UNIVERSIDADE FEDERAL DE SANTA CATARINA

“Monitoramento de servidores com scripts”

André Felipe Durieux

Florianópolis – SC

2012 / 2

UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE SISTEMAS DE INFORMAÇÃO

"Monitoramento de servidores com scripts"

André Felipe Durieux

Trabalho de conclusão de curso submetido ao Departamento de Informática e Estatística da Universidade Federal de Santa Catarina para a obtenção do Grau de Bacharelado em Sistemas de Informação.

Florianópolis – SC
2012 / 2
André Felipe Durieux

“Monitoramento de servidores com scripts”

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação

Orientador(a): Prof. Carla Merkle Westphall, Dra.

Banca Examinadora

Prof. Carlos Becker Westphall, Dr.
Universidade Federal de Santa Catarina

Prof. João Bosco Mangueira Sobral, Dr.
Universidade Federal de Santa Catarina

Agradecimentos

Agradeço primeira e especialmente aos meus pais por estarem sempre presentes.

Gostaria também de agradecer a todos os meus amigos e colegas da faculdade que estiveram presentes nos momentos em que mais necessitei, em especial Samuel Pierri Cabral, Marco Antônio Curi, Jonatan Matschulat, Rodrigo Becker e Leonardo Defenti.

Por último, quero agradecer à orientadora Prof. Carla Merkle Westphall e membros da banca por me guiarem durante este trabalho.

Sumário

Listas de ilustrações

Lista de abreviaturas e siglas

CSS – Cascading Style Sheets (Folhas de Estilo em Cascata)

HTML – Hypertext Markup Language (Linguagem de Marcação Hipertexto)

HTTP – Hypertext Transfer Protocol (Protocolo de Transferência de Hipertextos)

PHP – Hypertext Preprocessor (Pré-processador de Hipertexto)

SNMP – Simple Network Management Protocol (Protocolo simples para gerenciamento de redes)

Resumo

O monitoramento de redes, aplicações, servidores, ou seja, dispositivos importantes conectados à rede vem se tornando de suma importância nos dias de hoje para a gestão da tecnologia da informação. Organizações que oferecem serviços através da rede, necessitam que seus sistemas permaneçam funcionando. Logo, a necessidade por maneiras de se tentar garantir o máximo de disponibilidade para o ambiente vem crescendo consideravelmente.

Este monitoramento, em tempo real, permite obter as informações necessárias sobre estes equipamentos de modo rápido, sintético, preciso e confiável, facilitando a tomada de decisão do gestor no momento do planejamento, adequação, suporte e expansabilidade da infra-estrutura organizacional.

Informações em tempo real sobre o que está acontecendo no ambiente da empresa permitem que seja possível evitar problema maiores, visando garantir uma estrutura estável para a organização.

Palavras-chave: Monitoramento, SNMP, WMI

Abstract

The monitor of networks, applications and servers is becoming very important nowadays for the any organization that wishes to offer always better services and products. Those organizations need their products or services to stay online and working for the most possible time, not to say always. Therefore, the search for ways to maintain and improve the quality of those services keep growing.

Such real-time monitoring allows the system administrators to obtain all kind of informations about any device they may choose to monitor in a very fast, trusty and easy way. It helps a lot when the time to make a decision come.

Those informations about the health of the organization's environment can also help them to avoid bigger problems, which is very important to maintain the environment stability as well.

1 INTRODUÇÃO

No universo dos sistemas computacionais, é essencial que os administradores de sistemas sejam capazes de identificar e até prever as iminentes falhas que eventualmente virão a ocorrer. Dentro deste universo, existem ferramentas cuja exclusiva responsabilidade é realizar, de maneira precisa, o monitoramento de outras aplicações e dos servidores onde estas aplicações estão implantadas. Este tipo de software é extremamente importante para qualquer organização e visa garantir que suas aplicações e sistemas permaneçam funcionando e que forneçam aos administradores de redes informações necessárias para que estes possam tomar as providências necessárias.

Neste mundo complexo de roteadores, switches e servidores, a tarefa de gerenciá-los e principalmente garantir não só que estejam no ar e funcionando, mas que estejam configurados da melhor maneira possível, pode não parecer muito convidativa (MAURO, DOUGLAS, 2001)

Um sistema de informações eficiente facilita a comunicação e acelera a troca de informações e a tomada de decisão, o que acaba por aumentar a produtividade. É justamente a saúde destes sistemas que garantirá às organizações que seus ambientes permaneçam estáveis.

1.1 Motivação

O velho modelo de um computador atendendo a todas as necessidades computacionais de uma empresa foi substituído por um conjunto de computadores autônomos interconectados que podem trocar informações, conhecido como redes de computadores.

Estas redes tornaram-se um importante e impactante diferencial sobre a sociedade e suas medidas de negócios. Nesse sentido, é de fundamental importância que os administradores destas redes possuam ferramentas que os auxiliem a monitorar os componentes, especialmente os críticos, de sua infra-estrutura. A antecipação de futuras situações – tendências – que demandem a tomada de ações por partes destes administradores, tanto para correção de alguma configuração qualquer que esteja incorreta ou até mesmo para a substituição de um dispositivo que esteja em mau funcionamento.

Dito isso, a motivação deste trabalho é justamente o desenvolvimento de uma ferramenta flexível e customizável, que venha a colaborar e facilitar o trabalho dos administradores de redes, tentando antecipar situações consideradas críticas.

1.2 Objetivo Geral

O objetivo este trabalho tem como essência o desenvolvimento de uma aplicação para monitoramento de servidores e aplicações. Neste processo, serão descritas as ferramentas e protocolos envolvidos, para oferecer uma maior clareza e entendimento sobre o projeto.

1.3 Objetivos Específicos

Os objetivos específicos deste trabalho são:

1. Descrever conceitos básicos referentes a monitoramento de servidores
2. Modelar e descrever a aplicação a ser desenvolvida
3. Definir os requisitos para esta aplicação
4. Desenvolver uma aplicação de monitoramento que atenda aos objetivos especificados

1.4 Organização do Trabalho

O capítulo 1 apresentou a motivação e os objetivos, geral e específicos, do trabalho. O restante deste trabalho está organizado da seguinte forma: no capítulo 2, serão descritos os conceitos básicos de tudo que será abordado na proposta da aplicação.

Já no capítulo 3, outras ferramentas de monitoramento opensource serão brevemente descritas e caracterizadas. No 4º capítulo será, então, apresentada a proposta da aplicação. O capítulo 5 conterá as considerações finais a respeito deste trabalho, enquanto no 6º serão descrtias as referências.

2 CONCEITOS BÁSICOS

Com o crescimento do uso de serviços online, o monitoramento de servidores e das aplicações por trás torna-se cada vez mais necessário e importante. Neste capítulo serão abordados alguns tipos de servidores, de bancos de dados e de aplicações que serão utilizados no desenvolvimento da aplicação e que funcionam como base para várias empresas hoje em dia.

O Java é uma das tecnologias mais utilizadas atualmente para o desenvolvimento de serviços e aplicações. Começou a ser criada em 1991 com o nome de Green Project, mas sem muito sucesso. Em 1995 foi lançada pela SUN oficialmente como linguagem para a construção de aplicativos, já com o nome Java. Desde então, o seu uso e desenvolvimento vem crescendo bastante. Várias empresas a utilizam para desenvolver seus produtos e serviços. Em função disso, o monitoramento da plataforma Java é importante para garantir a estabilidade destes produtos e serviços, oferecidos através da internet. Uma das maneiras de se monitorar a JVM é através dos MBeans.

A figura 1 mostra a interface do JConsole, aplicativo para gerenciamento da Java Virtual Machine.

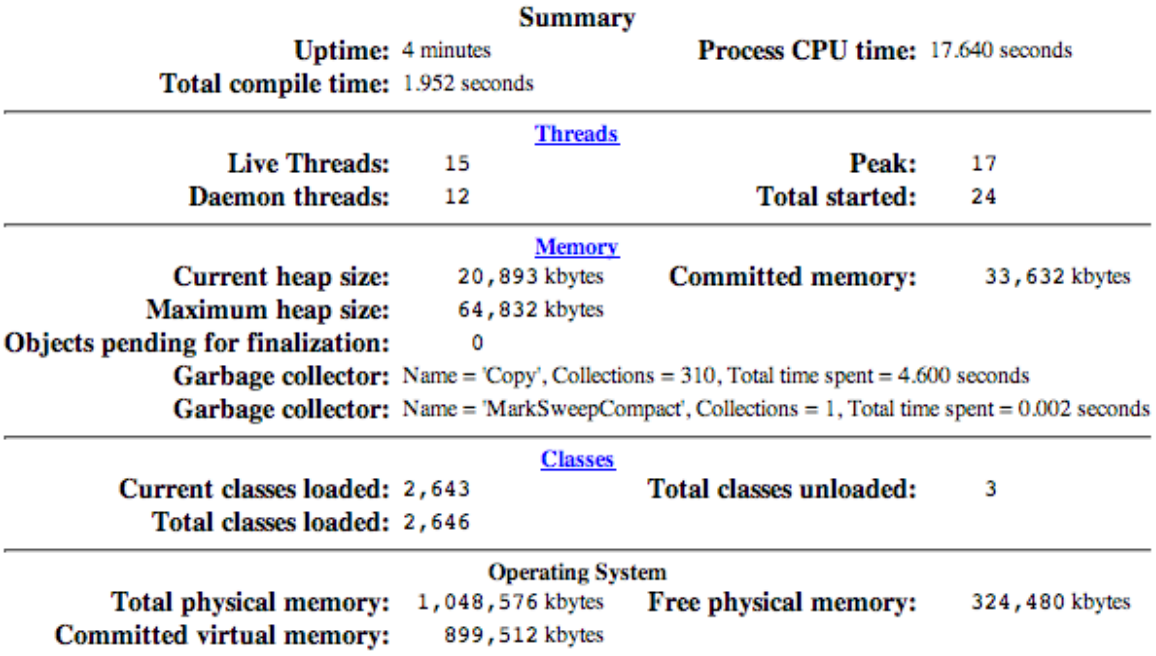

Figura 1 - Jconsole

Dados como memória alocada para a JVM, a memória heap utilizada, o número de threads ativas são exemplos importantes e devem ser monitorados. Com o monitoramento destes dados será possível identificar anomalias que poderão vir a causar problemas na aplicação.

Aplicações corporativas Java rodam sobre os chamados servidores de aplicação. O conhecido exemplo de servidor de aplicações corporativas Java é o JBoss.

O JBoss é um servidor de aplicação de código fonte aberto baseado na plataforma JEE – Java Enterprise Edition – implementado completamente na linguagem de programação Java. Por ser baseada em Java, o JBoss pode ser utilizado em qualquer sistema operacional que suporte o Java.

Atualmente está na versão 7, lançada em julho de 2011. Esta última versão implementa a mesma especificação da versão anterior, lançada 6 meses antes.

O JBoss oferece métricas que podem ser monitoradas através da JMX-Console, que funciona de maneira similar ao Jconsole, mas na web. JMX é a tecnologia Java que permite que aplicações, dispositivos que rodem java sejam monitorados remotamente.

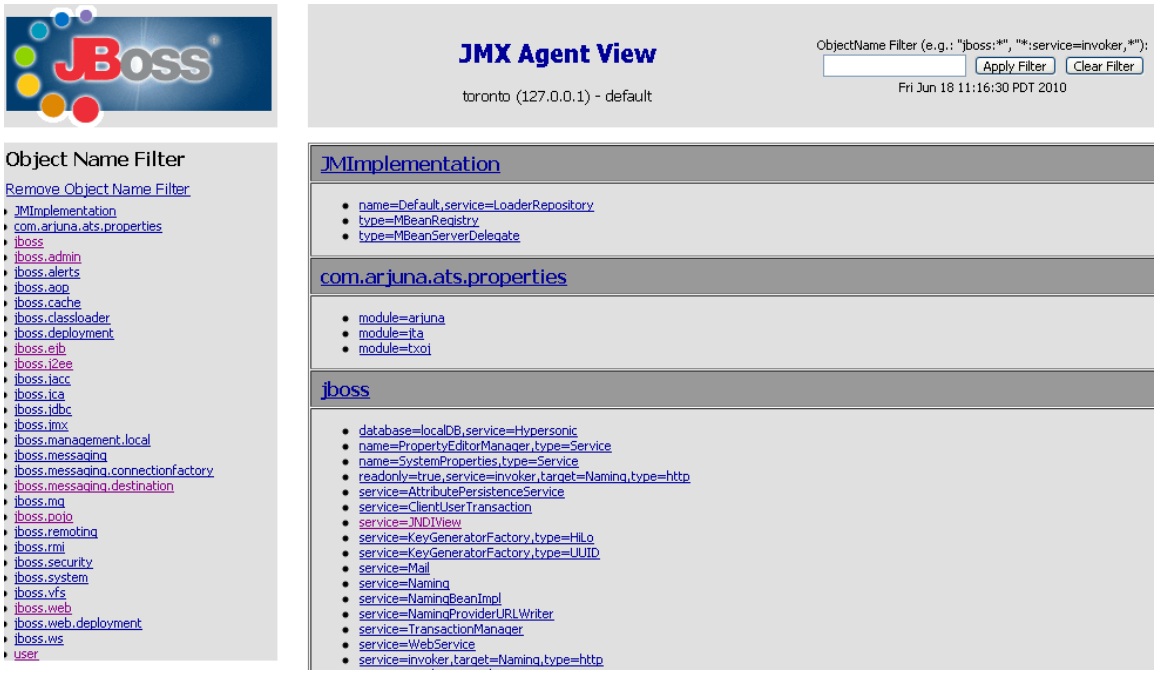

Figura 2 - Jmx console

O Linux é um sistema operacional desenvolvido e mantido por Linus Torvalds. Adota a licença GPL, isso quer dizer que todos podem usá-lo e redistribuí-lo nos termos da licença. Em qualquer uma de suas distribuições, é mais utilizado como servidor. Entretanto, sua utilização como desktop vem crescendo muito rapidamente, muito em função da liberdade que este sistema operacional oferece ao usuário.

Diferentemente do Linux, o Microsfot Windows, desenvolvido por Bill Gates, é um sistema operacional

proprietário. Ainda o mais utilizado como desktop, também é utilizado como servidor, com a sua versão server.

Já quando falamos de banco de dados, não podemos deixar de comentar sobre Oracle e MySQL. Este último, recentemente adquirido pela Oracle, ainda é um dos SGBD opensource mais utilizado mundialmente. Extremamente fácil de usar, está na versão 5.5.14. Já o Oracle, é um dos mais robustos, se não o mais robusto entre todos eles. Muitas organizações, de todo o mundo, o utilizam como banco de dados.

Estes SGBD's dispobilizam tabelas e view estatísticas de uso. Tais estatísticas, assim como nos servidores e aplicações, permitem uma interpretação em tempo real do que está acontecendo no banco de dados. Percentual de utilização da cache, da área de redo e o número de consultas que estão demorando mais de 10, 30 ou 60 segundos são exemplos importantes destas estatísticas.

2.1 Monitoramento de Redes

O gerenciamento e monitoramento são extremamente importantes para o correto funcionamento de uma rede de computadores, sendo que, sem operações de gerenciamento, uma rede local não tem como se manter operacional por muito tempo. Em especial, as grandes redes corporativas estão fadadas ao caos sem estas funções. Além de agirem relativamente, as tarefas gerenciais de rede também são proativas no sentido de detectar e antecipar eventuais falhas.

Para isso, devem ser utilizadas ferramentas de desempenho de rede que podem ajudar o administrador a determinar o status da rede e identificar as áreas que podem ser melhoradas para aumentar seu desempenho.

A idéia de gerenciamento de rede é fornecer ferramentas para que um administrador de rede seja capaz de monitorar equipamentos remotos, analisando os dados de modo a garantir o funcionamento e operação dentro dos limites especificados; controlar reativamente o sistema fazendo ajustes de acordo com as modificações ocorridas no sistema ou em seu ambiente e gerenciar pró-ativamente o sistema, detectando tendências ou comportamentos anômalos para que seja possível executar uma ação antes que surjam problemas mais sérios.

O gerenciamento de uma rede permite a detecção e correção de problemas que tornam a comunicação em uma empresa ineficiente ou impossível. Desta forma o controle sobre seus recursos é um investimento justificado na medida em que o administrador da rede deseja identificar imediatamente quando um ativo ou serviço da rede fica indisponível.

A utilização de software de gerência de rede é o mecanismo para que o administrador descubra problemas e isole sua causa. Esses tipos de softwares que são baseados principalmente no protocolo SNMP são capazes de monitorar o estado de serviços e equipamentos da rede.

Com o crescimento do número e da heterogeneidade dos equipamentos envolvidos nas redes, o número de problemas potenciais e a complexidade envolvida nestes problemas tornam-se críticos, e exigem que os gerentes de rede possuam uma vasta quantidade de informação sobre as redes manuseadas e os problemas destas. Assim, o gerenciamento de redes destina-se a auxiliar os gerentes a trabalhar com a complexidade dos dados envolvidos, de modo a garantir a máxima eficiência e transparência da rede para os seus usuários (Melchiors, 1999). O gerenciamento de uma rede inclui o oferecimento, a integração e a coordenação de elementos de hardware, software e humanos, para monitorar, testar, consultar, configurar, analisar, avaliar e controlar recursos da rede, além de elementos para satisfazer às exigências operacionais de desempenho e de qualidade de serviço, em tempo real a um custo razoável.

O monitoramento de redes tem a função de prover dados e ferramentas paraauxiliar o engenheiro de tráfego ou administrador de redes em basicamente quatroatividades, que são apresentadas a seguir.

a. Dimensionamento de infra-estrutura de TI: Este é um dos problemas mais difíceis para os administradores resolverem. O monitoramento fornece dados tais como a porcentagem do uso do link, o atraso inerente da rede, o tempo de resposta, etc. Esses dados podem ser valiosos para determinar os gargalos do sistema, que são pontos a serem alterados para uma melhor desempenho. Um exemplo seria de uma empresa que fosse lançar um novo produto com venda pela internet. Como deveria ser expandido sua infraestrutura afim de atender a nova demanda, que aumentaria com o lançamento do novo produto? Se não for feito um dimensionamento correto, os usuários podem não conseguir finalizar a compra no site da empresa. Como conseqüência, eles implesmente deixarão de comprar. O sucesso do produto depende é diretamente relacionado ao desempenho do sistema.

b. Segurança: Segurança atualmente não é mais uma opção, é uma necessidade para todos os tamanhos de empresas. Tanto para guardar os segredos comerciais e também para garantir a privacidade dos consumidores e funcionários. Imagine uma loja varejista online que tivesse todo o seu cadastro de clientes extraviado, incluindo dados como número de cartão de crédito. O onitoramento neste caso fornece marcas sobre atividades suspeitas na rede e pode ser usado para identificar origem de tentativas de ataques.

c. Caracterização do tráfego: Um administrador pode querer saber quais serviços estão sendo mais utilizados na rede. Para isso o monitoramento é feito para indicar o uso de cada serviço em relação a capacidade da rede. Um exemplo seria separar o tráfego por serviços (HTTP, FTP, SMTP) para dimensionar corretamente o sistema.

2.2 Monitoramento de servidores Linux

Em distribuições Linux, o monitoramento pode ser feito de várias maneiras. É possível recuperar informações relevantes sobre a utilização dos recursos dos servidores pode ser feito através de scripts que podem ser desenvolvidos em várias das linguagens de programação existentes. Além disso, pode-se utilizar o protocolo SNMP, que é amplamente utilizado para se monitorar tanto servidores Linux como qualquer outro dispositivo ligado à rede que implemente este protocolo.

O SNMP - Simple Network Management Protocol (Protocolo Simples de Gerência de Redes) é um protocolo da camada de aplicação utilizado no gerenciamento de dispositivos através do IP. Permite ao administrador de redes solicitar informações sobre um ou mais dispositivos IP conectados à rede. A simplicidade envolvida na troca de informações consolidou o SNMP como arquitetura dominante no que se diz respeito ao gerenciamento de dispositivos IP (PAUL E. SERVINÇ, 2004). Tornou-se um dos protocolos de gerenciamento de redes mais aceitos e utilizados.

Foi apresentado em 1988 pela IAB atráves da RFC 1157 para suprir a necessidade de um padrão de gerenciamento de dispositivos IP e oferece aos usuários uma pequena gama de operações que permitem que estes dispositivos sejam monitorados remotamente. Alguns dos objetivos do projeto SNMP era garantir o gerenciamento de rede integrado, ou seja, a capacidade de gerenciar redes incorporando componentes que venham de uma variedade de fabricantes com uma simples aplicação. Além disso, visava também a interoperabilidade, que era possibilitar que o equipamento de um vendedor pudesse ser gerenciado pelo equipamento de um outro vendedor. A padronização talvez tenha sido o objetivo principal. Métodos de comunicação e estruturas de dados foram definidas para que redes não similares pudessem ser integradas com o gerenciamento de rede.

O protocolo SNMP é baseado em dois elementos: o agente e as NMS. O agente é responsável por enviar informações sobre si mesmo para a NMS. Este também envia alertas, conhecidos como traps, para as estações de gerenciamento, ou NMS. Esta última, por sua vez, além de receber e decodificar estes alertas, tem por função coletar informações sobre todos os agentes configurados na rede.

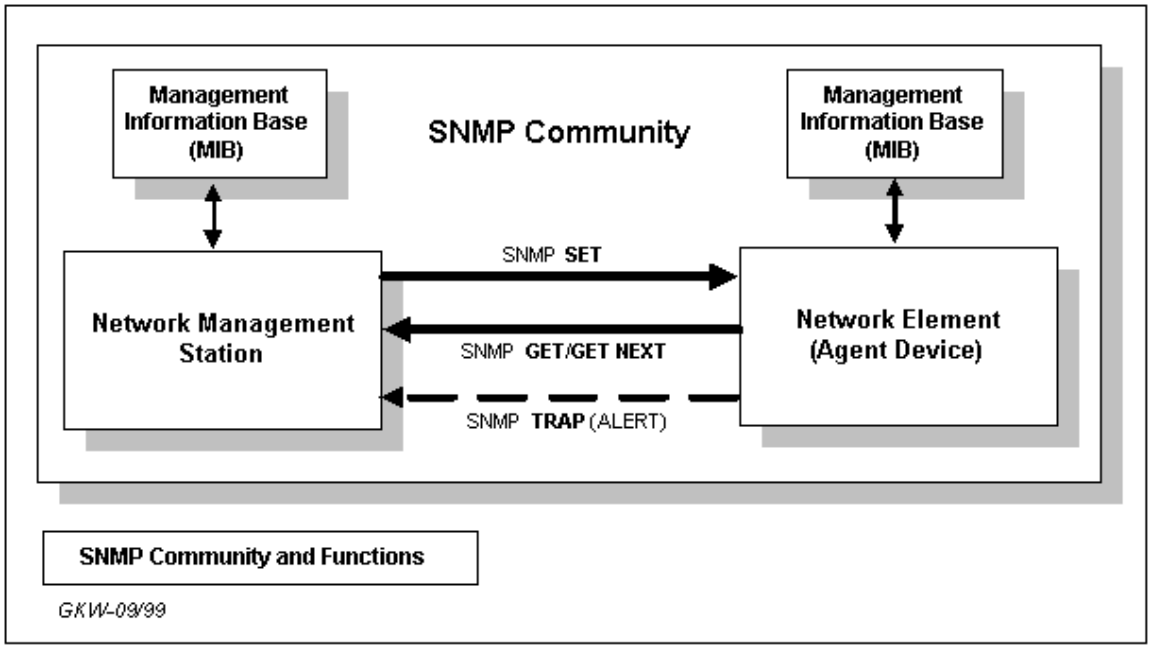

Figura 3 - Arquitetura do protocolo SNMP

Na verdade, qualquer dispositivo que esteja conectado à rede pode ser visto como um agente SNMP. Apesar do antecessor do protocolo SNMP, o SMGP – Simple Gateway Management Protocol – ter sido desenvolvido para gerenciar roteadores, o SNMP pode ser utilizado para monitorar sistemas Linux, sistemas Windows, impressoras, etc. Qualquer dispositivo que permita a recuperação de informações via SNMP pode ser gerenciado (SCHMIDT, KEVIN, 2001). Em função da grande utilização deste protocolo e da vasta variedade de dispositivos, os

próprios fabricantes tornaram-se responsáveis por desenvolver agentes SNMP para seus produtos. Isto acabou levando para o desenvolvimento de novas versões do protoloco, a SNMPv2 e SNMPv3.

Até agora foram especificadas 3 versões para o protocolo SNMP. A primeira versão trouxe como principais vantagens a simplicidade, flexibilidade e a consequente popularização. Entretanto, toda esta simplicidade oferece risco com relação à segurança.

A versão 2, SNMPv2, fo desenvolvida com o intuito de corrigir algumas deficiências da primeira versão. Utiliza a SMI2, que permite a presença de novos tipos ASN.1. Além disso, permite a criação e exclusão de objetos, juntamente com a comunicação entre gerentes através da MIB. As mensagens e a segurança foram melhor implementadas. Nas versões denominadas SNMPv2u e SNMPv2*, existe a segurança feita por usuário (user-based), o que permite a realização de operações apenas de usuários específicos.

A versão 3 do protocolo, SNMPv3, trouxe como como vantagens a introdução de mecaninsmos de criptografia como o DES e de algoritmos de autenticação que podem ser tanto o MD5 quanto o SHA. Toda essa preocupação busca evitar a alteração das mensagens enviadas. Além disso, barra-se o acesso à elementos estranhos à execução de operações de controle, que são realizadas através da operação SET. Evita-se também a leitura das mensagens por parte de estranhos, além de garantir ao gerente o direito de alteração da senha dos agentes.

Como foi comentado anteriormente, o agente irá monitorar o hardware e enviar alertas em caso de problemas. Por exemplo, se uma impressora que esteja conectada à rede estiver com a quantidade de toner abaixo da estipulada como crítica ou se um disco de um servidor apresentar falta de espaço, um alerta, ou trap, será enviado pelo agente para a NMS. O administrador, então, poderá configurar a estação para notificá-lo por e-mail ou até por sms.

O protocolo SNMP oferece uma pequena gama de comandos que são utilizados no gerenciamento dos dispositivos. Estes comando são descritos a seguir.

GET - A operação GET nada mais é que uma requisição enviada pela NMS ao dispositivo gerenciável. Este dispositivo, ou agente, responde à requisição enviando um ou mais valores.

GET NEXT - Esta operação é similar à operação GET. A grande diferença é que a operação GET NEXT recupera o valor do próximo OID da árvore da MIB.

GET BULK - A operação GET BULK é usada para recuperar um grande volume de dados de uma grande tabela da MIB.

SET - Esta operação é utilizada pelos administradores para modificar ou atribuir um valor para o dispositivo IP.

TRAPS - Diferentemente dos comandos descritos acima, que são iniciados pela NMS, as Traps são iniciadas pelos agentes. Uma trap é um sinal enviado para a NMS de um evento ocorrido no agente.

INFORM - Este comando é similar à uma trap. A diferença é o recebimento da confirmação da mensagem por parte da NMS.

RESPONSE - Comando utilizado para “carregar” o valor ou sinal de ações direcionadas à NMS.

Mas o que se fazia antes do SNMP? Vamos imaginar um cenário onde tenhamos uma rede com 100 máquinas rodando diferentes sistemas operacionais. Destas máquinas, várias são servidores de arquivos, algumas outras impressoras, uma rodando uma aplicação web responsável por verificar transações de cartão de crédito, e o resto das máquinas são estações de trabalho.

Além disso tudo, existem vários switches e roteadores que ajudam a manter a rede em pleno funcionamento. Há

um link conectando a organização à internet e um outro conectando-a ao sistema de verificação de cartão de crédito. O que acontece se um dos servidores de arquivos falhar? Se acontecer durante a semana, é provável que os usuários irão perceber e pedir uma solução ao adminstrador de redes. Mas e se acontecer após o expediente, quando todos já tiverem ido embora, inclusive o adminstrador, ou durante o final de semana? E se um dos links cair na madrugada de uma sexta-feira e por algum motivo a conexão só seja restaurada na segunda-feira pela manhã? Se o motivo do problema tiver sido o hardware e pudesse ter sido solucionado apenas trocando um dos roteadores, por exemplo, milhares de dólares em vendas pelo sistema web poderiam ter sido perdidos. Da mesma maneira, se o link que conecta a organização à internet tivesse saido, muito dinheiro poderia ter sido perdido.

Obviamente estes são problemas sérios por serem capazes de afetar o crescimento de uma organização. É exatamente ai que entra o SNMP. Ao invés de se esperar até que alguém perceba que há algo de errado e localize a pessoa responsável por resolver estes tipos de problemas (o que pode acontecer somente segunda de manhã, caso o problema ocorra no final de semana), este protocolo permite que se monitore os dispositivos IP da rede constantemente, mesmo quando não se está presente. É possível, por exemplo, verificar se a quantidade de pacotes problemáticos que entram por um dos roteadores está aumentando gradativamente, sugerindo que este roteador possa vir a apresentar problemas. Pode-se configurar o agente ou alguma aplicação que capture e trate estes dados para que o administrador da rede ou qualquer outra pessoa seja notificada de maneira automática quando a falha seja iminente, para que se possa resolver o problema antes deste acontecer, evitando, assim, maiores complicações. Outra possibilidade é uma configuração para ser notificado caso o processamento de uma transação de cartão de crédito demore para ser finalizada. Pode-se, inclusive, resolver tais problemas remotamente.

Caso nada disso aconteça, ou seja, nada dê errado após o expediente de sexta-feira ou durante o final de semana, retorna-se ao trabalho na segunda-feira sem

surpresas. Talvez não haja tanta glória em solucionar problemas antes que eles aconteçam, mas isto vai garantir que o ambiente continue funcionando como deveria. Além disso, o SNMP permite que se mantenha logs dos eventos ocorridos para que seja possível provar que o ambiente está funcionando de maneira consistente e mostrar quando ações foram tomadas para solucionar problemas, tanto que ocorreram como os que foram evitados. (MAURO, DOUGLAS, 2001).

2.3 Monitoramento de servidores Windows

No Windows é possível monitorar vários recursos relacionados à performance. Para isso, utiliza-se o NSClient, que funciona de maneira similiar à um agente SNMP.

O NSClient foi concebido para ser utilizado pelo Nagios, entretanto, é possível fazer uso por meio de qualquer outra aplicação capaz de realizar requisições remotas. A imagem abaixo mostra os vários objetos possíveis de serem monitorados.

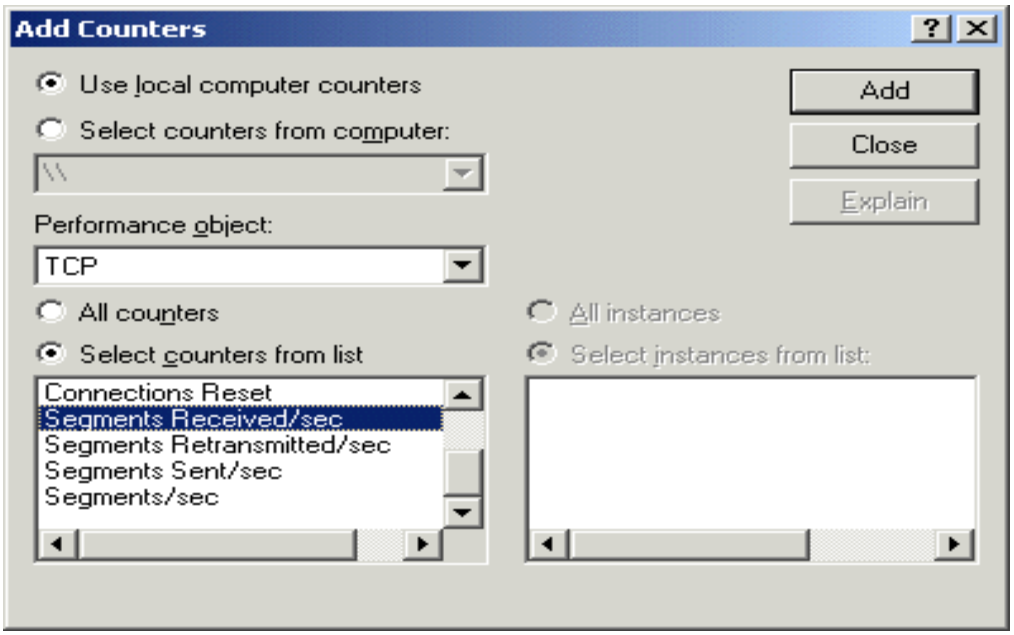

Figura 4 – Performance do Windows

Além disso, o Windows oferece uma poderosa tecnologia que pode ser utilizada para se monitorar recursos da máquina. Trata-se do WMI.

O Windows Management System, ou simplesmente WMI, é uma das várias tecnologias introduzidas pela Microsoft para suportar o gerenciamento de sistemas corporativos. Ela é fruto de um esforço entre companhias de computação interessadas em desenvolver uma camada de software padronizada para gerenciamento corporativo de sistemas e de dispositivos.

Esta iniciativa foi denominada WBEM, sigla para Web Based Enterprise Management (Gerenciamento Corporativo Baseado na Web). O objetivo era desenvolver um único conjunto de padrões para gerenciar qualquer componente de uma rede corporativa. Dessa forma, futuramente as companhias desenvolveriam hardware, software e sistemas que pudessem ser gerenciados da mesma maneira, através da adoção do padrão WBEM. A responsabilidade pela iniciativa WBEM foi assumida pela organização DTMF (Desktop Management Task Force), que é responsável em manter os padrões que irão ajudar a atingir os objetivos da iniciativa WBEM.

A iniciativa WBEM possui ainda algumas premissas que podem ser definidas assim:

- Definir um modelo para representação e classificação de informações enterprise gerenciáveis
- Prover uma maneira de compartilhar estas informações através de redes heterogêneas
- Permanecer independente no que diz respeito à sistemas operacionais e arquiteturas de rede

A quantidade de informações disponíveis através da interface WMI é imensa – configurações de hardware, informações de performance, configuração de drivers, informações sobre a BIOS, configurações de aplicações, informações de log de eventos, e muito mais. A interface WMI recupera essas informações utilizando vários conjuntos de APIs, mas apresenta essas informações seguindo um modelo de gerenciamento de objetos simples e padronizado. Isto torna desnecessário aos desenvolvedores de aplicação aprender os detalhes de cada API fornecida pelo Windows.

Visto isso, o WMI nada mais é que a infraestrutura por trás do gerenciamento de informações e operações em sistemas operacionais Microsoft Windows. Trata-se de uma tecnologia de instrumentação à nível de kernel da Microsoft que implementa as premissas do WBEM. Uma infraestrutura para o completo gerenciamento de sistemas operacionais Windows. Além disso, provê uma API – Application Programming Interface – para desenvolvedores acessarem e enviarem dados para seu local de armazenamento de informações. Permite a criação de scripts que podem executar complexas tarefas de gerenciamento através desta API. Além disso, permite aos desenvolvedores usar um mecanismo simples e consistente para pesquisar informações ou parâmetros de configuração em computadores dentro do ambiente da organização.

A Microsoft é comprometida com a simplificação da instrumentação de hardware e software para seus sistemas operacionais.

2.4 Monitoramento de Bandos de Dados

A performance em bancos de dados pode ser considerada como uma potencial área de risco e portanto exige controle por parte dos administradores.

Riscos específicos relacionados à performance envolvem disponibilidade e a própria continuidade do negócio. Uma perfomance não tão boa reflete na reputação da organização.

Neste universo de riscos e controle, ações preventivas são as mais aceitas para se detectar eventuais falhas. É sempre melhor tentar antecipar-se antes que estes problemas transformem-se em condições realmente adversas.

A degradação da performance geralmente pode ser identificada antes de ocorrer de fato. Isto é obtido através de um monitoramento efetivo. Algumas medidas de correção podem ser tomadas, como por exemplo:

- Utilização de índices
- Atualização do modelo de dados e normalização
- Particionamento dos dados físicos
- Melhor utilização do buffer
- Aumento do poder de processamento do servidor

Entretanto, deve-se, antes de mais nada, definir os critérios para ser possível saber o que é uma boa performance. Estes critérios, no que diz respeito à performance em bancos de dados, consistem em:

- Tempo de resposta, que é medido pelo tempo de resposta aceitável para usuários finais
- Vazão das consultas

- Disponibilidade que é medida pela duração do tempo de eventuais falhas

Estes são alguns dos critérios que podem ser utilizados para se definir os recursos da máquina onde estará o banco de dados. Estes recursos são: velocidade do processador, total de espaço em disco necessário, poder de I/O do disco, etc.

Quer dizer, o constante monitoramento da performance de um banco de dados deve ser levado muito a sério, considerando-se todos os riscos associoados.

2.5 Monitoramento da Java Virtual Machine

Em um ambiente corporativo, onde são criadas dezenas de aplicações para provimento de serviços diversos, é de suma importância que a organização tenha o cuidado em prover uma infraestrutura que mantenha estas aplicações disponíveis com a qualidade esperada pelo cliente do serviço.

O grande desafio neste cenário é manter o controle operacional de todas estas aplicações, coletando informações referentes à disponibilidade do serviço, desempenho, consumo de recursos de hardware e de rede, defeitos no software, dentre outros. Para que este monitoramento se torne uma realidade dentro do ambiente, é indispensável o uso de ferramentas que possam disponibilizar estas informações em tempo hábil para que sejam realizadas as intervenções necessárias. Manter um histórico destas informações traz um diferencial importante, pois com o seu uso, medidas pró-ativas podem

ser realizadas na aplicação e/ou ambiente evitando a indisponibilidade do serviço oferecido; conseqüentemente evitando que apenas intervenções corretivas sejam realizadas.

Para criar o monitoramento de um ambiente com este cenário, muitas vezes é comum integrar estas aplicações com outros produtos para prover recursos de monitoramento. Identificar e alarmar os pontos críticos de falha e repassar estas informações à equipe responsável pela operação da aplicação e/ou ambiente é um dos principais objetivos para manter o serviço operante. Uma solução muito comum para prover monitoramento é a utilização de ferramentas que implementam o protocolo SNMP (Simple Network Management Protocol), que já é adotado no mercado há muitos anos para este fim. Desenvolver aplicações Java com recursos de monitoramento baseados em SNMP nem sempre é uma tarefa fácil, visto que a grande maioria dos desenvolvedores não é especialista no protocolo e ainda têm que dividir seu tempo de desenvolvimento com o aprendizado destes recursos, juntamente com o processo de integração com o ambiente de monitoramento.

Alternativamente à tecnologia SNMP, no que se refere ao monitoramento e gerenciamento de aplicações Java, será apresentada a utilização dos recursos da API JMX (Java Management Extensions) que oferecem inúmeras vantagens ao desenvolvedor Java na criação de um ambiente de monitoramento.

Serão apresentados os recursos disponíveis pela tecnologia JMX baseados nas especificações da JSR (Java Specification Request) 3 e na JSR 160, que trazem novos recursos para o gerenciamento remoto, com uma breve comparação com os recursos oferecidos pelo protocolo SNMP.

O Java Management Extension – JMX – é um framework que disponibiliza um conjunto de especificações para o desenvolvimento de ferramentas utilizando o Java que possibilita efetuar o monitoramento e gerenciamento remoto de aplicações, sistema operacional, recursos de hardware, etc. As especificações trazidas pela tecnologia

JMX, inicialmente através da JSR 3, definem a sua arquitetura e padrões de projeto que permitem integrar estes recursos a um projeto de software Java, oferecendo uma forma de monitoramento e gerenciamento não só da aplicação, mas de todo o ambiente, proporcionando a execução da aplicação utilizando apenas Java.
Este processo também pode ser realizado remotamente.

A tecnologia JMX permite que sejam encapsuladas no software informações coletadas do ambiente, diferentemente da tecnologia SNMP. Através do quadro "SNMP – Breve descrição" podemos identificar as características deste protocolo e fazer comparativos com a tecnologia JMX.

3 Ferramentas de Monitoramento

Atualmente são muitas as ferramentas de monitoramento de redes, servidores, ou seja, recursos computacionais. Três destas, todas opensource, são brevemente descritas abaixo.

3.1 Nagios

O Nagios é uma aplicação de monitoramento de redes licenciado pelo sistema GPL. Tem como objetivo monitorar equipamentos como: hosts, switches, roteadores, aplicações, carga do processador, espaço livre em disco,

etc. É capaz de realizar notificações quando algum serviço e/ou equipamentos estiverem indisponíveis. Estas notificações podem ser enviadas por e-mail, sms ou qualquer outro serviço definido pelo usuário através de plugin.

Esta capacidade de extensibilidade por meio de plugins é sua principal característica, além do fato de possuir vaŕios mantenedores em todo o mundo, que continuam corrigindo bugs e desenvolvendo novos plugins à medida de necessidade.

Apesar de ser altamente flexível e configurável, o Nagios apresenta alguns pontos negativos. Por ter sido construído em uma arquitetura servidor/agente, torna-se bastante trabalhoso configurar os agentes em redes com muitas máquinas, pois é necessário configurar um agente para cada máquina da rede. A escassez de relatórios, a complexidade para configurar e adicionar um novo dispositivo na rede - muitos arquivos de configuração - também são exemplos de pontos negativos. Não é uma ferramenta simples de manipular para alguém sem experiência.

Service	Status	Last Check	Duration	Attempt	Status Information
CPU load	OK	2009-09-20 17:46:54	63d 4h 16m 24s	1/1	OK - 0.10
CPU utilization	OK	2009-09-20 17:46:54	63d 4h 16m 24s	1/1	OK - user: 1%, system: 2%, wait: 0%
Check_MK	OK	2009-09-20 17:46:54	63d 4h 16m 24s	1/1	OK - Agent Version 1.0.35rc, processed 12 host infos
Disk IO read	OK	2009-09-20 17:46:54	63d 4h 16m 24s	1/1	OK - 0.0MB/s (in last 60 secs)
Disk IO write	OK	2009-09-20 17:46:54	63d 4h 16m 24s	1/1	OK - 0.0MB/s (in last 60 secs)
Memory used	OK	2009-09-20 17:46:54	63d 4h 16m 24s	1/1	OK - 6.1% of RAM (74 MB) used by processes
NIC eth0 counters	OK	2009-09-20 17:46:54	63d 4h 16m 24s	1/1	OK - Receive: 0.00 MB/sec - Send: 0.00 MB/sec
NIC eth0 link	OK	2009-09-20 17:46:54	63d 4h 16m 24s	1/1	OK - Link is up
NIC eth0 parameter	OK	2009-09-20 17:46:54	63d 4h 16m 24s	1/1	OK - 1000Mb/s,Full,on
NIC pan0 counters	OK	2009-09-20 17:46:54	63d 4h 16m 24s	1/1	OK - Receive: 0.00 MB/sec - Send: 0.00 MB/sec
Number of threads	OK	2009-09-20 17:46:54	63d 4h 16m 24s	1/1	OK - 202 threads
fs_/	OK	2009-09-20 17:46:54	63d 4h 16m 24s	1/1	OK - 52.3% used (2.5 of 4.7 GB), (levels at 80.0/90.0%)

figura 5 - Dashboard do Nagios

3.2 OpenNMS

O OpenNMS é um projeto opensource focado na criação de uma plataforma de gerência de rede voltada principalmente para camada de aplicação. Este software de gerenciamento de redes é capaz de fornecer uma série de métricas que diretores, gerentes e administradores de rede podem utilizar para medir a qualidade, tempo de disponibilidade de serviços. Sua principal característica é o auto descobrimento e justamente por isso é extremamente fácil de ser configurado. Com apenas alguns cliques é possível iniciar o monitoramento de uma rede.

Esta ferramenta tem como outras características a possibilidade de se configurar a geração de alertas e a geração de relatórios sobre todos os dados analisados.

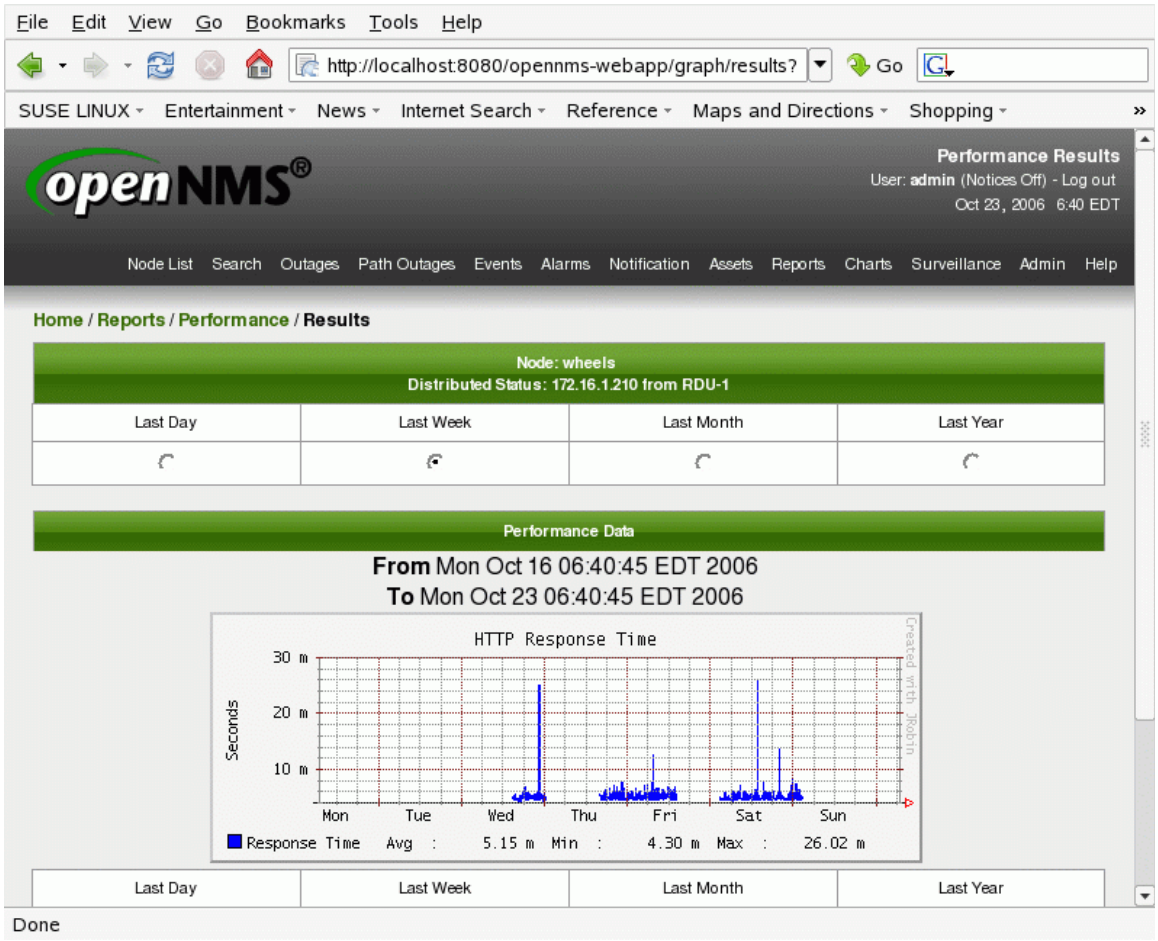

Figura 6 - Tempo de resposta HTTP

3.3 Cactisonar

É uma ferramenta front-end ao RRDtool. Com ela é possivel reproduzir em gráficos, informações referente ao estado dos elementos monitorados na rede através do SNMP. Estas informações são armazenadas em um banco de dados MySQL, e podem ser consultadas via WEB.

O Cacti é uma ferramenta com licença GPL e permite o monitoramento de informações como: tráfego na rede, uso de memória, espaço em disco, além de dispositivos como switches e roteadores.

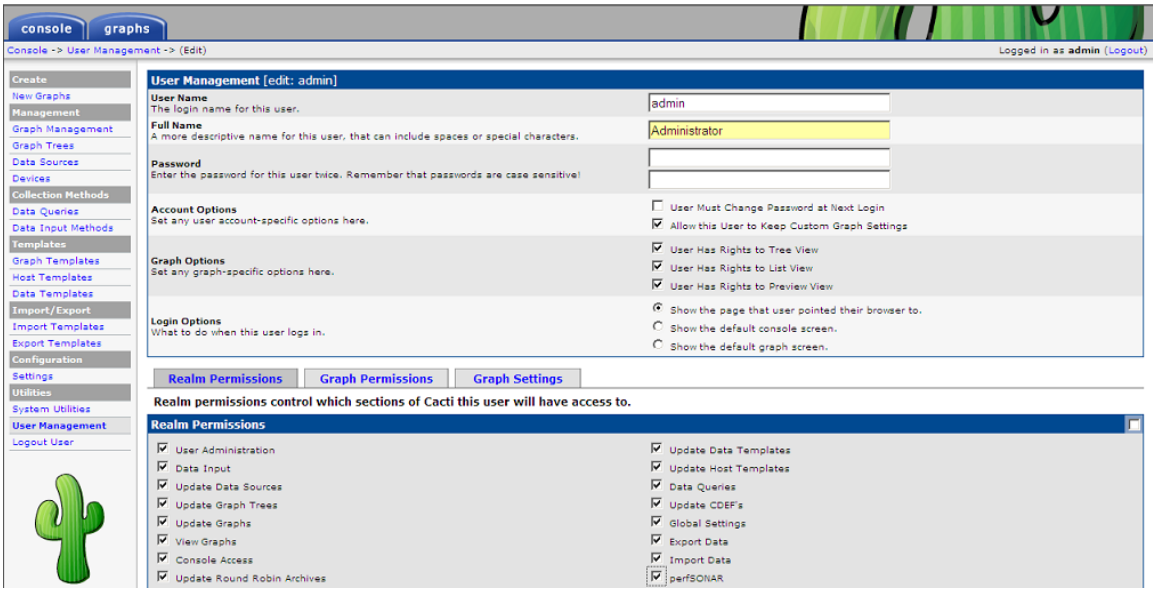

Figura 7 - Gerenciamento de usuários no Cactisonar

3.4 Trabalhos Relacionados

No artigo “OpenNMS: A powerhouse of an open source network monitoring tool”, aborda-se a ferramenta OpenNMS, comentando como os desenvolvedores da ferramenta continuam envolvidos no projeto, oferecendo suporte à empresas, apesar desta ferramenta ser opensource. O suporte é oferecido através do pacote OpenNMS Green Light, onde os criadores da ferramenta prestam treinamento às empresas, entre outras atividades.

Já no artigo “Building a Monitoring Infrastructure With Nagios”, David Josephsen enaltece e explica porque ele considera o Nagios a melhor ferramenta de monitoramento existente. Ele comenta que a maioria das ferramentas proprietárias oferecem soluções padrão, ou seja, não servirão para todas as necessidades. Já o Nagios, por ser composto por plugins, escritos por desenvolvedores e usuários do mundo todo, acaba com este problema. A cada necessidade encontrada, plugins são desenvolvidos e compartilhados com a comunidade.

No artigo “Gerência Corporativa de Sistemas Baseada na Web utilizando Agentes Móveis Inteligentes”, o protocolo SNMP é colocado em pauta. Explica-se o seu funcionamento, sua arquitetura - agente e NMS - descrevendo cada um destes elementos, dentro do contexto do crescimento de aplicações e serviços oferecidos pela internet e da necessidade de se monitorar tais serviços, afim de garantir a estabilidade destes.

4 Desenvolvimento da Aplicação

Neste trabalho foi desenvolvida uma aplicação capaz de monitorar recursos de servidores Linux e Windows, fornecer informações e estatísticas de bancos de dados como MySQL e Oracle, fornecer informações sobre a máquina virtual Java e aplicações que estejam rodando sobre a JVM. Na verdade, trata-se de uma ferramenta totalmente baseada em scripts e portanto capaz de monitorar praticamente qualquer informação que scripts em várias linguagens de programação forem capazes de coletar. Esta aplicação possui uma interface onde o usuário pode visualizar todos os monitoramentos configurados para os servidores do seu ambiente. A idéia foi centralizar, em um único local, todas as informações sobre todos os domínios configurados.

A linguagem de programação escolhida para desenvolvimento no lado servidor foi o PHP, enquanto a interface Web, que será acessada pelo usuário, foi desenvolvida utilizando HTML para a estrutura e apresentação das páginas, Javascript para requisições AJAX e CSS para o estilo destas páginas. A biblioteca jQuery foi escolhida para o desenvolvimento no lado do cliente.

A estrutura da aplicação foi dividida em duas partes: Uma contendo os serviços, bibliotecas, arquivos de configuração, templates e scripts pertencentes à lógica da aplicação utilizados na parte do servidor, e outra com os arquivos relacionados à visualização e interface pública, como imagens, arquivos javascript e páginas HTML.

Com o intuito de facilitar o desenvolvimento de novas funcionalidades, a aplicação foi desenvolvida de maneira modular, eliminando a necessidade de modificação de layouts, controle de sessões e conexão ao banco de dados. Neste caso, cada novo módulo deverá utilizar a classe TemplateHandler – classe responsável pela manipulação dos templates para apresentação das páginas e dos arquivos CSS e Javascript necessários em cada módulo -

que trata a inclusão dos arquivos dependentes à este módulo.

A arquitetura de funcionamento da aplicação consiste basicamente em três elementos principais:

- O usuário, através de um navegador web, realiza requisições http ao servidor. Estas requisições podem ser para cadastro ou remoção de scripts, servidores, monitoramentos ou análises. A aplicação então processa estas requisições ao executar operações com o banco de dados (MySQL) e retornando informações ao navegador.

- O serviço responsável executa todos os monitoramentos ativos, persistindo as informações recuperadas no banco de dados, tanto de sucesso como de falha.

- Estas informações, coletadas pelos monitoramentos, são, então, apresentadas ao usuário na interface web.

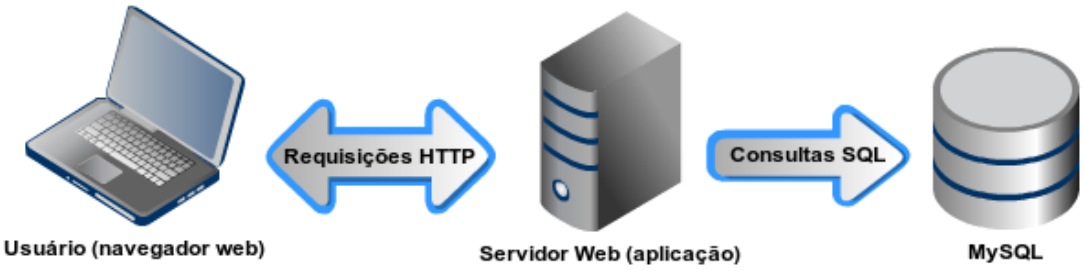

Figura 8 – Interação do usuário com a aplicação

A figura 8 mostra o fluxo da interação do usuário com a aplicação. Através de um navegador web, o usuário pode cadastrar e editar scripts, servidores, monitoramentos e análises, além de visualizar os dados recuperados pelos monitoramentos no dashboard.

4.1 Estrutura de desenvolvimento

Cada um dos módulos presentes na aplicação, tanto na área administrativa como na área pública, serão explicados.

Como já foi dito anteriormente, há uma divisão com relação a estrutura. Na pasta application estão os arquivos de configuração de acesso ao banco de dados, serviços, templates e todas as bibliotecas e arquivos extras utilizados na aplicação, como a biblioteca de logging *log4php* - utilizou-se o framework Apache Log4PHP, cujas principais características são a possibilidade de logar as informações tanto para um ou mais arquivos, como para console e a fácil configuração através de arquivos .xml - a biblioteca *nusoap*, biblioteca utilizada para desenvolver e consumir webservices, além do próprio TemplateHandler.

Já na public, estão todos os elementos referentes a parte pública da aplicação: arquivos HTML, arquivos CSS e arquivos Javascript, todos estes utilizados na apresentação das informações para o usuário.

Dentro de cada um dos módulo da área pública, temos três arquivos que são essenciais e comuns a todos estes módulos: index.php, view.php, além de um outro arquivo de mesmo nome do módulo.

O arquivo index.php funciona como o controle deste módulo. O controle é responsável por toda requisição feita pelo usuário através da interface web. Além disso, tem

como outras responsabilidades o carregamento dos arquivos CSS e Javascript necessários na interface deste módulo, a definição do arquivo que tem como função apresentar as informações ao usuário, neste caso o arquivo *view.php*. O arquivo responsável pela apresentação das informações contém o código HTML da página. Já o arquivo de mesmo nome do módulo contém o código Javascript responsável por montar a tabela que mostra os registros já cadastrados para cada um destes módulos.

A figura 9 mostra a parte do código do controle do módulo script responsável pelo cadastro de um novo script. Percebemos que o serviço é recuperado através de sua única instância e então o método *save* é chamado.

```php
<?php
session_start ();
require_once (dirname ( __FILE__ ) . '/../../application/config/general.php');
require_once (dirname ( __FILE__ ) . '/../../application/utils/TemplateHandler.php');
require_once (dirname ( __FILE__ ) . '/../../application/service/ScriptServiceImpl.php');

$action = isset ( $_GET ['action'] ) ? $_GET ['action'] : '';
$usuario_id = isset ( $_SESSION ['id'] ) ? $_SESSION ['id'] : '';

switch ($action) {

	case 'add' :

		$dominioID = isset ( $_POST ['dominio'] ) ? $_POST ['dominio'] : '';
		$descricao = isset ( $_POST ['descricao'] ) ? $_POST ['descricao'] : '';
		$binID = isset ( $_POST ['bin'] ) ? $_POST ['bin'] : '';
		$arquivo = isset ( $_POST ['arquivo'] ) ? $_POST ['arquivo'] : '';

		// Getting the service
		$service = ScriptServiceImpl::getInstance ();
		// Calling the service to save a new script
		exit ( $service->save ( $descricao, $binID, $arquivo, $dominioID ) );
		break;
```

Figura 9 – Controle do módulo script

Na parte do cliente, a biblioteca DataTables é utilizada em todos os módulos para a apresentação dos scripts, servidores, monitoramentos e análises já cadastradas. Trata-se de um plugin para JQuery, biblioteca javascript utilizada no projeto, bastante flexível para a criação de tabelas. Este plugin tem como principais

características a paginação automática, busca e filtro de dados na tabela em tempo real e internacionalização.

Tendo a plataforma já estabelecida, a próxima etapa foi o planejamento dos módulos a serem agregados à aplicação.

4.2 Configuração

Praticamente toda a configuração da aplicação reside na pasta config, distribuída em dois arquivos: general.php e constantes.php.

O arquivo general.php possui todas as informações referentes à conexão com o banco de dados, conexão SSH, configurações de logging e de tratamento de exceção.

Trata-se do principal arquivo de configuração. No caso de uma nova instalação, este arquivo deve ser alterado para contemplar as informações do novo ambiente no qual esta ferramenta será instalada.

```php
<?php
require_once(dirname(__FILE__) . '/constantes.php');

global $config;

/* MySQL Settings */
$config['mysql_host'] = 'localhost';                    # MySQL location
$config['mysql_user'] = 'root';                         # MySQL user
$config['mysql_password'] = '???';                      # MySQL password
$config['mysql_database'] = 'tcc';                      # MySQL database

/* SSH Settings */
$config['ssh_host'] = 'localhost';                      # SSH location
$config['ssh_user'] = 'andre';                          # SSH user
$config['ssh_password'] = '???';                        # SSH password
$config['ssh_port'] = '22';                             # SSH port

/* Logs Settings */
$config['log_verbosity'] = LOG_VERBOSITY_DEBUG;         # Log verbosity level
$config['log_enabled'] = true;                          # Log errors into DB table / E-mail
$config['log4php_log_enabled'] = true;                  # Log errors with Log4php
$config['firephp_log_enabled'] = true;                  # Log errors with FirePHP
$config['log_email'] = 'afdurieux@gmail.com';           # E-mail address to send the error logs (when log DB insert fails)

/* Error/Exception Handling */
set_exception_handler('handle_exception');              # Define the function to handle exceptions
set_error_handler('handle_error');                      # Define the function to handle errors
error_reporting(E_ALL);                                 # Sets which PHP errors are reported
error_reporting(0);                                     # Sets which PHP errors are reported
ini_set('display_errors', '1');                         # Turn error displaying on or off

date_default_timezone_set('America/Sao_Paulo');
?>
```

Figura 10 – O arquivo de configuração

A figura 10 mostra o principal arquivo de configuração da aplicação. Podemos ver a variável global *config*, que contém dados de configuração de acesso ao MySQL e da conexão SSH.

Já no arquivo constantes.php, ficam as definições de todas as constantes utilizadas na aplicação. Este arquivo não precisa ser alterado no caso de uma nova instalação, exceto no caso de o usuário da aplicação estiver desenvolvendo um novo módulo.

A figura 11 mostra como são definidas todas as constantes utilizadas na aplicação.

```php
// Defaults
define ( 'DEFAULT_TITLE', 'TCC' );
define ( 'DEFAULT_HEADER_FILE', 'header.php' );
define ( 'DEFAULT_BODY_FILE', 'body.php' );
define ( 'DEFAULT_FOOTER_FILE', 'footer.php' );
define ( 'DEFAULT_GLOBAL_FILE', 'global.php' );
define ( 'DEFAULT_FORBIDDEN_FILE', 'forbidden.php' );

// Paths
define ( 'APPLICATION_PATH', realpath ( dirname ( __FILE__ ) . '/../' ) );
define ( 'PUBLIC_PATH', realpath ( dirname ( __FILE__ ) . '/../../public/' ) );
define ( 'IMAGE_PATH', PUBLIC_PATH . '_img/' );
define ( 'SCRIPTS_PATH', PUBLIC_PATH . '/monitoramento/scripts/' );

define ( 'LINUX_SCRIPTS_PATH', SCRIPTS_PATH . 'linux/' );
define ( 'WINDOWS_SCRIPTS_PATH', SCRIPTS_PATH . 'windows/' );
define ( 'JAVA_SCRIPTS_PATH', SCRIPTS_PATH . 'java/' );
define ( 'MYSQL_SCRIPTS_PATH', SCRIPTS_PATH . 'mysql/' );
define ( 'ORACLE_SCRIPTS_PATH', SCRIPTS_PATH . 'oracle/' );
define ( 'POSTGRES_SCRIPTS_PATH', SCRIPTS_PATH . 'postgres/' );

// Global URIs:
$ROOT_URI = explode ( '/public', $_SERVER ['REQUEST_URI'] );
$ROOT_URI = $ROOT_URI [0] . '/public/';
define ( 'ROOT_URI', 'http://' . htmlentities ( $_SERVER ['SERVER_NAME'] . $ROOT_URI ) );

define ( 'CSS_URI', ROOT_URI . '_css/' );
define ( 'JS_URI', ROOT_URI . '_js/' );
define ( 'IMG_URI', ROOT_URI . '_img/' );

// URI's
define ( 'DASHBOARD_URI', ROOT_URI . 'dashboard/' );
```

Figura 11 - O arquivo das constantes

Tendo visto os arquivos de configuração, partimos agora para o arquivo responsável pela execução dos monitoramentos.

Na pasta core, também temos dois arquivos: run.php e start_msnlistener.sh. O arquivo run.php é executado pelo cron – agendador de tarefas padrão de sistemas operacionais Linux, que apenas instancia o serviço que é o real responsável pela execução de todos os monitoramentos configurados na aplicação. O arquivo é apresentando na figura 12.

```php
<?php
set_time_limit ( 60 );

require_once (dirname ( __FILE__ ) . '/../service/ExecutorServiceImpl.php');

$executorService = ExecutorServiceImpl::getInstance ();
$executorService->execute ();

?>
```

Figura 12 - O arquivo run.php

Na primeira linha, o comando set_time_limit garante o tempo limite para a execução deste arquivo. No caso, o tempo limite é de 60 segundos. Isto assegura que caso alguns dos monitoramentos não ocorram devido a eventuais problemas com os seus respectivos scripts, não haverá um bloqueio do script mostrado na figura 12, o que pode ocasionar problemas de má utilização de recursos por parte do servidor, uma vez que a cada novo minuto, o mesmo script será invocado.

Já na terceira linha, importamos o serviço responsável pela execução dos monitoramentos.

O arquivo start_msnlistener.sh é o script responsável pela execução do msnlistener, um webservice que funciona como um cliente de msn, utilizado pelo serviço de análise para a notificação do usuário em casos críticos. A figura 13 mostra o conteúdo do arquivo, onde primeiramente definimos o interpretador do script, no caso o bash, o diretório onde queremos armazenar o log do comando que é definido na última linha.

```
#!/bin/bash

log_dir=/works/projetos/php/tcc/application/logs

`/usr/bin/java -jar /works/projetos/php/tcc/application/utils/msnlistener.jar & > $log_dir/msnlistener.log`
```

Figura 13 – O arquivo start_msnlistenet.sh

4.3 Serviços

A pasta service é o principal componente da aplicação no lado servidor. Possui todos os serviços da aplicação. Estes serviços são responsáveis por quase toda a lógica da aplicação, isto porque há alguma lógica contida nos controles.

Todas as classes de serviço seguem o padrão Singleton, que restringe a instanciação da classe para apenas um objeto. Este padrão é extremamente útil quando um objeto é necessário para coordenar ações dentro do sistema.

O serviço dos scripts é responsável pelo gerenciamento relacionado aos scripts. Ações de cadastro e edição de scripts são responsabilidades desta classe. A figura 14 mostra o método chamado retrieve, que recupera um ou mais scripts, dependendo dos parâmetros.

```
public function retrieve($id = '', $fields = array(), $start = 0, $limit = 0) {
        $columns = '*';
        if ($fields) {
                // Junta os campos/colunas para a consulta SQL
                $columns = implode ( ',', $fields );
        }
        // Prepara a consulta SQL
        $query = "SELECT $columns FROM Script";
        // Caso o ID tenha sido especificado, retornará apenas os dados
        // referentes a aquele ID
        if ($id) {
                $query .= sprintf ( " WHERE id='%s' LIMIT 1", mysql_real_escape_string ( $id ) );
        }               // Verifica se os parâmetros para limite/paginação foram passados
        elseif (is_numeric ( $start ) && $start >= 0 && is_numeric ( $limit ) && $limit > 0) {
                $query .= sprintf ( " LIMIT %u,%u", mysql_real_escape_string ( $start ), mysql_real_escape_string ( $limit ) );
        }
        try {
                $result = $this->DB->query ( $query );
        } catch ( Exception $e ) {
                Logger::getLogger ( "ScriptServiceImpl" )->error ( "Error trying to retrieve a script", $e->getMessage () );
                return false;
        }
        // Retorna os resultados em forma de matriz
        return $this->DB->parse ( $result );
}
```

Figura 14 – Parte da classe do serviço de script

4.4 Módulos

A aplicação foi desenvolvida de maneira modular. Deste modo, torna-se mais fácil tanto a manutenção como o desenvolvimento de uma nova funcionalidade.

A seguir, os módulos da aplicação serão apresentados e descritos.

4.4.1 Script

Muitas vezes temos que disponibilizar serviços que não podem ficar fora do ar nem por alguns minutos durante o dia. Para alcançar esse objetivo, é necessário termos um sistema muito bem instalado e configurado. Além disso, é essencial que o administrador deste sistema tenha uma ou mais ferramentas capazes de lhe ajudar a visualizar em tempo real as informações do seu ambiente de trabalho.

Ambientes Linux são extremamente poderosos e disponibilizam várias ferramentas e comandos que podem ser utilizados para monitorar a maioria dos recursos de máquina, tanto local quanto remotamente.

A criação de scripts, que englobam estes comandos e monitoram algum recurso específico, fica sob responsabilidade do usuário da aplicação. Entretanto, uma vez desenvolvido, o usuário, eventualmente um administrador de sistemas, precisa cadastrar este script na aplicação.

Outra possibilidade é a utilização de scripts genéricos, que possam ser utilizados para monitorar mais

de um servidor. Neste caso, os parâmetros deste script, caso necessário, são configurados no monitoramento, como será descrito na seção 4.4.3.

Dentro da aplicação desenvolvida, os scripts tem um papel muito importante. Estas possibilidades tornam a ferramenta extremamente útil para administradores de sistemas. Vale lembrar que vários scripts já desenvolvidos são oferecidos pela aplicação, categorizados por domínios: Linux, Windows, a máquina virtual Java, MySQL e Oracle. Todo script cadastrado na aplicação pertence a um domínio. Entretanto, não estão cadastrados ainda, ou seja, estão pronto para serem usados. Para isso, precisam ser adicionados à aplicação, através da interface do módulo dos scripts, onde é possível cadastrar um novo script.

A figura 15 mostra a interface de cadastro de um novo script. Além disso, também mostra quatro scripts previamente cadastrados, em forma de tabela.

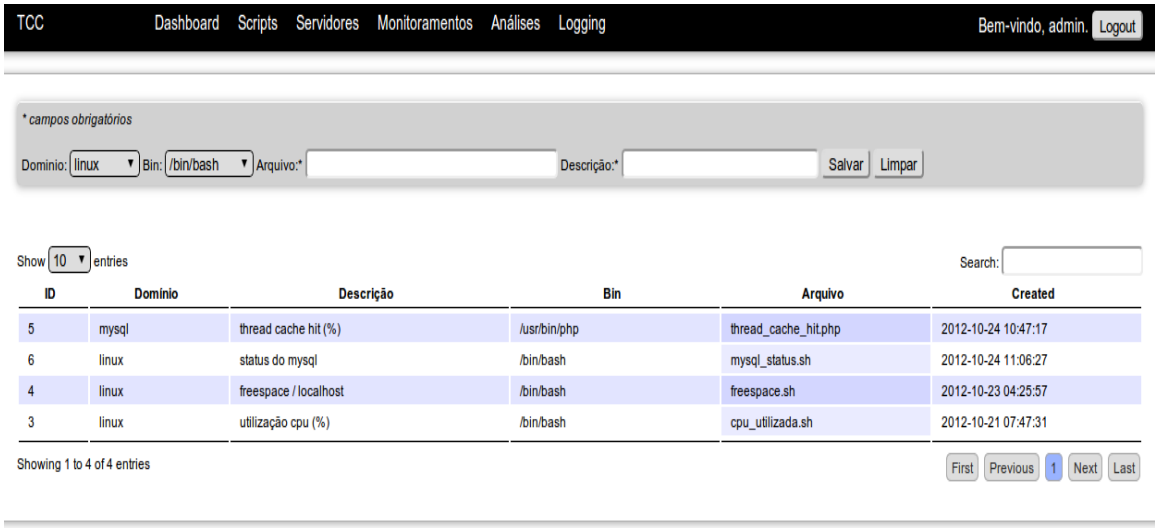

Figura 15 – Interface do módulo script

4.4.2 Servidor

No módulo dos servidores, o usuário dispõe da possibilidade de cadastro de um novo servidor sobre o qual poderá configurar novos monitoramentos. Para cadastrar um servidor com sucesso, o usuário precisa informar o nome, descrição e dizer se trata-se de um servidor Linux ou Windows.

O usuário da aplicação pode ter em seu ambiente mais de um servidor e cadastrando cada um deles, torna-se mais fácil a identificação, na aplicação, de quais servidores estão sendo monitorados.

O cadastro de um servidor seria o segundo passo do usuário. Podemos ver na figura 16, a interface de cadastro de um servidor. Neste caso, é possível ver também que já existem dois servidores cadastrados na aplicação, como podemos ver na tabela desta mesma figura.

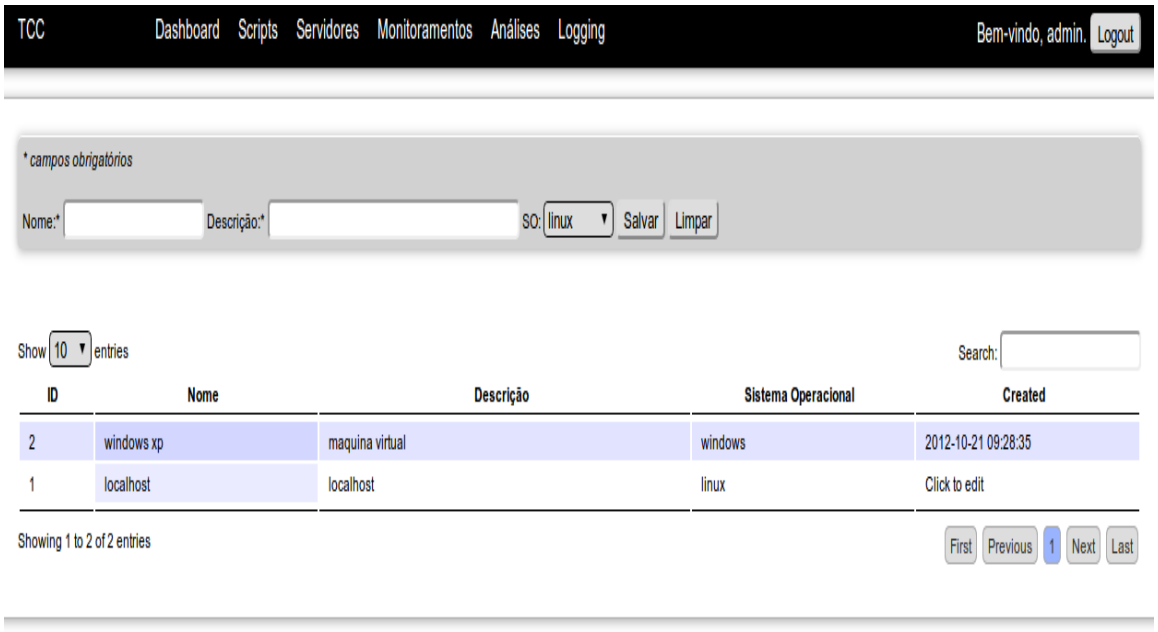

Figura 16 – Interface do módulo servidor

4.4.3 Monitoramento

Um monitoramento pode ser descrito um comando que será executado pelo serviço responsável pela execução dos monitoramentos.

Para cadastrar um monitoramento, é necessário escolher um script que já tenha sido cadastrado pelo usuário. Então, devemos adicionar parâmetros para o script selecionado, caso seja necessário. Os parâmetros são configurados para o monitoramento e não para o script porque o usuário pode desenvolver um script genérico, que pode ser usado no monitoramento de mais de um servidor. Desta maneira, não será preciso cadastrar o mesmo script mais de uma vez se quisermos monitorar um mesmo recurso de servidores diferentes. Assim, é preciso apenas passar os parâmetros para o monitoramento.

Neste cenário, se quisermos monitorar um recurso comum de duas ou mais máquinas, será preciso configurar o mesmo número de monitoramentos tanto quanto o número de máquinas que serão monitoradas. Entretanto, será possível utilizar o mesmo script, uma vez que os parâmetros são configuráveis por monitoramento. No caso de um script que não recebe parâmetros, basta configurar um novo monitoramento, sem parâmetros.

A figura 17 mostra a interface do módulo dos monitoramentos, onde é possível cadastrar novos monitoramentos, além de editar os já cadastrados, que são apresentados na tabela.

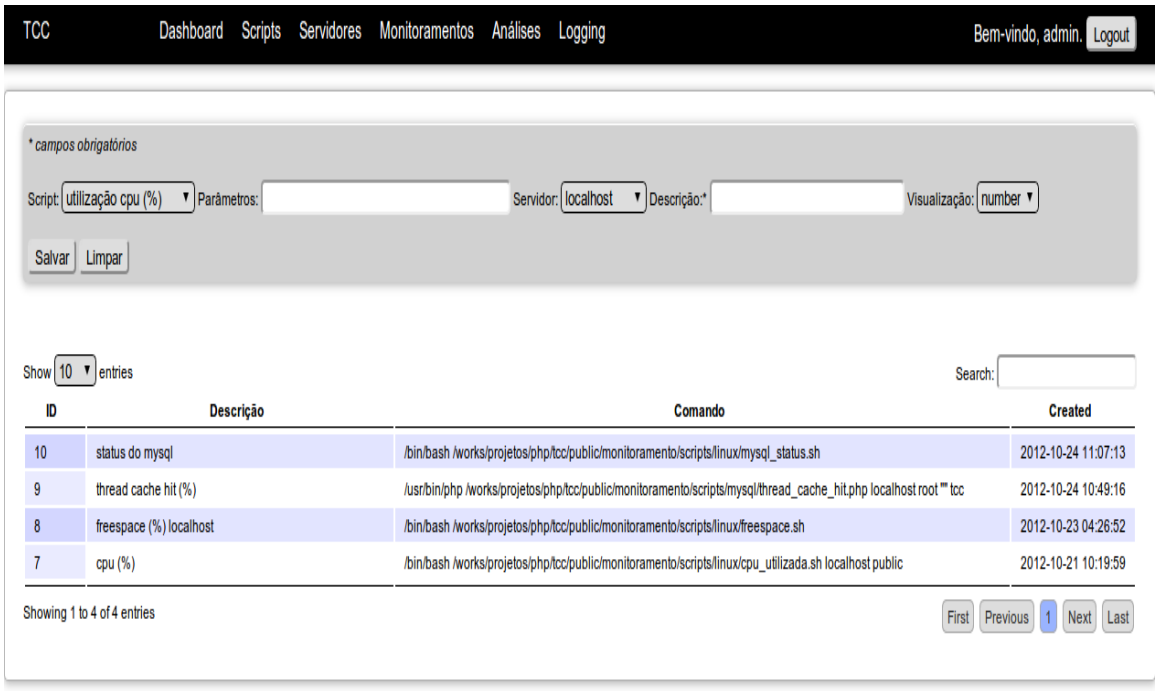

Figura 17 – Interface do módulo monitoramento

4.4.4 Análise

Uma análise seria o último passo para a uma configuração completa de um monitoramento. Vale ressaltar que a configuração de uma análise não é necessária para os monitoramentos sejam executados. Entretanto, se o usuário quiser ser notificado caso algum monitoramento recupere um dado que ele considere fora do padrão, uma análise deve ser configurada.

Para cadastrar uma nova análise, devemos fornecer uma descrição, informar o monitoramento referente à esta análise, uma regra de comparação, um valor crítico e um tipo de alerta. Uma simples leitura de uma análise seria basicamente a notificação do usuário caso o valor recuperado por um monitoramento seja maior, menor, diferente ou igual que o valor crítico configurado.

Para exemplificar, digamos que tenhamos configurado um monitoramento da utilização da CPU da máquina localhost. Imaginemos que o usuário queira ser notificado caso o dado coletado por este monitoramento seja maior que 90, no caso, 90%. Assim, configuramos o valor crítico como 90. Então, o tipo do alerta deve ser especificado. Atualmente são suportados apenas dois tipos de alerta: e-mail e MSN.

A figura 18 mostra a interface do módulo das análises, onde é possível cadastrar e editar análises. Percebemos na figura que uma análise já configurada. No caso, um e-mail será enviado toda vez que o dado coletado pelo monitoramento de utilização da CPU for maior que 10%.

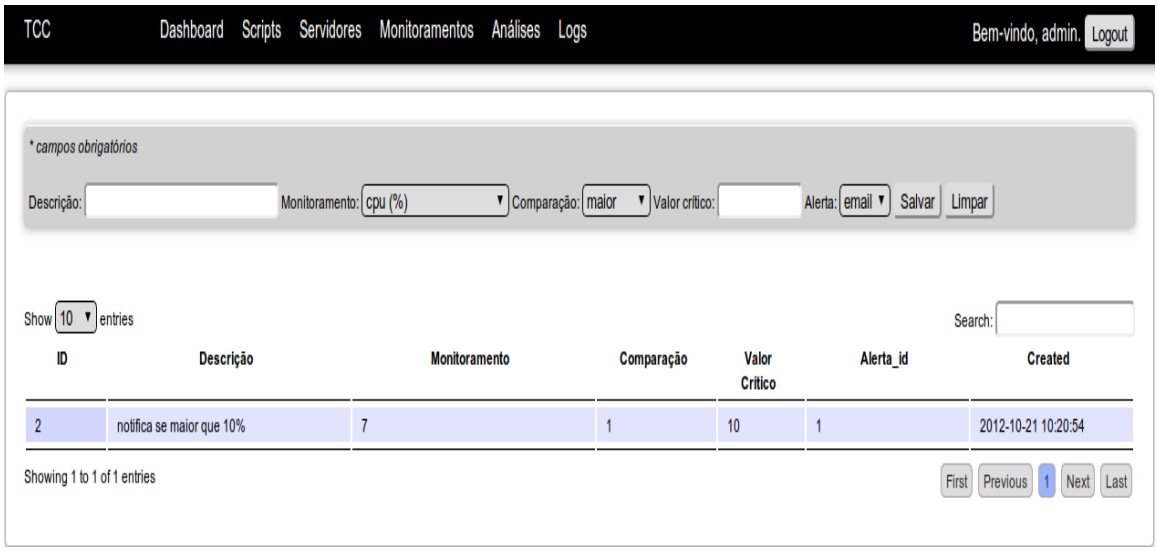

Figura 18 – Interface do módulo análise

Ainda no contexto das análises, temos também os alertas. Um alerta não foi definido como um módulo da aplicação. Trata-se da maneira pela qual a aplicação notifica o usuário sobre o status de algum monitoramento.

Alertas possuem um importante papel quando se trata de monitoramento. Quando dissemos que o sistema nos notifica quando algo crítico acontece, significa que estamos recebendo um alerta.

No cenário apresentado anteriormente, teríamos o monitoramento da utilização da CPU e uma análise para este monitoramento. No caso, o usuário seria notificado toda vez que o percentual de utilização da CPU superasse os 90%.

A notificação por e-mail é feita utilizando-se a biblioteca PHPMailer, que trata exclusivamente o envio de mensagens de e-mail.

Já a notificação via MSN é feita através de um webservice desenvolvido em Java que roda um cliente de MSN, o qual é acionado sempre que temos um evento crítico e cujo alerta configurado for o MSN. A figura abaixo mostra um pedaço do código do msnlistener.

4.4.5 Logging

O módulo de logging da aplicação utiliza uma terceira aplicação chamada *Log.io* para a visualização dos logs da aplicação. Trata-se de uma ferramenta de visualização de logs no browser em tempo real. A grande vantagem desta ferramenta é a capacidade de centralização de vários logs de diferentes máquinas. Além disso, oferece a possibilidade de busca por expressão regular nos logs, facilitando e agilizando o encontro de mensagens específicas.

A figura 19 mostra a interface do módulo logging da aplicação.

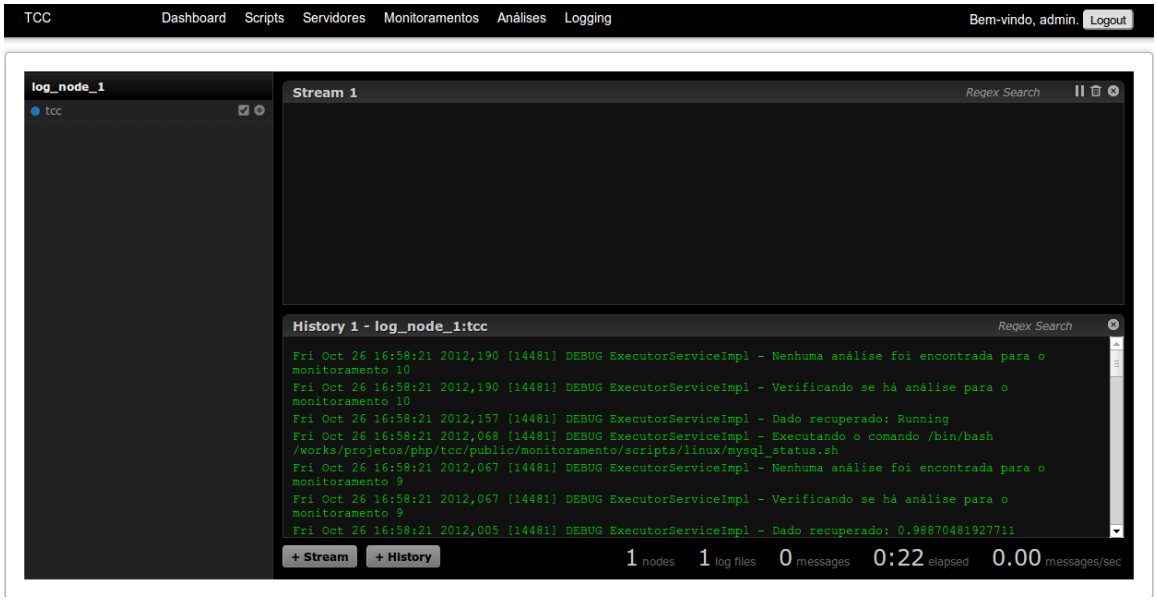

Figura 19 – Interface do módulo logging

4.4.6 Dashboard

O Dashboard contempla a apresentação de todos as informações coletadas pelos monitoramentos que estejam cadastrados e ativos. Estas informações, dependendo do tipo de visualização configurada no monitoramento, são apresentadas para o usuário no dashboard.

Nesta versão da aplicação, existem dois tipos de visualizações para o dado coletado pelo monitoramento: number e string. Na verdade trata-se de duas maneiras distintas de apresentação da informação para o usuário.

Não há nada de especial com relação a visualização das informações. Como nesta versão não há nenhum tipo de tratamento para os dados coletados, foram criadas estes dois tipos de visualização. Neste sentido, ao se cadastrar um novo monitoramento, deve-se indicar o tipo. Caso o dado coletado seja apenas um número, tanto absoluto como porcentagem, o elemento que exibirá o dado é criado de uma forma que este dado possa ser melhor visualizado. Já no caso do cadastro de um monitoramento com o tipo de visualização string, podendo ser um texto de qualquer tamanho, o elemento criado tem um outro estilo, mas com o mesmo intuito, de melhor visualização da informação. A figura 20 mostra o dashboard, com informações sobre os monitoramentos, inclusive o dado coletado.

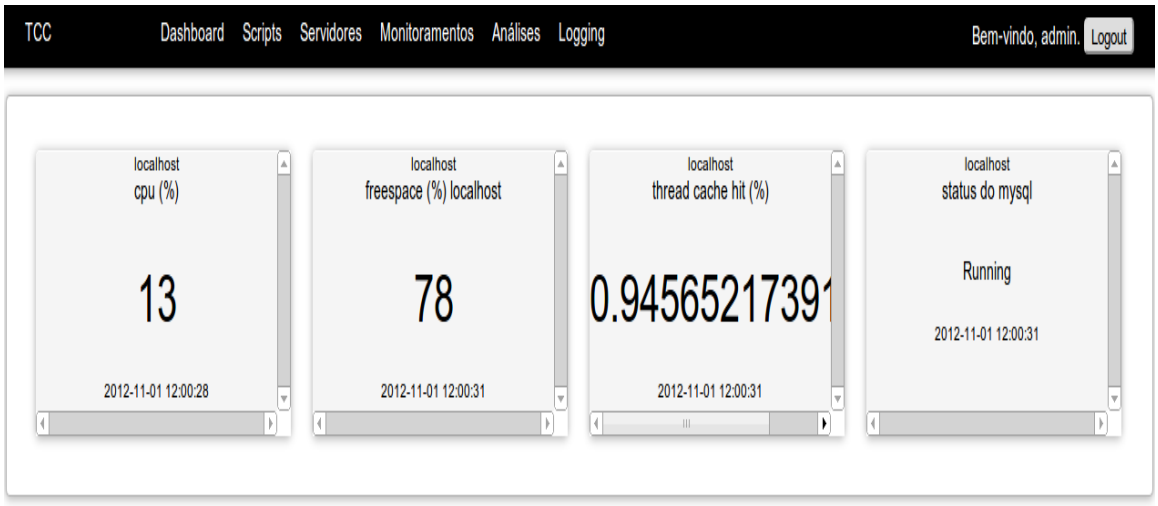

Figura 20 – Interface do módulo dashboard

4.5 Utilização da aplicação

Descreveremos como proceder para instalar a aplicação pela primeira vez.

O primeiro passo é a configuração do banco de dados MySQL. O arquivo bootstrap.sql deve ser executado para que o banco seja configurado e os dados iniciais da aplicação sejam carregados. Então, é preciso configurar o arquivo config.php. O usuário deve alterar os dados de acordo com o seu ambiente. Neste ponto, acessando http://localhost/tcc, a aplicação estará disponível.

Uma vez que a aplicação esteja disponível, já é possível iniciar a configuração do primeiro monitoramento.

O primeiro módulo que deve ser acessado é o servidor. Lá, será feito o cadastro de um servidor que iremos monitorar.

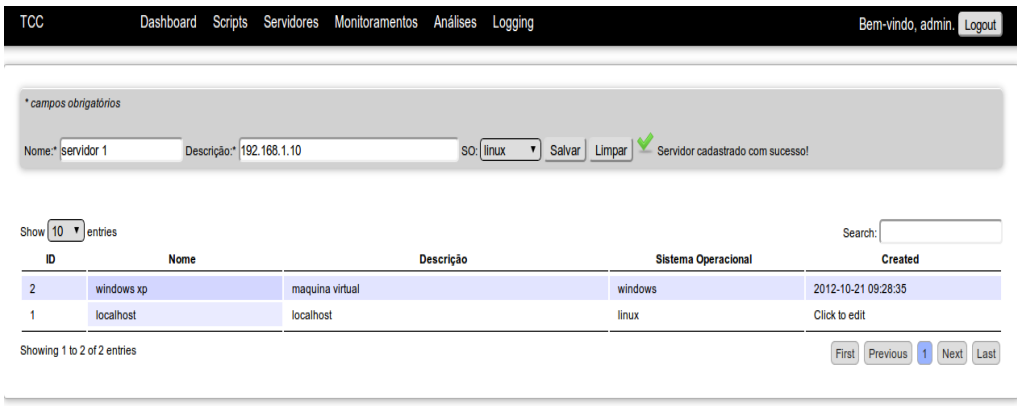

Figura 21 – Cadastro de servidor com sucesso

A figura 21 mostra como devemos proceder para cadastrar um novo servidor com sucesso. No exemplo, um servidor de nome *Servidor 1*, com a descrição *192.168.1.10*, que no caso é o endereço ip da máquina, e de sistema operacional Linux foi cadastrado com êxito.

Feito isso, devemos, então, cadastrar o primeiro script da aplicação. Todos os scripts estão divididos no sistema de arquivos por domínios. Os domínios existentes são: linux, windows, java, mysql, oracle e postgres. Cada uma destas pastas possue vários scripts já configurados.

Como dito anteriormente, estes já são oferecidos, mas ainda precisam ser cadastrados para que sua utilização seja possível. Para cadastrar um script, deve-se acessar o módulo script.

A figura 22 mostra o cadastro de um novo script bash, de descrição *utilização da memória (%)* e com arquivo de nome *utilização_memoria.sh*.

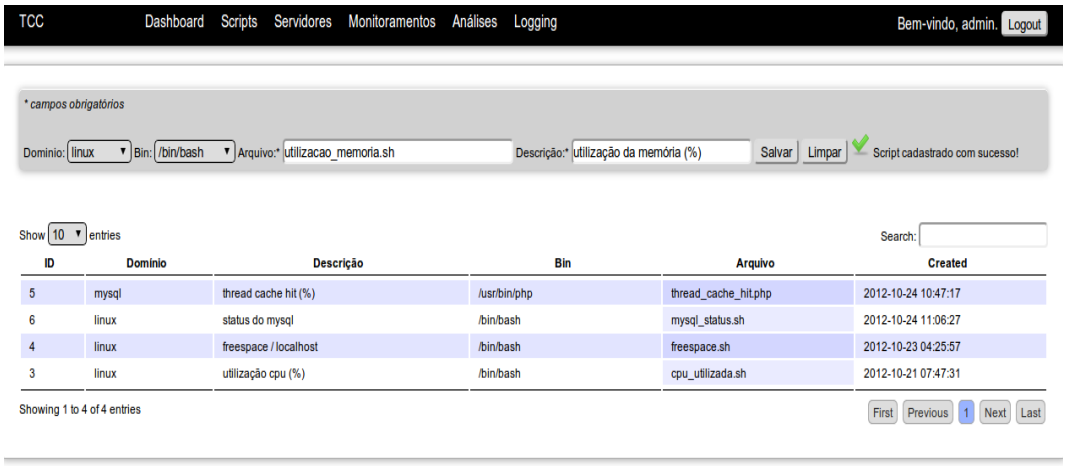

Figura 22 - Cadastro de script com sucesso

Tendo o primeiro script cadastrado, podemos acessar o módulo dos monitoramentos e configurar nosso primeiro monitoramento.

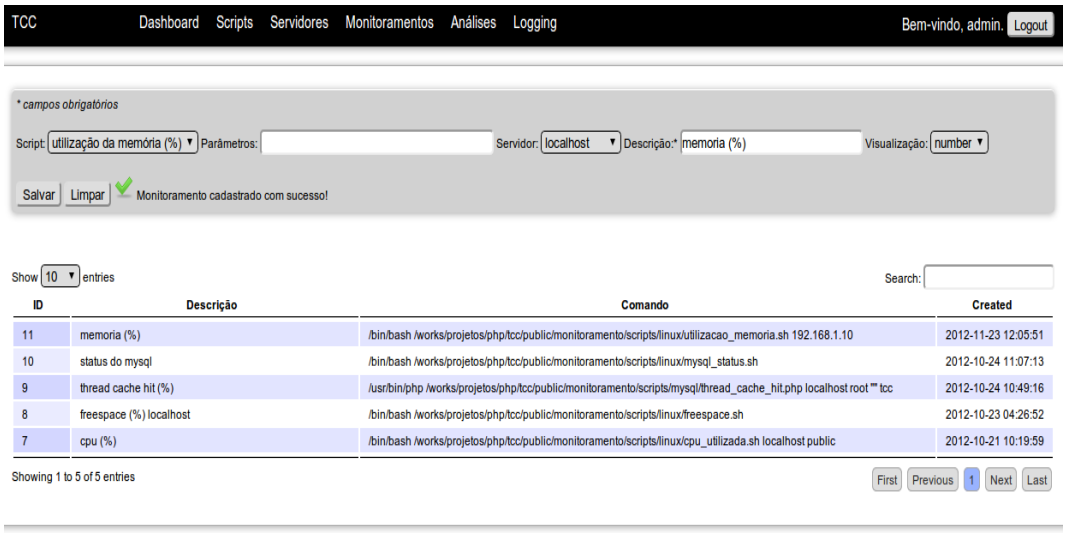

Figura 23 - Cadastro de monitoramento com sucesso

A figura 23 mostra o cadastro de um novo monitoramento com sucesso. Neste cadastro, utilizamos tanto o servidor como o script cadastrados anteriormente. No caso, o monitoramento, não recebe parâmetros. A visualização deste monitoramento é *number*.

O monitoramento recém configurado será executado a cada minuto, pelo serviço responsável pela execução dos monitoramentos. Entretanto, para o processo ficar completo, podemos configurar uma análise para este monitoramento que acabamos de criar. Acessando o módulo das análises, é possível configurar um valor crítico, que será comparado com o dado coletado pelo monitoramento. Também é preciso configurar o tipo de alerta pelo qual queremos ser notificados em caso de uma análise positiva.

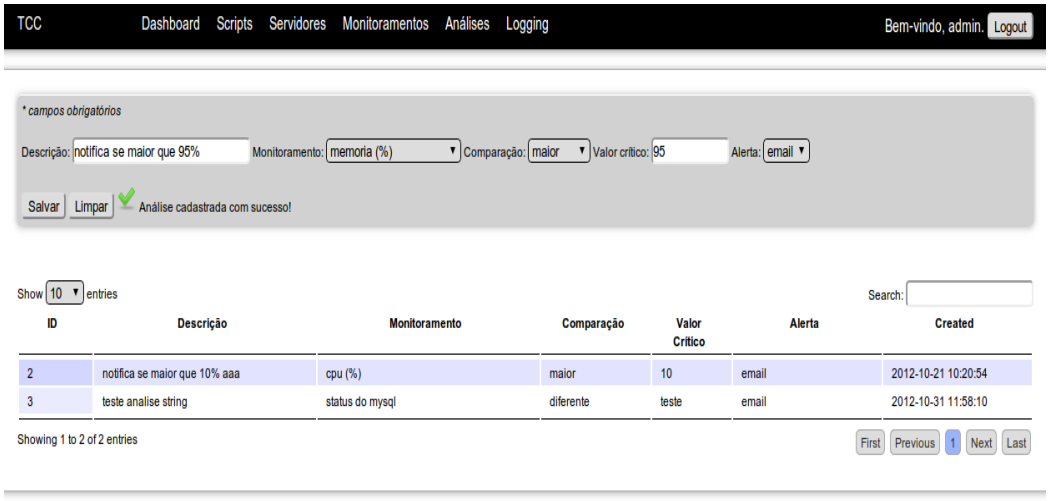

Figura 24 - Cadastro de análise com sucesso

A figura 24 mostra um cadastro de análise realizado com sucesso. A análise foi configurada para o monitoramento da memória. O valor crítico foi configurado em 95, com *email* como tipo de alerta. Isto significa que, toda vez que o valor coletado pelo monitoramento da memória do servidor *localhost*, que utiliza o script de nome *utilizacao_memoria.sh*, foi maior que 95, uma notificação por e-mail será enviada ao usuário.

Repetindo estes passos, é possível cadastrar vários monitoramentos para mais de um servidor do nosso ambiente.

Para exemplicar, a figura 25 mostra o script utilizaado para monitorar a memória utilizada.

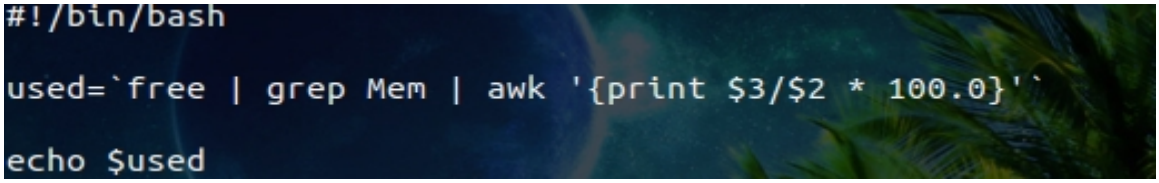

Figura 25 – Script utilização_memoria.sh

5 Considerações Finais

Este trabalho apresentou os elementos básicos que fazem parte da aplicação proposta. Cada um dos módulos foi descrito. Falamos de alguns protocolos utilizados pelos scripts já desenvolvidos e oferecidos pela aplicação.

Também foram abordadas algumas ferramentas de monitoramento, com foco nas opensource. Depois, a proposta da aplicação foi apresentada, uma ferramenta de monitoramento de recursos computacionais de servidores Linux, Windows, máquina virtual do Java, bancos de dados, entre outros, baseada em scripts.

5.1 Contribuições

Ferramenta / Característica	Nagios	OpenNMS	Cacti	TCC
Mapas	Sim	Sim	Sim, via plugin	Não
Agentes	Sim	Não	Não	Não
Relatórios	Sim, via plugin	Sim	Sim	Não
Alertas	Sim	Sim	Sim	Sim

A tabela acima mostra uma breve comparação entre algumas características das 3 ferramentas apresentadas no capítulo três e a aplicação proposta neste trabalho.

O que podemos perceber é que a aplicação proposta neste trabaho tem suas diferenças em comparação às outras ferramentas descritas. No caso dos agentes, em particular, o fato de não precisarmos configurar agentes em cada uma das máquinas que queremos monitorar, diferentemente do Nagios, é uma grande vantagem da minha proposta.

Já uma grande vantagem do Nagios, além de basear-se em scripts, é a grande comunidade que colobora com novos scripts e os disponibiliza para todos os usuários desta ferramenta.

Neste sentido, a aplicação proposta é semelhante ao Nagios, exceto pela comunidade, que inexiste até o momento. A principal vantagem da aplicação é o fato de basear-se em scripts para a execução dos monitoramentos. Estes scripts podem ser desenvolvidos em qualquer linguagem de programação suportada pelo Linux. Vale lembrar que uma das premissas de utilização desta ferramenta é que ela foi desenvolvida para rodar sobre distribuições Linux apenas, o que proporciona tamanha flexibilidade.

Hoje em dia, a maioria das distribuições Linux suportam a grande maioria de linguagens por padrão. Assim sendo, a utilização de script desenvolvidos em python, php, bash, java, perl, etc torna-se possível. O usuário terá apenas de garantir que estes scripts, desenvolvidos por ele ou por terceiros, estejam funcionando.

Como dito anteriormente, não há qualquer tratamento por parte da aplicação dos dados recuperados pelos monitoramentos. Esta responsabilidade é do desenvolvedor do script.

5.2 Trabalhos Futuros

Tendo uma plataforma para desenvolvimento estabelecida, a expansão torna-se mais fácil. Com base em possíveis novas demandas e por se tratar de uma aplicação modular, diversas possibilidades poderão ser exploradas no futuro, a fim de oferecer ao próprio usuário mais facilidades na utilização da aplicação.

Uma atual necessidade que não foi desenvolvida ainda, é a possibilidade de configuração individual, para cada monitoramento, do tempo de execução. Quer dizer, como está implementado hoje, todos os monitoramentos ocorrem sempre no mesmo tempo, a cada minuto, executados em seqüência. A idéia seria desenvolver a possibilidade de cada um destes monitoramentos ser executado em tempos distintos, dependendo da importância e necessidade, uma vez que há tipos de monitoramentos que não precisam ser feitos todo minuto.

Além disso, entendemos que a internacionalização da aplicação e a possibilidade de execução remota de

comandos seriam outras duas interessantes capacidades a serem atribuídas a ferrementa.

Referências

Sullins, B., Whipple, M. “JMX in Action”. Manning Publications, 2003.

Lindfors, J., Fleury, M. “JMX: Managing J2EE with Java Management Extensions”. Sams Publishing, 2002.

Tanenbaum, A. “Computer Networks, Fourth Edition”. Prentice Hall, 2003.

Mauro, D., Schmidt, K.. “Essential SNMP”. O'Reilly, 2001.

Kurose, J. e Ross, K. “Redes de computadores e a internet: uma abordagem top-down”.
Editora Addison Wesley. 3ªed., São Paulo. 2006

Melchiors, C.“Raciocínio Baseado em Casos Aplicado ao Gerenciamento de Falhas”.
151f. Tese de Mestrado em Ciência da Computação. Universidade Federal do Rio
Grande do Sul. UFRGS. Porto Alegre. 2003.

Oliveira, F. “Gerenciamento de Redes de Computadores com o uso do raciocínio
baseado em casos e ferramentas auxiliares”. 148f. Tese de Doutorado. Universidade
Federal do Rio de Janeiro. UFRJ. Rio de Janeiro. 2007

http://www.nagios.org/
Acesso em julho de 2011.

http://www.opennms.org/
Acesso em julho de 2011.

http://www.ibm.com/developerworks/websphere/library/techarticles/0304_polozoff/polozoff.html Acesso em maio de 2011.

http://www.manageengine.com/products/applications_manager/Applications-performance-management.pdf Acesso em maio de 2011.

Apêndices

```
<?php session_start(); ?>
<?php require_once (dirname(__FILE__) . '/../../application/config/constantes.php')?>
<!doctype html>
<!-- Consider adding a manifest.appcache: h5bp.com/d/Offline -->
<!--[if IE 7]>  <html class="no-js lt-ie9 lt-ie8" lang="en"> <![endif]-->
<!--[if IE 8]>  <html class="no-js lt-ie9" lang="en"> <![endif]-->
<!-- Consider adding a manifest.appcache: h5bp.com/d/Offline -->
<!--[if gt IE 8]><!--> <html class="no-js" lang="en"> <!--<![endif]-->
<head>
<meta charset="utf-8">
        <script type="text/javascript">
<!-- Use the .htaccess and remove these lines to avoid edge case issues.
       More info: h5bp.com/b/378 -->
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
</script>
<title><?php echo $this->get_title(); ?></title>
<meta name="description" content="">
</html>
<!-- Mobile viewport optimized: h5bp.com/viewport -->
<meta name="viewport" content="width=device-width">

<link rel="icon" type="image/png" href="<?php echo IMG_URI?>favicon.png">

<script src="http://code.jquery.com/jquery-latest.js"></script>

<link rel="stylesheet" href="<?php echo CSS_URI ?>bootstrap.min.css">
<link rel="stylesheet" href="<?php echo CSS_URI ?>estilo.css">

  <?php $this->import_css(); ?>

  <!-- All JavaScript at the bottom, except this Modernizr build.
       Modernizr enables HTML5 elements & feature detects for optimal performance.
       Create your own custom Modernizr build: www.modernizr.com/download/ -->
<script src="<?php echo JS_URI ?>modernizr-2.5.3.min.js"></script>
<script src="<?php echo JS_URI ?>bootstrap.min.js"></script>
<script src="<?php echo JS_URI ?>knockout-2.2.0.js"></script>
<script src="<?php echo JS_URI ?>jquery.json-2.4.min.js"></script>

</head>
<body>
        <!-- Prompt IE 6 users to install Chrome Frame. Remove this if you support IE 6.
       chromium.org/developers/how-tos/chrome-frame-getting-started -->
        <!--[if lt IE 7]><p class=chromeframe>Your browser is <em>ancient!</em> <a href="http://
browsehappy.com/">Upgrade to a different browser</a> or <a href="http://www.google.com/chromeframe/?
redirect=true">install Google Chrome Frame</a> to experience this site.</p><![endif]-->
        <header class="menu">
        <?php $this->show_header(); ?>
  </header>
        <div role="main" id="main">
        <?php $this->show_body(); ?>
  </div>
        <!-- <footer id="footer">
        <?php //$this->show_footer(); ?>
  </footer> -->

        <!-- JavaScript at the bottom for fast page loading -->

        <!-- Grab Google CDN's jQuery, with a protocol relative URL; fall back to local if offline -->
        <script
                src="//ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js"></script>
        <script>window.jQuery || document.write('<script src="<?php JS_URI?>jquery-1.7.1.min.js"><\/
script>')</script>

        <!-- scripts concatenated and minified via build script -->
        <script src="<?php echo JS_URI ?>plugins.js"></script>
        <script src="<?php echo JS_URI ?>script.js"></script>
        <script src="<?php echo JS_URI ?>flexigrid.js"></script>
```

```
<?php
$rResultTotal = mysql_query ( $sQuery, $gaSql ['link'] ) or die ( mysql_error () );
$aResultTotal = mysql_fetch_array ($rResultTotal );
$iTotal = $aResultTotal [0];
/*
 * Output
 */
$output = array(
"sEcho" => intval($_GET['sEcho']),
"iTotalRecords" => $iTotal,
"iTotalDisplayRecords" => $iFilteredTotal,
"aaData" => array ()
);
while ( $aRow = mysql_fetch_array ( $rResult ) ) {
$row = array ();
for ( $i = 0; $i < count ( $aColumns ); $i ++) {
if ( isset ( $_GET ['sSearch'] ) && $_GET ['sSearch'] != "" ) {
$row [] = $aRow [ $aColumns [$i] ];
}
}
$output ['aaData'] [] = $row;
}
echo json_encode ( $output );
?>
```

```
<?php    public function getByMonitoramento($monitoramentoID){
require_once (dirname ( __FILE__ ) . '/../utils/DB.php');
require_once (dirname ( __FILE__ ) . "WHERE Monitoramento_id = '%s'",
mysql_real_escape_string ( $monitoramentoID ));
                $mail->CharSet = 'utf-8';       $this->executaAlerta ( $analise ['Alerta_id'] );
/**             try {                           break;
 *              $mail->SetFrom(                 'A descrição não pode ser vazia';
 *              } catch (Exception $e) {
 * @author andre
 */             $mail->AddAddress(              'O valor crítico não pode ser vazio';
                }               if ($dadoRecuperado != $valorCritico) {
class AnaliseServiceImpl                        $this->executaAlerta ( $analise ['Alerta_id'] );
                return $this->DB->            'O assunto não pode ser vazio';
        protected $DB;          Logger::getLogger ( "AnaliseServiceImpl" )->info ( "Erro ao enviar email
para $destinatario" );
        public function getComparacoes() {
        /**             // string   Logger::getLogger ( "AnaliseServiceImpl" )->info ( "Email enviado com
sucesso!" );                                     limitando-a somente a um objeto
         * (Singleton)  }
         *      try VALIDATE    $dadoRecuperado = $monitoramento ['dado'];
         * @see http://www.php.net/          $analise ['valor_critico'];
        }*/     } catch ( Exception $e ) {   $comparacao = $analise ['Comparacao_id'];
        public static function getInstance() {   $e->getMessage(), __FILE__);
        private function notificaViaMSN($destinatario, $mensagem) {
                }       self::$instance = new self ();
                require_once (APPLICATION_PATH . '/utils/nusoap/nusoap.php');
                // Retornando os resultados em forma de matriz    $this->executaAlerta ( $analise ['Alerta_id'] );
        }       // Criando o DB->parser (Destinatario, mensagem)  $column%'";
        }       $param = array ('arg0' => $destinatario, 'arg1' => $mensagem );
        /**     $endpoint = "http://.../listener";
        public function ... obtenha a partir da função getInstance()
         */     // Chamando o cliente ... ( $result, 0, 0 );
        private function __construct() {
                $this->DB = DB::getInstance();    $dadoRecuperado != $valorCritico) {
                $this->DB->...                    $this->executaAlerta ( $analise ['Alerta_id'] );
        }               $result = $this->DB->query ( $query );
        }       $where = ("Exception ... mysql_real_escape_string ( $id ) . "'";
        /**             }
}        * Previne ... clonagem do objeto
?>      }*/     }       $this->DB->update ( 'Analise', $fields, $where, false );
        private function __clone() {    em forma de matriz
        public function update(...)   Logger::getLogger ( "AnaliseServiceImpl" )->error ( "Erro ao atualizar", $e );
                }       return json_encode ( array ('error' => 'Erro ao atualizar no banco de dados' ) );
        public function save($descricao, $valorCritico, $comparacao, $alerta, $monitoramentoID) {
        public function executaAnalise($analise, $monitoramento) {
                return ... ($descricao, $valorCritico, $monitoramentoID );
                $visualizacao = $monitoramento ['Visualizacao_id'];
        }       if ($invalidFields)  Logger::getLogger ( "AnaliseServiceImpl" )->info ( "Notificando via E-
mail" );        switch (   return json_encode ( array ('error' => $invalidFields ) );
        public function remove($ids) {   ->notificaViaEmail ( '', 'teste', 'teste' );
                        // number  break;
                if (tuas campos do Banco de Dados exemplo: $fields['nome_da_coluna'] =
                // 'valor'  return json_encode ( array ('error' => 'IDs inválidos' ) );
                $fields = array ();   $dadoRecuperado = $monitoramento ['dado'];
                $fields ['descricao']   Logger::getLogger ( "AnaliseServiceImpl" )->info ( "Notificando via MSN"
);              $ids_str = implode ( ...   $analise ['Comparacao_id'];
                $query = sprintf ( ... notificaViaMSN ( ... 'teste' );
                $fields ['Comparacao_id'] = $comparacao;   WHERE id IN ('%s')", $ids_str );
                $fields ['Alerta_id']  if ($comparacao == 1) {
        }       try { $fields ['created'] = date ( ... time () );   $dadoRecuperado ... $valorCritico) {
                        $this->DB->query ( $query );   $this->executaAlerta ( $analise ['Alerta_id'] );
        private function notificaViaEmail($destinatario, $assunto, $mensagem) {
                        // Deixar o último parâmetro como FALSE (permitir updates de somente
                require_once (APPLICATION_PATH . '/utils/PHPMailer/class.phpmailer.php'); de dados' ) );
                }       $this->DB->insert ( 'Analise', $fields );
                } catch ( Exception $e ) {   $comparacao == 2) {
                $mail->IsSMTP ();   Logger::getLogger ( ... "Erro ao registro da análise", $e );
                return json_encode ( array ('error' => ... $this->executaAlerta ( $analise ['Alerta_id'] ) dados' ) );
        }       $mail->SMTPSecure = "tls";      break;
                $mail->Host = "smtp.gmail.com";
```

```
<?php   }               return json_encode ( array ('error' => 'Query: $query' ) __FILE__ );
require_once (dirname ( __FILE__ ) . '/../config.php' );          'available' => false ) );
require_once (dirname ( __FILE__ ) . '/Dominio.php' );  $fields = array(), $start = 0, $limit = 0) {
                return json_encode ( array ('error' => null ) );
/**     }       if (mysql_result ( $result_sql, 0, 'total' ) > 0) {
 * Classe controle de Configurações                 return json_encode ( array ('available' => false ) );
 *      private function validate($nome, $sobrenome, $email, $login, $senha, $repete_senha) {
 * @author andre <durieux@inf.ufsc.br>
 *              return json_encode ( array ('available' => true ) );
 */     }       $query = "SELECT * FROM Dominio";  'O nome não pode ser vazio';
                // Caso o ID tenha sido especificado, retornará apenas os dados
class ConfigService {   // referentes a aquele ID
         * Verifica a disponibilidade de um login (nome de usuário)   não pode ser vazio';
        protected $DB;  $query .= sprintf ( " WHERE id='%s' LIMIT 1", mysql_real_escape_string ( $id ) );
        private static $instance;  // Verifica se os parâmetros para limite/paginação foram passados
         *      } elseif (is_numeric ( $start ) && $start >= 0 && is_numeric ( $limit ) && $limit > 0) {
        /** @return string JSON available = TRUE caso não exista ...  mysql_real_escape_string ( $start ),
mysql_real_escape_string ( $limit ) ); FALSE, restringindo-a somente a um objeto
         * (Singleton) $result = json_decode($this->email_available($email), true);
         */     try {  if (!$result['available'] && $update == 0){
        public function login_available($login) {              'E-mail já cadastrado';
         */     $query = sprintf ( "SELECT COUNT(*) as total
        public static function getInstance() {  FROM Usuario
                if (! self::$instance) {   WHERE login='%s'", mysql_real_escape_string ( $login ) ); e );
                        self::$instance = new self ();   'O login não pode ser vazio';
                }       try {
                return self::$instance;  $result = $this->DB->query ( $query );
        }       return $this->DB->query ( $query );  $this->login_available($login), true);
                // if (!$result['available'] && $update)"{(Query: $query)", __FILE__);
        /**     // $invalid = json_encode ( array ('login já cadastrado' => false ) );
         * Construtor privado, chamado a partir da função getInstance()
        public function update($id, $nome, $sobrenome, $email, $login, $senha, $repete_senha, $msn = null,
$ddd = null, private function __construct() {  ( $result_sql, 0, 'total' ) > 0) {
                $this->DB = DBConnection::getInstance();  'available' => false ) );
                $this->DB->connect();  $campos_invalidos ['senha'] = 'A senha deve conter ao menos 5 caracteres';
        }       } elseif (return json_encode ( array ('error' => 'ID não especificado' ) );
                return json_encode ( array ('repete_senha' => 'Confirme a senha';
        /**     } elseif ($senha && $senha != $repete_senha) {
}        * Previne a cópia do objeto  $this->validate($nome, $sobrenome, $email, $login, $senha,
?>$repete_senha );}
        private function __clone() {
        }       if ($campos_invalidos) {
        }               return json_encode ( array ('error' => $campos_invalidos ) );
        public function retrieveBin($id = '', $fields = array(), $start = 0, $limit = 0) {
        /**     $columns = '*';
         *      // Atualização no Banco de Dados (exemplo: $fields['nome_da_coluna'] =
         *      // 'valor'; Junta os campos/colunas para a consulta SQL
         *      $fields = array(); implode ( ',', $fields );
         *      $fields ['nome'] = $nome;
         *      // Prepara a consulta SQL  $sobrenome;
         *      $query = "SELECT $columns FROM Bin";
         *      // Caso o ID tenha sido especificado, retornará apenas os dados
         *      // referentes a aquele ID
         *      if ($id) {   com MD5
         *      if ($senha) $query .= sprintf ( " WHERE id='%s' LIMIT 1", mysql_real_escape_string ( $id ) );
         *      }       $fields // Verifica se os parâmetros para limite/paginação foram passados
         *      elseif (is_numeric ( $start ) && $start >= 0 && is_numeric ( $limit ) && $limit > 0) {
         *              $query .= sprintf ( " LIMIT %u,%u", mysql_real_escape_string ( $start ),
mysql_real_escape_string ( $limit ) );
         *      $fields ['ddd'] = $ddd;
         * @param $fields[unknown_type $celular;
         */             $result = $this->DB->query ( $query );
        public function isEmailAvailable($email) {  mysql_real_escape_string ( $id ) . "'";
                $query = sprintf ( "SELECT COUNT(*) as total
                try {   Logger::getLogger ( "ConfigServiceImpl" )->error ( "Erro ao recuperar bin", e );
                        return ... último parâmetro como 'FALSE'  mysql_real_escape_string ( $email ) );
                }       // alguns campos)
                try {   Retornamos o DB->update( formar de $fields, $where, false );
                return $this->DB->query ( $query );
                } catch (Exception $e) {  getMessage(), __FILE__);
```

```php
<?php
require_once (dirname(__FILE__) . '/../dao/DadoDAO.php');
require_once (dirname(__FILE__) . '/../dao/DescricaoDAO.php');

/**
 * Classe controlesdo Dashboard.
 *
 * @author Andre Duarte Bueno <duarte@inf.ufsc.br>
 *
 */
class DashboardServiceImpl {

	protected $DB;
	private static $instance;

	/**
	 * Obtém uma instância da classe, restringindo a somente a um objeto
	 * (Singleton)
	 *
	 * @see http://en.wikipedia.org/wiki/Singleton_pattern
	 */
	public static function getInstance() {
		if (! self::$instance) {
			self::$instance = new self ();
		}
		return self::$instance;
	}

	/**
	 * Construtor privado da função getInstance()
	 */
	private function __construct() {
		$this->DB = DBFactory::getInstance();
		$this->DB->connect();
	}

	/**
	 * Previne cópia do objeto
	 */
	private function __clone() {
	}

	/**
	 * Monta o layout de exibição dos monitoramentos
	 *
	 * @param $monitoramentos matriz
	 */
	function createDashboard($monitoramentos) {
		foreach ( $monitoramentos as $monitoramento ) {
			$ret = $monitoramento ['visualizacao_id'];
			if ($ret == 1) {
				$this->createNumberBox ( $monitoramento );
			}
			if ($ret == 2) {
				$this->createStringBox ( $monitoramento );
			}
		}
	}

	public function createGaugeBox($monitoramento) {
	}

	public function createNumberBox($monitoramento) {
		$descricao = $monitoramento ['descricao'];
		$datahora = $monitoramento ['datahora_coleta'];
		$dado = $monitoramento ['dado'];
		$id = $monitoramento ['id'];
		$descricao = $monitoramento ['descricao'];
		$datahora = $monitoramento ['datahora_coleta'];
		$id = "div" . $monitoramento ['id'];
		$styleDescricao = "font-size: 14px; margin-top: 30px;";
		$styleDado = "font-size: 12px; margin-top: 20px;";
		$styleDatahoraColeta = "font-size: 10px; margin-top:20px;";
		$css = '
		width: 260px;
		height: 150px;
		text-align: center;
		margin: auto;
		margin: 1em;
		background-color: #F5F5F5;
		border-radius: 4px;
		box-shadow: 0 1px 10px #a7a7a7, inset 0 1px 0 #fff;';
		echo "<div id='$id' style='$css'>";
		echo "<p>$servidor</p>";
		echo "<p style='$styleDescricao'>$descricao</p>";
		echo "<p style='$styleDado'>$dado</p>";
		echo "<p style='$styleDatahoraColeta'>$datahora</p>";
		echo "</div>";
		echo "
		<script type='text/javascript'>
		var g1;
		window.onload = function(){
		  var g1 = new JustGage({
		    id: '$id',
		    value: $dado,
		    min: 0,
		    max: 100,
		    title: '$descricao',
		    label: '$servidor'
		  });
		};
		</script>";
		echo "$javascript";
	}
}
```

```php
<?php           $comparacoes = $analiseService->getComparacoes ();
require_once (dirname ( __FILE__ ) . '/../config/general.php');
require_once (APPLICATION_PATH . "/service/MonitoramentoServiceImpl.php');
require_once (APPLICATION_PATH . '/service/AnaliseServiceImpl.php');
require_once (APPLICATION_PATH . '/utils/SSH2Light.php');
require_once (APPLICATION_PATH . '/utils/Logger.php');
require_once (APPLICATION_PATH . '/service/ExecutorServiceImpl.php');
                        $comando = $monitoramento ['comando'];
/**
 *                      $logger->debug ( "Executando o comando $comando" );
 * @author andre <durieux@inf.ufsc.br>  $shell->exec ( $comando );
 *
 */                     if ($dado == "" || $dado == null) {
                                $dado = $shell->getLastError ();
class ExecutorServiceImpl {
                        $logger->debug ( "Dado coletado: $dado" );
        protected $DB;
        private static $ins  if ($dado == $shell->getLastError ()) {
                                $logger->debug ( "Erro ao executar monitoramento $monitoramentoID" );
        /**                     $dado = "erro";
         * Obtém a única instância da classe, restringindo-a somente a um objeto
         * (Singleton)          $monitoramentoService->saveColeta ( $monitoramentoID, $dado );
         *
         * @see http://en.wikipedia.org/wiki/Singleton_pattern
         */                     $analise = $analiseService->getByMonitoramento ( $monitoramentoID );
        public static function getInstance() {
                if (! se$logger->debug ( "Verificando se há análise para o monitoramento
$monitoramentoID" );    self::$instance = new self ();
                }               if ($analise != null || ! (empty ( $analise ))) {
                return self::$instance;
        }                       foreach ( $analise as $umaAnalise ) {
                                        $analiseID = $umaAnalise ['id'];
        /**                             $logger->debug ( "Análise encontrada: $analiseID" );
         * Construtor privado, chamado a partir da função getInstance();
         */                             $analiseService->executaAnalise ( $umaAnalise, $monitoramento );
        private function __construct() {
                $this->DB = DB::getInstance ();
                $this->DB->connect ();
        }                       $logger->debug ( "Nenhuma análise foi encontrada para o monitoramento
$monitoramentoID" );
        /**                     continue;
         * Previne cópia do objeto
         */             }
        private function __clone() {
}       }

?>      private static function connect() {

        }

        public static function execute() {

                $logger = Logger::getLogger ( 'ExecutorServiceImpl' );

                global $config;

                $shell = new SSH2Light ( $config ['ssh_host'], $config ['ssh_user'], $config
['ssh_password'] );
                try {
                        $shell->connect ();
                } catch ( Exception $e ) {
                        $logger->error ( "Erro ao conectar em $config ['ssh_host']" );
                        exit ( $e->getMessage () );
                }

                // Services
                $monitoramentoService = MonitoramentoServiceImpl::getInstance ();
                $analiseService = AnaliseServiceImpl::getInstance ();

                $monitoramentos = $monitoramentoService->retrieve ();
```

```
<?php    }
}* Output
/*/
$output = array ( "sEcho" => intval ( $_GET ['sEcho'] ), "iTotalRecords" => $iTotal,
"iTotalDisplayRecords" => $iFilteredTotal, "aaData" => array () );
 */does it word by word on any field. It's possible to do here, but concerned
while ( $aRow = mysql_fetch_array ( $rResult ) ) { MySQL's regex functionality is
 * very limited array (); database columns which should be read and sent back to DataTables. Use a space where
 */     for ( $i = 0 ; $i < count ( $aColumns ) ; $i ++ ) { example a counter or static image)
$sWhere = "";    if ($aColumns [$i] == "version") {
if (isset ( $_GET ['sSearch'] ) && $_GET ['sSearch'] != "" ) {
        $sWhere = "WHERE ("; Special output formatting for 'version' column
/*      for($i = 0; $i < count ( $aColumns ); $i ++) {
 * Indexed column (used for fast and accurate table cardinality) mysql_real_escape_string ( $_GET
['sSearch'] ) . "%' OR "; else if ($aColumns [$i] != ' ') {
$sIndexColumn = "id";    /*
        $sWhere = substr_replace ( $sWhere, "", - 3 );
/*      $sWhere .= ')';  */
}* DB table to use       $row [] = $aRow [$aColumns [$i]];
 */              }
$sTable = "Monitoramento";
 * Individual column filtering = $row;
}*/
for ( $i = 0; $i < count ( $aColumns ); $i ++) {
echo json_encode ( $output ); bSearchable_' . $i] ) && $_GET ['bSearchable_' . $i] == "true" && $_GET
$gaSql ['user'] = "root";
$gaSql ['password'] if ($sWhere == "") {
$gaSql ['db'] = "tcc";   $sWhere = "WHERE ";
$gaSql ['server'] } else { alhost";
                 $sWhere .= " AND ";
/*              }
 * * * * * * * * $sWhere .= "`" . $aColumns [$i] . "` LIKE '%" . mysql_real_escape_string ( $_GET
['sSearch_' . $i] ) . "%' "; want to use the basic configuration for DataTables
 * with PHP server-side, there is no need to edit below this line
}*/

/*      /*
 * SQL queries MySQL connection display
 */     */
$sQuery = $gaSql ['link'] = mysql_pconnect ( $gaSql ['server'], $gaSql ['user'], $gaSql ['password'] ) or
die ( 'Could not SELECT SQL_CALC_FOUND_ROWS `". str_replace ( " , ", " ", implode ( "`, `", $aColumns )
) . "`
mysql_select_db ( FROM  $sTable ], $gaSql ['link'] ) or die ( 'Could not select database ' . $gaSql ['db']
);               $sWhere
                 $sOrder
/*               $sLimit
 * Paging        ";
$rResult = mysql_query ( $sQuery, $gaSql ['link'] ) or die ( mysql_error () );
$sLimit = "";
if (isset ( $_GET ['iDisplayStart'] ) && $_GET ['iDisplayLength'] != '-1') {
 * Data set length after filtering mysql_real_escape_string ( $_GET ['iDisplayStart'] ) . ", " .
mysql_real_escape_string ( $_GET ['iDisplayLength'] );
$sQuery = "
                 SELECT FOUND_ROWS()
/*      ";
$rResultFilterTotal = mysql_query ( $sQuery, $gaSql ['link'] ) or die ( mysql_error () );
$aResultFilterTotal = mysql_fetch_array ( $rResultFilterTotal );
$iFilteredTotal = $aResultFilterTotal [0];
if (isset ( $_GET ['iSortCol_0'] )) {
/*      $sOrder = "ORDER BY  ";
 * Total data set length intval ( $_GET ['iSortingCols'] ); $i ++) {
 */              if ($_GET ['bSortable_' . intval ( $_GET ['iSortCol_' . $i] )] == "true") {
$sQuery = "          $sOrder .= "`" . $aColumns [intval ( $_GET ['iSortCol_' . $i] )] . "` " .
mysql_real_escape_string ( $_GET ['sSortDir_' . $i] ) . ", ";
                 FROM   $sTable
        ";
$rResultTotal = mysql_query ( $sQuery, $gaSql ['link'] ) or die ( mysql_error () );
$aResultTotal = mysql_fetch_array ( $rResultTotal );
$iTotal = $aResultTotal [0]; "ORDER BY") {
                 $sOrder = "";
```

```php
<?php
require_once (dirname ( __FILE__ ) . '/../persistence/DB.class.php');
require_once (dirname ( __FILE__ ) . '/../utils/Utils.php');
require_once (dirname ( __FILE__ ) . '/../utils/Logger.class.php');

/**
 * Classe controladora para a entidade monitoramento_coleta
 *
 * @author andre <durieux@inf.ufsc.br>
 *
 */
class MonitoramentoColetaService {
	protected $DB;
	protected $Logger;

	/**
	 * Obtém a única instância desse serviço (Singleton)
	 *
	 * @see http://pt.wikipedia.org/wiki/Singleton
	 */
	public function getInstance() {
		static $instance = null;
		if (null === $instance) {
			$instance = new self();
		}
		return $instance;
	}

	/**
	 * Construtor privado para impedir instanciação direta
	 */
	private function __construct() {
		$this->DB = DB::getInstance ();
		$this->Logger = Logger::getLogger ( "MonitoramentoColetaService" );
	}

	/**
	 * Retorna a versão do serviço
	 */
	public function getVersion() {
		return "1.0.0";
	}

	/**
	 * Impede clonagem da instância
	 */
	private function __clone() {
	}

	/**
	 * Impede unserialize da instância
	 */
	private function __wakeup() {
	}
}
?>
```

```
<?php
/*
 * Script:    DataTables server-side script for PHP and MySQL
 * Copyright: 2010 - Allan Jardine
 * License:   GPL v2 or BSD (3-point)
 * Contact:   http://datatables.net/forums/discussion/113/php-data-get-php-script-for-server-side-processing/0
 */

/* Array of database columns which should be read and sent back to DataTables. Use a space where
 * you want to insert a non-database field (for example a counter or static image)
 */
$aColumns = array( 'descricao', 'b.path', 'arquivo', 'created' );

/* Indexed column (used for fast and accurate table cardinality) */
$sIndexColumn = "id";

/* DB table to use */
$sTable = 'Dominio d INNER JOIN Bin b ON b.id = s.Dominio_id INNER JOIN Bin b ON b.id = s.Bin_id ';

/*
 * Paging
 */
$sLimit = "";
if ( isset( $_GET['iDisplayStart'] ) && $_GET['iDisplayLength'] != '-1' )
{
	$sLimit = "LIMIT ".intval( $_GET['iDisplayStart'] ).", ".
		intval( $_GET['iDisplayLength'] );
}

/*
 * Ordering
 */
$sOrder = "";
if ( isset( $_GET['iSortCol_0'] ) )
{
	$sOrder = "ORDER BY  ";
	for ( $i=0 ; $i<intval( $_GET['iSortingCols'] ) ; $i++ )
	{
		if ( $_GET[ 'bSortable_'.intval($_GET['iSortCol_'.$i]) ] == "true" )
		{
			$sOrder .= $aColumns[ intval( $_GET['iSortCol_'.$i] ) ]."
				".mysql_real_escape_string( $_GET['sSortDir_'.$i] ).", ";
		}
	}

	$sOrder = substr_replace( $sOrder, "", -2 );
	if ( $sOrder == "ORDER BY" )
	{
		$sOrder = "";
	}
}


/*
 * Filtering
 * NOTE this does not match the built-in DataTables filtering which does it
 * word by word on any field. It's possible to do here, but concerned about efficiency
 * on very large tables, and MySQL's regex functionality is very limited
 */
$sWhere = "";
if ( $_GET['sSearch'] != "" )
{
	$sWhere = "WHERE (";
	for ( $i=0 ; $i<count($aColumns) ; $i++ )
	{
		$sWhere .= $aColumns[$i]." LIKE '%".mysql_real_escape_string( $_GET['sSearch'] )."%' OR ";
	}
	$sWhere = substr_replace( $sWhere, "", -3 );
	$sWhere .= ')';
}

/* Individual column filtering */
for ( $i=0 ; $i<count($aColumns) ; $i++ )
{
	if ( isset( $_GET['bSearchable_'.$i] ) && $_GET['bSearchable_'.$i] == "true" && $_GET['sSearch_'.$i] != '' )
	{
		if ( $sWhere == "" )
		{
			$sWhere = "WHERE ";
		}
		else
		{
			$sWhere .= " AND ";
		}
		$sWhere .= $aColumns[$i]." LIKE '%".mysql_real_escape_string($_GET['sSearch_'.$i])."%' ";
	}
}

/*
 * SQL queries
 * Get data to display
 */
$sQuery = "
		SELECT SQL_CALC_FOUND_ROWS ".str_replace(" , ", " ", implode(", ", $aColumns))."
		FROM   $sTable
		$sWhere
		$sOrder
		$sLimit
	";
$rResult = mysql_query( $sQuery ) or die(mysql_error());

/* Data set length after filtering */
$sQuery = "
		SELECT FOUND_ROWS()
	";
$rResultFilterTotal = mysql_query( $sQuery ) or die(mysql_error());
$aResultFilterTotal = mysql_fetch_array($rResultFilterTotal);
$iFilteredTotal = $aResultFilterTotal[0];

/* Total data set length */
$sQuery = "
		SELECT COUNT(".$sIndexColumn.")
		FROM   $sTable
	";
$rResultTotal = mysql_query( $sQuery ) or die(mysql_error());
$aResultTotal = mysql_fetch_array($rResultTotal);
$iTotal = $aResultTotal[0];


/*
 * Output
 */
$output = array(
	"sEcho" => intval($_GET['sEcho']),
	"iTotalRecords" => $iTotal,
	"iTotalDisplayRecords" => $iFilteredTotal,
	"aaData" => array()
);

while ( $aRow = mysql_fetch_array( $rResult ) )
{
	$row = array();
	for ( $i=0 ; $i<count($aColumns) ; $i++ )
	{
		if ( $aColumns[$i] == "version" )
		{
			/* Special output formatting for 'version' column */
			$row[] = ($aRow[ $aColumns[$i] ]=="0") ? '-' : $aRow[ $aColumns[$i] ];
		}
		else if ( $aColumns[$i] != ' ' )
		{
			/* General output */
			$row[] = $aRow[ $aColumns[$i] ];
		}
	}
	$output['aaData'][] = $row;
}

echo json_encode( $output );
?>
```

```php
<?php
require_once (dirname ( __FILE__ ) . '/../utils/DB.php');
require_once (dirname ( __FILE__ ) . '/../utils/Auth.php');
require_once (dirname ( __FILE__ ) . '/../utils/Logger.php');
require_once (dirname ( __FILE__ ) . '/../utils/Validate.php');
// oracle
/**
 *
 * @author andre
 *
 */
class ScriptServiceImpl {
	protected $DB;
	private $binID;
}
?>
```

```
<?php
require_once (dirname(__FILE__) . '/../utils/Auth.php');
require_once (dirname(__FILE__) . '/../utils/...');
require_once (dirname(__FILE__) . '/../utils/...');

/**
 *
 *
 * @author andre...
 */
class ServidorServiceImpl ...

	protected $DB;
	private static $instance;
	}
	/**
	 * ... (Singleton)
	 *
	 * @see http://en.wikipedia.org/wiki/Singleton_pattern
	 */
	public static function getInstance() {
		if (! self::$instance) {
			self::$instance = new self ();
		}
		return self::$instance;
	}

	/**
	 * Construtor ...
	 */
	private function __construct() {
		$this->DB = DB::getInstance ();
		$this->DB->connect ();
	}

	/**
	 * Previne cópia/clonagem do objeto
	 */
	private function __clone() {
	}

	public function getSistemasOperacionais() {
	}
	private function ...
	}

	public function updateDB(...) {
	}

}
?>
```

```
		// Verifica se os parâmetros para limite/paginação foram passados
		if (is_numeric($start) && $start >= 0 && is_numeric($limit) &&
		    $limit ...
		$query .= sprintf(" LIMIT %u, %u",
		    mysql_real_escape_string($start),
		// mysql_real_escape_string($limit));
		$where = "id = '" . mysql_real_escape_string ( $id ) . "'";
		try {
			$this->DB->update( 'Servidor', $fields, $where, false );
		} catch ( Exception $e ) {
			Logger::getLogger ( "ServidorServiceImpl" )->error ( "Erro ao atualizar", $e );
			return json_encode ( array ('error' => 'Erro ao atualizar no banco de dados' ) );
		}
		// Retorna os resultados em forma de matriz
		return $this->DB->parse ( $result );
		return json_encode ( array ('error' => null ) );

		...->validate ( $nome, $descricao );
		if (! is_array( $ids ) || ! $ids) {
			return json_encode( array ('error' => 'Invalid ids' ));
		}
		// Atualiza o Banco de Dados (exemplo: $fields['nome_da_coluna'] =
		$query = sprintf( "DELETE FROM Servidor WHERE id IN ('%s')", $ids_str );
		$fields = array ();
		$fields ['nome'] = $nome;
		try {
			$this->DB->query( $query );
		} catch ( Exception $e ) {
			...->error("", $e->getMessage(), __FILE__);
			return json_encode ( array ('error' => 'Erro ao excluir no banco de dados' ) );
		}
		// ... último parâmetro como FALSE (permitir updates de somente
		// Nenhum ... Retorna nulo.
		return json_encode( $this->DB->insert('Servidor', $fields) );
		} catch ( Exception $e ) {
			... log("error", $e->getMessage(), __FILE__);
			return json_encode ( array ('error' => 'Erro ao atualizar no banco de dados' ) );
		}
		return json_encode ( array ('error' => null ) );
		$query = "SELECT ... FROM ...";
		// Caso o ID tenha sido especificado, retornará apenas os dados
		if ($id) {
			$query .= sprintf ( " WHERE id='%s' LIMIT 1", mysql_real_escape_string ( $id ) );
			$result = $this->DB->query ( $query );
		} catch ( Exception $e ) {
			... 'não pode ser vazio';
			... 'descrição não pode ser vazia';
		} catch ( Exception $e ) {
			...->getMessage(), __FILE__);
		// Retorna os resultados em forma de matriz
		return $this->DB->parse ( $result );
		}
		// Retorna os resultados em forma de matriz
		... $start = 0, $limit = 0) {
		$columns = '*';
		if (VALIDATE) {
		// MELHORAR ... campos/colunas para a consulta SQL
			$columns = implode ( ',', $fields );
		$tabela = "Servidor";
		... removeAcentos( $column );
		$query = "SELECT $columns FROM Servidor";
		// Caso o ID tenha sido especificado, retornará apenas os dados
		FROM ... columns
		WHERE table_name='$tabela' AND column_name LIKE '%$column%'";
			$query .= sprintf ( " WHERE id='%s' LIMIT 1", mysql_real_escape_string ( $id ) );
		$result = $this->DB->query ( $consulta );
```

```php
??php
require_once (dirname ( __FILE__ ) . '/../utils/DB.php');
require_once (dirname ( __FILE__ ) . '/../utils/Auth.php');

/**
 * Classe controle do Usuário.
 *
 * @author andre <durieux@inf.ufsc.br>
 */

class UserServiceImpl {

	protected $DB;
	private static $instance;

	/**
	 * Obtém a única instância da classe, restringindo-a somente a um objeto
	 * (Singleton)
	 *
	 * @see http://en.wikipedia.org/wiki/Singleton_pattern
	 */
	public static function getInstance() {
		if (! self::$instance) {
			self::$instance = new self ();
		}
		return self::$instance;
	}

	/**
	 * Construtor privado, chamado a partir da função getInstance()
	 */
	private function __construct() {
		$this->DB = DB::getInstance ();
		$this->DB->connect ();
	}

	/**
	 * Previne cópia do objeto
	 */
	private function __clone() {
	}

	public function retrieve($id = '', $fields = array(), $start = 0, $limit = 0) {
		$columns = '*';
		if ($fields) {
			// Junta os campos/colunas para a consulta SQL
			$columns = implode ( ',', $fields );
		}
		// Prepara a consulta SQL
		$query = "SELECT $columns FROM Usuario";
		// Caso o ID tenha sido especificado, retornará apenas os dados
		// referentes a aquele ID
		if ($id) {
			$query .= sprintf ( " WHERE id='%s' LIMIT 1", mysql_real_escape_string ( $id ) );
		}		// Verifica se os parâmetros para limite/paginação foram passados
		elseif (is_numeric ( $start ) && $start >= 0 && is_numeric ( $limit ) && $limit > 0) {
			$query .= sprintf ( " LIMIT %u,%u", mysql_real_escape_string ( $start ),
mysql_real_escape_string ( $limit ) );
		}
		try {
			$resultado = $this->DB->query ( $query );
		} catch ( Exception $e ) {
			// Logger::log($e->getMessage() . " (Query: $query)", __FILE__);
			return false;
		}

		// Retorna os resultados em forma de matriz
		return $this->DB->parse_result ( $resultado );
	}
}
```

```php
<?php
session_start ();
require_once (dirname ( __FILE__ ) . '/../../application/config/general.php');
require_once (dirname ( __FILE__ ) . '/../../application/utils/TemplateHandler.php');
require_once (dirname ( __FILE__ ) . '/../../application/service/MonitoramentoServiceImpl.php');
require_once (dirname ( __FILE__ ) . '/../../application/service/AnaliseServiceImpl.php');

$action = isset ( $_GET ['action'] ) ? $_GET ['action'] : '';
$usuarioID = isset ( $_SESSION ['id'] ) ? $_SESSION ['id'] : '';

$monitoramentoService = MonitoramentoServiceImpl::getInstance ();
$analiseService = AnaliseServiceImpl::getInstance ();

switch ($action) {

	case 'getMonitoramentos' :
		exit ( json_encode ( $monitoramentoService->retrieve () ) );
		break;

	case 'getComparacoes' :
		exit ( json_encode ( $analiseService->getComparacoes () ) );
		break;

	case 'getAlertas' :
		exit ( json_encode ( $analiseService->getAlertas () ) );
		break;

	case 'add' :

		// Estes dados não mudam
		$descricao = isset ( $_POST ['descricao'] ) ? $_POST ['descricao'] : '';
		$monitoramento = isset ( $_POST ['monitoramento'] ) ? $_POST ['monitoramento'] : '';
		$valorCritico = isset ( $_POST ['valor_critico'] ) ? $_POST ['valor_critico'] : '';
		$comparacao = isset ( $_POST ['comparacao'] ) ? $_POST ['comparacao'] : '';
		$alerta = isset ( $_POST ['alerta'] ) ? $_POST ['alerta'] : '';

		exit ( $analiseService->save ( $descricao, $valorCritico, $comparacao, $alerta,
$monitoramento ) );
		break;

	case 'update' :

		$id = $_REQUEST ['id'];
		$value = $_REQUEST ['value'];
		$column = $_REQUEST ['columnName'];
		$columnPosition = $_REQUEST ['columnPosition'];
		$columnId = $_REQUEST ['columnId'];
		$rowId = $_REQUEST ['rowId'];

		// Calling the service to save a new script
		exit ( $analiseService->update ( $id, $column, $value ) );
		break;

	case 'del' :

		$items = isset ( $_POST ['items'] ) ? $_POST ['items'] : '';
		$items = substr ( $items, 0, - 1 ); // Remove a última vírgula

		$ids = explode ( ",", $items ); // Transforma em array

		exit ( $analiseService->remove ( $ids ) );
		break;

	default :

		$template = new TemplateHandler ();
		$template->set_css_files ( array ('datatable.css' ) );
		$template->set_js_files ( array ('jquery-1.7.1.js', 'jquery.dataTables.js',
'jquery.dataTables.editable.js', 'jquery.jeditable.js', '/analise/analise.js', '/analise/datatable.js' )
```

```
<?php if (! Auth::check ()) { exit( 'Acesso negado' ); } ?>
                                <th></th>
<?php                           <th></th>
require_once (dirname ( __FILE__ ) . '/../../application/config/general.php');
?>              </tfoot>
        </table>
<fieldset>
        <div>
                <form id="addAnaliseForm" class="form-inline"
                        action="<?php echo ANALISE_URI ?>?action=add" method="post">

                        <div id="status" class="hide">
                                <button type="button" class="close" data-dismiss="alert">×</button>
                                <h4>Erro!</h4>
                                Verifique os campos obrigatórios
                        </div>
                        <br />
                        <p>
                                <label for="descricao">Descrição:*</label> <input type="text"
                                        id="descricao" name="descricao" size="30" /> <label
                                        for="monitoramento">Monitoramento:</label> <select
                                        id="monitoramento" name="monitoramento"
                                        data-bind="options: monitoramentos, value: selectedMonitoramento,
optionsText: 'descricao', optionsValue: 'id'">

                                </select> <label for="comparacao">Comparação:</label> <select
                                        id="comparacao" name="comparacao"
                                        data-bind="options: comparacoes, value: selectedComparacao,
optionsText: 'operador', optionsValue: 'id'">

                                </select> <label for="valor_critico">Valor crítico:</label> <input
                                        type="text" id="valor_critico" name="valor_critico" size="10" />
<label
                                        for="alerta">Alerta:</label> <select id="alerta" name="alerta"
                                        data-bind="options: alertas, value: selectedAlerta, optionsText:
'regra', optionsValue: 'id'">

                                </select>
                        </p>
                        <br /> <input type="submit" value="Salvar" id="botao_add" class="btn" />
                        <input type="reset" value="Limpar" id="clearButton" class="btn" /> <em>*
                                campos obrigatórios</em>
                </form>
        </div>
</fieldset>
<br />
<fieldset>
        <table class="display" id="datatable" style="text-align: center;">
                <thead>
                        <tr>
                                <th width="3%">ID</th>
                                <th width="15%">Descrição</th>
                                <th width="15%">Monitoramento</th>
                                <th width="5%">Comparação</th>
                                <th width="10%">Valor Crítico</th>
                                <th width="10%">Alerta</th>
                                <th width="10%">Created</th>
                        </tr>
                </thead>
                <tbody>
                        <tr>
                                <td colspan="5" class="dataTables_empty">Carregando</td>
                        </tr>
                </tbody>
                <tfoot>
                        <tr>
                                <th></th>
                                <th></th>
                                <th></th>
                                <th></th>
```

```php
<?php
require_once (dirname ( __FILE__ ) . '/../../application/config/general.php');
require_once (dirname ( __FILE__ ) . '/../../application/utils/TemplateHandler.php');

$template = new TemplateHandler ();
$template->set_css_files ( array () );
$template->set_js_files ( array ('justgage.1.0.1.js','raphael.2.1.0.min.js') );
$template->set_content_file ( 'dashboard/view.php' );
$template->set_authenticated_only ( true );
$template->show ();
```

```
<head>
<!-- Atualização da página a cada 45 segundos -->
<meta http-equiv="refresh" content="25" />
</head>
<?php if (!Auth::check()) { exit('Acesso negado'); } ?>
<?php

require_once (APPLICATION_PATH . '/service/MonitoramentoServiceImpl.php');
require_once (APPLICATION_PATH . '/service/DashboardServiceImpl.php');

$userID = isset ( $_SESSION ['id'] ) ? $_SESSION ['id'] : '';

if (! $userID) {
        exit ( 'ID não especificado' );
}

// Getting the services
$monitoramentoService = MonitoramentoServiceImpl::getInstance ();
$dashboardService = DashboardServiceImpl::getInstance ();

$monitoramentos = $monitoramentoService->retrieve ();

if (! $monitoramentos) {
        exit ( 'Nenhum monitoramento encontrado!' );
}

$monitoramentos = $monitoramentoService->atualizaMonitoramentos ( $monitoramentos );

?>
<div id="monitoramentos">
        <?php $dashboardService->createDashboard($monitoramentos); ?>
</div>
```

```php
<?php
session_start ();
require_once (dirname ( __FILE__ ) . '/../../application/configuration/Configuration.php');
require_once (dirname ( __FILE__ ) . '/../../application/service/AuthenticatedOnly.php');
require_once (dirname ( __FILE__ ) . '/../../application/utils/TemplateHandler.php');
require_once (dirname ( __FILE__ ) . '/../../application/service/MonitoramentoServiceImpl.php');
require_once (dirname ( __FILE__ ) . '/../../application/service/ServidorServiceImpl.php');
require_once (dirname ( __FILE__ ) . '/../../application/service/ScriptServiceImpl.php');

$action = isset ( $_GET ['action'] ) ? $_GET ['action'] : '';
$usuarioID = isset ( $_SESSION ['id'] ) ? $_SESSION ['id'] : '';

// Services
$monitoramentoService = MonitoramentoServiceImpl::getInstance ();
$servidorService = ServidorServiceImpl::getInstance ();
$scriptService = ScriptServiceImpl::getInstance ();

switch ($action) {

	case 'getScripts' :
		exit ( json_encode ( $scriptService->retrieve () ) );
		break;

	case 'getServidores' :
		exit ( json_encode ( $servidorService->retrieve () ) );
		break;

	case 'getVisualizacoes' :
		exit ( json_encode ( $monitoramentoService->getVisualizacoes () ) );
		break;

	case 'add' :

		$scriptID = isset ( $_POST ['script'] ) ? $_POST ['script'] : '';
		$descricao = isset ( $_POST ['descricao'] ) ? $_POST ['descricao'] : '';
		$servidorID = isset ( $_POST ['servidor'] ) ? $_POST ['servidor'] : '';
		$visualizacaoID = isset ( $_POST ['visualizacao'] ) ? $_POST ['visualizacao'] : '';
		$params = isset ( $_POST ['params'] ) ? $_POST ['params'] : '';

		exit ( $monitoramentoService->save ( $descricao, $scriptID, $usuarioID, $servidorID,
$visualizacaoID, $params ) );
		break;

	case 'update' :

		$id = $_REQUEST ['id'];
		$value = $_REQUEST ['value'];
		$column = $_REQUEST ['columnName'];
		$columnPosition = $_REQUEST ['columnPosition'];
		$columnId = $_REQUEST ['columnId'];
		$rowId = $_REQUEST ['rowId'];

		// Calling the service to save a new script
		exit ( $monitoramentoService->update ( $id, $column, $value ) );
		break;

	case 'del' :

		$items = isset ( $_POST ['items'] ) ? $_POST ['items'] : '';
		$items = substr ( $items, 0, - 1 ); // Remove a última vírgula

		$ids = explode ( ",", $items ); // Transforma em array

		exit ( $monitoramentoService->remove ( $ids ) );
		break;

	default :

		$template = new TemplateHandler ();
		$template->set_css_files ( array ('datatable.css' ) );
		$template->set_js_files ( array ('jquery-1.7.1.js', 'jquery.dataTables.js',
'jquery.dataTables.editable.js', 'jquery.jeditable.js', '/monitoramento/monitoramento.js', '/
datatable.js' ) );
		$template->set_content_file ( 'monitoramento.php' );
		$template->set_authenticated_only ();
		$template->show ();
}
```

```
<?php if (!Auth::check()) { exit('Acesso negado'); } ?>

<?php
require_once (dirname ( __FILE__ ) . '/../../application/config/general.php');
?>

<fieldset>
	<div>
		<form id="addMonitoramentoForm" class="form-inline"
			action="<?php echo MONITORAMENTO_URI ?>?action=add" method="post">
			<div id="status" class="hide">
				<button type="button" class="close" data-dismiss="alert">×</button>
				<h4>Erro!</h4>
				Verifique os campos obrigatórios
			</div>
			<p>
				<br /> <label for="script">Script:</label> <select id="script"
					name="script"
					data-bind="options: scripts, value: selectedScript, optionsText:
'descricao', optionsValue: 'id'">

				</select> <label for="params">Parâmetros:</label> <input type="text"
					id="params" name="params" size="40" /> <label
for="servidor">Servidor:</label>
				<select id="servidor" name="servidor"
					data-bind="options: servidores, value: selectedServidor,
optionsText: 'nome', optionsValue: 'id'">

				</select> <label for="descricao">Descrição:*</label> <input
					type="text" id="descricao" name="descricao" size="30" /> <label
					for="visualizacao">Visualização:</label> <select id="visualizacao"
					name="visualizacao"
					data-bind="options: visualizacoes, value: selectedVisualizacao,
optionsText: 'nome', optionsValue: 'id'">

				</select>
			</p>
			<br /> <input type="submit" value="Salvar" id="botao_add" class="btn" />
			<input type="reset" value="Limpar" id="clearButton" class="btn" /> <em>*
				campos obrigatórios</em>
		</form>
	</div>
</fieldset>
<br />
<fieldset>
	<table class="display" id="datatable" style="text-align: center;">
		<thead>
			<tr>
				<th width="5%">ID</th>
				<th width="15%">Descrição</th>
				<th width="65%">Comando</th>
				<th width="15%">Created</th>
			</tr>
		</thead>
		<tbody>
			<tr>
				<td colspan="5" class="dataTables_empty">Carregando</td>
			</tr>
		</tbody>
		<tfoot>
			<tr>
				<th></th>
				<th></th>
				<th></th>
				<th></th>
			</tr>
		</tfoot>
	</table>
</fieldset>
```

```php
<?php
session_start ();
require_once (dirname ( __FILE__ ) . '/../../application/config/general.php');
require_once (dirname ( __FILE__ ) . '/../../application/utils/TemplateHandler.php');
require_once (dirname ( __FILE__ ) . '/../../application/service/ScriptServiceImpl.php');
require_once (dirname ( __FILE__ ) . '/../../application/service/ConfigServiceImpl.php');

$action = isset ( $_GET ['action'] ) ? $_GET ['action'] : '';
$usuarioID = isset ( $_SESSION ['id'] ) ? $_SESSION ['id'] : '';

$configService = ConfigServiceImpl::getInstance ();

switch ($action) {

	case 'getDominios' :

		exit ( json_encode ( $configService->retrieveDominio () ) );
		break;

	case 'getBins' :

		exit ( json_encode ( $configService->retrieveBin () ) );
		break;

	case 'add' :

		$dominioID = isset ( $_POST ['dominio'] ) ? $_POST ['dominio'] : '';
		$descricao = isset ( $_POST ['descricao'] ) ? $_POST ['descricao'] : '';
		$binID = isset ( $_POST ['bin'] ) ? $_POST ['bin'] : '';
		$arquivo = isset ( $_POST ['arquivo'] ) ? $_POST ['arquivo'] : '';

		// Getting the service
		$service = ScriptServiceImpl::getInstance ();
		// Calling the service to save a new script
		exit ( $service->save ( $descricao, $binID, $arquivo, $dominioID ) );
		break;

	case 'update' :

		$id = $_REQUEST ['id'];
		$value = $_REQUEST ['value'];
		$column = $_REQUEST ['columnName'];
		$columnPosition = $_REQUEST ['columnPosition'];
		$columnId = $_REQUEST ['columnId'];
		$rowId = $_REQUEST ['rowId'];

		// Getting the service
		$service = ScriptServiceImpl::getInstance ();
		// Calling the service to save a new script
		exit ( $service->update ( $id, $column, $value ) );
		break;

	case 'del' :

	default :

		$template = new TemplateHandler ();
		$template->set_css_files ( array ('datatable.css' ) );
		$template->set_js_files ( array ('jquery-1.7.1.js', 'jquery.dataTables.js',
'jquery.dataTables.editable.js', 'jquery.jeditable.js', '/script/script.js', '/script/datatable.js' ) );
		$template->set_content_file ( 'script/view.php' );
		$template->set_authenticated_only ( true );
		$template->show ();

}
```

```
<?php if (!Auth::check()) { exit('Acesso negado'); } ?>

<?php
require_once (dirname ( __FILE__ ) . '/../../application/config/general.php');
?>
<fieldset>
	<div>
		<form id="addScriptForm" class="form-inline"
			action="<?php echo SCRIPT_URI ?>?action=add" method="post">
			<div id="status" class="hide">
				<button type="button" class="close" data-dismiss="alert">×</button>
				<h4>Erro!</h4>
				Verifique os campos obrigatórios
			</div>
			<p>
				<br /> <label for="dominio">Dominio:</label> <select id="dominio"
					name="dominio"
					data-bind="options: dominios, value: selectedDominio,
optionsText: 'nome', optionsValue: 'id'">
				</select> <label for="bin">Bin:</label> <select id="bin" name="bin"
					data-bind="options: bins, value: selectedBin, optionsText:
'path', optionsValue: 'id'">
				</select> <label for="arquivo">Arquivo:*</label> <input type="text"
					id="arquivo" name="arquivo" size="40" /> <label
for="descricao">Descrição:*</label>
				<input type="text" id="descricao" name="descricao" size="30" /> <input
					type="submit" value="Salvar" class="btn" /> <input type="reset"
					value="Limpar" id="clearButton" class="btn" /> <em>* campos
					obrigatórios</em>
			</p>
		</form>
	</div>
</fieldset>
<br />
<fieldset>
	<table class="display" id="datatable" style="text-align: center;">
		<thead>
			<tr>
				<th width="5%">ID</th>
				<th width="10%">Domínio</th>
				<th width="20%">Descrição</th>
				<th width="15%">Bin</th>
				<th width="15%">Arquivo</th>
				<th width="15%">Created</th>
			</tr>
		</thead>
		<tbody>
			<tr>
				<td colspan="5" class="dataTables_empty">Carregando</td>
			</tr>
		</tbody>
		<tfoot>
			<tr>
				<th></th>
				<th></th>
				<th></th>
				<th></th>
				<th></th>
				<th></th>
			</tr>
		</tfoot>
	</table>
</fieldset>
```

Monitoramento de servidores com scripts

Andre Felipe Durieux

Departamento de Informática e Estatística - Universidade Federal de Santa Catarina (UFSC)

Caixa Postal 476 – 88040-900, Florianópolis – SC – Brasil

durieux@inf.ufsc.br

Abstract. *The monitor of networks, applications and servers is becoming very important nowadays for the any organization that wishes to offer always better services and products. Those organizations need their products or services to stay online and working for the most possible time, not to say always. Such real-time monitoring allows the system administrators to obtain all kind of informations about any device they may choose to monitor in a very fast, trusty and easy way. It helps a lot when the time to make a decision come.*

Those informations about the health of the environment can also help them to avoid bigger problems, which is very important to maintain the environment stability as well.

Resumo. *O monitoramento de redes, aplicações, servidores, ou seja, dispositivos importantes conectados à rede vem se tornando de suma importância nos dias de hoje para a gestão da tecnologia da informação. Organizações que oferecem serviços através da rede, necessitam que seus sistemas permaneçam funcionando. Este monitoramento, em tempo real, permite obter as informações necessárias sobre estes equipamentos de modo rápido, sintético, preciso e confiável, facilitando a tomada de decisão do gestor no momento do planejamento, adequação, suporte e expansabilidade da infra-estrutura organizacional.*

1. Introdução

O monitoramento de redes, aplicações, servidores, ou seja, dispositivos importantes conectados à rede vem se tornando de suma importância nos dias de hoje para a gestão da tecnologia da informação. Organizações que oferecem serviços através da rede, necessitam que seus sistemas permaneçam funcionando. Logo, a necessidade por maneiras de se tentar garantir o máximo de disponibilidade para o ambiente vem crescendo consideravelmente.

Este monitoramento, em tempo real, permite obter as informações necessárias sobre estes equipamentos de modo rápido, sintético, preciso e confiável, facilitando a tomada de decisão do gestor no momento do planejamento, adequação, suporte e expansabilidade da infra-estrutura organizacional.

Informações em tempo real sobre o que está acontecendo no ambiente da empresa permitem que seja possível evitar problema maiores, visando garantir uma estrutura estável para a organização.

2. Conceitos Básicos

Com o crescimento do uso de serviços online, o monitoramento de servidores e das aplicações por trás torna-se cada vez mais necessário e importante. Neste capítulo serão abordados alguns tipos de servidores, de bancos de dados e de aplicações que serão utilizados no desenvolvimento da aplicação e que funcionam como base para várias empresas hoje em dia.

O Java é uma das tecnologias mais utilizadas atualmente para o desenvolvimento de serviços e aplicações. Começou a ser criada em 1991 com o nome de Green Project, mas sem muito sucesso. Em 1995 foi lançada pela SUN oficialmente como linguagem para a construção de aplicativos, já com o nome Java. Desde então, o seu uso e desenvolvimento vem crescendo bastante. Várias empresas a utilizam para desenvolver seus produtos e serviços. Em função disso, o monitoramento da plataforma Java é importante para garantir a estabilidade destes produtos e serviços, oferecidos através da internet. Uma das maneiras de se monitorar a JVM é através dos Mbeans.

O Linux é um sistema operacional desenvolvido e mantido por Linus Torvalds. Adota a licença GPL, isso quer dizer que todos podem usá-lo e redistribuí-lo nos termos da licença. Em qualquer uma de suas distribuições, é mais utilizado como servidor. Entretanto, sua utilização como desktop vem crescendo muito rapidamente, muito em função da liberdade que este sistema operacional oferece ao usuário.

Já quando falamos de banco de dados, não podemos deixar de comentar sobre Oracle e MySQL. Este último, recentemente adquirido pela Oracle, ainda é um dos SGBD opensource mais utilizado mundialmente. Extremamente fácil de usar, está na versão 5.5.14. Já o Oracle, é um dos mais robustos, se não o mais robusto entre todos eles. Muitas organizações, de todo o mundo, o utilizam como banco de dados.

3. Ferramentas de Monitoramento

Atualmente são muitas as ferramentas de monitoramento de redes, servidores, ou seja, recursos computacionais. Três destas, todas opensource, são brevemente descritas abaixo.

O Nagios é uma aplicação de monitoramento de redes licenciado pelo sistema GPL. Tem como objetivo monitorar equipamentos como: hosts, switches, roteadores, aplicações, carga do processador, espaço livre em disco, etc. É capaz de realizar notificações quando algum serviço e/ou equipamentos estiverem indisponíveis. Estas

notificações podem ser enviadas por e-mail, sms ou qualquer outro serviço definido pelo usuário através de plugin.

O OpenNMS é um projeto opensource focado na criação de uma plataforma de gerência de rede voltada principalmente para camada de aplicação. Este software de gerenciamento de redes é capaz de fornecer uma série de métricas que diretores, gerentes e administradores de rede podem utilizar para medir a qualidade, tempo de disponibilidade de serviços. Sua principal característica é o auto descobrimento e justamente por isso é extremamente fácil de ser configurado. Com apenas alguns cliques é possível iniciar o monitoramento de uma rede.

Esta ferramenta tem como outras características a possibilidade de se configurar a geração de alertas e a geração de relatórios sobre todos os dados analisados.

O Cacti é uma ferramenta com licença GPL e permite o monitoramento de informações como: tráfego na rede, uso de memória, espaço em disco, além de dispositivos como switches e roteadores.

4. Desenvolvimento da Aplicação

Neste trabalho foi desenvolvida uma aplicação capaz de monitorar recursos de servidores Linux e Windows, fornecer informações e estatísticas de bancos de dados como MySQL e Oracle, fornecer informações sobre a máquina virtual Java e aplicações que estejam rodando sobre a JVM. Na verdade, trata-se de uma ferramenta totalmente baseada em scripts e portanto capaz de monitorar praticamente qualquer informação que scripts em várias linguagens de programação forem capazes de coletar. Esta aplicação possui uma interface onde o usuário pode visualizar todos os monitoramentos configurados para os servidores do seu ambiente. A idéia foi centralizar, em um único local, todas as informações sobre todos os domínios configurados.

Dentro da aplicação desenvolvida, os scripts tem um papel muito importante. Estas possibilidades tornam a ferramenta extremamente útil para administradores de sistemas. Vale lembrar que vários scripts já desenvolvidos são oferecidos pela aplicação, categorizados por domínios: Linux, Windows, a máquina virtual Java, MySQL e Oracle. Todo script cadastrado na aplicação pertence a um domínio. Entretanto, não estão cadastrados ainda, ou seja, estão pronto para serem usados. Para isso, precisam ser adicionados à aplicação, através da interface do módulo dos scripts, onde é possível cadastrar um novo script.

O usuário da aplicação pode ter em seu ambiente mais de um servidor e cadastrando cada um deles, torna-se mais fácil a identificação, na aplicação, de quais servidores estão sendo monitorados.

O cadastro de um servidor seria o segundo passo do usuário. Podemos ver na figura 16, a interface de cadastro de um servidor. Neste caso, é possível ver também que já existem dois servidores cadastrados na aplicação.

Um monitoramento pode ser descrito um comando que será executado pelo serviço responsável pela execução dos monitoramentos.

Para cadastrar um monitoramento, é necessário escolher um script que já tenha sido cadastrado pelo usuário. Então, devemos adicionar parâmetros para o script selecionado, caso seja necessário. Os parâmetros são configurados para o monitoramento e não para o script porque o usuário pode desenvolver um script genérico, que pode ser usado no monitoramento de mais de um servidor. Desta maneira, não será preciso cadastrar o mesmo script mais de uma vez se quisermos monitorar um mesmo recurso de servidores diferentes. Assim, é preciso apenas passar os parâmetros para o monitoramento.

Uma análise seria o último passo para a uma configuração completa de um monitoramento. Vale ressaltar que a configuração de uma análise não é necessária para os monitoramentos sejam executados. Entretanto, se o usuário quiser ser notificado caso algum monitoramento recupere um dado que ele considere fora do padrão, uma análise deve ser configurada.

O módulo de logging da aplicação utiliza uma terceira aplicação chamada Log.io para a visualização dos logs da aplicação. Trata-se de uma ferramenta de visualização de logs no browser em tempo real. A grande vantagem desta ferramenta é a capacidade de centralização de vários logs de diferentes máquinas. Além disso, oferece a possibilidade de busca por expressão regular nos logs, facilitando e agilizando o encontro de mensagens específicas. A figura 19 mostra a interface do módulo logging da aplicação.

Figura 19 – Interface do módulo logging

O Dashboard contempla a apresentação de todos as informações coletadas pelos monitoramentos que estejam cadastrados e ativos. Estas informações, dependendo do tipo de visualização configurada no monitoramento, são apresentadas para o usuário no dashboard.

5. Contribuições

Ferramenta / Característica	Nagios	OpenNMS	Cacti	TCC
Mapas	Sim	Sim	Sim, via plugin	Não
Agentes	Sim	Não	Não	Não
Relatórios	Sim, via plugin	Sim	Sim	Não
Alertas	Sim	Sim	Sim	Sim

A tabela acima mostra uma breve comparação entre algumas características das 3 ferramentas apresentadas no capítulo três e a aplicação proposta neste trabalho.

O que podemos perceber é que a aplicação proposta neste trabaho tem suas diferenças em comparação às outras ferramentas descritas. No caso dos agentes, em particular, o fato de não precisarmos configurar agentes em cada uma das máquinas que queremos monitorar, diferentemente do Nagios, é uma grande vantagem da minha proposta.

Já uma grande vantagem do Nagios, além de basear-se em scripts, é a grande comunidade que colobora com novos scripts e os disponibiliza para todos os usuários desta ferramenta.

Neste sentido, a aplicação proposta é semelhante ao Nagios, exceto pela comunidade, que inexiste até o momento. A principal vantagem da aplicação é o fato de basear-se em scripts para a execução dos monitoramentos. Estes scripts podem ser desenvolvidos em qualquer linguagem de programação suportada pelo Linux. Vale lembrar que uma das premissas de utilização desta ferramenta é que ela foi desenvolvida para rodar sobre distribuições Linux apenas, o que proporciona tamanha flexibilidade.

6. Conclusão

Este artigo apresentou os elementos básicos que fazem parte da aplicação proposta. Cada um dos módulos foi descrito. Falamos de alguns protocolos utilizados pelos scripts já desenvolvidos e oferecidos pela aplicação.

Também foram abordadas algumas ferramentas de monitoramento, com foco nas opensource. Depois, a proposta da aplicação foi apresentada, uma ferramenta de monitoramento de recursos computacionais de servidores Linux, Windows, máquina virtual do Java, bancos de dados, entre outros, baseada em scripts.

Referências

Sullins, B., Whipple, M. "JMX in Action". Manning Publications, 2003.

Lindfors, J., Fleury, M. "JMX: Managing J2EE with Java Management Extensions". Sams Publishing, 2002.

Tanenbaum, A. “Computer Networks, Fourth Edition”. Prentice Hall, 2003.

Mauro, D., Schmidt, K.. “Essential SNMP”. O'Reilly, 2001.

Kurose, J. e Ross, K. “Redes de computadores e a internet: uma abordagem top-down”.

Editora Addison Wesley. 3ªed., São Paulo. 2006

Melchiors, C.“Raciocínio Baseado em Casos Aplicado ao Gerenciamento de Falhas”.

151f. Tese de Mestrado em Ciência da Computação. Universidade Federal do Rio

Grande do Sul. UFRGS. Porto Alegre. 2003.

Oliveira, F. “Gerenciamento de Redes de Computadores com o uso do raciocínio

baseado em casos e ferramentas auxiliares”. 148f. Tese de Doutorado. Universidade

Federal do Rio de Janeiro. UFRJ. Rio de Janeiro. 2007

http://www.nagios.org/

Acesso em julho de 2011.

http://www.opennms.org/

Acesso em julho de 2011.

http://www.ibm.com/developerworks/websphere/library/techarticles/0304_polozoff/polozoff.html Acesso em maio de 2011.

http://www.manageengine.com/products/applications_manager/Applications-performance-management.pdf Acesso em maio de 2011.