Universidade Católica de Santos

Centro de Ciências Exatas e Tecnológicas

Curso de Sistemas de Informação

Trabalho de Conclusão de Curso – II

SimulaRSO – Simulador de Recursos de Sistemas Operacionais

Autor(es): André de Araújo Rodrigues

Caio Ribeiro Pereira

Orientador(a): Prof. André Luiz Vizine Pereira

Santos

2011

SimulaRSO – Simulador de Recursos de Sistemas Operacionais

Trabalho apresentado ao Curso de Sistemas de Informação do Centro de Ciências Exatas e Tecnologia da Universidade Católica de Santos como requisito obrigatório para aprovação na disciplina Trabalho de Conclusão de Curso – II.

Aluno(a): André de Araújo Rodrigues

Aluno(a): Caio Ribeiro Pereira

Orientador(a): Prof. André Luiz Vizine Pereira

Agradecimentos

Agradecemos primeiramente a Deus por nos dar condições de executar esse trabalho.

Os nossos familiares pelos apoios incondicionais durante toda jornada do nosso curso.

O professor e nosso orientador André Vizine pela sugestão do tema do projeto e seu auxílio sobre o respectivo desenvolvimento.

Resumo

O objetivo do projeto SimulaRSO é simular o comportamento dos principais algoritmos que escalonam processos, que escalonam requisições de disco e realizam paginação de memória virtual, através de animações gráficas sobre os recursos do sistema operacional e apresentando suas informações relevantes para estudos analíticos. O resultado desse trabalho gerou uma ferramenta online de ensino que auxilia o aluno durante as aulas de Sistemas Operacionais.

Palavras-chave: Sistemas Operacionais. Simulador. Escalonamento de Processos. Escalonamento de Disco. Paginação de Memória Virtual.

Abstract

The goal of the project SimulaRSO is to simulate the behavior about the main algorithms of process's scheduling, disk scheduling and virtual memory paging through graphic animations about the operating system resources and providing relevant information for analytical studies. The result of this work has created an online educational tool which helps the student during classes of Operating Systems.

Key-words: Operating Systems. Simulator. Process Scheduling. Disk Scheduling. Virtual Memory Paging.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

CPU – Central Processing Unit

C-SCAN – Circular SCAN

CSS3 – Cascading Style Sheets Version 3

EAD – Ensino a Distância

FCFS – First-Come, First-Served

FIFO – First-In, First-Out

GUI – Graphical User Interface

HTML5 - HyperText Markup Language Version 5

IDE – Integrated Drive Electronics

I/O – Input/Output

LRU – Least Recently Used

MMU – Memory Management Unit

MRU – Most Recently Used

MVC – Model-View-Controller

OPT – Optimum

SCSI – Small Computer System Interface

SJF – Shortest Job First

SRT – Shortest Remaining Time

SSTF – Shortest Seek Time First

TI – Tecnologia da Informação.

UML – Unified Modeling Language

W3C – World Wide Web Consortium

SUMÁRIO

1. INTRODUÇÃO

1.1 Contextualização

Os avanços tecnológicos na área da TI (Tecnologia da Informação) foram fundamentais para o fenômeno conhecido por Revolução Digital que vem transformando a sociedade (Calil, 2007). O computador que antigamente custava milhões de dólares e ficava restrito à indústria bélica e a pouquíssimas Universidades, hoje é vendido em prateleiras de supermercado com preços altamente convidativos. Graças ao *WWW* (World Wide Web) no início da década de 90, a internet tornou-se um dos principais meios de comunicação da atualidade, reduzindo custos, disponibilizando recursos e informação para pessoas das mais diversas áreas e interesses (Calil, 2007).

Devido a este cenário surgiram novas profissões da era digital, fazendo com que as instituições de ensino superior reformulassem suas grades curriculares oferecendo novos cursos na área de computação com o objetivo de formar egressos capacitados ao mercado de trabalho (Rodrigues & Pereira, 2010).

Um dos conteúdos essenciais que o aluno deve conhecer para poder entender não só como funciona o computador, mas também o mundo da Internet, e claro as suas configurações mais complexas, são os sistemas operacionais que governam desde um simples dispositivo de poucos recursos até a uma grande máquina que realiza processamento complexo.

1.2 Problema

Sistemas operacionais é uma disciplina que possui conteúdos importantíssimos e essenciais que ampliam o conhecimento do aluno, porém grande parte desses conceitos existe apenas na teoria e isso dificulta o aluno visualizar como seriam esses conceitos teóricos na prática.

O problema principal identificado foi a ausência de ferramentas de fácil acesso que auxilie o aluno na prática compreender todo conteúdo teórico apresentado em sala de aula, exibindo de forma clara e objetiva através de animações e interações, por exemplo, o comportamento dos algoritmos responsáveis pelo escalonamento de processos, escalonamento de disco ou o funcionamento dos algoritmos que realizam a substituição de páginas de uma memória virtual, a paginação de memória.

1.3 Problematização

Tendo em vista o problema apresentado, questiona-se:

- Como desenvolver uma ferramenta que apresente na prática de forma clara e objetiva os principais conteúdos teóricos da disciplina?
- Como oferecer um aplicativo de fácil acesso através da internet?
- Qual será o custo de desenvolvimento para esta solução?

1.4 Objetivos

1.4.1 Objetivo Geral

O objetivo geral desse trabalho é propor uma ferramenta interativa para auxiliar o professor no processo de aprendizagem do aluno.

Pretende-se com isso facilitar a assimilação do conteúdo pelo aluno, eliminando suas dúvidas em relação à disciplina de sistema operacionais.

1.4.2 Objetivos Específicos

Desenvolver uma ferramenta de ensino online para o auxílio nas aulas de sistemas operacionais que abordem assuntos relacionados a escalonamento de processos, escalonamento de disco e algoritmos de paginação de memória virtual.

A aplicação deverá apresentar o comportamento dos principais algoritmos de cada um dos assuntos através de animações gráficas.

1.4.3 Relevância

A ferramenta visa auxiliar o professor na tarefa de complementar e tornar claro os conceitos teóricos abordados em sala de aula sobre a disciplina de sistemas operacionais.

2. BIBLIOGRAFIA

2.1 Revisões Bibliográficas

1. wxProc – Simulador de Políticas de Escalonamento Multiplataforma.

É um aplicativo *desktop* [1] construído sobre linguagem de programação C/C++ com uso das bibliotecas *Standard Template Library* [2]e *wxWindow* [3]. O trabalho possibilita o aluno ter uma visão clara dos principais algoritmos de escalonamento de processos, sua GUI (interface gráfica) é simples e objetiva, exibe o comportamento dos algoritmos através de apresentações gráficas em um painel mostrando os estados finais dos processos após término da simulação (Rocha & Schneider & Alves & Silva, 2011).

2. S²O – Uma Ferramenta de Apoio ao Aprendizado de Sistemas Operacionais.

O protótipo foi desenvolvido para atender às funcionalidades de gerência de processos, foi projetado sob diagramas UML (UML, 2011) e linguagem de programação Java (Oracle, 2011), suas funcionalidades: simular as principais políticas de escalonamento, adicionar processos antes e durante a execução, manter processos organizados em fila, aumentar e diminuir a velocidade da simulação e visualizar os estados dos processos (Carvalho & Balthazar & Dias & Araújo & Monteiro, 2011).

3. SAE – Simulador de Algoritmos de Escalonamento.

É uma ferramenta didática que simula os principais algoritmos de escalonamentos vistos nas aulas de sistemas operacionais. Foi programado sob linguagem de programação C# (Microsoft, 2011). As funcionalidades do projeto são: avaliar cada ocorrência de I/O (entrada e saída), interromper um processo atual, gerar um novo processo de forma dinâmica e realizar execuções comparativas entre diferentes algoritmos (Almeida & Lima & Rabelo, 2011).

---

1 Desktop: aplicação executada em uma área de trabalho do sistema operacional.
2 Standard Template Library: biblioteca padrão da linguagem de programação C++.
3 wxWindows: biblioteca de componentes gráficos para criação de GUIs em aplicações desktop.

Abaixo segue uma tabela comparando os recursos do SimulaRSO com os projetos wxProc, S²O e SAE:

Projetos	SAE	wxProc	S²O	SimulaRSO
Escalonamento de Processos	Sim	Sim	Sim	Sim
Escalonamento de Disco	Não	Não	Não	Sim
Paginação de Memória Virtual	Não	Não	Não	Sim
Simulação Comparativa	Sim	Não	Não	Sim
Simulação Dinâmica	Sim	Não	Sim	Não
Plataforma	C#	C/C++	Java	Java, HTML5, CSS3 e JS
Suporte Multi-idioma (internacionalização)	Não	Não	Não	Sim
Idiomas	Português	Português	Português	Português e Inglês

Tabela 1 – Comparativo entre os projetos SAE, wxProc, S²O e SimulaRSO.

2.2 Embasamentos Teóricos

2.2.1 O Sistema Computacional

O computador possui diversos recursos, tanto de hardware 4quanto de software 5. O sistema operacional atua como gerenciador desses recursos, alocando programas e usuários específicos conforme a necessidade de execução de tarefas que são solicitadas pelo operador (Silberschatz & Galvin, 2008).

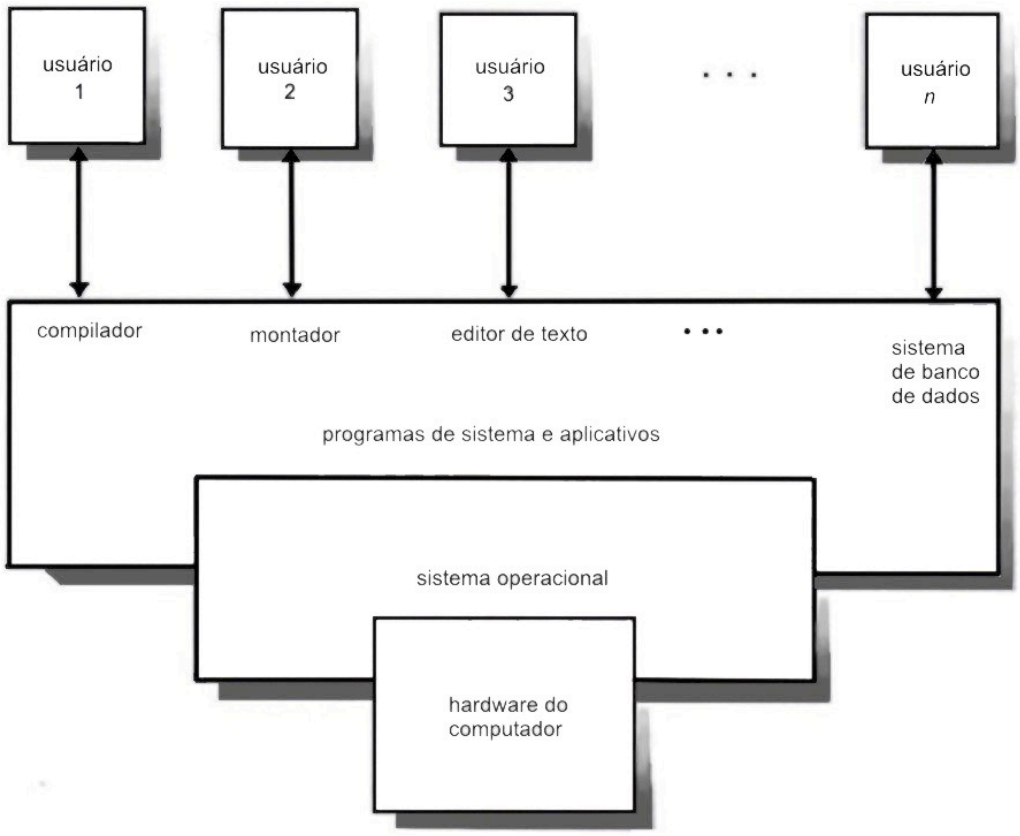

Figura 1 - Arquitetura de um sistema computacional.
Fonte: Silberschatz & Galvin, 2008.

De maneira geral, um sistema operacional é formado pelos seguintes componentes: gerenciador de processos, de memória, de arquivos, de dispositivos de I/O, de redes, sistema de proteção, um interpretador de comandos e gerenciador de serviços (Silberschatz & Galvin, 2008).

4 Hardware: material físico, componente de um computador.
5 Software: Conjunto de programas, processos e regras, e, eventualmente, de documentação, relativos ao funcionamento de um conjunto de tratamento da informação.

2.2.2 Processos

Processos nada mais são do que programas em execução, o programa em si é apenas uma entidade, ou seja, um conjunto de instruções passível de ser executado, o processo é uma instância do programa, pois ele é alocado no processador – responsável por dar "vida ao programa" – executando o aplicativo para interagir com o usuário (Tanenbaum, 2009).

Quando um processo é criado ele carrega vários atributos específicos para sua identificação e manipulação, entre eles, seus principais atributos são os estados, contadores e registradores:

- **Estados:** eles são classificados como:
    - **Novo**: quando o processo acabou de ser criado.
    - **Em espera**: o processo está na fila de prontos, esperando ser executado.
    - **Suspenso**: a execução do processo foi suspensa.
    - **Pronto**: o processo está pronto para ser executado.
- **Contador do programa:** o contador que indica o endereço da próxima instrução a ser executada pelo processo.
- **Registradores**: incluem informações de acumuladores, registradores de índice, ponteiro de pilha e informações de código de condição.

Os processos podem ser cooperativos ou independentes. Na tese os processos cooperativos são aqueles que podem ser afetados por outro processo, já os independentes não interagem com outros processos durante sua execução (Silberschatz & Galvin, 2008).

Quando um processo é criado a partir de outro processo, ele passa a ser chamado de processo filho, sendo o seu gerador considerado como processo pai, essa técnica faz com que ambos compartilhem recursos durante sua execução (Silberschatz & Galvin, 2008).

2.2.3 Algoritmos de Escalonamento de Processos

O conceito da multiprogramação é dar aos usuários a experiência de que o sistema operacional consegue executar vários processos ao mesmo tempo (multitarefa), isto é, maximizando a utilização da CPU (unidade de processamento central), principalmente para os computadores que possuem um processador de apenas um núcleo – que executam um processo de cada vez – os sistemas operacionais modernos conseguem realizar essa técnica através de algoritmos de escalonamento de processos (Tanenbaum, 2009).

Para cada sistema operacional é implementada uma política de escalonamento, ou seja, um algoritmo gerenciador de processos concorrentes.

2.2.3.1 Tipos de Escalonamento

Atualmente existem dois tipos de escalonamento de processos, os não-preemptivos e os preemptivos:

- **Não-preemptivos**: apenas ordenam os processos que serão executados, ou seja, eles alocam um processo na CPU e só aloca o próximo somente quando o processo atual terminar. São muito utilizados em sistemas antigos, que realizam uma tarefa de cada vez, os sistemas do tipo mono-tarefa (Silberschatz & Galvin, 2008).
- **Preemptivos**: são os mais utilizados nos sistemas operacionais modernos, pois eles realizam a troca de contexto (troca de processos) no tempo que for necessário e se o processo atual ainda não estiver terminado, ele interrompe sua execução levando-o para o fim da fila, para alocar outro processo. Geralmente os escalonamentos deste tipo seguem o conceito de fila circular até o término total de todos os processos (Silberschatz & Galvin, 2008).

2.2.3.2 Critérios de avaliação

Ao implantar uma política de escalonamento de processos, primeiro são analisados alguns parâmetros que auxiliam na tomada de decisão sobre qual algoritmo possui o melhor desempenho (Silberschatz & Galvin, 2008), essas características avaliadas são:

- ***Burst CPU* (tempo de surto)**: é considerado o tempo de surto do processo, tempo em que ele necessita se alocar na CPU.
- ***Throughput* (Vazão)**: é o número de processos completos por unidade de tempo.
- **Tempo de espera**: é o tempo que processo leva na fila de processos prontos.
- ***Turnaround* (Tempo de retorno)**: o tempo total que o processo aguarda na fila de prontos, desde a submissão até o seu término, geralmente este atributo possui mais relevância com os algoritmos do tipo preemptivos.
- **Tempo de resposta**: tempo que o processo leva para produzir algum resultado.

Com base nesses atributos, busca-se aperfeiçoar o desempenho de um algoritmo de escalonamento através dos seguintes critérios:

- Maximizar a utilização do processador;
- Obter vazão máxima na execução do escalonamento;
- Gerar um tempo de retorno mínimo para cada processo;
- Realizar no menor tempo de espera;
- Conseguir um tempo de resposta mínimo;

2.2.3.3 Algoritmos Não-Preemptivos

FCFS *(First-Come First-Served [6])*: Esse algoritmo é do tipo fila, em que o primeiro processo a solicitar a CPU é o primeiro a ser atendido, fazendo com que os demais entrem na fila, aguardando o término do processo atual para serem executados, esse algoritmo é do tipo não-preemptivo (Tanenbaum, 2009). Abaixo segue um exemplo na prática:

- Entradas:

Processo	***Burst CPU***
P1	**24 milissegundos**
P2	**3 milissegundos**
P3	**3 milissegundos**

Tabela 2: Entrada de processos para o algoritmo FCFS.

- Execução gráfica:

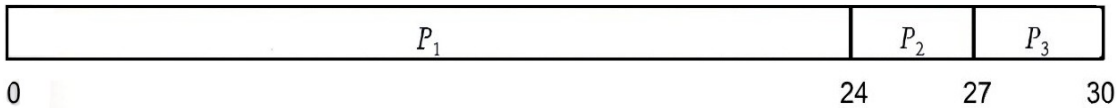

Figura 2: Linha do tempo do algoritmo FCFS.
Fonte: Silberschatz & Galvin, 2008.

- Resultado:

Processo	**Tempo de espera**	**Tempo de resposta**	***Turnaround***
P1	**0 milissegundo**	**24 milissegundos**	**24 milissegundos**
P2	**24 milissegundos**	**27 milissegundos**	**27 milissegundos**
P3	**27 milissegundos**	**30 milissegundos**	**30 milissegundos**

Tabela 3: Resultado referente ao escalonamento FCFS.

Como resultado o escalonamento gerou os tempos médios: tempo de espera de 17 milissegundos, tempo de resposta e *turnaround* de 27 milissegundos.

6 First-come First-served: Primeiro que vem é o primeiro a ser servido, é o conceito conhecido como fila.

SJF *(Shortest Job First [7])*: por ser um escalonador não-preemptivo, o algoritmo segue os mesmos conceitos do FCFS, seu único diferencial é que ele ordena antes os processos da fila priorizando os que possuem menor burst CPU para serem executados primeiro (Tanenbaum, 2009).

- Entradas:

Processo	*Burst CPU*	Tempo de chegada
P1	**7 milissegundos**	**0 milissegundo**
P2	**4 milissegundos**	**2 milissegundos**
P3	**1 milissegundo**	**4 milissegundos**
P4	**4 milissegundos**	**5 milissegundos**

Tabela 4: Entrada de processos para analisar o algoritmo SJF.

- Execução gráfica:

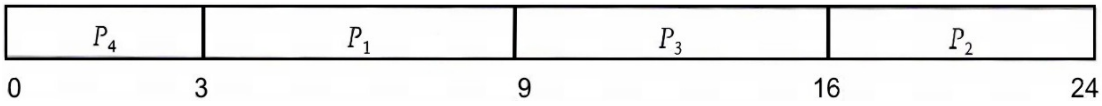

Figura 3: Linha do tempo do algoritmo SJF.
Fonte: Silberschatz & Galvin, 2008.

- Resultado:

Processo	Tempo de espera	Tempo de resposta	*Turnaround*
P1	**0 milissegundo**	**7 milissegundos**	**7 milissegundos**
P2	**7 milissegundos**	**11 milissegundos**	**11 milissegundos**
P3	**11 milissegundos**	**12 milissegundos**	**12 milissegundos**
P4	**12 milissegundos**	**16 milissegundos**	**16 milissegundos**

Tabela 5: Resultado de saída do escalonamento SJF.

As médias de tempo geradas pelo SJF foram: tempo de espera de 7,5 milissegundos, tempo de resposta e *turnaround* de 11,5 milissegundos.

7 Shortest Job First: O trabalho mais curto primeiro é o conceito aplicado no algoritmo que prioriza os processos de menor tempo para serem executados primeiro.

Prioridade: Semelhante ao SJF, seu único diferencial é o critério de ordenação. Ele utiliza as prioridades de cada processo para organizá-los, sendo assim, cada processo recebe um valor de prioridade, pelo qual é concedida a execução do processo (Tanenbaum, 2009).

- Entradas:

Processo	*Burst CPU*	Prioridade
P1	**10 milissegundos**	**3 milissegundos**
P2	**1 milissegundo**	**1 milissegundo**
P3	**2 milissegundos**	**4 milissegundos**
P4	**1 milissegundo**	**5 milissegundos**
P5	**5 milissegundos**	**2 milissegundos**

Tabela 6: Entrada para avaliação do algoritmo de prioridade.

- Execução gráfica:

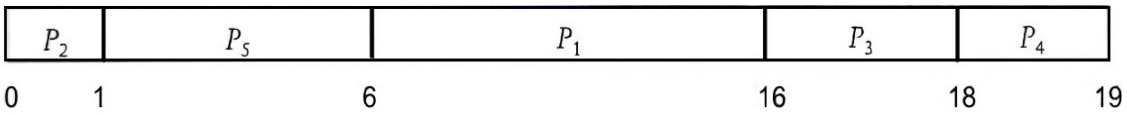

Figura 4: Linha do tempo do algoritmo de Prioridade.
Fonte: Silberschatz & Galvin, 2008.

- Resultado:

Processo	Tempo de espera	Tempo de resposta	*Turnaround*
P1	**6 milissegundos**	**16 milissegundos**	**16 milissegundos**
P2	**0 milissegundo**	**1 milissegundo**	**1 milissegundo**
P3	**16 milissegundos**	**18 milissegundos**	**18 milissegundos**
P4	**18 milissegundos**	**19 milissegundos**	**19 milissegundos**
P5	**1 milissegundo**	**6 milissegundos**	**6 milissegundos**

Tabela 7: Resultado de execução do algoritmo de prioridade.

O resultado gerou os seguintes tempos médios: tempo de espera de 8,2 milissegundos, tempo de resposta e *turnaround* de 12 milissegundos.

2.2.3.4 Algoritmos Preemptivos

SRT *(Shortest Remaining Time[8]):* o algoritmo interrompe o processo em andamento, caso entre na fila um processo com menor tempo de surto durante o instante de tempo de execução do atual. O processo interrompido somente volta a se alocar na CPU quando todos os processos menores terminarem (Tanenbaum, 2009).

- Entradas:

Processo	*Burst CPU*	Tempo de chegada
P1	**8 milissegundos**	**0 milissegundo**
P2	**4 milissegundos**	**1 milissegundos**
P3	**9 milissegundos**	**2 milissegundos**
P4	**5 milissegundos**	**3 milissegundos**

Tabela 8: Exemplo de entrada para simulação do algoritmo SRT.

- Execução gráfica:

Figura 5: Linha do tempo do algoritmo SRT.
Fonte: Silberschatz & Galvin, 2008.

- Resultado:

Processo	Tempo de espera	Tempo de resposta	*Turnaround*
P1	**9 milissegundos**	**0 milissegundo**	**17 milissegundos**
P2	**1 milissegundos**	**1 milissegundos**	**5 milissegundos**
P3	**17 milissegundos**	**17 milissegundos**	**26 milissegundos**
P4	**5 milissegundos**	**5 milissegundos**	**10 milissegundos**

Tabela 9: Resultado gerado pelo algoritmo SRT.

8 Shortest Remaining Time: Algoritmo que prioriza processos de menor tempo restante.

Como resultado o algoritmo gerou em média os seguintes valores: tempo de espera média de 8 milissegundos, tempo de resposta média de 5,75 milissegundos e um *turnaround* médio de 14,5 milissegundos.

Round Robin: é um algoritmo preemptivo, que associa um *quantum* (tempo de corte) como parâmetro de interrupção do processo atual, realizando com frequência a troca de contexto entre os processos, de forma abstrata, o escalonamento executa cada processo por partes, através de uma fila circular. É o algoritmo mais utilizado em sistemas operacionais multitarefas e também o seu conceito é aplicado para solucionar problemas, como o agendamento de dados de pacotes nas redes de computadores (Tanenbaum, 2009). A utilização do tempo de corte neste algoritmo faz com que o algoritmo realize trocas de contexto com maior freqüência, de fato, quanto menor o tempo de corte, maior o número de substituição de processos que usarão os recursos do processador, a figura 6 ilustra perfeitamente o comportamento deste escalonamento:

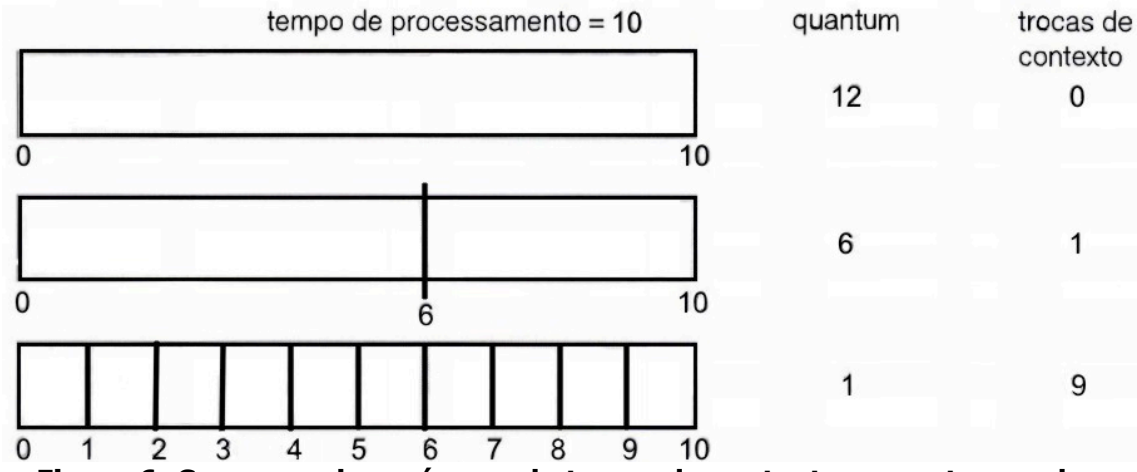

Figura 6: Comparando o número de trocas de contexto com o tempo de corte.

Fonte: Silberschatz & Galvin, 2008.

Neste exemplo prático, o algoritmo será apresentado utilizando um tempo de corte de 4 milissegundos:

- Entradas:

Processo	***Burst CPU***
P1	**24 milissegundos**
P2	**3 milissegundos**
P3	**3 milissegundos**

Tabela 10: Entrada para análise do algoritmo Round Robin.

Reparem na linha do tempo gerada pelo Round Robin, é perceptível que cada processo foi particionado em pequenos fragmentos baseado no tempo de corte pré-configurado pelo algoritmo. Isso faz com que cada processo utilize a CPU de forma justa até o término do escalonamento.

- Execução gráfica:

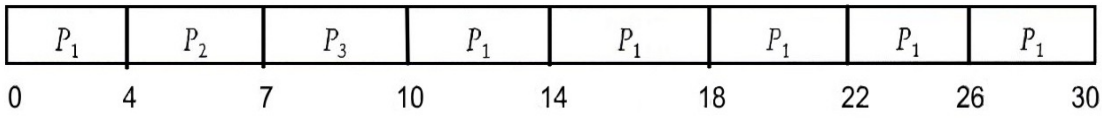

Figura 7: Linha do tempo do algoritmo Round Robin.
Fonte: Silberschatz & Galvin, 2008.

- Resultado:

Processo	**Tempo de espera**	**Tempo de resposta**	***Turnaround***
P1	**6 milissegundos**	**0 milissegundo**	**30 milissegundos**
P2	**4 milissegundos**	**4 milissegundos**	**7 milissegundos**
P3	**7 milissegundos**	**7 milissegundos**	**10 milissegundos**

Tabela 11: Resultado final do algoritmo Round Robin.

A execução deste algoritmo gerou os seguintes tempos médios: tempo de espera de 5,66 milissegundos, tempo de resposta de 3,66 milissegundos e um *turnaround* de 15,66 milissegundos.

2.2.4 Discos Magnéticos

Os discos magnéticos conhecidos como discos rígidos são unidades de armazenamento secundário, são dispositivos compostos por componentes eletrônicos e mecânicos, basicamente ele é constituído por uma lâmina de material magnético chamado prato e por cabeça acoplada a um braço mecânico. Sua cabeça flutua em cima da lâmina e através do braço, fazendo com seja executado os movimentos no disco de leitura e gravação de dados. São eles os responsáveis por manterem persistidos todos os arquivos pessoais e do sistema operacional de forma que seja possível desligar o computador e acessá-los novamente a qualquer momento que ligá-lo novamente (Tanenbaum, 2009).

Essa lâmina é dividida logicamente em trilhas, que por sua vez, são subdivididas em setores que variam de 512 bytes a 1024 bytes, o conjunto de trilhas concêntricas são chamados de cilindros e podem existir milhares deles numa unidade, os dados são armazenados através de uma escrita magnética (Silberschatz & Galvin, 2008).

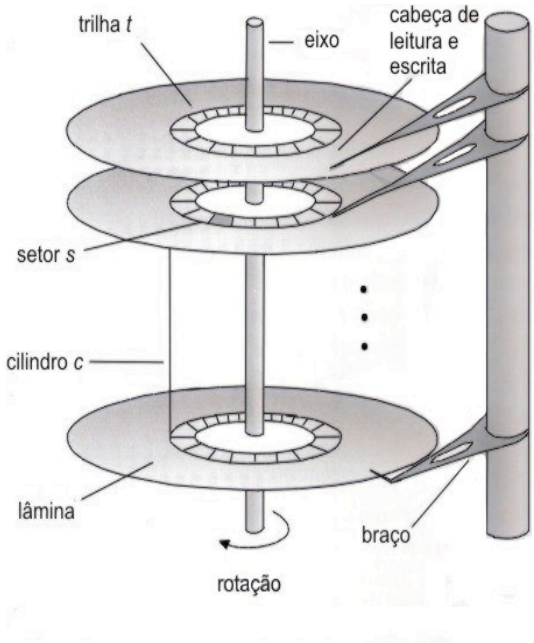

Figura 8: Formato mecânico de um disco magnético.
Fonte: Silberschatz & Galvin, 2008.

As unidades de disco se comunicam com a CPU através de cabos chamados barramentos de I/O, os barramentos mais conhecidos são: *IDE (Integrated Drive Electronics)* ou *SCSI (Small Computer System Interface)*, um controlador gerência a comunicação do disco rígido com o processador.

2.2.5 Algoritmos de Escalonamento de Disco

Para se ter eficiência na utilização do disco é necessário obter um tempo mínimo de acesso (Tanenbaum, 2009), na utilização dos dois componentes principais:

- **Tempo de busca**: é o tempo utilizado pelo braço do disco para mover as cabeças para o cilindro que contém o setor desejado.

- **Latência rotacional**: é o tempo adicional de espera para o que o disco gire o setor até a cabeça.

A largura de banda é o total de bytes transferidos, divididos pelo tempo entre a primeira solicitação até o término da última transferência.

Esse tempo mínimo de acesso é conseguido por meio de algoritmos de escalonamento de disco. O objetivo é gerenciar o movimento da cabeça do disco entre os cilindros do disco magnético. Um bom algoritmo visa percorrer os cilindros da fila de requisições realizando um menor número de movimentos (Silberschatz & Galvin, 2008).

Os principais algoritmos encontrados na literatura de sistemas operacionais são:

Para uma melhor análise entre os algoritmos que serão citados abaixo, todos os exemplos utilizarão os seguintes dados de entrada:

- Posição do cilindro cabeça: 53

- Fila de requisições: 98, 183, 37, 122, 14, 124, 65 e 67

FCFS (*First-Come, First-Served*): é o algoritmo clássico, seu conceito é simples, apenas atende os cilindros da fila de requisições em ordem semelhante a uma fila – o primeiro que chega é o primeiro a ser atendido – até o finalizar todos os movimentos no disco rígido.

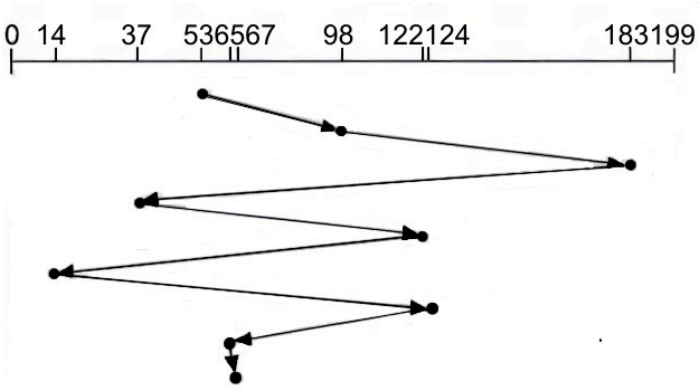

Figura 9: Apresentação gráfica do escalonamento FCFS.

Fonte: Silberschatz & Galvin, 2008.

O total de movimentos realizados pela cabeça por intermédio deste algoritmo foi de 640 cilindros.

SSTF *(Shortest-Seek Time First [9])*: este algoritmo que prioriza as requisições de menor tempo de busca com base de referência a posição do cilindro cabeça. Este tipo de escalonamento pode gerar estagnação durante algumas requisições da fila de cilindros.

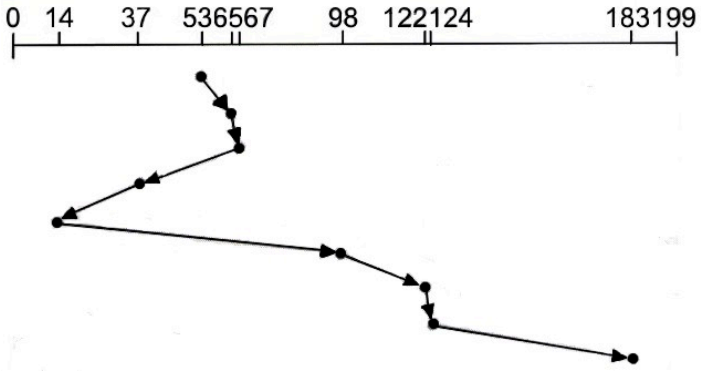

Figura 10: O total de movimentos foi de 236 cilindros.

Fonte: Silberschatz & Galvin, 2008.

9 Shortest-Seek Time First: Algoritmo que busca o cilindro de menor tempo primeiro.

SCAN: mais conhecido como o algoritmo do elevador, ele faz com que o braço do disco se movimente de uma extremidade para outra, atendendo todos os cilindros que surgirem.

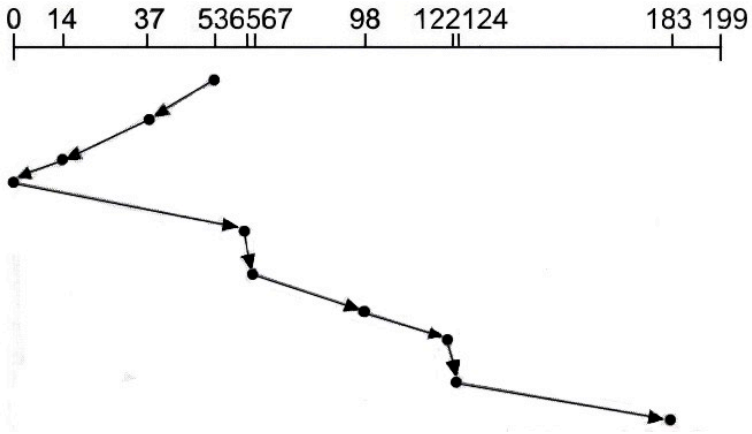

Figura 11: Neste exemplo o algoritmo resultou um número de 275 movimentações de cilindros.
Fonte: Silberschatz & Galvin, 2008.

C-SCAN: comparado com o SCAN ele oferece um tempo mais uniforme e trata todas as requisições como se fosse uma fila circular.

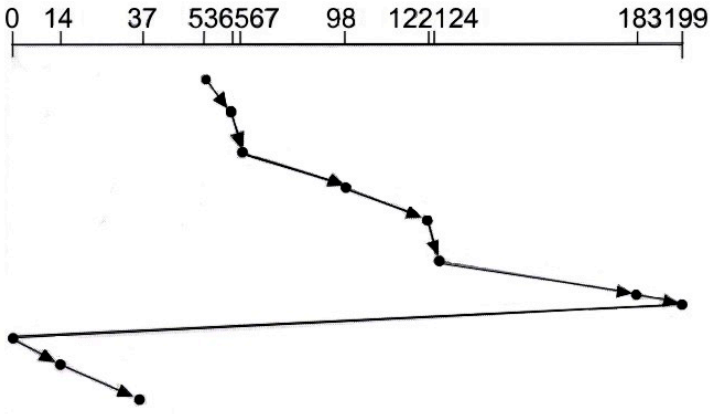

Figura 12: O algoritmo C-SCAN gerou um total de 382 movimentos.
Fonte: Silberschatz & Galvin, 2008.

Look: é uma versão aperfeiçoada do C-SCAN em que o braço começa em uma direção atendendo todos os cilindros e em seguida inverte sua movimentação sem passar pelas extremidades do disco.

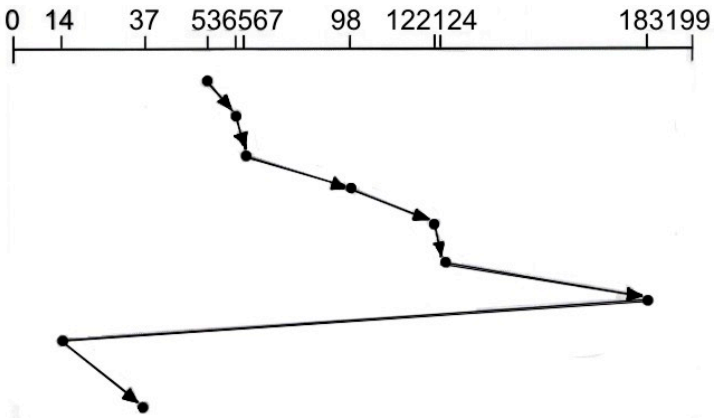

Figura 13: Este escalonamento gerou como resultado um total de 322 movimentações de cilindros.

Fonte: Silberschatz & Galvin, 2008.

2.2.6 Memória Virtual

A memória física é um *hardware* que mantém os dados enquanto o sistema operacional esta ligado, todas as informações são guardadas através de uma matriz de endereços que permite realizar acessos rápidos as informações recentes, já a memória lógica é uma matriz de endereços criada pelo próprio processador e está vinculada com a memória física, ou seja, a memória lógica é um mapeamento da memória física, a CPU faz acesso a memória física através da MMU (Unidade de gerenciamento de memória) que faz todo processo de resolução de endereço (Tanenbaum, 2009).

Por exemplo: para que um processo seja escalonado é necessário fazer uma carga do mesmo na memória física e tal execução é limitada pelo tamanho da memória física instalada no computador.

Alguns sistemas operacionais buscam resolver o problema de limitação de memória, implementando uma memória virtual. A memória virtual abstrai a memória física em segmentos do mesmo tamanho chamado de páginas, quando a CPU busca uma página, ela faz referência a um determinado fragmento de um processo qualquer. A memória virtual é um tipo de alocação dinâmica, pois utiliza uma pequena parte do disco rígido e o acesso a esta área é mais lento do que a de uma memória física, geralmente utiliza-se esta técnica quando a memória principal não possui espaço livre, ou seja, um processo somente é alocado nela quando realmente for necessário, caso contrário ele será alocado na memória principal para obter um melhor desempenho, o mesmo também ocorre quando um determinado programa exige um tamanho maior do que a memória física, com isso ocorre uma fragmentação do mesmo, para alocar o máximo possível desses fragmentos (páginas) na memória física e o que sobrar é alocado na memória virtual (Silberschatz & Galvin, 2008).

2.2.7 Paginação sob Demanda

Objetivo da paginação sob demanda é utilizar a memória física de forma eficiente, pois de forma resumida, o processo é carregar as páginas mais acessadas em uma tabela de páginas conhecida como *frame* (quadro) de páginas, evitando assim o acesso a memória lógica do sistema operacional (Tanenbaum, 2009). Essa técnica visa trazer os seguintes benefícios:

- Redução de I/O na memória virtual;
- Menor utilização da memória lógica, devido ao seu baixo desempenho;
- Geração de respostas mais rápidas, pois mantém as páginas mais utilizadas em uma tabela de páginas na memória física;

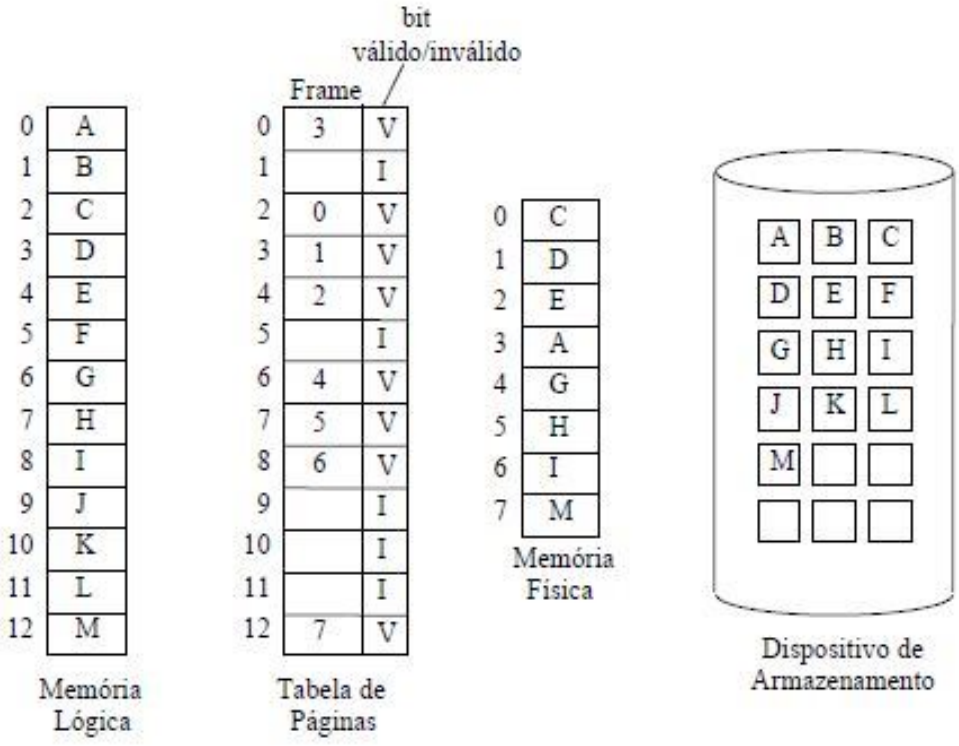

Figura 14: Tabela de páginas realizando um intermédio entre memória lógica e a memória física.
Fonte: Silberschatz & Galvin, 2008.

Sua implementação necessita de um bit incluído em cada página para controlar a tabela de páginas, pois essa tabela irá manter as mesmas páginas de maior acesso que estão presentes na memória virtual.

A MMU realiza a conversão do endereço lógico utilizando o estado dos bits da página. Basicamente se o bit da página for válido a página será acessada normalmente pelo sistema operacional e caso contrário, poderá ocorrer um dos dois possíveis casos:

1° - O sistema operacional verifica se a referência da página é inválida, caso sim o processo será abordado.

2° O sistema operacional busca sua referência válida na memória virtual, gerando um *page-fault* (falta de página), e a técnica para o tratamento disso, é simplesmente buscar essa página na memória virtual e alocá-la em um quadro livre na tabela de páginas. Como não está garantido que sempre haverá quadros livres para alocação, às vezes, é necessário substituir uma página existente da tabela de páginas e esse tratamento é conhecido como Substituição de Páginas (Silberschatz & Galvin, 2008).

Resumidamente uma string de referência (vetor de referências das páginas), mantém todas as palavras ou bytes de cada página que serão futuramente acessados. O processo é simples:

Inicialmente verifica o primeiro byte referente a uma página presente da string de referências, se a tabela de páginas estiver vazia, ocorre uma *page-fault* e é alocada a página na tabela. Depois verifica o próximo byte do vetor se o byte já existir na tabela o acesso a página é direto pela mesma, não ocorrendo uma falta de página. O processo ocorrerá repetidamente até acabar a leitura. Durante todo o processo, surgirão casos em que a tabela de páginas estará cheia, e o acesso a uma página da string de referência não existirá na tabela. Com isso, acontecerá uma substituição de uma página qualquer da tabela pela página acessada pela string de referência (Silberschatz & Galvin, 2008).

2.2.8 Algoritmos de Substituição de Páginas

Os algoritmos de substituição de páginas são classificados em duas categorias: algoritmos de espaço fixo e algoritmos de espaço variável. A diferença entre ambos é que o algoritmo que trabalha com espaços fixos utiliza uma área de memória de tamanho estático e a que trabalha com espaço variável possui a característica de modificar sua área de memória dinamicamente. Um algoritmo eficiente é aquele que menos gera *page-faults* (Silberschatz & Galvin, 2008).

As principais políticas de substituição de páginas são:

FIFO *(First-In, First-Out [10]):* é o algoritmo de fácil implementação, a substituição é sempre pela primeira página do quadro, seguindo o conceito de uma fila, ou seja, trocasse pela primeira página que entrou no quadro (a primeira página a entrar é a última a sair).

No exemplo abaixo utiliza-se o algoritmo FIFO configurado para manter um quadro de páginas com alocação de no máximo até 3 páginas:

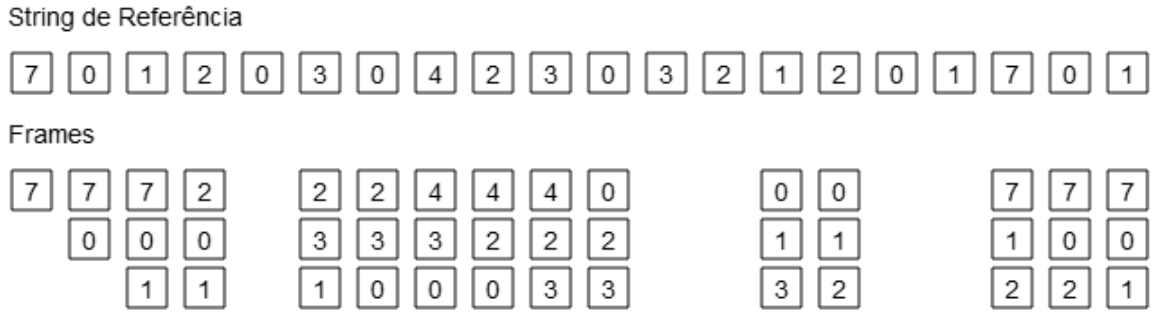

Figura 15: O algoritmo FIFO gerou 15 *page-faults.*
Fonte: Silberschatz & Galvin, 2008.

[10] **First-In, First-Out:** Algoritmo de fila, o primeiro que entra é primeiro que sai.

OPT (Algoritmo Ótimo): esse algoritmo é baseado no conhecimento futuro da string de referência, seria necessário prever todas as páginas que virão, é um algoritmo utilizado apenas para simulações e estudos acadêmicos, pois ainda não existe a possibilidade de prever os futuros bytes de uma string de referência sendo também impossível implementá-lo em um sistema operacional (Silberschatz & Galvin, 2008). Abaixo segue uma imagem ilustrando o comportamento do algoritmo com um quadro de tamanho 3:

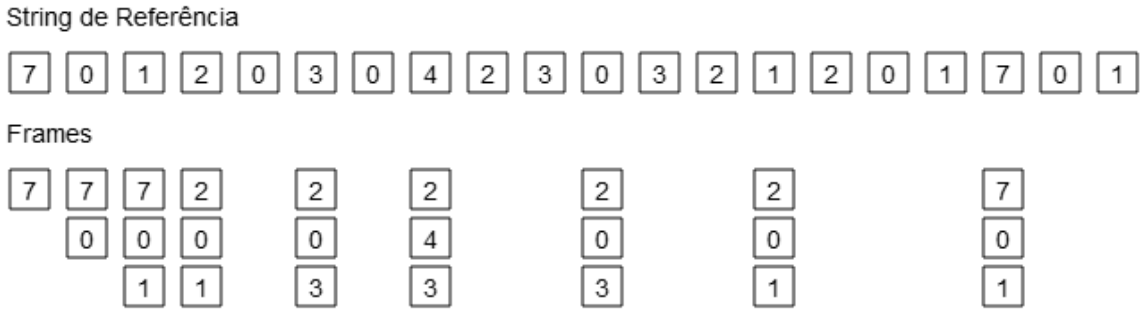

Figura 16: O algoritmo ótimo gerou 9 *page-faults.*
Fonte: Silberschatz & Galvin, 2008.

LRU *(Last Recently Used* [11]*):* Esse algoritmo realiza substituição da página que estiver mais tempo no quadro. Existem duas abordagens para implementação desse algoritmo: LRU com contadores e o LRU de pilha. A figura abaixo apresenta o comportamento do algoritmo seguindo o mesmo exemplo de tamanho de quadros e bytes da string de referências:

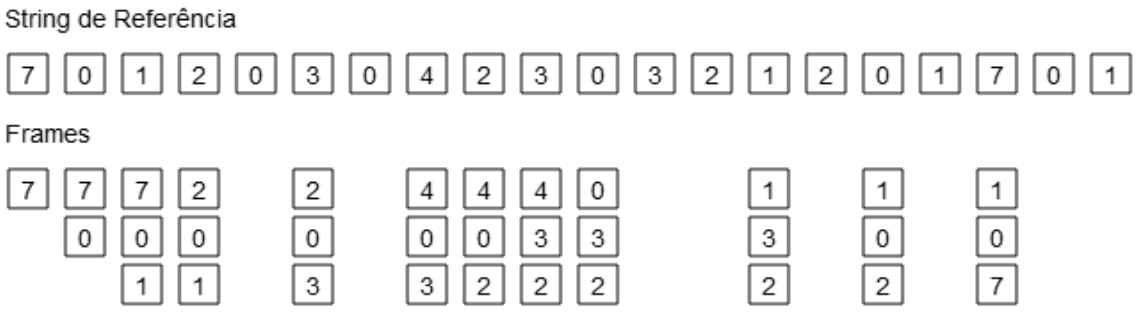

Figura 17: Neste caso o algoritmo LRU resultou 12 *page-faults.*
Fonte: Silberschatz & Galvin, 2008.

[11] **Last Recently Used:** Algoritmo que substitui os últimos utilizados recentemente.

LRU com contadores: São associados ao algoritmo, relógios lógicos, ou seja, a substituição de páginas é feita através de contadores. Toda vez que ocorre um falta de página o contador será incrementado, a página que estiver mais tempo alocada será substituída.

LRU com pilha: Esse algoritmo utiliza a estrutura de pilha, para selecionar as páginas que serão sobrescritas. A página que for alocada no quadro e for a mais referenciada será alocada para o topo da pilha. O exemplo abaixo exibe com clareza o comportamento dessa estrutura:

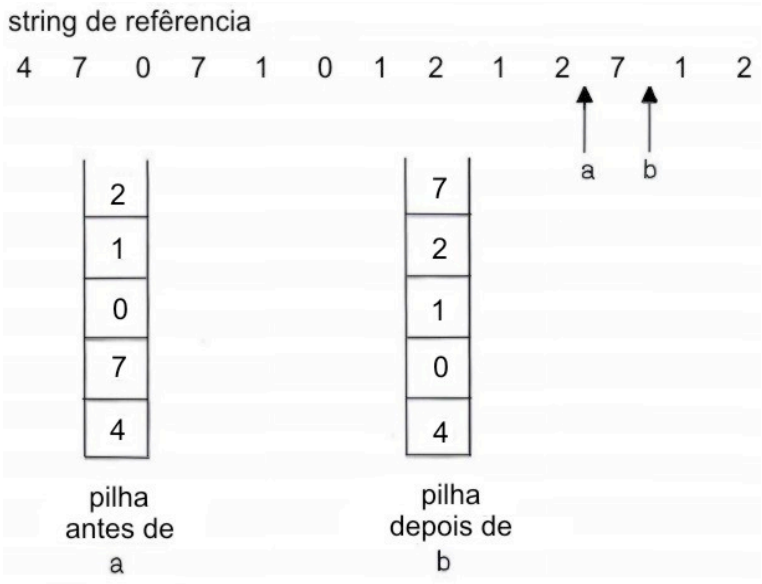

Figura 18: Algoritmo LRU com estrutura de pilha.

Fonte: Silberschatz & Galvin, 2008.

Algoritmos de substituição baseados em contagem: utiliza contadores para contar o número de referências, a página que tiver o menor número de contagem é substituída.

Algoritmo de bits de referência adicional: É mantida um byte de 8 bits em cada página, nos intervalos regulares, uma interrupção transfere o controle para o sistema operacional e este passa o bit de referência de cada página para o bit mais significativo entre os seus 8 bits deslocando os outros bits para direita para descartar o bit menos significativo no final.

MRU *(Most Recently Used*[12] *):* Segue o mesmo conceito do algoritmo LRU, sua única diferença é que ele substitui as páginas de menor tempo alocadas no quadro, ou seja, esta política do algoritmo substitui a última página alocada.

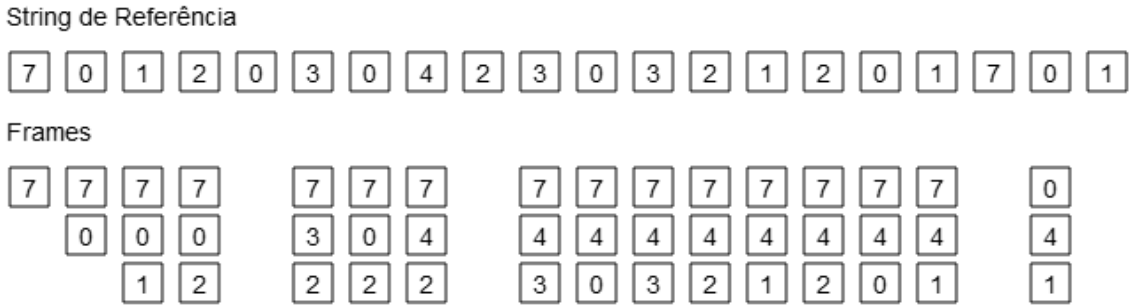

Figura 19: Algoritmo MRU gerando um total de 16 *page-faults.*

Fonte: Silberschatz & Galvin, 2008.

Algoritmo segunda chance: Os quadros são dispostos em fila circular, cada quadro recebe um bit de referência, durante a execução o CPU verifica o bit de cada quadro, o que tiver bit de valor 0, esse será sobrescrito e os que tiverem valor 1 recebe uma segunda chance.

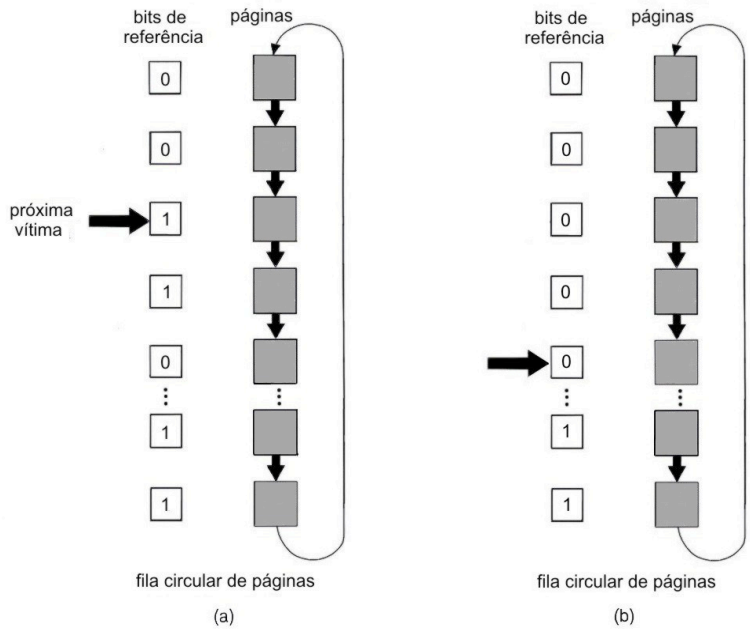

Figura 20: Fila circular do algoritmo segunda chance.

Fonte: Silberschatz & Galvin, 2008.

[12] **Most Recently Used:** Algoritmo de substituição das páginas recentemente utilizadas.

Algoritmo de segunda chance otimizado: Nesse algoritmo são utilizados dois bits: um bit de referência e um bit para checagem de modificação com isso têm a seguintes classificações realizadas pelo algoritmo:

Bit (0,0): a página não foi utilizada e nem modificada recentemente, isso significa que ela é a melhor página para ser substituída.

Bit (0,1): a página não foi usada recentemente, mas houve modificação, isso significa que a página precisará ser gravada antes de ser substituída.

Bit (1,0): a página foi usada recentemente e provavelmente será usada novamente.

Bit (1,1): a página foi usada recentemente e modificada, logo será usada novamente, mas antes precisará ser gravada para ser substituída.

Ao percorrer a fila de páginas, a CPU levará em consideração essas quatro categorias de bit para fazer uma substituição de página (Silberschatz & Galvin, 2008).

3. PROJETO

3.1 Conhecendo o Projeto

O projeto SimulaRSO (Simulador de Recursos de Sistemas Operacionais) é uma ferramenta para simulações e estudos sobre os principais e essenciais componentes que são implementados em uma arquitetura de sistema operacional moderno.

O objetivo desse simulador é apresentar de forma gráfica e intuitiva o comportamento desses recursos, de forma a agregar conhecimentos para os alunos que cursam disciplinas dos cursos de TI (Tecnologia da Informação) que abordem em sua ementa, os estudos sobre os algoritmos de escalonamento de processos, algoritmos escalonamento de disco e algoritmos paginação de memória, que são geralmente vistos na disciplina de Sistemas Operacionais.

Esse simulador foi projetado para ser acessado pela internet, não exigindo que o usuário faça *download* e instalação local para utilizá-lo. A aplicação foi desenvolvida sobre arquitetura *cliente-servidor*[13] para gerenciar as requisições de simulação dos algoritmos, sendo gerenciadas pelo servidor e apresentadas para o cliente os resultados. Tecnologias emergentes tais como HTML5, CSS3 e Javascript (W3CSchools, 2011) foram utilizada para desenvolver uma *interface* usuário prática e interativa, através da implementação gráfica sobre tecnologias no projeto e soluções gratuitas foram encontradas para hospedagem do simulador no que resultou em um custo zero para publicação da aplicação.

Todas as simulações ocorrem através de gráficos programados e personalizados pelos autores, para assemelhar e manter fidelidade aos que são vistos nas aulas e nos livros sobre Sistemas Operacionais.

O projeto é código-aberto para que futuramente outros colaboradores possam facilmente agregar novas funcionalidades e novos módulos de

[13] **Cliente-Servidor:** Arquitetura clássica utilizada em sistemas web, que separa a lógica da aplicação e a apresentação do sistema, permanecendo as regras de negócio (lógicas da aplicação) no computador servidor e as apresentações do sistema no computador cliente, pelo qual o cliente solicita recursos do sistema requisitando informações para o computador servidor através da páginas do sistema (apresentação da aplicação).

simulação. Uma das vantagens do projeto é que ele pode ser facilmente integrado a qualquer sistema EAD (Ensino a Distância), pelo fato da aplicação ser totalmente acessada pela internet.

Para complementar alguns detalhes sobre o simulador, abaixo serão apresentadas imagens do projeto em funcionamento, abordando detalhadamente todas as suas características, funcionalidades e como usar a aplicação:

Figura 21: Home Page do SimulaRSO.

1. Menu de navegação principal sobre as funcionalidades do projeto.
2. Área para seleção do idioma.
3. Seção de informações sobre o projeto e suas últimas atualizações.
4. Rodapé da página com informações sobre os autores.

Figura 22: Módulo de escalonamento de processos.

1. **Modo de simulação**:
    a. **Única**: Realiza simulação com um único algoritmo.
    b. **Comparativa**: Realiza simulação entre dois distintos algoritmos.
2. **Total de processos:** permite instanciar no mínimo dois até no máximo vinte processos a serem escalonados.
3. **Troca de contexto:** define o tempo hipotético para o algoritmo realizar a troca de processos durante o escalonamento.
4. **Algoritmo 1:** define o primeiro algoritmo a ser simulado.
5. **Algoritmo 2:** define o segundo algoritmo a ser simulado.
6. **Tempo de corte:** define a política de tempo de corte, exclusivo e funcional apenas para o algoritmo *Round Robin.*

7. **Painel de configuração dos processos:**
    a. ***Burst*:** define o tempo de surto do processo - tempo mínimo 1 milissegundo e tempo máximo 99 milissegundos.
    b. **Chegada:** define o instante de tempo que o processo surgiu na fila de chegada do escalonamento - tempo mínimo 0 milissegundo e tempo máximo 99 milissegundos.
    c. **Prioridade:** determina a prioridade do processo – prioridade 1 é a mais alta e prioridade 10 é a mais baixa.
8. **Botão Ajuda:** Exibe uma janela explicando detalhadamente todos os recursos deste módulo de simulação.
9. **Botão Configuração automática:** Faz geração aleatória dos tempos de cada processo instanciado no painel de configuração dos processos.
10. **Botão Executar:** Realiza a simulação do escalonamento de processos.

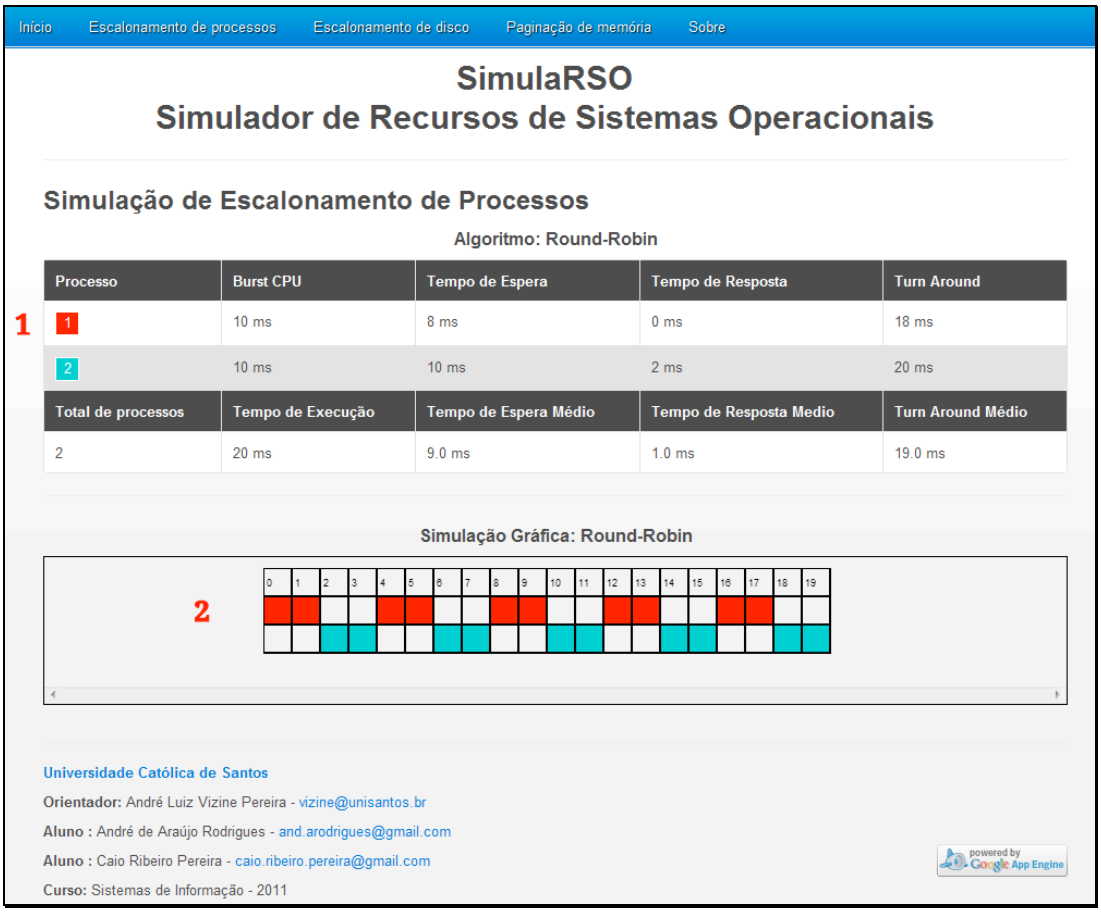

Figura 23: Resultado de um escalonamento de processos.

1. Resultado apresentando os principais tempos medidos pelo algoritmo:
    a. *Burst CPU;*
    b. Tempo de espera;
    c. Tempo de resposta;
    d. Tempo de *turn arounds;*
    e. Total de processos instanciados;
    f. Tempo de execução do algoritmo;
    g. Tempo de espera médio;
    h. Tempo de resposta médio;
    i. Tempo de *turn around* médio;
2. Animação gráfica sobre o comportamento do escalonamento;

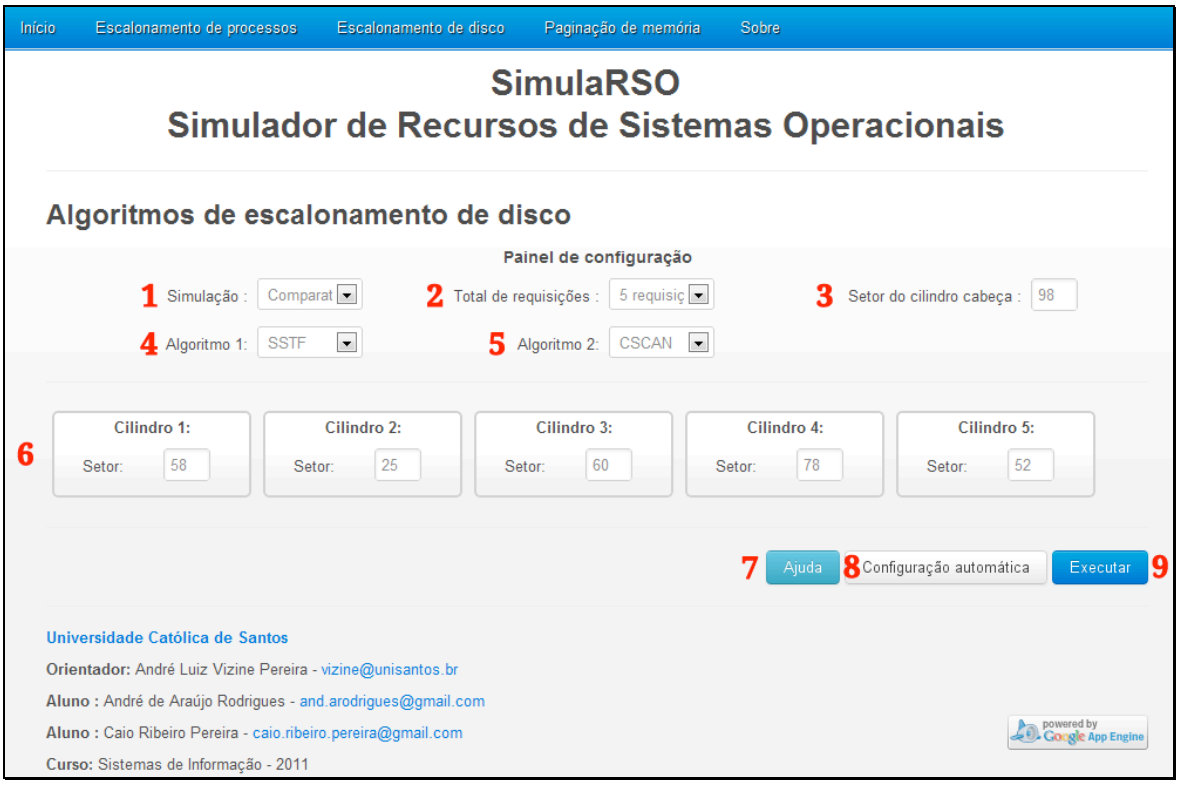

Figura 24: Módulo de escalonamento de disco.

1. **Modo de simulação**:
    a. **Única**: Realiza simulação com um único algoritmo.
    b. **Comparativa**: Realiza simulação entre dois distintos algoritmos.
2. **Total de requisições:** permite realizar no mínimo duas até no máximo trinta requisições de *I/O* nos cilindros a serem escalonados.
3. **Setor do cilindro cabeça:** define em que setor o cilindro cabeça irá iniciar – valor mínimo 0 e valor máximo 99.
4. **Algoritmo 1:** define o primeiro algoritmo a ser simulado.
5. **Algoritmo 2:** define o segundo algoritmo a ser simulado.
6. **Painel de configurações de cilindros:** permite definir o valor dos setores de cada cilindro que serão escalonados – setor mínimo 0 e máximo 99.

7. **Botão Ajuda:** Exibe uma janela explicando detalhadamente todos os recursos deste módulo de simulação.
8. **Botão Configuração automática:** Faz geração aleatória dos setores no painel de configuração dos cilindros do disco.
9. **Botão Executar:** Realiza a simulação do escalonamento de disco.

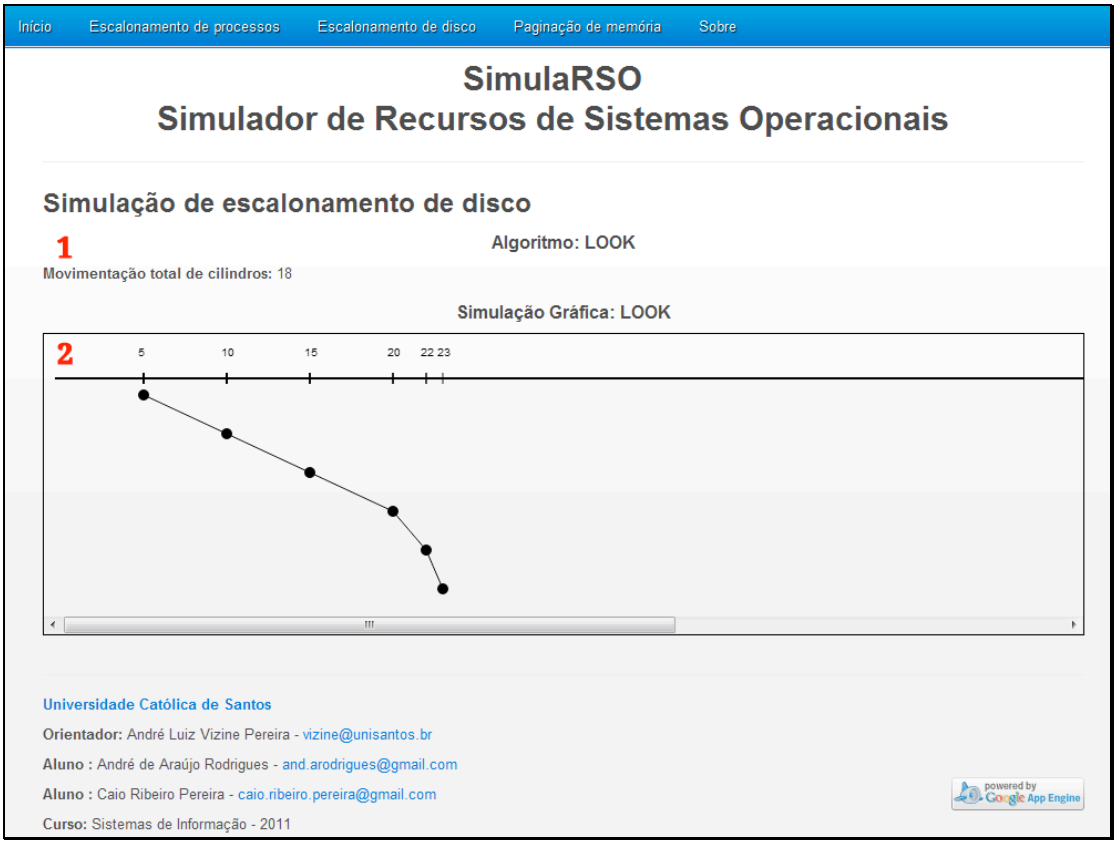

Figura 25: Resultado de um escalonamento de disco.

1. Resultado apresentando o total de movimentações entre os cilindros.
2. Animação gráfica exemplificando o comportamento da simulação.

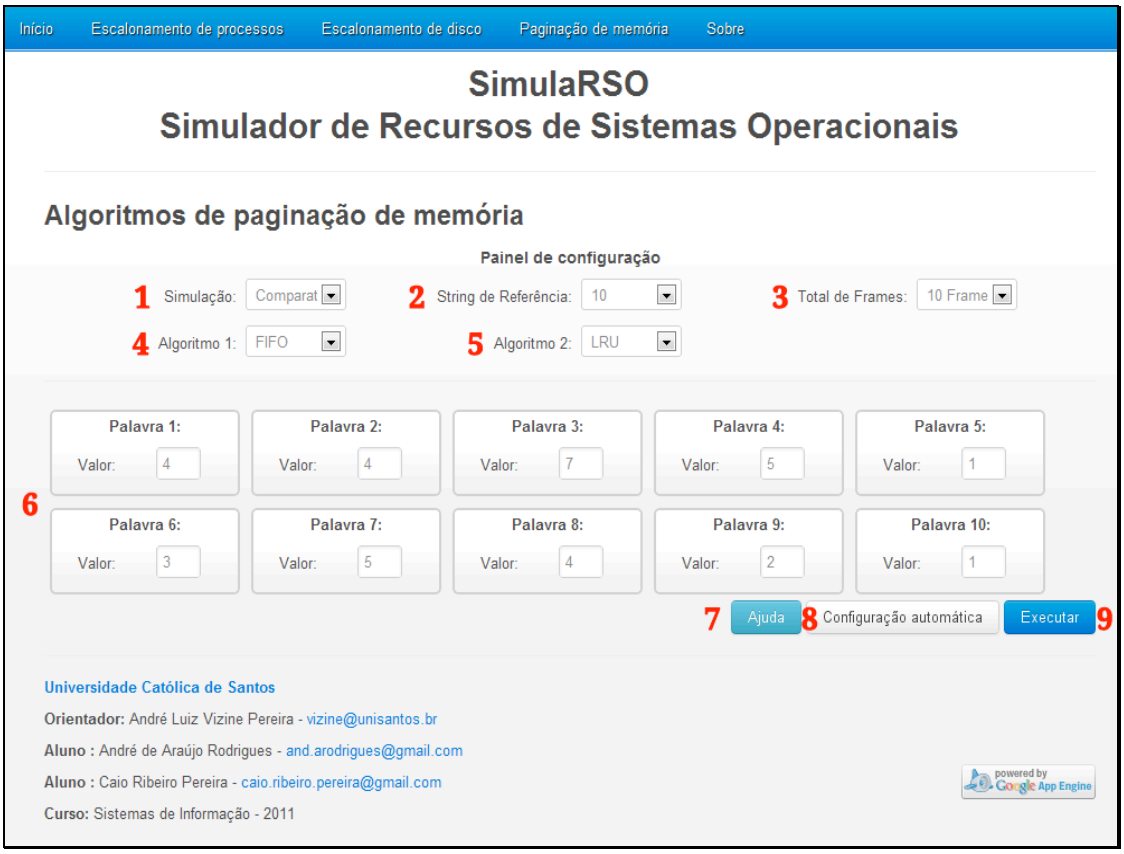

Figura 26: Módulo de paginação de memória virtual.

1. **Modo de simulação**:
    a. **Única**: Realiza simulação com um único algoritmo.
    b. **Comparativa**: Realiza simulação entre dois distintos algoritmos.
2. **String de referência:** determina o tamanho da *String de Referência* que será usada na simulação – valor mínimo 2 e máximo 30.
3. **Total de Frames:** determina o tamanho do quadro de paginação – valor mínimo 2 e máximo 10.
4. **Algoritmo 1:** define o primeiro algoritmo a ser simulado.
5. **Algoritmo 2:** define o segundo algoritmo a ser simulado.
6. **Painel de configurações de palavras:** permite definir o valor das palavras que serão incorporadas na *String de Referência.*

7. **Botão Ajuda:** Exibe uma janela explicando detalhadamente passo a passo como realizar este módulo de simulação.
8. **Botão Configuração automática:** Faz geração aleatória dos valores no painel de configuração de palavras da *String de Referência*.
9. **Botão Executar:** Realiza a simulação da paginação de memória.

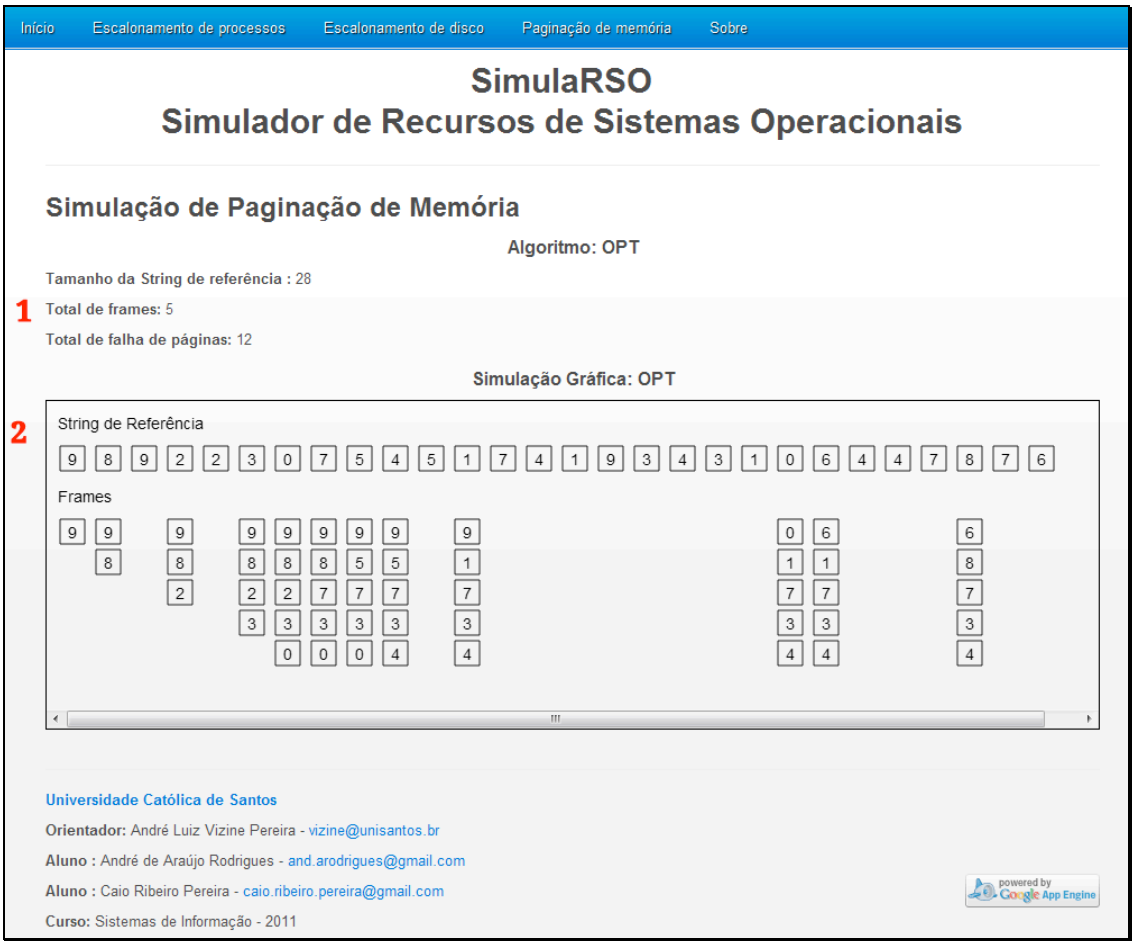

Figura 27: Resultado de uma paginação de memória.

1. Resultado final da simulação de um algoritmo de paginação de memória virtual, apresentando as seguintes medidas analíticas:
    a. Tamanho da String de Referência;
    b. Total de Frames;
    c. Total de falhas de páginas ocorridas pelo algoritmo;
2. Simulação gráfica da execução de um algoritmo de paginação;

Figura 28: Página sobre detalhes do projeto.

3.2 Especificações do Projeto

3.2.1 Diagrama de Caso de Uso

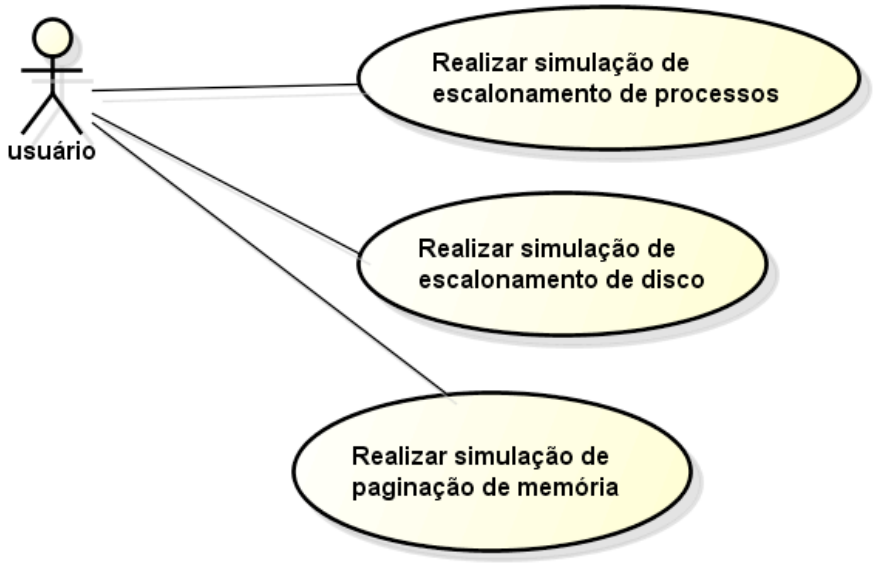

Figura 29: Diagrama de caso de uso sobre as funcionalidades do projeto.

3.2.2 Diagrama de Componentes

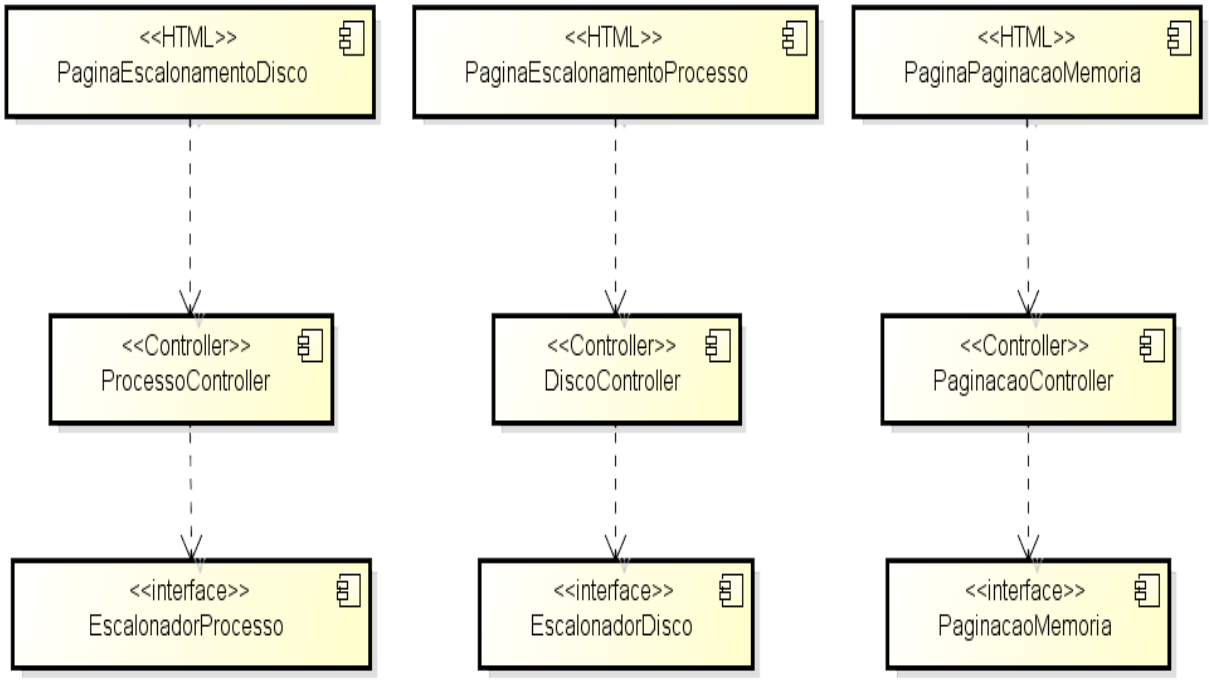

Figura 30: Diagrama de componentes ilustrando a arquitetura do projeto.

3.2.3 Diagrama de Classes

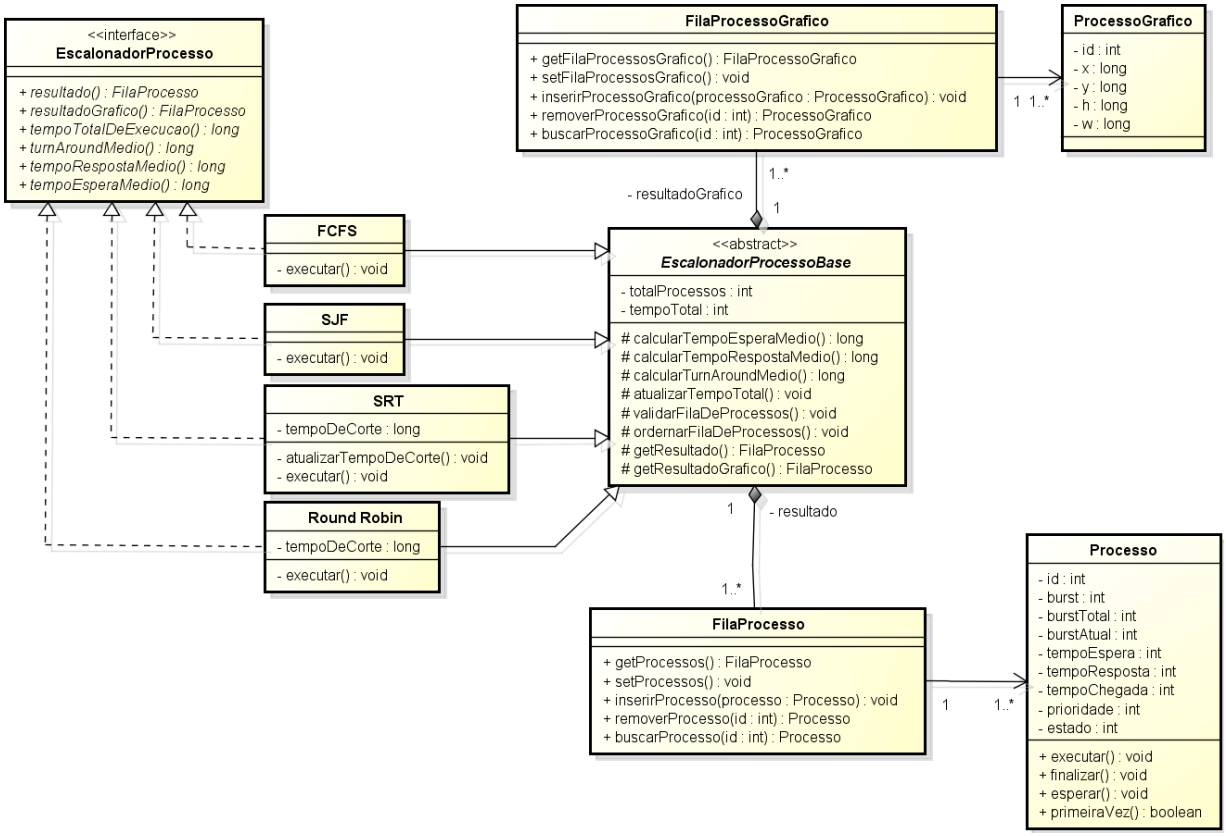

Figura 31: Diagrama de classes do módulo: Escalonamento de processos.

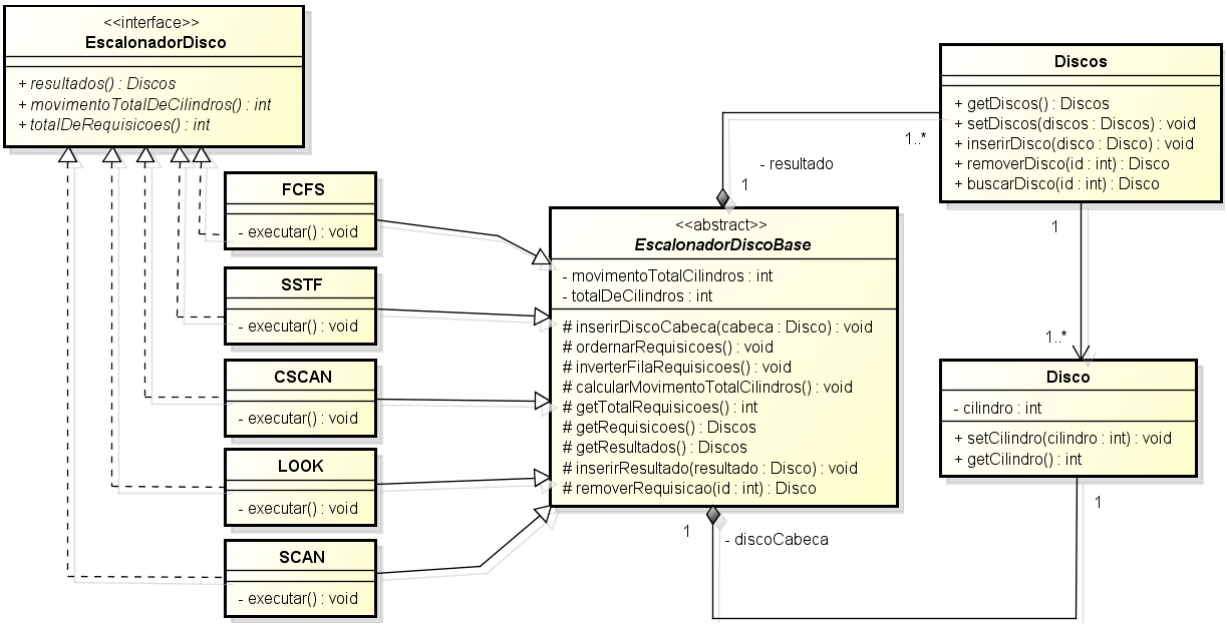

Figura 32: Diagrama de classes do módulo: Escalonamento de disco.

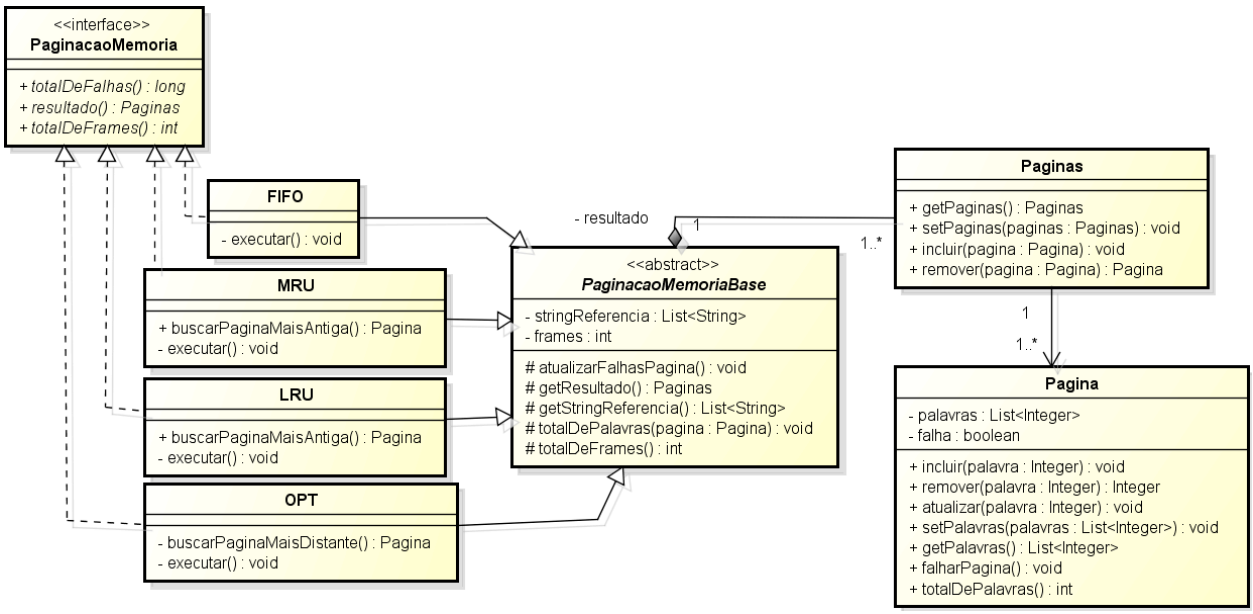

Figura 33: Diagrama de classes do módulo: Paginação de memória.

3.2.4 Características Técnicas do Projeto

O projeto SimulaRSO - Simulador de Recursos de Sistemas Operacionais será uma aplicação web e as tecnologias que utilizadas são:

- Plataforma Java EE (Java Enterprise Edition) (Oracle, 2011);
- *Framework MVC* (Model-View-Controller[14]) VRaptor (Caelum, 2011);
- Hospedagem do projeto no *Google App Engine* (Google, 2011);
- Publicação do código-fonte no repositório *Github Social Coding* (Github, 2011);
- Interface gráfica da aplicação seguirá as normas e boas práticas pelo órgão mundial W3C (W3CSchools, 2011);
- Desenvolvimento do layout utilizará o conjunto de tecnologias web: HTML5*,* CSS3 *e* Javascript (W3CSchools, 2011);
- Será implementado técnicas de testes automatizados utilizando o framework de testes unitários JUnit (JUnit, 2011) que irá testar a funcionalidade dos algoritmos de simulação;
- Os diagramas de classes, componentes e caso de uso do projeto será elaborado seguindo as normas técnicas de modelagem UML (Linguagem de Modelagem Unificada) (UML, 2011);

[14] **Model-View-Controller:** Conceito aplicado para desenvolvimento de projetos com 3 camadas, Modelo (Classes de entidade de negócio), *View* (Interface gráfica de visualização para o usuário) e *Controller* (Classes que interligam os modelos com as *views*).

3.2.5 Funcionalidade do Projeto

As funcionalidades do projeto SimulaRSO são:

- Simular os algoritmos de escalonamento de processos com até 20 processos simultâneos;
- Simular os algoritmos de escalonamento de disco com até 30 requisições de *I/O* em disco;
- Simular os algoritmos de substituição de página de memória virtual com até 30 palavras na *String de Referência;*
- Realizar simulação única de um algoritmo do módulo;
- Realizar simulação comparativa entre até no máximo 2 algoritmos distintos permitindo analisar os resultados e comportamentos deles;
- Geração aleatória de valores para os algoritmos;
- Exibição do comportamento dos algoritmos através de gráficos intuitivos, semelhantes aos que são vistos em sala de aula.
- Projeto com camada de internacionalização de idiomas, com suporte e apresentação do aplicativo entre os idiomas inglês e português.

Os módulos e algoritmos que foram desenvolvidos nessa primeira versão do projeto são:

- Módulo: Escalonamento de processos
  - Algoritmo FCFS
  - Algoritmo SJF
  - Algoritmo SRT
  - Algoritmo Round Robin
- Módulo: Escalonamento de disco
  - Algoritmo FCFS
  - Algoritmo SSTF
  - Algoritmo SCAN
  - Algoritmo C-SCAN
  - Algoritmo LOOK
- Módulo: Paginação de memória virtual
  - Algoritmo FIFO
  - Algoritmo Ótimo
  - Algoritmo LRU
  - Algoritmo MRU

Atualmente o projeto esta concluído, já publicado na internet para o usuário final. Para conhecer o SimulaRSO e seu código-fonte visite os sites:

Código fonte: https://github.com/caio-ribeiro-pereira/SimulaRSO

Site oficial: http://www.simula-rso.appspot.com

4. CONCLUSÃO

Com o avanço constante das tecnologias, compreender o comportamento lógico de um sistema operacional passou a ser um diferencial para entender e manipular os dispositivos e computadores de forma correta e eficiente.

Sistemas Operacionais é uma disciplina em que há muito a ser explorado, pois existe uma profunda teoria em seus conceitos que raramente são vistos na prática durante as aulas, pois existem poucas ferramentas conhecidas e de fácil acesso que apresentem de forma clara e intuitiva para o aluno o que ocorre com esses recursos internos do sistema.

O problema estudado lida com o cenário da criação de mais uma ferramenta de ensino voltada para a disciplina de Sistemas Operacionais, que através de simulações gráficas auxilie professores e principalmente alunos no processo de aprendizagem.

Para isso, foram realizados estudos com alguns usuários da turma do 5° semestre de Sistemas de Informação e Ciências da Computação do ano de 2011, para que fosse aplicada uma *interface* amigável no simulador e neste projeto foram implementados apenas alguns dos algoritmos – os que geralmente são apresentados em sala de aula – deixando espaço para adicionar futuras extensões.

Na modelagem do sistema foram utilizados alguns *Padrões de Projetos* na arquitetura do sistema, já citados durante a especificação do projeto para permitir a modularização do projeto e facilitar no trabalho de futuras extensões e ou manutenções neste simulador.

5. REFERÊNCIAS BIBLIOGRÁFICAS

Caelum, Caelum – Ensino e Inovação, Documentação do VRaptor, Disponível em <http://vraptor.caelum.com.br>, acessado em fevereiro de 2011.

Calil, L. E. Silingowschi. A Revolução Digital. Mundo dos Filósofos, 10 maio 2007, disponível em <http://www.mundodosfilosofos.com.br/lea20.htm>, acessado em setembro de 2011.

Carvalho, D. S.; Balthazar, G. R.; Dias, C. R.; Araújo, M. A. P.; Monteiro, P. H. R. S²O: Uma Ferramenta de Apoio ao Aprendizado de Sistemas Operacionais. Disponível em <http://www.natalnet.br/sbc2006/pdf/arq0107.pdf>, acessado em maio de 2011.

Google, Google App Engine, Guia do Desenvolvedor, disponível em <http://code.google.com/intl/pt-BR/appengine/docs/>, acessado em março de 2011.

JUnit, JUnit Resources for Test Driven Development, JUnit JavaDoc, disponível em <http://kentbeck.github.com/junit/javadoc/latest/>, acessado em março de 2011.

Microsoft, Linguagem de programação C#, Centro para iniciantes, disponível em <http://msdn.microsoft.com/pt-br/beginner/bb308730.aspx>, acessado em setembro de 2011.

Oracle, Java Platform Enterprise Edition, v6.0, API Specifications, disponível em <http://download.oracle.com/javaee/6/api/>, acessado em fevereiro de 2011.

Renata de Andrade P. Almeida; Tiago F. Lima; Paulo S. Rabelo, SAE - Simulador para Algoritmos de Escalonamento, disponível em <https://sistemas.usp.br/siicusp/cdOnlineTrabalhoVisualizarResumo?numeroInscricaoTrabalho=3488&numeroEdicao=18>, acessado em agosto de 2011.

Rocha, A. R.; Schneider, A.; Alves, J. C.; Silva, R., M. A. wxProc – Um Simulador de Políticas de Escalonamento Multiplataforma, disponível em <http://www.ic.unicamp.br/~rocha/pub/papers/wxProcUmSimuladorPoliticasEscalanonamento.pdf>, acessado em maio de 2011.

Rodrigues, A. A.; Pereira, C. R. SimulaGP – Simulador de Gerenciamento de Processos, 10º Congresso Nacional de Iniciação Científica, Conic-Semesp, 2010

Silberschatz, Abraham; Galvin, Peter Baer. Sistemas Operacionais com Java, 7ª Edição, Brasil, Campus, 2008.

Tanenbaum, Andrew S. Sistemas Operacionais Modernos, 2ª Edição, Brasil, Prentice Hall, 2009.

UML, Unified Modeling Language, Introduction To OMG's Unified Modeling Language™, disponível em <http://www.uml.org>, acessado em agosto de 2011.

W3CSchools, The World Wide Web Consortium (W3C), W3CSchools Online Web Tutorials, disponível em <http://www.w3schools.com>, acessado em fevereiro de 2011.