

# Serviço para execução de processamento em lotes (batch) com agendamento e gerenciamento pela web

Douglas M. Dornelles da Silva<sup>1</sup>, Guilherme Bertoni Machado<sup>1</sup>

<sup>1</sup> Curso de Análise e Desenvolvimento de Sistemas –  
Faculdade de Tecnologia Senac RS (FATEC/RS) – Porto Alegre – RS – Brasil  
dornelles.poa@gmail.com, gb.machado@sinprors.org.br

**Abstract.** Companies aims to automate manual tasks through the execution of batch processes (jobs that run on specific operating system and do not require user intervention) at a given frequency, usually when computing resources are less busy. To do this they must be scheduled by the administrator. The system proposed in this paper is a set of integrated applications to a Windows Service, which the user can schedule and monitor the implementation of batch processes registrations, view the executions summary and order it on your e-mail (for situations of remote monitoring) through a web application. As the main contribution of this work can be said that the developed tool helps in the management of batch processes safely and effectively, regardless of the complexity of the jobs execution flow.

**Keywords:** Processes, Windows Service, Scheduling, Management, Batch.

**Resumo.** As empresas têm a necessidade de automatizar tarefas manuais através da execução de processos batch (jobs específicos que rodam no sistema operacional e que não precisam da intervenção do usuário) com uma determinada frequência, normalmente quando os recursos computacionais são menos ocupados. Para isso estes devem ser agendados pelo administrador. O sistema proposto neste trabalho é um conjunto de aplicações integradas a um Windows Service, onde através de uma aplicação web o usuário pode agendar e acompanhar a execução dos processos batch cadastrados, visualizar o resumo das execuções e solicitá-lo em seu e-mail (para situações de monitoração remota). Como principal contribuição deste trabalho pode-se afirmar que a ferramenta desenvolvida contribui na administração dos processos batch de forma segura e eficaz, independente da complexidade dos fluxos de execução dos jobs.

**Palavras-chave:** Processos, Windows Service, Agendamento, Gerenciamento, Processamento em Lote.

## 1. Introdução

O processamento em lote tem sido associado com computadores de grande porte desde os primórdios da computação, pois inicialmente, os computadores não eram capazes de ter vários programas carregados na memória principal.

Quando surgiram os sistemas em lote, nos anos 60, o programador submetia um *job* através de cartões perfurados carregando-os para a sala de máquinas. Quando vários *jobs* já haviam sido montados, o operador lia todos eles como um único lote [Tanenbaum 2008].

Sistemas de informação evoluem de maneira acelerada a fim de acompanhar as evoluções tecnológicas, onde os sistemas *web* acabam adquirindo cada vez mais essa fatia no mercado. Mesmo nestas condições, as empresas ainda necessitam executar processos repetitivos e até mesmo com grande volume de informações, utilizando para isso as aplicações *batch*.

Processos *batch* são *jobs* específicos executados no sistema operacional e que não necessitam da intervenção do usuário. Normalmente são executados em períodos onde os recursos computacionais são menos ocupados. Gerenciar e agendar, de forma flexível, é a chave para o sucesso da execução dos processos em lote.

O objetivo deste trabalho é disponibilizar um conjunto de aplicações integradas a um *Windows Service* (denominada Controla Processos), onde através de uma aplicação *web* o usuário administrador pode agendar e acompanhar a execução dos processos *batch* cadastrados. É necessário salientar que existe uma limitação de escopo na ferramenta desenvolvida.

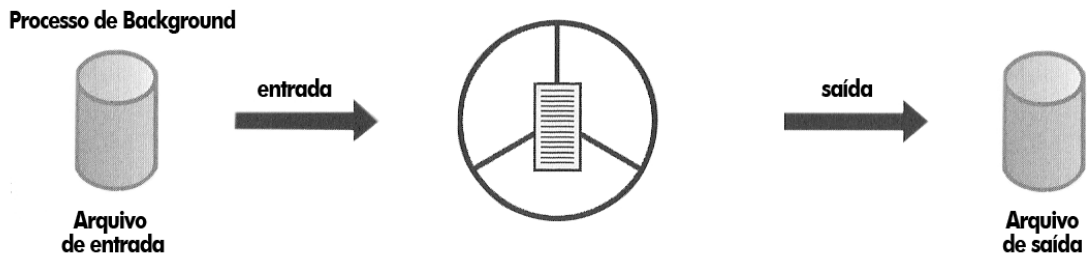
Para que os processos sejam completamente gerenciáveis, os consoles devem seguir uma estrutura padrão. Portanto, fica como motivação o controle de processos independentes, utilizando componentes fornecidos pela própria linguagem de programação, mas que não serão aplicados neste trabalho.

Ao longo deste artigo será explicado como este conjunto de ferramentas foi criada, incluindo as tecnologias envolvidas e a arquitetura utilizada.

O Artigo está organizado da seguinte forma: na Seção 2 é apresentado o referencial teórico do trabalho, na Seção 3 serão apresentados os trabalhos relacionados. Na Seção 4 é esplanada a descrição sobre o sistema proposto. Na Seção 5 é mostrada a solução. Na Seção 6 são apresentados os testes, com os respectivos resultados, executados para comprovar seu funcionamento. Na Seção 7, são apresentadas as conclusões e as sugestões para trabalhos futuros.

## **2. Referencial Teórico**

Um sistema operacional executa uma variedade de programas, entre eles temos os sistemas *batch* (*jobs*) e sistemas *time-sharing* (programas de usuário os tarefas). Um processo *background* é aquele onde não existe a comunicação com o usuário durante o seu processamento. O processamento do tipo *batch* é realizado através de processos *background*, conforme Figura 1 [Machado e Maia 2008].



**Figura 1. Processos em segundo plano extraído de Machado e Maia (2008).**

Nos primeiros sistemas computacionais, somente um programa era executado em cada momento (dominando todos os recursos), enquanto nos sistemas atuais, temos vários programas executando concorrentemente (multiprogramação), exigindo maior controle e compartimentalização dos vários programas [Tanenbaum 2008].

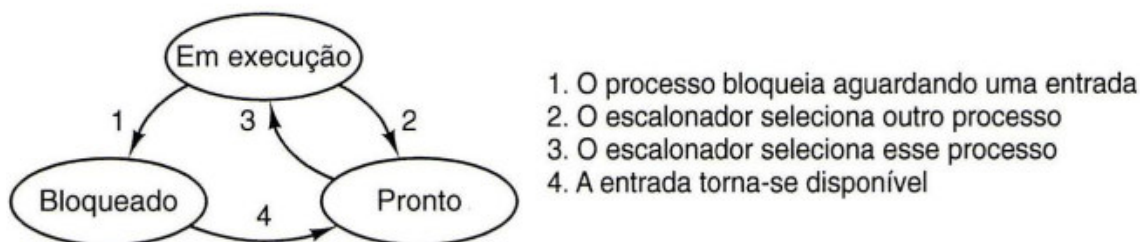
Um processo é uma abstração utilizada para representar um programa em execução, o mesmo contém toda informação necessária para completar uma computação, mas não devemos confundir processo com programa, já que um programa é apenas parte do estado de um processo, ou seja, um programa pode gerar diversos processos.

A gerência de processos é uma das principais funções de um sistema operacional, possibilitando aos programas alocar recursos, compartilhar dados, trocar informações e sincronizar suas execuções [Machado e Maia 2008].

O sistema operacional deve assegurar que os processos não interfiram uns com os outros, já que vários processos podem compartilhar uma única CPU (unidade central de processamento), isto significa que o mesmo deve garantir que todos os processos tenham acesso aos recursos, como memória e CPU (escalonamento), além de controlar o estado dos processos, de tal modo que um processo não possa modificar arbitrariamente o estado de outro processo (proteção). O estado dos processos, durante sua existência, passa pelos estados:

- **Executando:** utilizando a CPU de fato naquele instante;
- **Pronto:** executável; temporariamente parado para dar lugar a outro processo;
- **Bloqueado:** não-executável até que um evento externo aconteça.

As transições entre esses estados aparecem ilustradas, conforme a Figura 2.



**Figura 2. Estado dos Processos extraído de Tanenbaum [2008].**

Quatro transições são possíveis entre esses três estados, conforme se vê na figura 1. A transição 1 ocorre quando um processo descobre que ele não pode prosseguir. Em alguns sistemas, o processo precisa executar uma chamada ao sistema, como *block* ou *pause*, para entrar no estado bloqueado. Em outros sistemas, inclusive no Unix, quando um processo lê de um *pipe* ou de um arquivo especial (por exemplo, um terminal) e não há entrada disponível, o processo é automaticamente bloqueado.

As transições 2 e 3 são causadas pelo escalonador de processos - uma parte do sistema operacional -, sem que o processo saiba disso. A transição 2 ocorre quando o escalonador decide que o processo em execução já teve tempo suficiente de CPU e é momento de deixar outro processo ocupar o tempo da CPU.

A transição 3 ocorre quando todos os outros processos já compartilharam a CPU, de uma maneira justa, e é hora de o primeiro processo obter novamente a CPU. O escalonamento - isto é, a decisão sobre quando e por quanto tempo cada processo deve executar.

A transição 4 ocorre quando acontece um evento externo pelo qual um processo estava aguardando (como a chegada de alguma entrada). Se nenhum outro processo estiver executando naquele momento, a transição 3 será disparada e o processo executará. Caso contrário, ele poderá ter que aguardar em estado de pronto por um pequeno intervalo de tempo, até que a CPU esteja disponível e sua vez chegue [Tanenbaum 2008].

Quando um computador é multiprogramado, ele muitas vezes tem variados processos que competem pela CPU ao mesmo tempo, essa situação ocorre sempre que dois ou mais processos estão simultaneamente no estado de pronto. O escalonador é a parte do sistema operacional que faz a escolha de qual processo deve ser executado, e o algoritmo que é usado, é chamado de algoritmo de escalonamento. O algoritmo de escalonamento pode ser [Silberschatz et al 2001]:

- **Não preemptivo:** O processo executa até o fim, sem ser interrompido;
- **Preemptivo:** O processo executa em fatias de tempo determinado pelo sistema operacional.

É obtido maior desempenho e rapidez na execução dos processos, uma vez que é tarefa do escalonador definir quais processos os processadores do computador devem executar, determinado os que devem aguardar e os que serão executados naquele momento.

Como existem diversos sistemas operacionais, temos diferentes algoritmos de escalonamento, onde destacamos os seguintes ambientes:

- **Lote:** não temos o usuário aguardando um retorno do sistema, portanto algoritmos com grande intervalo de tempo normalmente são viáveis;
- **Interativo:** um processo não pode ocupar completamente a CPU, pois assim negaria serviços aos outros, então neste caso é necessário que tenhamos a preempção, evitando assim este comportamento;

- **Tempo real:** estes visam o progresso da aplicação, onde a preempção algumas vezes não é necessária, já que os processos não executam por muito tempo, realizam o que devem fazer e logo bloqueiam.

Segundo Tanenbaum [2008], os gerentes de grandes centros de computação – que executam muitos jobs em lote – observaram, em geral, três métricas para verificar se os sistemas deles estão executando bem ou não: vazão, tempo de retorno e utilização da CPU.

**Vazão** é o número de *jobs* por hora que o sistema termina. Considerando tudo que foi discutido, terminar 50 *jobs* por hora é melhor do que terminar 40 no mesmo período.

O **tempo de retorno** é estatisticamente o tempo médio do momento em que um job em lote é submetido até o momento em que ele é terminado. Ele indica quanto tempo, em média, o usuário tem de esperar pelo fim de um trabalho. Aqui a regra é quanto menor, melhor.

Resumindo, as principais métricas que um gerente busca em um sistema em lote são:

- **Vazão** (*throughput*): maximizar o número de jobs por hora;
- **Tempo de retorno:** minimizar o tempo entre a submissão e o término;
- **Utilização de CPU:** manter a cpu ocupada o tempo todo.

O escalonador do Windows é preemptivo com prioridades. As prioridades são organizadas em duas classes: tempo real e variável. Cada classe possui 16 níveis de prioridades, sendo que as *threads* da classe tempo real têm precedência sobre as *threads* da classe variável, isto é, sempre que não houver processador disponível, uma *thread* de classe variável é preemptada em favor de uma *thread* da classe tempo real [Teixeira 2007].

Serviço no sistema operacional windows é conceituado como um programa, rotina ou processo que executa uma função específica do sistema para dar suporte a outros programas, especificamente em um nível baixo. Um serviço é um tipo de aplicativo que é executado no plano de fundo do sistema sem uma interface de usuário e é similar a um processo UNIX *daemon* [Microsoft 2010].

### 3. Trabalhos Relacionados

Existem no mercado ferramentas para agendamento de processos, mas grande parte dessas aplicações possui funcionalidades e até mesmo programas relacionados à mesma que acabam dificultando o manuseio pelo usuário, que necessita apenas o módulo de agendamento de processos *batch*.

Durante a pesquisa foram estabelecidos 3 atributos (agendamento ilimitado, *log* por email e aplicação web) que serviram de guia funcional em relação ao que o agendador de tarefas do Windows não oferece e ao que a ferramenta desenvolvida dispõe. O sistema mais correlato ao Controla Processos é a ferramenta Z-Cron Scheduler [Baumann 2010].

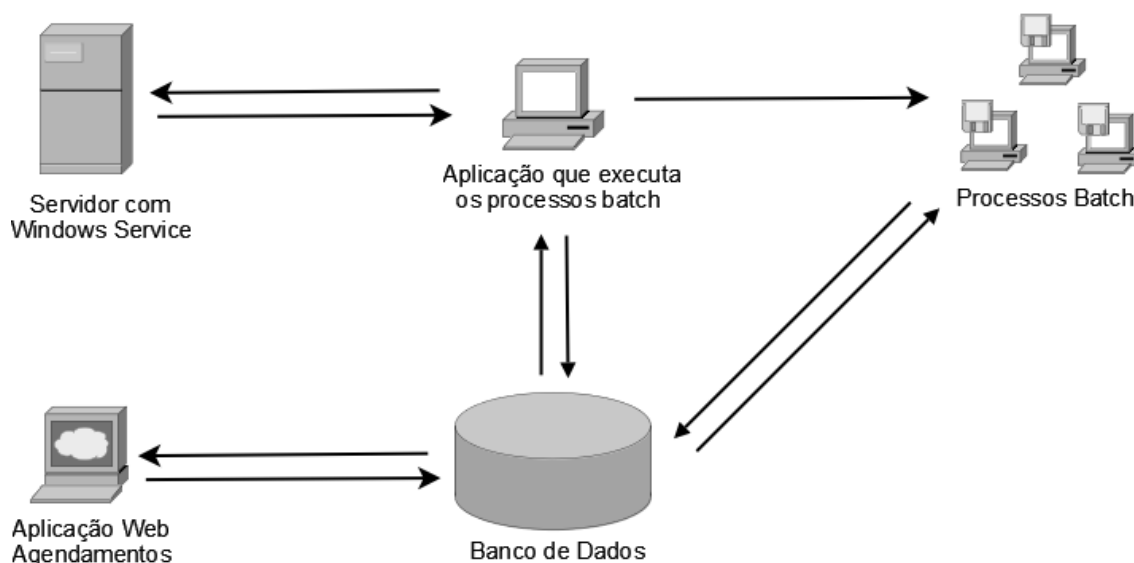
Outras ferramentas no qual são focadas no gerenciamento, por maioria acabam sendo gratuitas, ou até mesmo incorporadas no sistema operacional, como é o caso da agenda de tarefas do Windows, e tem limitações ao gerenciamento dos processos, conforme Tabela 1.

**Tabela 1. Trabalhos correlatos e seus principais atributos**

Nome do Sistema	Agendamento Ilimitado	Log por Email	Aplicação Web
Tarefas Agendadas do Windows			
Z-Cron Scheduler	X	X	
Senac Controla Processos	X	X	X

#### 4. Sistema Proposto

O sistema é um conjunto de aplicações integradas a um Windows Service, onde através de uma aplicação web o usuário pode agendar e acompanhar a execução dos processos *batch* cadastrados, visualizar o resumo das execuções e solicitá-lo em seu e-mail (para situações de monitoração remota). A solução contempla três diferentes tipos de aplicações, conforme Figura 3.



**Figura 3. Arquitetura do Sistema Proposto**

- **Net:** Aplicação Windows Service, na qual será instalada em computadores com sistema operacional windows (NT, 2000, XP, 2003, Vista e 2007). Por se tratar de um serviço do windows, utiliza o recurso de gravar mensagens ou eventos no *Log* de Eventos do mesmo. Seu objetivo consiste que a cada minuto seja executado a aplicação Starter em *background* no sistema operacional;
- **Starter:** Aplicação console que realiza um processo de verificação e decisão de qual processo *batch* foi configurado para ser executado naquele determinado momento;

- **ControlaProcessos:** Ferramenta Web onde o usuário vai realizar o agendamento dos processos *batch* no qual ele deseja que sejam disparados automaticamente, possibilitando o mesmo a gerenciar as seguintes informações:
  - Definir que um processo seja executado em determinados dias da semana ou do mês, em determinados horários e de forma única ou repetitiva;
  - Executar qualquer programa que possa ser disparado via linha de comando;
  - Informar sobre as atividades executadas via *log* e via e-mail;
  - Cada processo pode ser configurado para enviar e-mails distintos para atividades distintas.

## 5. Solução

O sistema foi desenvolvido utilizando-se tecnologias da Microsoft Corporation que auxiliam a construção de aplicações. Essas tecnologias envolvidas no desenvolvimento do sistema são descritas nas próximas subseções.

### 5.1. Persistência de dados

Para persistência (armazenamento) dos dados foi utilizado o banco de dados Microsoft SQL Server 2005 Express Edition, um produto disponível gratuitamente. O SQL Server é um sistema de administração relacional multicomponente centrado em alta-performance, com um sistema de banco de dados altamente disponível [Watt 2006].

Trata-se de uma ferramenta confiável para gerenciamento de dados, na qual fornece recursos robustos, proteção de dados e desempenho para clientes de aplicativos incorporados, aplicativos Web simples e armazenamentos de dados locais, conforme Figura 4.

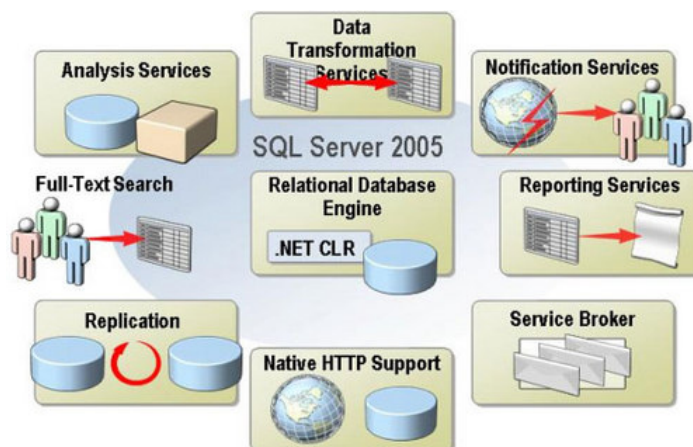


Figura 4. Microsoft Sql Server 2005 Express extraído de Agarwal (2006)

### 5.2. Servidor Web

Como servidor Web foi utilizado o IIS (*Internet Information Services*) na sua versão 5.1. IIS é um servidor web criado pela Microsoft para seus sistemas operacionais para servidores.

Sua primeira versão foi introduzida com o Windows NT Server versão 4, e passou por várias atualizações. Neste trabalho foi utilizada a versão distribuída junto com o Windows XP versão *Professional* no qual só pode servir a 10 conexões simultâneas e não aproveita todos os recursos do servidor.

### 5.3. Padrão de Projeto

O sistema segue o padrão de projeto MVC (*Model View Controller*), no qual tem como base a divisão em camadas, separando as classes dos *layouts*, ficando assim independentes, ou seja, podem ser alterados sem se afetarem. As camadas do MVC são referidas conforme Figura 5.

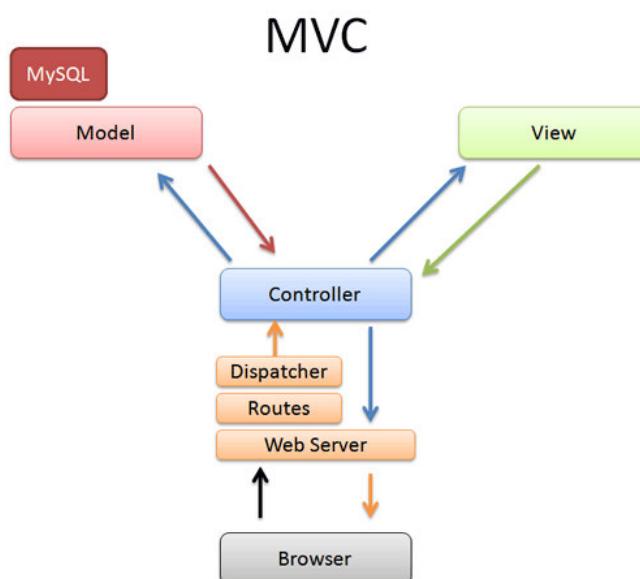


Figura 5. MVC (Modelo Visão Controlador) extraída de Santos (2009)

O MVC define que uma aplicação seja dividida em três camadas:

- **Modelo** (*model*) – Responsável pelo acesso aos dados com segurança e persistência;
- **Visão** (*view*) – É a interface com o usuário, recebe os dados de entrada fornecidos pelo mesmo e renderiza os dados para apresentá-los;
- **Controlador** (*controller*) – Responsável pelas regras de negócio, verificações e métodos de ação da aplicação.

### 5.4. Linguagem de Desenvolvimento

A linguagem de programação utilizada para o desenvolvimento deste trabalho foi o Visual Basic.Net, distribuída com o Microsoft Visual Studio 2005, que permite o desenvolvimento orientado a objetos.

A ferramenta disponibiliza um *template* específico para a criação de um serviço Windows desde a chegada da plataforma .NET, fazendo com que não seja necessário recorrer a ferramentas de terceiros para o criar projetos deste tipo.

## 6. Testes e Resultados

Para a validação, o sistema foi disponibilizado para 3 usuários distintos (funcionário de uma empresa de desenvolvimento de software para gestão de pessoas, um administrador de rede e um testador de software certificado).

Foram desenvolvidos alguns processos em lote para testar a aplicação. Dentre estes, um processo que criava uma quantidade de arquivos texto especificado por parâmetro, um processo que realizava a exclusão de arquivos, outro processo no qual realizava a leitura de uma determinada tabela do banco de dados e exportava as informações para um arquivo texto, e, por último, um processo para simular um processamento de maior consumo de recursos, denominado de processamento de folha de pagamento.

Após o período de 1 semana de testes, os usuários responderam um questionário com questões sobre o software e espaços para expor suas opiniões sobre a utilização do sistema. Com as questões disponibilizadas - utilizando um escala de 1 (grau mais baixo – muito ruim) a 5 (grau mais alto – muito bom) - foi possível consolidar os seguintes resultados ilustrados pela Figura 6.

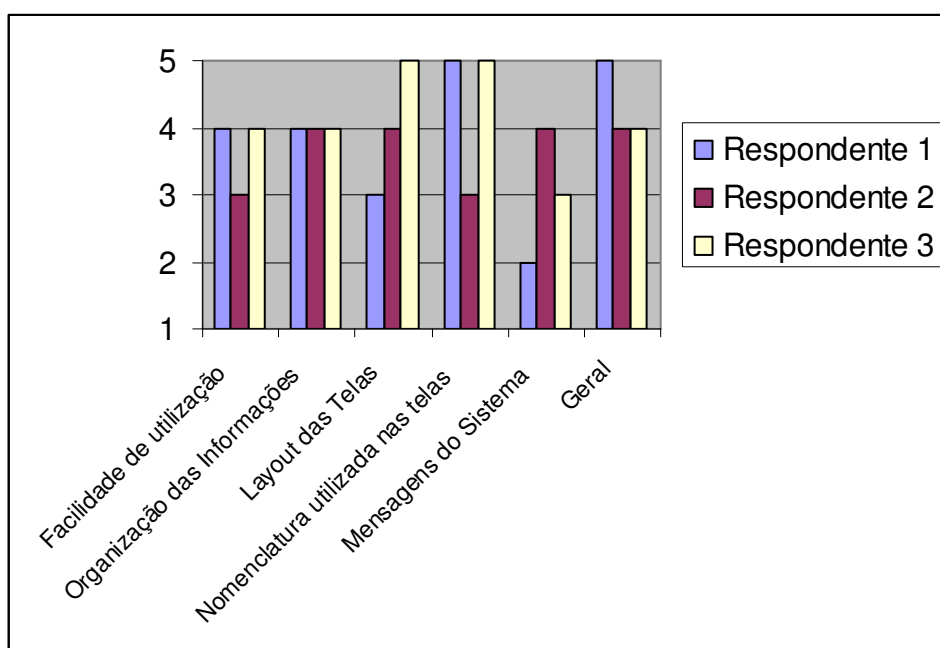


Figura 6. Resultados da avaliação do sistema

## 7. Conclusões

No desenvolvimento deste trabalho, foi possível confirmar que apesar do avanço tecnológico, ainda existe a necessidade e certa dificuldade para o controle e gerenciamento do processamento em lote. Estes processos ainda estão muito sujeitos a erros, pois mesmo não necessitando da intervenção manual para operações dos mesmos, sofrem atrasos nas tarefas diárias das empresas.

Um exemplo desta situação é o fechamento da folha de pagamento de uma empresa, que por ser um processo demorado onde não existe a comunicação com o

usuário durante o seu processamento deve ser agendado para executar após o horário de expediente e caso ocorra algum problema na execução deste processo, o usuário só vai ficar a par quando necessitar destas informações no dia seguinte. É importante ressaltar que esta ferramenta não está preparada para garantir que haja recursos computacionais, assim como erros de fatores externos, como por exemplo, falha de energia e afins.

Este trabalho visa contribuir na administração dos processos *batch* de forma segura e eficaz, independente da complexidade dos fluxos de execução dos *jobs*, atendendo os requisitos mais importantes para o controle e gerenciamento do processamento em lote, possibilitando benefícios aos seus usuários.

Durante o processo de desenvolvimento deste software, surgiram novos requisitos (agregação de funcionalidades na ferramenta) para que esta solução seja homologada de maneira eficaz e confiável. Ao analisar as sugestões enviadas no final dos testes com os usuários, foram mapeados os seguintes trabalhos futuros:

- A ferramenta gerenciar qualquer processo *batch*, sem a necessidade do *job* seguir o fluxo de controle, verificando se os processos terminaram normalmente;
- Implementar o conceito de dependência de processo, visando que um *job* só possa ser executado após a execução de outro como premissa;
- Realizar o controle de fila na execução dos processos, evitando que um *job* não tenha duas instancias;
- Permitir o gerenciamento mais eficaz na administração dos *log's* e e-mails gerados;
- Geração de estatísticas de controle geral sobre os *jobs* agendados.

## Referências

- Agarwal, Vidya Vrat. (2006) “SQL Server 2005 Overview”. Disponível em: <http://www.programmersheaven.com/2/SQL-server-2005-school-lesson-1>
- Baumann, Andreas. (2010) “Z-Cron :: Task and backup scheduler”. Disponível em: <http://www.z-cron.com/index.html>.
- Machado, Francis B., Maia Luiz P. (2008) “Arquitetura de Sistemas Operacionais”, 4nd Edição. LTC.
- Microsoft. (2010) “Microsoft TechNet Home Page”. Disponível em: <http://technet.microsoft.com>
- Santos, Ubiracy. (2009) “O Padrão MVC”. Disponível em: <http://usantos.wordpress.com/category/mvc/>
- Silberschatz, Abraham, Galvin, Peter, Gagne, Greg. (2001) Sistemas Operacionais - Conceitos e Aplicações, Ed. Campus.
- Tanenbaum, Andrew S. (2008) “Modern Operation Systems”, 2nd Edition. Pearson Education do Brasil.
- Watt, Andrew. (2006) “Microsoft SQL Server 2005 for Dummies”. Indianapolis, IN, EUA: Wiley Publishing, Inc.