da memória lógica, pois o usuário poderá escolher uma das memórias lógicas ativas que são apresentadas, no caso da Figura 36, estão ativas as memórias lógicas dos processos **A** e **B**.

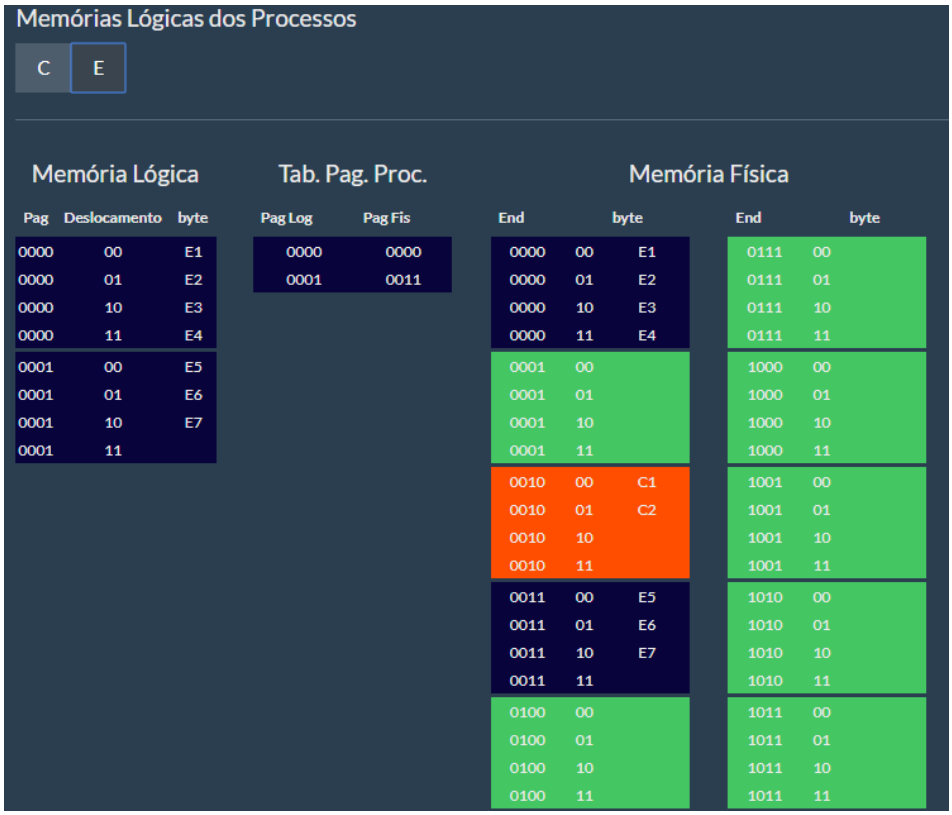
Na associação das páginas lógicas com físicas, a dificuldade está em saber onde determinado processo está alocado, o que pode acabar confundindo o entendimento do aluno. Por exemplo, a Figura 32 apresenta o modo como as páginas lógicas e físicas são vistas em aula, na maioria das vezes o esboço é feito no quadro.

Figura 32. Representação das páginas lógicas e físicas utilizadas



A Figura 33 apresenta como a ferramenta exibe as páginas lógicas e físicas para o usuário através da simulação, isso, relacionado com a Figura 32 apresentada anteriormente para demonstrar a diferença.

Figura 33. Representação das páginas lógicas e físicas feitas as pela ferramenta



A Figura 33 mostra como a ferramenta apresenta para o usuário a representação das páginas lógicas e físicas, algo que é difícil para compreender se utilizada a Figura 32 para explicação. Na Figura 33 é mais fácil localizar onde está alocado um determinado processo apenas pela sua cor. Por exemplo, o processo **E** na cor azul escuro está alocado nas páginas 0000 e 0011. Com a divisão em cores, o usuário localiza facilmente as áreas da memória física ocupadas pelo processo selecionado. Também, o que é apresentado na Figura vai ser alterado sempre que um processo for criado ou removido, assim, o usuário visualiza o que acontece em cada momento.

Diante o que foi apresentado sobre a ferramenta, consta-se que, o aluno pode simular alguma situação apresentada em sala, para reforçar ou melhorar a compreensão e, também, poderá usar a ferramenta para criar situações diversas. Com isso, o aluno poderá trabalhar em casos específicos em que possui dificuldades e, assim, sanar suas dúvidas que estejam lhe impedindo de progredir na disciplina.

5 CONSIDERAÇÕES FINAIS

O presente trabalho teve o objetivo de desenvolver um *software* que fosse capaz de simular o funcionamento do mecanismo de paginação da gerência de memória. Tendo também, como objetivos específicos, apresentar ferramentas que simulem a gerência de memória, apresentar comparativo das ferramentas estudadas, criar protótipos com base nos conceitos didáticos e ferramentas estudadas e por fim, a implementação da ferramenta proposta. Todos os objetivos propostos foram realizados, o que tornou válida a hipótese afirmada no início do trabalho, que foi "Usando as tecnologias *AngularJS, Bootstrap* e *HTML* é possível desenvolver uma ferramenta virtual capaz de simular o funcionamento do gerenciamento de memória, de maneira a facilitar a compreensão deste conteúdo".

Buscou-se desenvolver uma ferramenta didática, atrativa, fiel aos conteúdos didáticos e interativa, que sirva como ferramenta para auxiliar os alunos nos estudos. Os recursos visuais, como cores e mensagens, utilizados na ferramenta, tem o propósito de contribuir com avanços para o aprendizado do aluno, pois, pode gerar motivação no discente. Estas características da ferramenta ainda precisam ser verificadas, o que será feito em outro momento.

Vale destacar que os resultados produzidos e apresentados são de autoria própria do autor do trabalho, que o código foi produzido sem cópias. Para a apresentação do mecanismo de paginação foi seguido o padrão que, normalmente, está presente nos livros didáticos utilizados como referência.

Referente ao resultado obtido, acredita-se que a experiência que o aluno terá na utilização da ferramenta durante a simulação, será de grande valia, no sentido do aluno não simplesmente visualizar as simulações e sim produzi-las de forma a acompanhar seu próprio raciocínio lógico. Por exemplo, imaginar uma sequência de alocações e remoções e, executar ações de alocação e remoção na ferramenta a fim de verificar se o que será apresentado pela ferramenta está de acordo com o que imaginou. Ou então, como ficará a memória física se for removido um ou dois processos.

A ferramenta desenvolvida será avaliada com uma turma da disciplina de Sistemas Operacionais do CEULP/ULBRA, levando-se em consideração os parâmetros estudados, que são apresentados na seção 2.3.4 e outros critérios de usabilidade que serão definidos pelos pesquisadores que realizarão a avaliação. A finalidade da avaliação é verificar se a ferramenta desenvolvida neste trabalho corresponde ao que foi esperado e se está apta para ser utilizada como ferramenta de apoio na disciplina de Sistemas Operacionais e se ela é capaz de facilitar o estudo e a compreensão dos alunos.

Como trabalhos futuros, têm-se as seguintes possibilidades:

- avaliação do trabalho, explicada anteriormente;
- se na avaliação for verificada a necessidade de ajustes na ferramenta desenvolvida, estes ajustes podem ser realizados, por exemplo, alteração de posição dos componentes do ambiente e acréscimo de novas funcionalidades, isso, com o objetivo de melhorar a experiência do aluno com a ferramenta;
- implementação da ferramenta de simulação de paginação por demanda utilizando como base a ferramenta desenvolvida nesse trabalho;
- e, por fim, disponibilizar na ferramenta o conteúdo e exercícios relacionados à simulação do mecanismo de paginação.

REFERÊNCIAS

DEITEL, Harvey M.; DEITEL, Paul J.; CHOFFNES, David R.. **Sistemas Operacionais.** 3. ed. São Paulo: Pearson Prentice Hall, 2005. 760 p.

GADELHA, Renê N. S. et al. OS Simulator: Um Simulador de Sistema de Arquivos para Apoiar o Ensino/Aprendizagem de Sistemas Operacionais. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO - SBIE, 2010, João Pessoa. **Anais... .** Recife: Cin-ufpe, 2010. p. 1 - 9. Disponível em: <http://br-ie.org/pub/index.php/sbie/article/view/1470/1235>. Acesso em: 18 abr. 2017.

MAZIERO, Carlos Alberto. **Sistemas Operacionais:** Conceitos e Mecanismos. Curitiba: Dinf – Ufpr, 2011. 371 p. Disponível em: <http://wiki.inf.ufpr.br/maziero/lib/exe/fetch.php?media=so:so-livro.pdf>. Acesso em: 17 abr. 2017.

MINISTÉRIO DA EDUCAÇÃO CONSELHO NACIONAL DE EDUCAÇÃO. **PARECER CNE/CES N° 136/2012**: Parecer CNE/CES nº 136/2012, aprovado em 8 de março de 2012 - Diretrizes [...]. Distrito Federal: Ministério da Educação, 2012. Disponível em: <http://portal.mec.gov.br/index.php?option=com_docman&view=download&alias=11205-pces136-11-pdf&category_slug=julho-2012-pdf&Itemid=30192>. Acesso em: 03 jul. 2017.

SANTOS, Fábia Magali. **Avaliação de Software Educativo:** Reflexões para uma Análise Criteriosa. Disponível em: <http://www.educacaopublica.rj.gov.br/biblioteca/tecnologia/0001.html>. Acesso em: 14 jun. 2017.

SCHULZE, Eliete; PAULA, Simone de. **Fichas de Avaliação de Software.** Disponível em: <https://pt.slideshare.net/SchulzeEliete/avaliao-de-software-educativo-47629427>. Acesso em: 14 jun. 2017.

SILBERSCHATZ, Abraham; GALVIN, Peter Baer; GAGNE, Greg. **Fundamentos de Sistemas Operacionais.** 8. ed. Rio de Janeiro: Ltc, 2013. 515 p.

OLIVEIRA, Rômulo Silva de; CARISSIMI, Alexandre da Silva; TOSCANI, Simão Sirineo. **Sistemas Operacionais.** 4. ed. Porto Alegre: Bookman, 2010. 371 p. Disponível em: <https://books.google.com.br/books?hl=pt-BR&lr;=&id=9SdddkKFtlYC&oi=fnd&pg=PR7&dq=sistemas+operacionais&ots=kWvLm00N1B&sig=C1_BG9534HBMNswtW2_tVmhYHpo#v=onepage&q=sistemas operacionais&f=false>. Acesso em: 02 jun. 2017.

OLIVEIRA, Rômulo Silva de; CARISSIMI, Alexandre da Silva; TOSCANI, Simão Sirineo. **Sistemas Operacionais.** 2. ed. Porto Alegre: Sagra Luzzatto, 2001.

TANENBAUM, Andrew S.. **Sistemas Operacionais Modernos.** 3. ed. São Paulo: Pearson, 2009. 625 p.

TOSCANI, Simão Sirineo. **Gerência de Processador.** Disponível em: <http://www.inf.pucrs.br/~ramos/sop_gerprocessador.pdf>. Acesso em: 16 jun. 2017.

VESCE, Gabriela E. Possolli. **Ensino-aprendizagem por meio do computador.** 2017. Disponível em: <https://www.infoescola.com/educacao/ensino-aprendizagem-por-meio-do-computador/>. Acesso em: 02 nov. 2017.