

UNIVERSIDADE REGIONAL DE BLUMENAU
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

**SISTEMA DE WEB SERVICES PARA INVENTÁRIO DE
ESTAÇÕES EM REDE**

CHARLES BAMBINETI

BLUMENAU
2008

2008/1-07

CHARLES BAMBINETI

SISTEMA DE WEB SERVICE PARA INVENTÁRIO DE ESTAÇÕES EM REDE

Trabalho de Conclusão de Curso submetido à
Universidade Regional de Blumenau para a
obtenção dos créditos na disciplina Trabalho
de Conclusão de Curso II do curso de Ciências
da Computação — Bacharelado.

Prof. Francisco Adell Péricas , Mestre - Orientador

**BLUMENAU
2008**

2008/1-07

SISTEMA DE WEB SERVICE PARA INVENTÁRIO DE ESTAÇÕES EM REDE

Por

CHARLES BAMBINETI

Trabalho aprovado para obtenção dos créditos
na disciplina de Trabalho de Conclusão de
Curso II, pela banca examinadora formada
por:

Presidente: _____
Prof. Francisco Adell Péricas , Mestre – Orientador, FURB

Membro: _____
Prof. Paulo Fernando da Silva, Mestre – FURB

Membro: _____
Prof. Sérgio Stringari, Mestre – FURB

Blumenau, 15 julho 2008

Dedico este trabalho a minha família, meus amigos e em especial a minha namorada Gabi pelo incentivo e compreensão durante a realização deste.

"Os livros não mudam o mundo, quem muda o mundo são as pessoas. Os livros só mudam as pessoas."

Mario Quintana

RESUMO

Este trabalho visa apresentar a tecnologia de *Web Services* com uma alternativa para o desenvolvimento de um sistema para inventário de estações de computadores em rede. Para isto, será utilizado a linguagem de programação Python para desenvolver um servidor o *WebService* e dois sistemas clientes, um responsável por alimentar o servidor com os dados da estações e outro para consultas e gerência destes dados.

Palavras-chave: *Web Services*. Gerência de redes. Python.

ABSTRACT

This work aims to expose the technology of Web Services with an alternative for developing a system for managing stations in a network of computers, will use the programming language Python to develop a server WebService and the two systems clients, one responsible for the food server with data from stations and other for consultation and management of these data.

Key-words: *Web Services. Network Manager.* Python.

LISTA DE ILUSTRAÇÕES

Figura 1 – Cliente acessando <i>Web Service</i>	14
Quadro 1 – Exemplo de arquivo XML.....	15
Figura 2 – Envelope Soap.....	16
Figura 3 – Esquema de gerência de rede	18
Quadro 2 – Exemplo de código em Python.....	19
Figura 4 – Diagrama de atividades do Cliente	24
Figura 5 – Diagrama de atividades do Servidor	25
Figura 6 – Modelo de dados cliente.....	26
Figura 7 – Modelo de dados servidor	27
Figura 8 – Diagrama de classes do cliente	28
Figura 9 – Diagrama de classes do servidor.....	29
Quadro 3 – Código <i>Web Service</i>	31
Quadro 4 – Código cliente conecta <i>Web Service</i>	32
Quadro 5 – Métodos detectaPacotes.....	33
Quadro 6 – Sintaxe da ferramenta wscmd.....	34
Figura 10 – Saída do wscmd listando pacotes Windows.....	34
Figura 11 – Saída do wscmd listando pacotes Linux.....	35
Quadro 7 – Instruções de ajuda do wscmd	35
Quadro 8 – Instruções de ajuda do módulo grupos do wscmd	36
Quadro 9 – Lista XML de hardware.....	49

LISTA DE SIGLAS

HTTP - *HyperText Transfer Protocol*

IP - *Internet Protocol*

NAT - *Network Address Translation*

SMTP - *Simple Mail Transfer Protocol*

SNMP - *Simple Network Management Protocol*

SOA - *Service Oriented Architecture*

SOAP - *Simple Object Access Protocol*

WSDL - *Web Services Description Language*

XML - *eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO.....	10
1.1 OBJETIVOS DO TRABALHO	12
1.2 ESTRUTURA DO TRABALHO	12
2 FUNDAMENTAÇÃO TEÓRICA	13
2.1 <i>WEB SERVICES</i>	13
2.1.1 XML	15
2.1.2 SOAP	16
2.2 GERÊNCIA DE REDES	17
2.3 PYTHON	18
2.4 TRABALHOS CORRELATOS	20
3 DESENVOLVIMENTO.....	22
3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO.....	22
3.2 ESPECIFICAÇÃO	22
3.2.1 Diagrama de Atividades.....	23
3.2.2 Modelo conceitual da base de dados	26
3.2.3 Diagramas de Classes.....	28
3.3 IMPLEMENTAÇÃO	30
3.3.1 Técnicas e ferramentas utilizadas.....	30
3.3.2 Implementação do sistema	30
3.3.2.1 Diferenças entre sistemas operacionais	32
3.3.3 Operacionalidade do sistema.	34
3.4 RESULTADOS E DISCUSSÃO	36
4 CONCLUSÕES.....	38
4.1 EXTENSÕES	38
REFERÊNCIAS BIBLIOGRÁFICAS	40
ANEXO A.....	42

1 INTRODUÇÃO

Com o constante aumento das redes de computadores, fica imprescindível a sua administração. Isso ocorre porque cada vez mais o computador vem tornando-se uma ferramenta essencial em todos os setores das empresas, sejam elas grandes corporações ou até mesmo empresas de médio e pequeno porte.

As redes estão ficando cada vez maiores, atingindo mais pessoas, transformando-se em algo mais heterogêneo, pois possuem uma mesclagem de tecnologias e de fornecedores. As tecnologias estão em constante desenvolvimento, exigindo assim mais recursos dos equipamentos, um melhor desempenho, por conseguinte uma capacidade maior de administração dos mesmos.

Um fator que contribui acentuadamente para que o computador passe a ser uma ferramenta cada vez mais utilizada por um maior número de pessoas é o uso da rede mundial de computadores, a internet, que passa por um processo de disseminação acelerado. Em função deste crescimento, surgem necessidades quanto ao controle de diversas tarefas que devem ser executadas.

A eficiência e até a sobrevivência de empresas, desde as grandes corporações ou até as empresas de médio e pequeno porte, pode ser determinada pelo bom funcionamento de seus recursos de informática. O constante aumento do tamanho das redes de computadores provoca um aumento da complexidade e diversidade de tecnologias usadas.

Dentro deste contexto, ferramentas que ajudem os administradores de redes no controle de seus equipamentos, seja na coleta de informações, no monitoramento ou na execução de tarefas automatizadas, são de vital importância para que os administradores de rede desempenhem um bom trabalho.

Uma empresa pode ter vários departamentos, cada um com necessidades específicas, podendo haver grande diferença em relação às necessidades dos usuários, de software instalado aos recursos dos equipamentos, aos sistemas operacionais e até à arquitetura de hardware. Neste contexto, quanto maior e mais heterogênea uma rede, maior a necessidade de uma ferramenta que ajude a gerenciar estes equipamentos individualmente ou agrupando-os em setores, salas, modelos de equipamento, entre outros, conforme a necessidade do administrador da rede.

Este documento apresenta o desenvolvimento de um software que possibilita o inventário de estações em uma rede heterogênea, utilizando-se da tecnologia de *Web Services*, que irá coletar, processar, persistir e disponibilizar informações sobre as estações da rede.

Inicialmente foi desenvolvido a coleta de informações das estações como memória, processador, disco rígidos, dispositivos de rede, carga do sistema, softwares instalados, sistema operacional e configurações, entre outros.

Foram feitos relatórios a partir destas informações. Para isso, os dados coletados foram persistidos no banco de dados relacional Postgresql no servidor, podendo desta forma armazenar um grande histórico sobre a rede, tornando o sistema um importante software para o planejamento dos recursos de informática. Nas estações foi usado o banco de dados SQLite para persistência temporária, até que a informação seja enviada para o servidor,

Foram implementados alarmes quando uma estação atingir determinada condição, como por exemplo, espaço em disco livre estiver muito baixo, ou ainda alguma modificação nos recursos da estação e também caso a estação fique um determinado tempo sem se comunicar. Estas funções são importantes para detectar e antever problemas no parque de equipamentos.

Utilizou-se a linguagem de programação Python para o desenvolvimento do sistema. Segundo Catunda (2001, p. 6), a linguagem Python é fácil e poderosa, possui mecanismos eficientes e com um bom nível de abstração para manipulação de estruturas de dados. É uma linguagem interativa, interpretada e orientada a objetos, com uma sintaxe elegante e tipagem dinâmica. Pode ser facilmente estendida com novas funções e novos tipos de dados implementados em C ou C++.

Foi utilizado a solução de *Web Services* pela sua flexibilidade e interoperabilidade, utilizando o protocolo *Simple Object Access Protocol* (SOAP) através *HyperText Transfer Protocol* (HTTP) para transporte, visando facilitar a comunicação entre a estação e o servidor, evitando barreiras de topologias e *firewall*, possibilitando que o servidor esteja até mesmo em outro ponto da internet, fora da rede local.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho foi desenvolver um sistema *Web Service* para inventário de estações.

Os objetivos específicos do trabalho são:

- a) criar um *Web Service* responsável pela centralização do gerenciamento;
- b) criar um processo que ficará residente nas estações será para contactar com o *Web Service*, enviando as informações da estação;
- c) persistir os dados em banco de dados relacional;
- d) gerar relatórios atuais e de histórico;
- e) disparar alertas caso alguma condição seja atendida;
- f) gerar inventário do parque de equipamentos;
- g) facilitar o agrupamento de estações por setores, salas, modelos de equipamento entre outros, conforme a necessidade do administrador de rede.

1.2 ESTRUTURA DO TRABALHO

É exposto neste trabalho o funcionamento de tecnologia de *Web Services* e também os padrões *Simple Object Access Protocol* (SOAP) e *eXtensible Markup Language* (XML) que integram parte do *Web Services*, conceito de gerencia de redes.

Além das ferramentas utilizadas, a linguagem de programação Python e os bancos de dados relacionais Postgresql e Sqlite são apresentadas.

Na segunda parte deste documento é apresentado o desenvolvimento do *Web Services* e dos processos residentes nas estações responsáveis por concentrar e encaminhar as informações de gerenciamento.

Ao final do documento é apresentada uma conclusão focado nos resultados obtidos no trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Nas seções seguintes são apresentados conceitos, tecnologias, padrões e ferramentas que são essenciais para o entendimento do trabalho.

2.1 WEB SERVICES

World Wide Web Consortium (W3C) define *Web Service* da seguinte forma: “Um sistema *Web Service* é implementado para fornecer interoperabilidade entre *hosts* em uma determinada rede. Possui uma interface descrita no formato conhecido como *Web Services Description Language* (WSDL). Sistemas interagem com *Web Services* usando mensagens SOAP, através do *HyperText Transfer Protocol* (HTTP) com uma serialização XML em conjunto com outros padrões relacionados”.

Para Snell, Tidwell, Kulchenko (2002, p. 1), um *Web Service* é uma interface acessível pela rede de funcionalidades de aplicações, construída usando padrões da internet. Em outras palavras, se um aplicativo pode ser acessado através de uma rede utilizando uma combinação de protocolos como o HTTP, XML, *Simple Mail Transfer Protocol* (SMTP), ou Jabber, então é um *Web Services*. Apesar de todos os meios de conexão e protocolos em torno do *Web Service*, ele é realmente muito simples. *Web Service* não tem nenhuma novidade. Na verdade, representa a evolução de princípios empregados a muitos anos na internet.

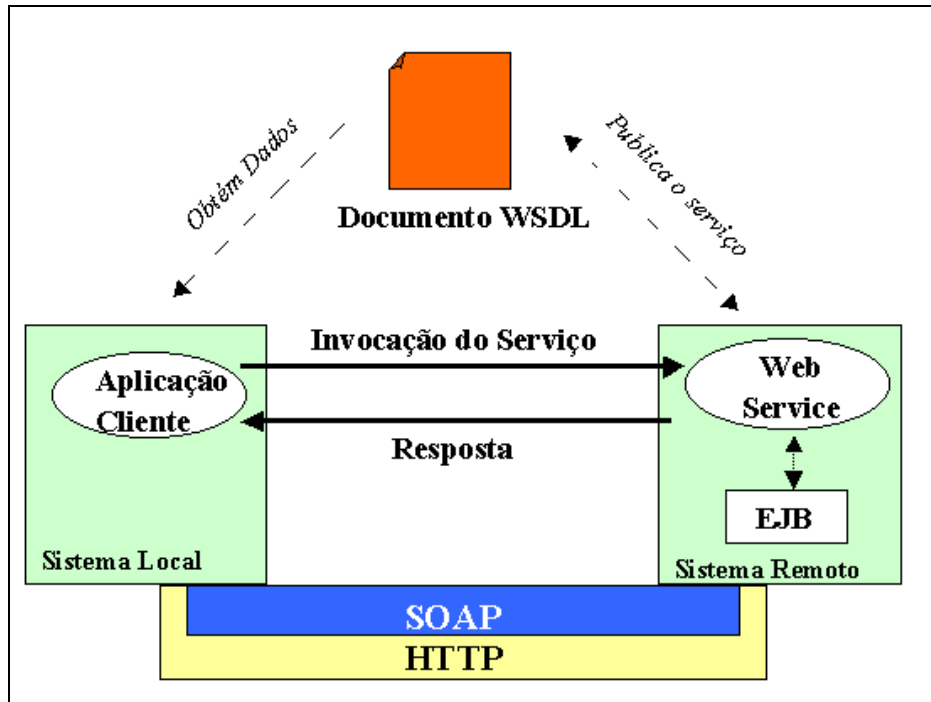
Segundo Singh (2006, p. 4), como qualquer aplicativo, os aplicativos baseados em *Web Services* podem executar uma gama variada de funções. Alguns podem lidar somente com requisições simples para informações, enquanto outros podem implementar processos complexos de negócios e interação.

Interoperabilidade é um dos benefícios chave ganhos com a implementação de *Web Service*. Por causa da abstração provida pelos padrões, não importa saber se a aplicação servidora é escrito em Java e o cliente é escrito em C++, ou a aplicação servidora implantada em um Unix enquanto a aplicação cliente executa no Windows.

Singh (2006, p. 5), cita que talvez a razão primordial para o uso crescente dos *Web Services*, seja que eles promovam a interoperabilidade através de diferentes plataformas, sistemas e linguagens, além disso, eles reduzem custos operacionais permitindo que as

organizações se estendam e reutilizem sua funcionalidade de sistemas existentes.

Menéndez (2002, p. 21) define que um *Web Service* é uma aplicação que aceita solicitações de outros sistemas através da Internet, sempre baseado em padrões para facilitar a comunicação entre as máquinas. Esses padrões são baseados em XML, que será visto posteriormente.



Fonte: Cunha (2002).

Figura 1 – Cliente acessando *Web Service*

Conforme a Figura 1, Cunha (2002) explica que a aplicação cliente pode acessar diretamente um *Web Service*, seja ele um *Enterprise Java Bean* (EJB) ou mesmo uma aplicação Java. O *Web Service* processa a chamada (possivelmente acessando o seu Banco de Dados) e retorna uma resposta ao cliente. A aplicação cliente, após localizar o serviço remoto (definido por um documento WSDL), invoca os seus serviços através de RPC. O *Web Service* recebe a chamada, a processa e envia uma resposta. É válido lembrar, que ambos (cliente e *Web Service*) "conversam" usando SOAP em cima de HTTP.

Golçalves (2005, p. 22) fala da utilização de *Web Services* em gerência de redes, ainda dentro do universo *Web* e, particularmente, da emergente família de tecnologias XML, mais recentemente uma nova tecnologia surgiu com a promessa de abrir um novo capítulo no gerenciamento de redes: os *Web Services*. Criada com o objetivo de permitir que aplicações sejam acessadas através de protocolos *Web* ao invés de protocolos distribuídos baseados em objetos, a tecnologia *Web Service* tem chamado a atenção de diversas empresas líderes de

mercado que tem anunciado estratégias de introduzir o suporte a *Web Service* em seus produtos.

2.1.1 XML

Segundo Menéndez (2002, p. 21), XML é um padrão da indústria de informática que tem como maior objetivo o intercâmbio de dados. Ele permite que sistemas possam trocar informações de uma forma mais abrangente que arquivos texto, já que podemos dar semântica aos dados que estão sendo manipulados.

De acordo Singh (2006, p. 34), XML é uma linguagem de marcação simples, flexível, e baseado em textos. Os dados XML são marcados utilizando *tags* anexadas entre colchetes angulares. As *tags* contêm o significado de dados que elas marcam. Tal marcação permite que sistemas diferentes facilmente façam o intercâmbio de dados uns com os outros.

Menéndez (2002, p. 21) explica que, da mesma forma que HTML (*Hyper Text Markup Language*), o XML se utiliza de etiquetas para representar os dados; porém existem várias diferenças entre esses dois padrões. Enquanto HTML especifica como os dados devem ser exibidos, XML se preocupa com o significado dos dados armazenados. A seguir há um exemplo de um documento XML que representa um curso e dois alunos com a sua média.

```
<curso>
  <aluno>
    <nome>Danielle</nome>
    <MGP>8.0</MGP>
  </aluno>
  <aluno>
    <nome>Cinthia</nome>
    <MGP>6.5</MGP>
  </aluno>
</curso>
```

Fonte: Menendez (2002, p. 21).

Quadro 1 – Exemplo de arquivo XML

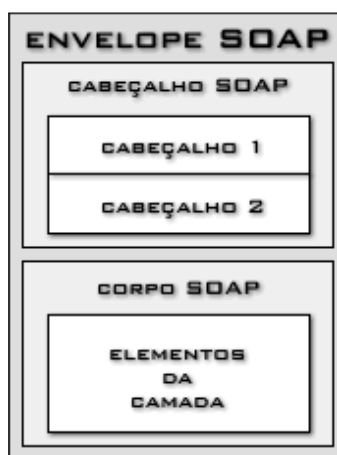
Pelo exemplo pode-se notar que as etiquetas que foram usadas indicam a estrutura e o conteúdo armazenado. Uma outra grande diferença entre XML e HTML é que as etiquetas XML são extensíveis, permitindo que documentos XML possam representar fielmente os dados por ele armazenados. Já o HTML possui um conjunto de etiquetas pré-definidas e fixas, não permitindo adicionar novas. Cada etiqueta do documento XML é chamado de *elemento*. Dessa forma, há vários elementos no XML mostrado no exemplo: curso, aluno, nome e média. Os elementos localizados dentro de outros elementos são chamados de sub-elementos.

2.1.2 SOAP

Para Pilgrim (2005, p. 153) o *Simple Object Access Protocol* (SOAP) é uma especificação complexa, e é de certa forma enganador dizer que o SOAP se resume à chamada de funções remotas. Algumas pessoas adicionariam que SOAP permite a transmissão assíncrona de mensagens em uma via, além de permitir serviços web orientados a documentos. E estas pessoas estariam corretas: o SOAP pode ser usado dessa e de muitas outras maneiras.

Singh (2006, p. 37), a XML soluciona a necessidade de uma linguagem comum, e o SOAP supre a necessidade de um formato de serviços de mensagens comum. O SOAP habilita que objetos não conhecidos uns dos outros se comuniquem; isto é, troquem mensagens. SOAP é um protocolo baseado em textos que utiliza um formato de codificação dados baseado na XML e HTTP/SMTP para transportar mensagens.

Reckziegel (2006a) esclarece o envelope SOAP é a parte obrigatória de uma mensagem SOAP. Ele funciona como um recipiente de todos os outros elementos da mensagem, possivelmente o cabeçalho e o corpo, assim como os *namespaces* de cada um. Assim como o nome e o endereço de uma carta entregue pelo correio, o envelope SOAP precisa das informações específicas do protocolo de transporte que está ligado a ele, com o intuito de garantir a chegada ao local certo. Especificamente no HTTP, há um cabeçalho que se chama SOAPAction, indicador do endereço de entrega da mensagem. Um dos principais motivos de implementar-se o cabeçalho desta maneira é porque administradores de sistemas podem configurar seus *firewalls* para filtrar as mensagens baseadas nas informações dos cabeçalhos, sem consultar o XML.



Fonte: Reckziegel (2006a).
Figura 2 – Envelope Soap

2.2 GERÊNCIA DE REDES

Kurose (2005, p. 572) diz que, nos primórdios das redes de computadores, quando elas eram artefatos de pesquisa e não uma infra-estrutura usada por milhões de pessoas diariamente, ‘gerenciamento de rede’ era algo que nunca se tinha ouvido falar. Se alguém descobrisse um problema de rede, poderia realizar alguns testes, como o *ping*, para localizar a fonte do problema. Com a internet pública e as intranets privadas cresceram e se transformaram de pequenas redes em grandes infra-estruturas globais, a necessidade de gerenciar mais sistematicamente a enorme quantidade de componentes de hardware e software dentre dessas redes também se tornou mais importante.

De acordo com Lopes, Sauvé, Nicoletti (2003, p. 4), o objetivo da Gerência de Redes é monitorar e controlar os elementos da rede (sejam eles físicos ou lógicos), assegurando certo nível de qualidade de serviço. Para realizar esta tarefa, os gerentes de redes são geralmente auxiliados por um sistema de gerência de redes. Um sistema de gerência de rede pode ser definido como uma coleção de ferramentas integradas para a monitoração e o controle da rede.

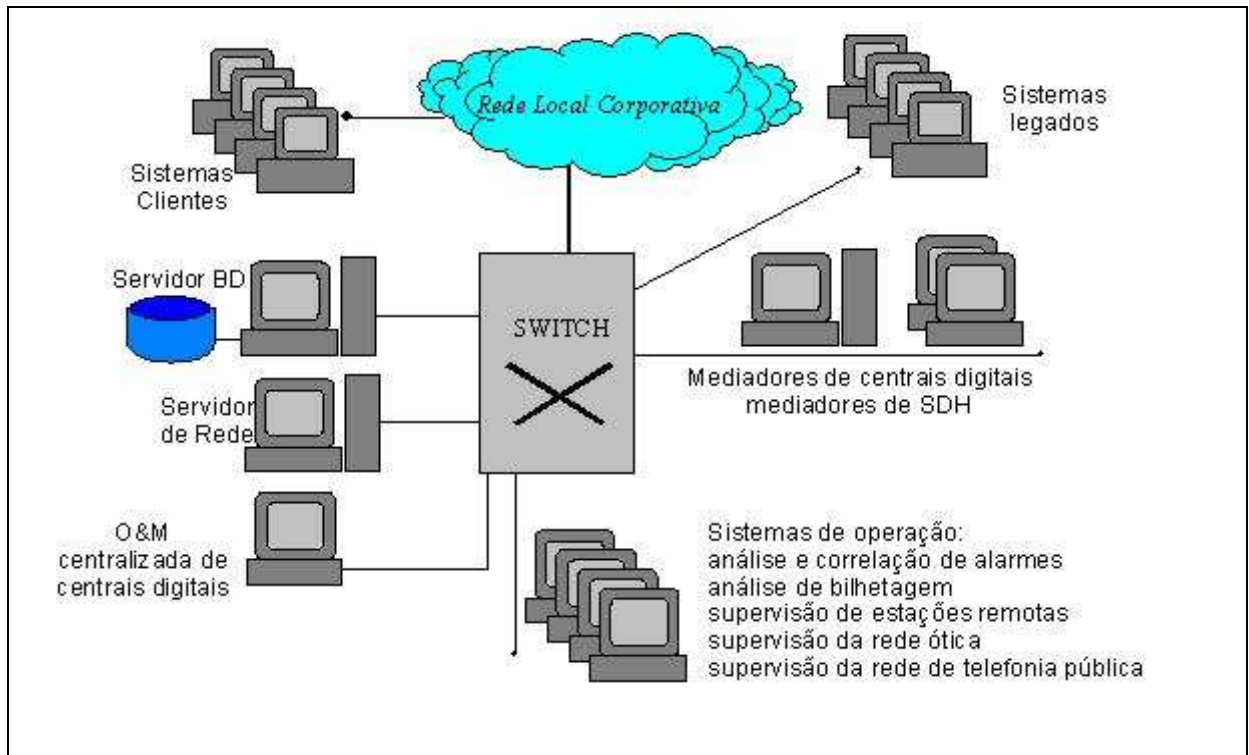
Segundo Kurose (2006, p. 575), gerenciamento de rede exige a capacidade de “monitorar, testar, consultar, configurar (..) e controlar” os componentes de hardware e software de uma rede. Como os dispositivos da rede são distribuídos, fazer isso exigirá, no mínimo, que o administrador possa coletar dados (por exemplo, para a finalidade de monitoração) de uma entidade remota e efetuar mudanças nessa entidade (por exemplo, controle).

De acordo Lopes, Sauvé, Nicoletti (2003, p. 6), um dos objetivos da gerência de redes é prevenir e solucionar problemas na rede. Não existe uma regra rígida sobre os profissionais que fazem parte desta equipe, porém, é comum que nesta equipe existam profissionais que executem quatro tarefas distintas: o pessoal do *help desk*, o operador da rede, a equipe de suporte técnico e o gerente da equipe.

O *help desk* atua no auxílio direto ao usuário, tirando dúvidas, auxiliando resolver problemas, por telefone ou remotamente, já a equipe de suporte técnico é que põe a mão na massa para solucionar os problemas, responsável pela configuração, operação e manutenção

da rede. O operador de sistema é o profissional encarregado de acompanhar os alarmes gerados pelo sistema de gerência, e o gerente da equipe administra os recursos, avaliando o desempenho de sua equipe, solicitando a compra de equipamentos, aplicações e outros recursos necessários, solicita treinamentos adequados para a equipe, etc.

Na figura 3 vê se um esquema representando a gerência de redes, onde os mais diversos tipos de equipamentos são gerenciados por um ou vários sistemas gerentes de redes



Fonte: Sortica (2008)

Figura 3 – Esquema de gerência de rede

2.3 PYTHON

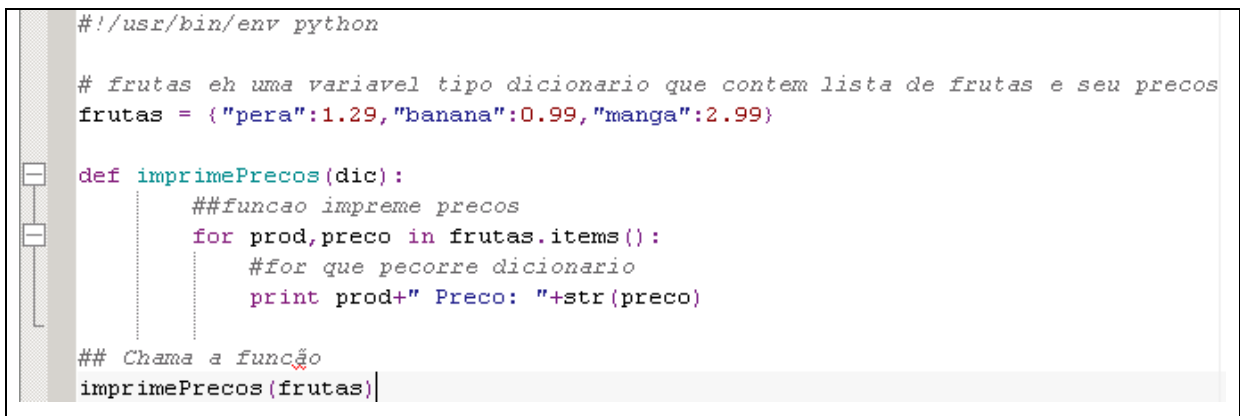
Segundo Lutz, Ascher (2007, p. 31), o Python é uma linguagem de propósito geral, freqüentemente aplicada em função de *script*. Ela é comumente definida como uma linguagem de *script* orientada a objetos – uma definição que combina suporte para POO com orientação global voltada para função de *script*. Como *script* tem significados diferentes, ferramenta de *shell*, linguagem de controle e de facilidade de uso, sendo a última a melhor maneira de pensar em Python como uma linguagem de fácil uso. O Python permite que os

programas sejam desenvolvidos mais rapidamente do que outras linguagens compiladas.

Segundo Catunda (2001, p. 6), a linguagem Python é fácil e possui mecanismos eficientes e com um bom nível de abstração para manipulação de estrutura de dados. É uma linguagem interativa, interpretada e orientada a objetos, com uma sintaxe elegante e tipagem dinâmica. Pode ser facilmente estendida com novas funções e novos tipos de dados implementados em C ou C++.

Segundo Lutz, Ascher (2007, p. 30), o Python implementa uma sintaxe deliberadamente simples e legível, em um modelo de programação altamente coerente, ele adota uma estratégia minimalista. Isso significa que, embora exista várias maneiras de executar uma tarefa, há apenas uma maneira óbvia e algumas alternativas menos óbvias.

Dois aspectos mais que se destacam em Python é delimitar os blocos usando indentação e tipagem dinâmica de variáveis com liberação automática de memória, como explica Catunda (2001, p. 9).



```
#!/usr/bin/env python

# frutas eh uma variavel tipo dicionario que contem lista de frutas e seu precos
frutas = {"pera":1.29,"banana":0.99,"manga":2.99}

def imprimePrecos(dic):
    ##funcao imprime precos
    for prod,preco in frutas.items():
        #for que percorre dicionario
        print prod+" Preço: "+str(preco)

## Chama a função
imprimePrecos(frutas)
```

Quadro 2 – Exemplo de código em Python

No quadro 2 se vê um pequeno exemplo de código em python, onde a primeira indentação define o bloco da função `imprimePrecos` e o segundo o bloco do `for`, podemos ver também que não precisamos definir os tipos das variáveis elas são tipados conforme o dado atribuído e que podemos começar a utilizar elas em qualquer parte do código sem precisar defini-las anteriormente.

O Python usa a indentação para demilitar os blocos de código. Não existe o conceito de abre “{“ e fecha “}”, ou “begin” e “end”. Quando se endenta um bloco de código, você define-se o grupo das estruturas de controle. Isso leva o programador a estar sempre organizado sob o ponto de vista da indentação do código.

Python é uma linguagem com tipagem dinâmica, portanto não precisa declarar qual o tipo da variável em questão. Dependendo do valor que se armazena na variável, esta altera seu

tipo automaticamente. Python possui mecanismo de contador de referência em suas variáveis. Isso quer dizer que não se precisa liberar memória das variáveis. Quando não existir mais referências para a variável em questão, esta será liberada.

Pilgrim (2005, p. 153) descreve o uso de SOAP em Python, onde SOAP permite simular a chamada de funções que retorna tipo de dados nativos, a ilusão é quase perfeita: você pode “chamar” uma função através de uma biblioteca SOAP, com a sintaxe padrão do Python para chamadas, e a função parecerá retornar objetos e valores do Python. Mas internamente, a biblioteca SOAP na verdade realiza uma complexa transação envolvendo múltiplos documentos XML e um servido remoto.

2.4 TRABALHOS CORRELATOS

Em Golçalvez (2005) é feito uma análise comparativa do uso de *Web Services* para gerência de redes e o protocolo (*Simple Network Management Protocol*) SNMP onde conclui que o modelo SNMP, amplamente utilizado no passado para o gerenciamento de redes de computadores, tem apresentado limitações frente à evolução destas redes nos aspectos de escalabilidade e eficiência. A simplicidade do SNMP, principal responsável pelo seu sucesso inicial, tem sido um dos fatores determinantes para a procura de tecnologias alternativas capazes de endereçar novos requisitos de gerenciamento como configuração e gerenciamento integrado. Por fim, o forte apelo comercial e a ampla difusão das tecnologias *Web* parecem definir a tendência a ser seguida: a utilização de tecnologias de uso geral em detrimento às de domínio específico.

A utilização de XML no gerenciamento de redes endereça algumas das dificuldades apresentadas pelo modelo SNMP no que tange a escalabilidade e eficiência. A insegurança inicial quanto ao possível aumento do *overhead* causado pela representação das informações de gerenciamento em XML tem sido superada pela utilização de métodos de compressão de dados. Estes mecanismos de compressão de informações colocam ainda XML em posição de destaque quando se trata da transferência de grandes massas de dados. Algumas questões, entretanto, estão fora do escopo de XML que não é capaz de solucionar problemas do SNMP que não estejam vinculados apenas à representação e manipulação de informações.

Web Service, como uma tecnologia baseada em XML, surge então para impulsionar o

uso de XML que agora não mais se restringe à representação de informações, mas também oferece uma arquitetura completa de processamento distribuído. O desenvolvimento do protótipo de gerenciamento de redes deixou claro a simplicidade e o grande potencial de *Web Services* aliada a outras tecnologias XML no gerenciamento de redes.

Em outro trabalho, Reis (2001) faz uma comparação entre o uso do protocolo SNMP e a abordagem de utilização de agentes móveis, onde algumas das conclusões é que, a abordagem alternativa gera maior trafico de dados e um aumento no tempo de resposta, e a principal vantagem da utilização de agentes móveis seria a flexibilidade proporcionada por esta abordagem.

Uma sistema que possui o mesmo propósito é o Cacic, *software* livre desenvolvido pelo governo federal, é capaz de fornecer um diagnóstico preciso do parque computacional e disponibilizar informações como o número de equipamentos e sua distribuição nos mais diversos órgãos, os tipos de softwares utilizados e licenciados, configurações de hardware, entre outras. Também pode fornecer informações patrimoniais e a localização física dos equipamentos, ampliando o controle do parque computacional e a segurança na rede, o Cacic foi desenvolvido usando as linguagens Php, Perl, Python, Delphi e banco de dados Mysql.

3 DESENVOLVIMENTO

O sistema desenvolvido neste trabalho é um *Web Service* que dá suporte à gerência de estações em rede, armazenando e fornecendo dados sobre estas estações sob a forma de chamadas remotas aos métodos do *Web Service* através do protocolo SOAP.

Nas estações é iniciada uma aplicação cliente responsável por catalogar e enviar informações sobre a estação ao servidor.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Abaixo são descritos os principais requisitos do sistema:

- a) criar um *Web Service* responsável pela centralização do gerenciamento do inventário (Requisito Funciona – RF);
- b) criar um cliente responsável por catalogar as informações da estação e enviar-las ao servidor (RF);
- c) persistir os dados em banco de dados relacional (Requisito não funcional – RNF);
- d) criar relatórios a partir desta informações (RF) ;
- e) disparar alertas caso alguma condição seja atendida (RF);
- f) gerar inventário do parque de equipamentos (RF);
- g) facilitar o agrupamento de estações por setores, salas, modelos de equipamento entre outros, conforme a necessidade do administrador de rede (RF);
- h) implementar o sistema utilizando a linguagem Python (RNF).

3.2 ESPECIFICAÇÃO

Utilizou-se para especificação deste sistema a orientação a objetos representado através da *Unified Modeling Language* (UML). Foi utilizada a ferramenta Enterprise Architect da Sparx System.

3.2.1 Diagrama de Atividades.

O diagrama de atividades é um diagrama definido pela UML, e representa os fluxos conduzidos por processamentos. É essencialmente um gráfico de fluxo, mostrando o fluxo de controle de uma atividade para outra. Comumente isso envolve a modelagem das etapas sequenciais em um processo computacional.

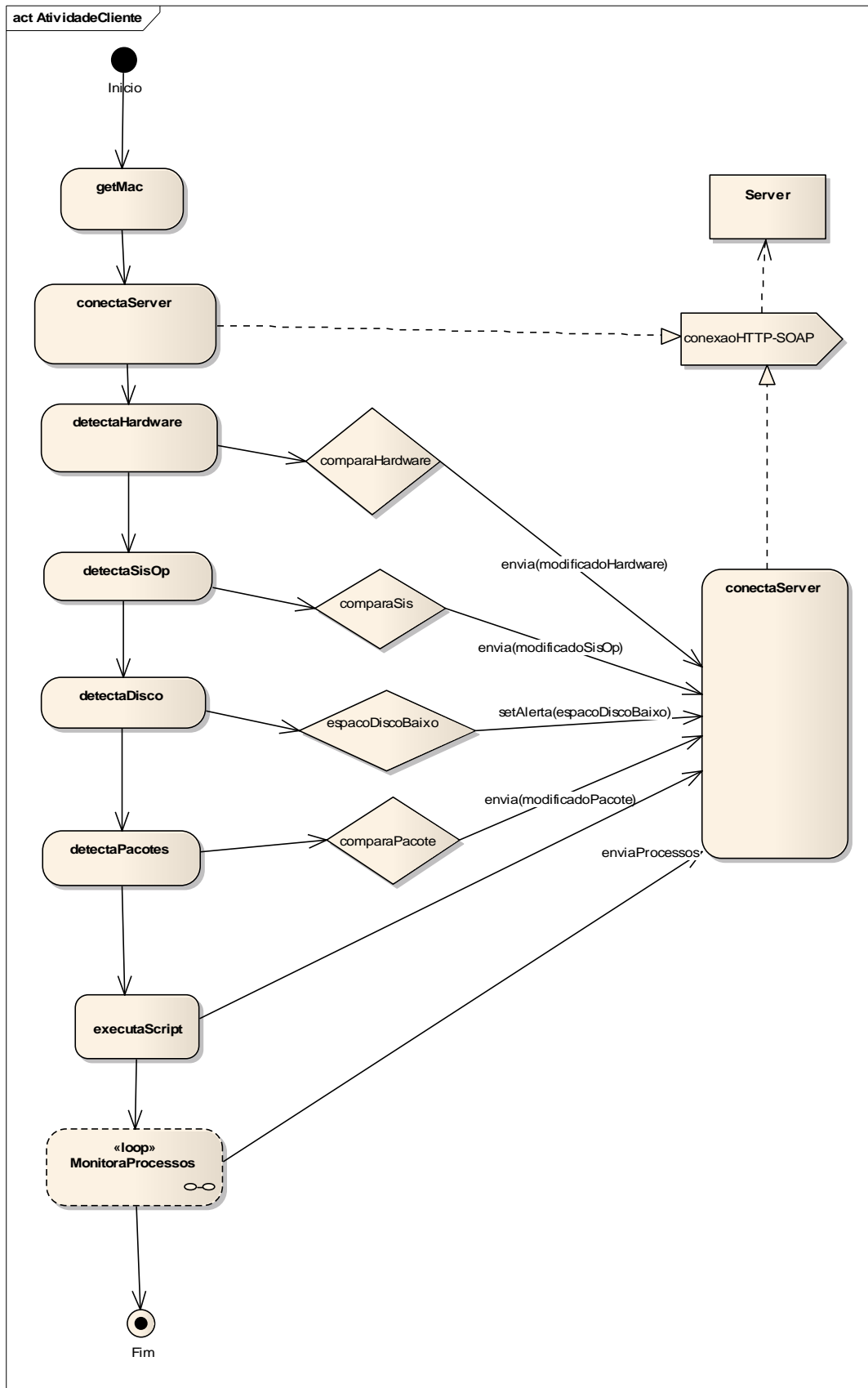


Figura 4 – Diagrama de atividades do Cliente

Na figura 4 demonstra-se o diagrama de atividades, onde se vêem todos os processos que o cliente realiza quando é iniciado, que são:

- a) pega seu número da interface de rede, que é único e por isto utilizado como o identificador da máquina;
- b) conecta o servidor avisando que ele esta ativo;
- c) detecta o hardware e verifica se houve alguma alteração: caso houver, envia aviso ao servidor, juntamente com a nova configuração de hardware ;
- d) detecta o sistema operacional e, se houver alteração, avisa ao servidor;
- e) verifica o espaço do disco principal e, caso o espaço livre estiver abaixo de valor mínimo definido, envia alerta ao servidor;
- f) verifica se houve instalação ou exclusão de algum pacote do sistema e envia alterações para o servidor caso houver;
- g) consulta o servidor se há algum *script* para ser executado;
- h) consulta servidor se está agendado o monitoramento da estação: caso esteja, fica ativo enviado o status dos processos em execução no sistema, assim como a utilização da memória e CPU.

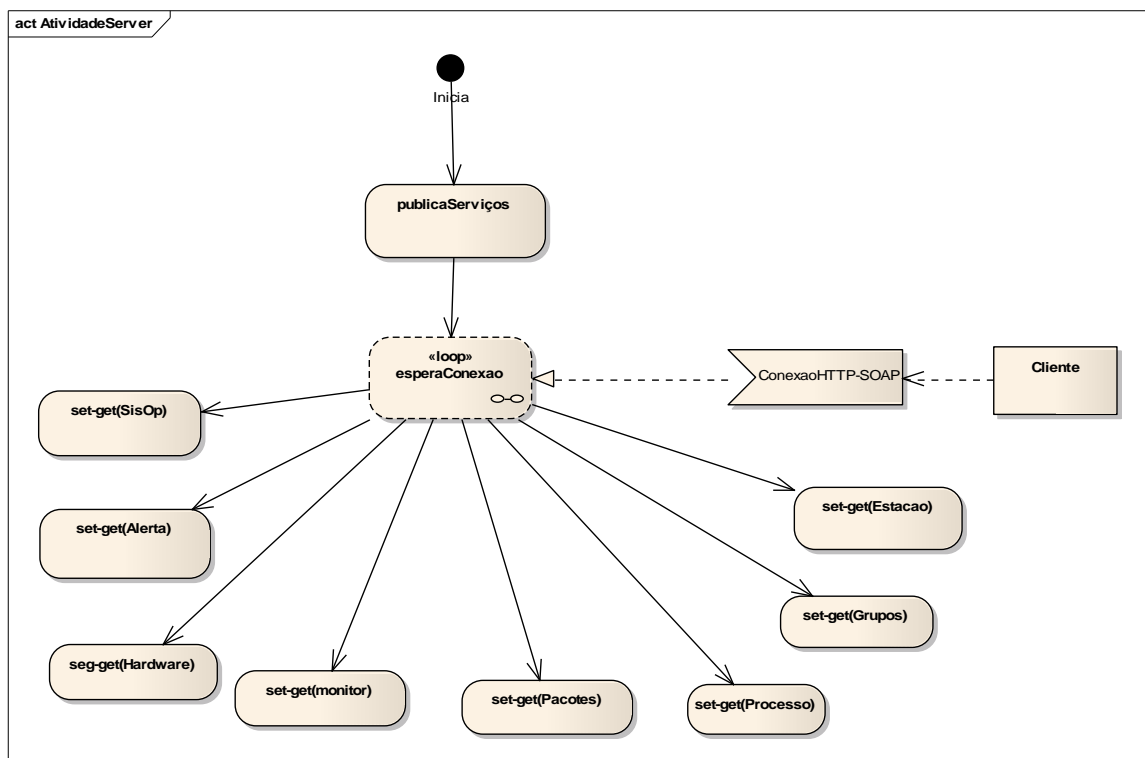


Figura 5 – Diagrama de atividades do Servidor

Na figura 5 está representado o diagrama de atividade do servidor *Web Service*.

Quando ele é iniciado, publica seus serviços e aguarda alguma conexão solicitando a chamada de um serviço, como exemplo a notificação que um pacote foi instalado em uma determinada estação através do serviço (setPacote).

Foram unidas as funções *set* e *get* de uma mesma classe em uma representação de atividade para simplificar o diagrama.

3.2.2 Modelo conceitual da base de dados

Os dados dos sistemas são persistidos em um banco de dados relacional.

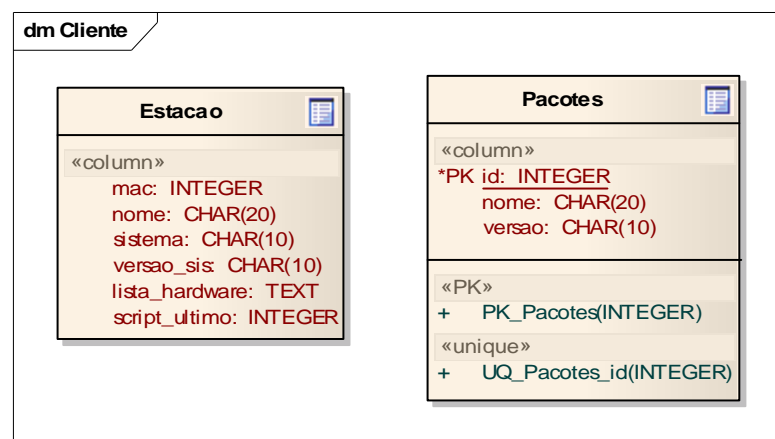


Figura 6 – Modelo de dados cliente

Na figura 6 estão as tabelas do cliente, que são a tabela Estação (que guarda alguns dados básicos na máquina como endereço físico e nome de rede da estação, qual o sistema operacional e sua versão, listagem do hardware e último *script* executado com sucesso), e as tabelas Pacotes (que guarda a listagem completa da última execução). Estes dados são utilizados para comparar se houve alguma modificação da execução anterior com a atual então. Havendo alguma alteração, será notificado ao servidor.

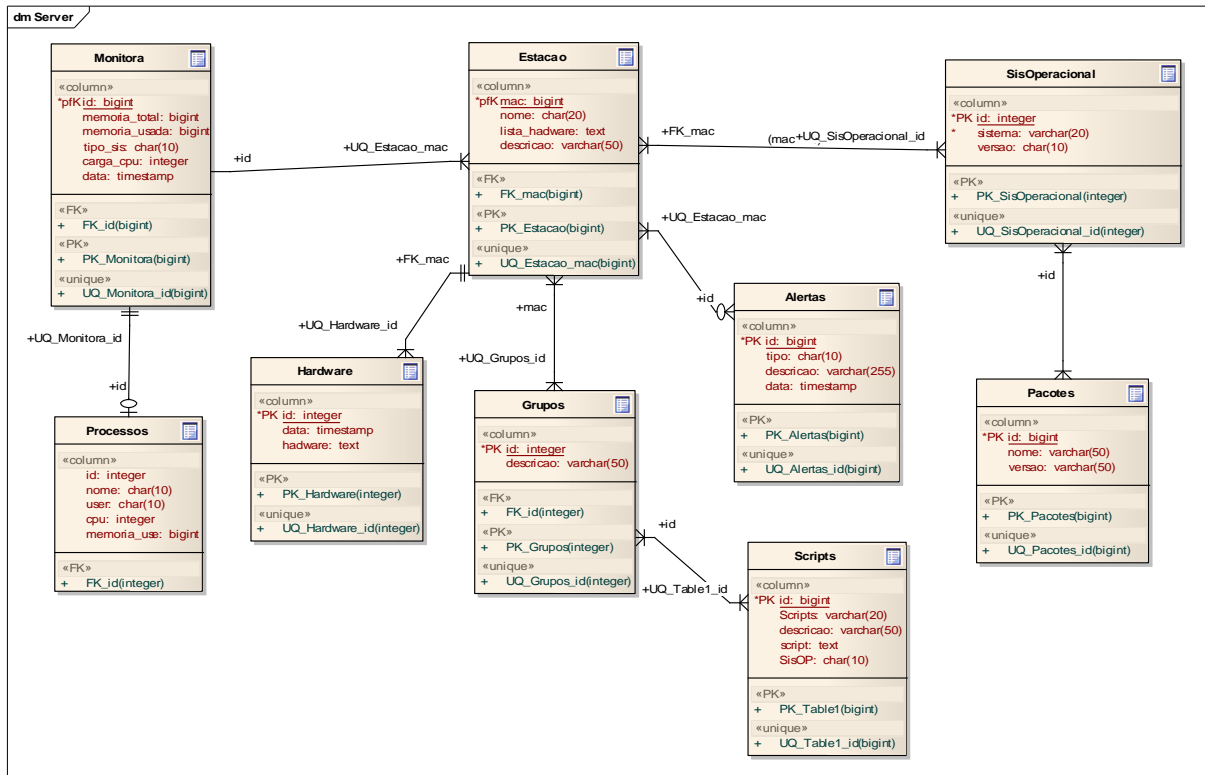


Figura 7 – Modelo de dados servidor

Na figura 7, tem-se a descrição de todos os dados que são persistidos no servidor, que estão distribuídos nas seguintes tabelas:

- tabela estação – guarda estação com endereço físico da interface de rede, nome e uma descrição da estação;
- tabela sistema operacional – guarda informações do sistema operacional: nome e versão.
- tabela pacotes – guarda a lista de todos os pacotes instalados no sistema operacional específico.
- tabela alertas – guarda alertas gerados pelas estações, com uma descrição, data do alerta e o tipo do alerta (hardware, software, urgente, etc.);
- tabela grupos – guarda os grupos das estações;
- tabela scripts – guarda *scripts* que são definidos por grupos.
- tabela hardware – guarda arquivo XML com a descrição do hardware e a data do seu cadastro;
- tabela monitora – guarda monitoramentos que são feitos nas estações;
- tabela processos – guarda listagem dos processos por monitoramentos.

3.2.3 Diagramas de Classes

Diagramas de Classes é uma representação da estrutura e relações das classes que servem de modelo para objetos, uma modelagem muito útil para o sistema. Define todas as classes que o sistema necessita possuir e é a base para a construção dos diagramas de comunicação, sequência e estados.

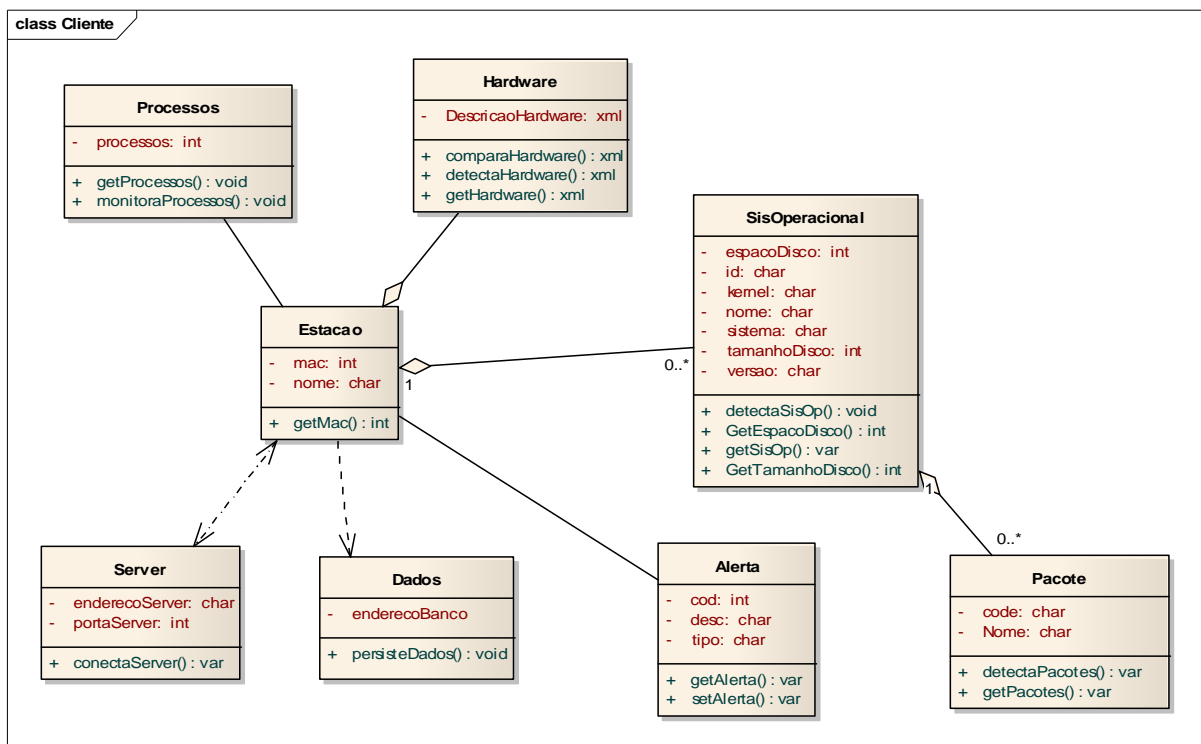


Figura 8 – Diagrama de classes do cliente

A função de cada classe do sistema cliente representada na figura 8, está descrito abaixo:

- classe Estacao – classe responsável por fazer o reconhecimento inicial do endereço físico de rede e nome da estação, que é utilizado para identificação da estação com o servidor;
- classe sisOperacional – classe responsável por reconhecer e comparar os dados principais do sistema operacional;
- classe Pacote – classe responsável por detectar todos os pacotes instalados na máquina e comparar com os dados do ultimo reconhecimento, para descobrir se houve alterações;
- classe Alerta – classe responsável por enviar os alertas ao servidor;

- e) classe Hardware – classe responsável por detectar e comparar modificações no hardware da estação;
- f) classe Processos – classe responsável de monitorar os processos executando nas estações e os recurso de memória e cpu consumidos;
- g) classe Server – classe responsável por gerenciar a conexão com o servidor;
- h) classe Dados – classe responsável de comunicar com a base de dados local.

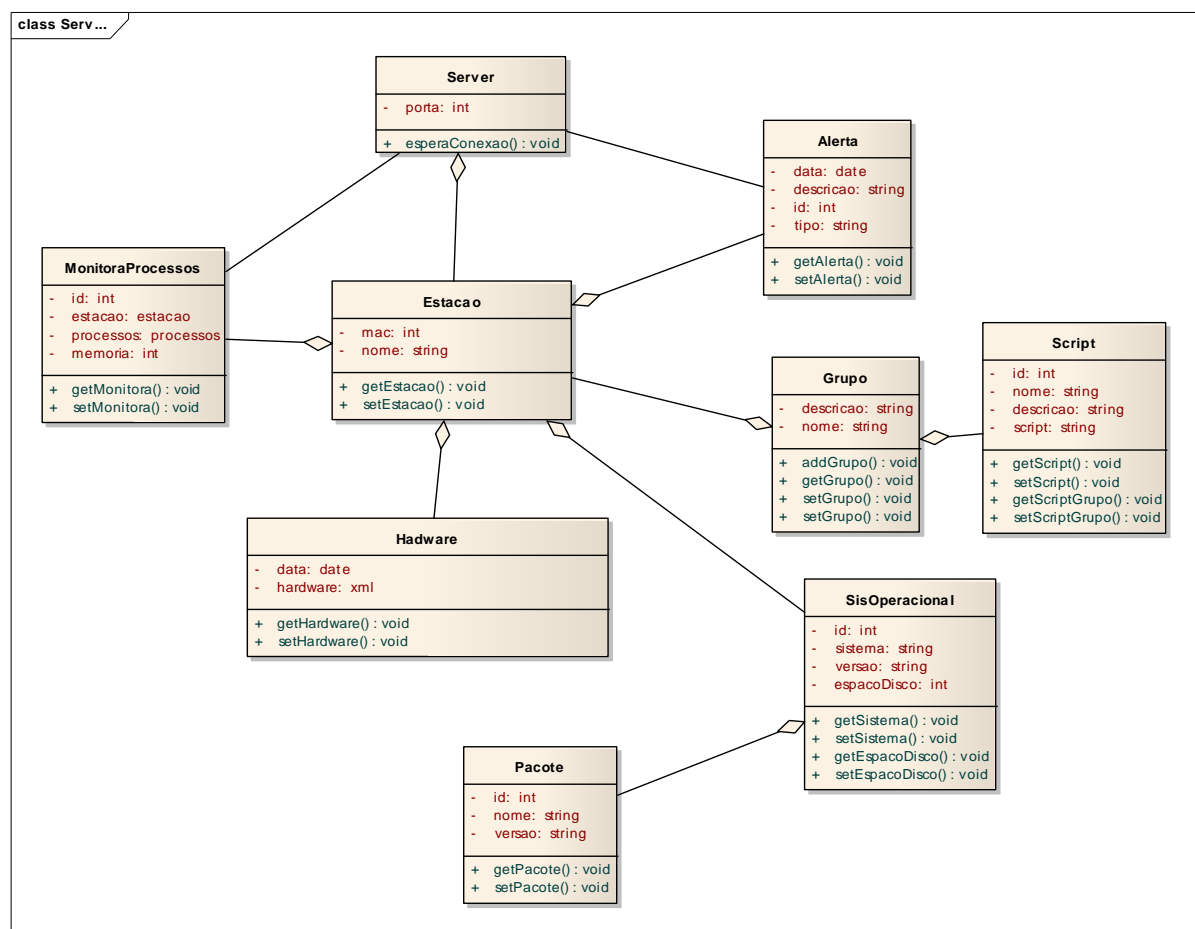


Figura 9 – Diagrama de classes do servidor

A função de cada classe do sistema servidor, representada pela figura 9, esta descrito abaixo:

- a) classe Estacao – classe responsável por tratar da identificação de cada estação, através do endereço físico de rede e nome da estação;
- b) classe sisOperacional – classe responsável por manipular os dados principais do sistema operacional;
- c) classe Pacote – classe responsável por guardar e informar os pacotes instalados nas estações
- d) classe Alerta – classe responsável por receber, guardar e gerenciar os alertas

- enviados pelas estações;
- e) classe *Hardware* – classe responsável por manter a listagem do hardware das estações.
- f) classe *MonitoraProcessos* – classe responsável por manter dados gerados pelo monitoramento das estações;
- g) classe *Server* – classe responsável por iniciar o *Web Service*, publicando seus serviços e aguardado conexões.

3.3 IMPLEMENTAÇÃO

Esta seção demonstra as técnicas e ferramentas utilizados para o desenvolvimento do sistema.

3.3.1 Técnicas e ferramentas utilizadas

Tanto o servidor *Web Service* quanto o cliente foram desenvolvidos com a linguagem de programação Python. Foi utilizado a biblioteca *xmlrpc* para a realizar a tecnologia *Web Service* e também o protocolo de comunicação SOAP entre o servidor e os clientes.

Foi utilizado o banco de dados relacional PostgreSQL para a persistência dos dados no *Web Services* e o *sqlite* no cliente por sua simplicidade.

3.3.2 Implementação do sistema

O objetivo deste trabalho é implementar um sistema de *Web Service*, cujo objetivo era receber e fornecer informações a estações através de uma aplicativo cliente, e também disponibilizar esses serviços a qualquer outra aplicação que possa tirar proveito destes, como uma sistema web que disponibiliza para usuários informações de quais softwares estão instalados em determinada estação ou ainda um *script* feito por um administrador da rede que poderia pegar o nome e o endereço físico de um determinado grupo de estações e gerar um arquivo de configuração para outro serviço.

No quadro 4 é apresentado um exemplo de como funciona o *Web Service* através da biblioteca `xmlrpc`, onde é instanciado o servidor, publicado o serviço `setEstacao` e em seguida iniciado o servidor.

```
import xmlrpclib

PORTA = 80

def setEstacao(mac, name):
    ## se mac e nome não estiver vazio adiciona ao banco
    if mac and name :
        .....
        cbase.execute("insert into estacao values (null,'%s','%s')" % nome, versao)
        conn.commit()
    ##
    # Para simplificar demonstração outras funções foram omitidas
    ##

## instancia servidor
server = SimpleXMLRPCServer.SimpleXMLRPCServer(("localhost", PORTA))
## registra função disponível no web service
server.register_function(setEstacao)
## inicia servidor
server.serve_forever()
```

Quadro 3 – Código *Web Service*

No quadro 4 é apresentado o código do Cliente que conecta ao servidor chamando a função `setEstacao` passando o endereço físico e o nome de rede da estação como parâmetro.


```
#!/usr/bin/env python

import uuid
import socket
import xmlrpclib

SERVER = "192.168.1.200"
PORTA = "80"

## instancia conexão com servidor
server = xmlrpclib.ServerProxy("http://" + SERVER + ":" + PORTA + "/")

class estacao:

    def conectaServer():
        mac = uuid.getnode()
        name = socket.gethostname()
        ## conecta servidor chamando a função setEstacao
        server.setEstacao(mac, name)

## instancia objeto estacaoE da classe estacao e chama metodo connectServer()
estacaoE = estacao()
estacaoE.conectaServer()
```

Quadro 4 – Código cliente conecta *Web Service*

3.3.2.1 Diferenças entre sistemas operacionais

Python é feito para rodar o mesmo código em diferentes sistemas operacionais, mas pela natureza do sistema que busca informações específicas do sistema operacional e nem todas são implementadas por algumas biblioteca, algumas partes do código tiveram que ter implementação específica para cada sistema operacional, um bom exemplo seria a função de identificar todos os pacotes de software instalados no sistema.

```

def detectaPacotesWin():

    BASE = "SOFTWARE\Microsoft\Windows\CurrentVersion\Installer\UserData\S-1-5-18\Products"
    REG = _winreg.HKEY_LOCAL_MACHINE

    explorer = _winreg.OpenKey(REG, BASE)

    index = 0
    L = []
    while True:
        try:
            name = _winreg.EnumKey(key, index)
        except EnvironmentError:
            break
        index += 1
        L.append(name)

    Pacd = {}
    for x in L:
        nodo = _winreg.OpenKey(REG, BASE+'\\'+x+'\\'+ "InstallProperties")
        index = 0
        l = {}
        while True:
            try:
                name = _winreg.EnumValue(nodo, index)
            except EnvironmentError:
                break
            index += 1
            a,b,c = name
            if a == "DisplayName" :
                n = b
            if a == "DisplayVersion" :
                v = b
            pacd[n] = v
        return pacd

def detectaPacotesLinux():
    d = {}
    p = (commands.getoutput('dpkg-query -W').split('\n'))
    for x in p:
        n,v = x.split('\t')
        d[n] =v
    return d

```

Quadro 5 – Métodos detectaPacotes

Como se vê na Quadro 5 no Linux, tratou-se a saída do comando “dpkg-query -W” que existe nas distribuições que usa os sistema de pacotes deb. Este comando lista todos os pacotes instalados. Já no Windows precisou-se abrir o sistema de registro e percorrer as chaves que guardam essa informação. Ambas as funções retornam um dicionário onde o nome do pacote é a chave e a versão o valor.

Para a detecção do hardware foi utilizado no Linux o programas lshw, que já é instalado como padrão no Ubuntu Linux. Com o comando “lshw -xml” é listado uma arquivo XML como se vê no Anexo A, com todas as informações detalhadas do hardware.

3.3.3 Operacionalidade do sistema.

Como o sistema sendo um *Web Service*, poder-se-ia criar uma interface *front-end*, para interagir com o sistema, poderia ser uma pagina web, interface gráfica ou linha de comando. Fez-se uma pequena ferramenta em linha de comando para interagir com o *Web Service*, chama de wscmd.

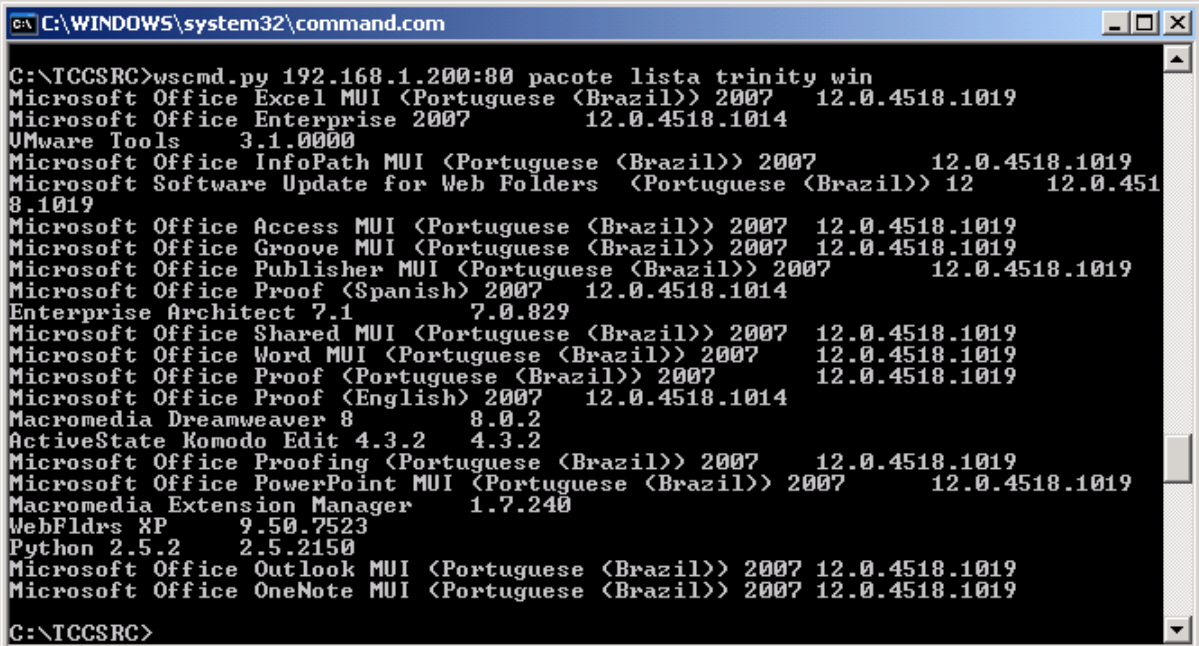
Esta ferramenta tem a seguinte sintaxe:

```
wscmd servidor:porta modulo ação opções
```

Quadro 6 – Sintaxe da ferramenta wscmd

Como exemplo de utilização, pode-se fazer o comando `wscmd 192.168.1.200:80 pacotes lista trinity win` onde 192.168.1.200:80 é nome e a porta do servidor, pacotes é o modulo do qual se quer a ação, lista a ação que se quer realizar: listar todos os pacotes instalados.

Na Figura 10 vê-se a saída da consulta da estação trinity do sistema operacional Windows. Este comando basicamente conecta o servidor solicitando a função `getPacotes`, passando como parâmetro a estação e sistema operacional. Na Figura 11 vê-se a mesma consulta agora de uma estação Linux, exibida parcialmente pois no Linux há uma lista muito maior de pacotes, por o Linux ser um sistema bastante granular com muitos pequenos pacotes com funções específicas.



```

C:\WINDOWS\system32\command.com
C:\TCCSRC>wscmd.py 192.168.1.200:80 pacote lista trinity win
Microsoft Office Excel MUI <Portuguese <Brazil>> 2007 12.0.4518.1019
Microsoft Office Enterprise 2007 12.0.4518.1014
VMware Tools 3.1.0000
Microsoft Office InfoPath MUI <Portuguese <Brazil>> 2007 12.0.4518.1019
Microsoft Software Update for Web Folders <Portuguese <Brazil>> 12 12.0.451
8.1019
Microsoft Office Access MUI <Portuguese <Brazil>> 2007 12.0.4518.1019
Microsoft Office Groove MUI <Portuguese <Brazil>> 2007 12.0.4518.1019
Microsoft Office Publisher MUI <Portuguese <Brazil>> 2007 12.0.4518.1019
Microsoft Office Proof <Spanish> 2007 12.0.4518.1014
Enterprise Architect 7.1 7.0.829
Microsoft Office Shared MUI <Portuguese <Brazil>> 2007 12.0.4518.1019
Microsoft Office Word MUI <Portuguese <Brazil>> 2007 12.0.4518.1019
Microsoft Office Proof <Portuguese <Brazil>> 2007 12.0.4518.1019
Microsoft Office Proof <English> 2007 12.0.4518.1014
Macromedia Dreamweaver 8 8.0.2
ActiveState Komodo Edit 4.3.2 4.3.2
Microsoft Office Proofing <Portuguese <Brazil>> 2007 12.0.4518.1019
Microsoft Office PowerPoint MUI <Portuguese <Brazil>> 2007 12.0.4518.1019
Macromedia Extension Manager 1.7.240
WebFldrs XP 9.50.7523
Python 2.5.2 2.5.2150
Microsoft Office Outlook MUI <Portuguese <Brazil>> 2007 12.0.4518.1019
Microsoft Office OneNote MUI <Portuguese <Brazil>> 2007 12.0.4518.1019
C:\TCCSRC>

```

Figura 10 – Saída do wscmd listando pacotes Windows

```

C:\TCCSRC>
C:\TCCSRC>wscmd.py 192.168.1.200 pacote lista persephone linux
libgcc1 1:4.2.3-2ubuntu7
java-common 0.28ubuntu3
zope3 3.3.1-5ubuntu2
x11-xkb-utils 7.3+1
gstreamer0.10-alsa 0.10.18-3
python-twisted-conch 1:0.8.0-1build1
ttf-lao 0.0.20060226-2
libmono-cairo1.0-cil 1.2.6+dfsg-6ubuntu3
libfs6 2:1.0.0-4ubuntu2
libavahi-common3 0.6.22-2ubuntu4
ghostscript-x 8.61.dfsg.1-1ubuntu3
libgweather-common 2.22.1.1-0ubuntu2
xserver-xorg-video-rendition 1:4.1.3.dfsg.1-4
totem 2.22.1-0ubuntu2
python-gmenu 2.22.2-0ubuntu1
libreadline5 5.2-3build1
libcdparanoia0 3.10+debian~pre0-6
libqtcore4 4.4.0-1ubuntu5~hardy1
update-manager-core 1:0.87.27
linux-restricted-modules-common 2.6.24.13-18.41
libpth20 2.0.7-8
libtiff4 3.8.2-7ubuntu3
gimp-python 2.4.5-1ubuntu2
libtotem-plparser10 2.22.3-0ubuntu2
libmysqlclient15off 5.0.51a-3ubuntu5.1
compiz-gnome 1:0.7.4-0ubuntu7
python-pysqlite2 2.4.0-2build1
gnome-netstatus-applet 2.12.1-1ubuntu1

```

Figura 11 – Saída do wscmd listando pacotes Linux

Caso o wscmd for executado com sintaxe errada ou com o parâmetro *help*, serão exibidas instruções de ajuda conforme o quadro 7, informando como deve ser usado.

Use:		
wscmd.py [server:port] estacao	<command>	Para consultar estações
wscmd.py [server:port] sistema	<command>	Para consultar sistema operacional da estação
wscmd.py [server:port] hardware	<command>	Para consultar hardware da estação
wscmd.py [server:port] pacotes	<command>	Para consultar pacotes instalados
wscmd.py [server:port] grupos	<command>	Gerencia pacotes de software
wscmd.py [server:port] scripts	<command>	Gerencia grupos de estações
wscmd.py [server:port] monitor	<command>	Gerencia monitor de processos

Quadro 7 – Instruções de ajuda do wscmd

Poderá ser consultada também a ajuda de um módulo específico, listando os comandos e as opções para aquele módulo. No quadro 8 vê-se o exemplo para o módulo grupos, quando executa `wscmd.py server:80 grupos help`.

Use:		
cria		
criar	[grupo] <descrição>	Cria Grupo
lista	[grupo]	Para lista as estações pertecente ao grupo
apagar	[grupo]	Apagar Grupo
add	[grupo] <estações>	Adiciona estações ao grupo
del	[grupo] <estações>	Remover estações de um grupo

Quadro 8 – Instruções de ajuda do módulo grupos do wscmd

3.4 RESULTADOS E DISCUSSÃO

A utilização de *Web Services* demonstra ser uma alternativa viável e interessante para o inventário de estações em rede, principalmente pela flexibilidade capaz de ser utilizada de muitas formas como ferramenta de apoio ao administrador de redes.

A utilização do HTTP como protocolo de comunicação traz vantagem de a estação e o servidor poderem estar separados por sub redes, *firewalls*, *Network Address Translation* (NAT). Pode-se exemplificar uma situação onde uma empresa tenha vários pequenos escritórios espalhados geograficamente: estas estações poderiam apenas ter um acesso a *internet* doméstico sem *internet protocol* (IP) fixo e atrás de NAT. Mesmo assim, esta estação conseguiria se comunicar com um *Web Service* público da empresa responsável pelo inventário das estações, assim ficando muito mais fácil e rápido detectar algum problema nesta estação, como remoção de hardware ou instalação de algum programa sem autorização.

Uma outra situação seria a possibilidade de mesmo remotamente o administrador consultar ou executar alguma operação nas suas estações, sem se preocupar com portas e tráfego gerado por exemplo de acessar um terminal remotamente.

Em trabalhos correlatos foram citados dois trabalhos que faziam a comparação de alternativas ao protocolo SNMP para gerência de redes. Baseado nestes trabalhos e na experiência obtida no desenvolvimento deste, verificou-se que a utilização de *Web Services* para gerência estações em rede trás como grande benefício a flexibilidade em relação ao SNMP, em contra partida de uma maior carga do sistema hospedeiro e maior tráfego gerado, mas como cada vez tem-se computadores mais potentes e redes mais velozes, parece ser um preço justificável em troca de uma maior flexibilidade para a gerência de redes.

Uma questão essencial que não foi contemplado neste projeto são as questões de segurança, que inviabilizaria a utilização prática em uma grande rede ou a disponibilidade

pública do *Webservice*, fator que foi indicado na seção de extensões.

Também não foram analisadas as questões de carga da aplicação Cliente na estação ou do *Web Service*, nem o tráfego de rede gerado pelas solicitações de serviços ou disponibilidade do servidor caso haja um grande número de solicitações.

Inicialmente o sistema foi desenvolvido usando a biblioteca para *WebServices* SOAPpy, enquanto foi utilizado apenas o Linux não houve problema, mas ao desenvolver e testar o cliente no Windows, não conseguiu-se instalar o SOAPpy que não dá suporte ao Python versão 2.5 na plataforma Windows. Estudando alternativas foi escolhida a xmlrpc que já vem embutida no Python, assim não precisa de instalação extra e atendeu a aplicação.

Não foi implementada a função de reconhecimento do hardware no Windows deixou de ser implementada primeiramente por não encontrar uma maneira trivial como no Linux e também por falta de conhecimento para fazer um estudo mais aprofundado das estruturas e recursos que o Windows poderia dispor para este fim.

A gama de informações e variáveis que podem ser extraídas de uma estação são muitas, mas buscou-se limitar e simplificar algumas, o suficiente para exemplificar o funcionamento e a viabilidade do sistema proposto.

4 CONCLUSÕES

A principal proposta de desenvolver um sistema de *Web Service*, que funcionasse com uma ferramenta de auxílio a um administrador de redes, na tarefa de gerenciar estações seja, em uma rede local ou distribuídos em várias redes.

Uma característica importante é a transparência da comunicação disponibilizada pelos protocolos HTTP e SOAP, assim transpondo de maneira fácil possíveis barreiras quando as estações não estão numa mesma rede local.

Outra ponto a se destacar é a flexibilidade fornecida pelo *Web Service*, possibilitando que o administrador de redes, crie sua próprias ferramentas que acessem informações do servidor, como *scripts* para automatizar sua tarefas ou uma pagina web onde seus usuários poderiam consultar informações sobre um determinada estação.

Desta forma concluiu-se que os objetivos foram atendidos, implementando os requisitos propostos e demonstrando a viabilidade da utilização da tecnologia de *Web Services* para o inventário de estações, além de abordagem *Web Services* que por si só é um tecnologia que começa a ser cada vez mais usada, principalmente por ser um dos principais componentes da arquitetura *Service Oriented Architecture* (SOA), a qual cada dia mais encontra-se artigos em revista especializadas.

Um último ponto para salientar, é a utilização da linguagem Python que é bastante utilizada mundialmente inclusive pelo indexador Google, e muito pouco utilizado nas empresas e menos ainda nas universidades da região de Blumenau.

4.1 EXTENSÕES

Este trabalho não contempla questões de segurança, como a confiança entre cliente e servidor ou a criptografia dos dados transmitidos e nem mesmo qualquer tipo de autenticação para consulta ou alterações das informações, questões essenciais para uma utilização real do sistema.

Outra sugestão importante é o desenvolvimento de uma interface web ou gráfica para interagir com o sistema de maneira mais simples e ágil, e a implementação do reconhecimento

de hardware na plataforma Windows.

Também podem se adicionar muitas funcionalidades ao sistema, buscando informações mais detalhadas ou adicionando novas funções como poder comandar instalação de pacotes através do *Web Service*, ainda criando um *Web Service* na estação tornando a comunicação nos dois sentidos e ampliando a gama de possibilidades.

REFERÊNCIAS BIBLIOGRÁFICAS

- CATUNDA, Marco. **Python: guia de consulta rápida**. São Paulo: Novatec, 2001.
- CUNHA, David; **Web Services, SOAP e Aplicações Web**. [S.I]: [2002]. Disponível em: <http://devedge-temp.mozilla.org/viewsource/2002/soap-overview/index_pt_br.html>. Acesso em: 29 abr. 2008.
- GONÇALVES, Frederico. **Gerenciamento de redes através de web services: uma análise comparativa**. 2005. 160 f. Trabalho final (mestrado profissional) - Instituto de Computação, Campinas Universidade Estadual de Campinas, Campinas.
- LOPES, Raquel V.; SAUVÉ, Jacques P.; Nicolletti, Pedro S. **Melhores práticas para gerência de redes de computadores**. Rio de Janeiro: Campus, 2003.
- LUTZ, Mark; ASCHER, David. **Aprendendo Python**. 2. ed. Porto Alegre: Bookman, 2007.
- KUROSE, James F; ROSS, Keith W. **Redes de computadores e a Internet: uma abordagem top-down**. 3. ed. São Paulo: Pearson Addison Wesley, 2005.
- MENÉNDEZ, Andrés I. M. **Uma ferramenta de apoio ao desenvolvimento de Web Services**. 2002. 97f. Dissertação (mestrado), Universidade Federal de Campina Grande, Coordenação de Pós-Graduação em Informática, Campina Grande,.
- PILGRIM, Mark. **Mergulhando no Python: o guia rápido e prático para dominar o Python**. São Paulo: Alta Books, 2005.
- RECKZIEGEL, Mauricio. **Protocolo de transporte padrão - SOAP**. [S.I]: [2006]. Disponível em: <http://imasters.uol.com.br/artigo/4379/webservices/protocolo_de_transporte_padrao_soap/>. Acesso em: 22 abr. 2008.
- REIS, Ana L. A. **Comparação SNMP x agentes móveis para gerência de redes**. 2001. 137f. Dissertação (mestrado) - Centro Tecnológico, Universidade Federal de Santa Catarina, Florianópolis.
- SINGH, Inderjeet et al. **Projetando web services com a plataforma J2EE 1.4: tecnologias JAX-RPC, SOAP e XML**. Rio de Janeiro: Ciência Moderna, 2006.

SNELL, James; TIDWELL, Doug; KULCHENKO, Pavel. **Programming web services with SOAP**. Beijing : O'Reilly, 2002.

SORTICA, Eduardo. **Esquema simples para gerência de redes**. [S.I]: Disponível em: http://pt.wikipedia.org/wiki/Imagem:Esquema_simples_de_rede_de_ger%C3%Aancia.JP

Acesso em: 29 abr. 2008

ANEXO A – Saído do comando “lshw -xml”

Vê-se no quadro 9 o arquivo XML gerado pelo comando “lshw -xml” listando o hardware de uma estação.

```
<?xml version="1.0" standalone="yes" ?>
- <!-- generated by lshw-B.02.12.0 -->
- <!-- GCC 4.2.3 (Ubuntu 4.2.3-2ubuntu7) -->
- <!-- Linux 2.6.24-18-server #1 SMP Wed May 28 21:25:52 UTC 2008 i686 -->
- <!-- GNU libc 2 (glibc 2.7) -->
=<node id="naufregados-1" claimed="true" class="system" handle="">
<description>Computer</description>
<width units="bits">32</width>
=<node id="core" claimed="true" class="bus" handle="">
<description>Motherboard</description>
<physid>0</physid>
=<node id="memory" claimed="true" class="memory" handle="">
<description>System memory</description>
<physid>0</physid>
<size units="bytes">4185190400</size>
</node>
=<node id="cpu:0" claimed="true" class="processor" handle="">
<product>Intel(R) Xeon(TM) CPU 3.40GHz</product>
<vendor>Intel Corp.</vendor>
<physid>1</physid>
<businfo>cpu@0</businfo>
<version>15.4.3</version>
<serial>0000-0F43-0000-0000-0000-0000</serial>
<size units="Hz">18446744072814584320</size>
<width units="bits">64</width>
=<configuration>
<setting id="id" value="1" />
</configuration>
=<capabilities>
<capability id="fpu">mathematical co-processor</capability>
<capability id="fpu_exception">FPU exceptions reporting</capability>
<capability id="wp" />
<capability id="vme">virtual mode extensions</capability>
<capability id="de">debugging extensions</capability>
<capability id="pse">page size extensions</capability>
<capability id="tsc">time stamp counter</capability>
<capability id="msr">model-specific registers</capability>
<capability id="pae">4GB+ memory addressing (Physical Address Extension)</capability>
<capability id="mce">machine check exceptions</capability>
<capability id="cx8">compare and exchange 8-byte</capability>
<capability id="apic">on-chip advanced programmable interrupt controller (APIC)</capability>
<capability id="sep">fast system calls</capability>
<capability id="mtrr">memory type range registers</capability>
<capability id="pge">page global enable</capability>
<capability id="mca">machine check architecture</capability>
<capability id="cmov">conditional move instruction</capability>
<capability id="pat">page attribute table</capability>
<capability id="pse36">36-bit page size extensions</capability>
<capability id="clflush" />
<capability id="dts">debug trace and EMON store MSRs</capability>
<capability id="acpi">thermal control (ACPI)</capability>
<capability id="mmx">multimedia extensions (MMX)</capability>
<capability id="fxsr">fast floating point save/restore</capability>
<capability id="sse">streaming SIMD extensions (SSE)</capability>
<capability id="sse2">streaming SIMD extensions (SSE2)</capability>
<capability id="ss">self-snoop</capability>
<capability id="ht">HyperThreading</capability>
<capability id="tm">thermal interrupt and status</capability>
<capability id="pbe">pending break event</capability>
<capability id="x86-64">64bits extensions (x86-64)</capability>
<capability id="constant_tsc" />
<capability id="pebs" />
<capability id="bts" />
<capability id="sync_rdtsc" />
<capability id="pni" />
<capability id="monitor" />
<capability id="ds_cpl" />
```

```

<capability id="est" />
<capability id="cid" />
<capability id="cx16" />
<capability id="xtpr" />
</capabilities>
- <node id="logicalcpu:0" claimed="true" class="processor" handle="CPU:1.0">
  <description>Logical CPU</description>
  <physid>1.1</physid>
  <width units="bits">64</width>
- <capabilities>
  <capability id="logical">Logical CPU</capability>
  </capabilities>
  </node>
- <node id="logicalcpu:1" claimed="true" class="processor" handle="CPU:1.1">
  <description>Logical CPU</description>
  <physid>1.2</physid>
  <width units="bits">64</width>
- <capabilities>
  <capability id="logical">Logical CPU</capability>
  </capabilities>
  </node>
</node>
- <node id="cpu:1" claimed="true" class="processor" handle="">
  <product>Intel(R) Xeon(TM) CPU 3.40GHz</product>
  <vendor>Intel Corp.</vendor>
  <physid>2</physid>
  <businfo>cpu@1</businfo>
  <version>15.4.3</version>
  <serial>0000-0F43-0000-0000-0000-0000</serial>
  <size units="Hz">18446744072814584320</size>
  <width units="bits">64</width>
- <configuration>
  <setting id="id" value="1" />
  </configuration>
- <capabilities>
  <capability id="fpu">mathematical co-processor</capability>
  <capability id="fpu_exception">FPU exceptions reporting</capability>
  <capability id="wp" />
  <capability id="vme">virtual mode extensions</capability>
  <capability id="de">debugging extensions</capability>
  <capability id="pse">page size extensions</capability>
  <capability id="tsc">time stamp counter</capability>
  <capability id="msr">model-specific registers</capability>
  <capability id="pae">4GB+ memory addressing (Physical Address Extension)</capability>
  <capability id="mce">machine check exceptions</capability>
  <capability id="cx8">compare and exchange 8-byte</capability>
  <capability id="apic">on-chip advanced programmable interrupt controller (APIC)</capability>
  <capability id="sep">fast system calls</capability>
  <capability id="mtrr">memory type range registers</capability>
  <capability id="pge">page global enable</capability>
  <capability id="mca">machine check architecture</capability>
  <capability id="cmov">conditional move instruction</capability>
  <capability id="pat">page attribute table</capability>
  <capability id="pse36">36-bit page size extensions</capability>
  <capability id="clflush" />
  <capability id="dts">debug trace and EMON store MSRs</capability>
  <capability id="acpi">thermal control (ACPI)</capability>
  <capability id="mmx">multimedia extensions (MMX)</capability>
  <capability id="fxsr">fast floating point save/restore</capability>
  <capability id="sse">streaming SIMD extensions (SSE)</capability>
  <capability id="sse2">streaming SIMD extensions (SSE2)</capability>
  <capability id="ss">self-snoop</capability>
  <capability id="ht">HyperThreading</capability>
  <capability id="tm">thermal interrupt and status</capability>
  <capability id="pbe">pending break event</capability>
  <capability id="x86-64">64bits extensions (x86-64)</capability>
  <capability id="constant_tsc" />
  <capability id="pebs" />
  <capability id="bts" />
  <capability id="sync_rdtsc" />
  <capability id="pni" />
  <capability id="monitor" />
  <capability id="ds_cpl" />
  <capability id="est" />
  <capability id="cid" />
  <capability id="cx16" />
  <capability id="xtpr" />

```

```

</capabilities>
- <node id="logicalcpu:0" claimed="true" class="processor" handle="CPU:1.0">
  <description>Logical CPU</description>
  <physid>1.1</physid>
  <width units="bits">64</width>
- <capabilities>
  <capability id="logical">Logical CPU</capability>
  </capabilities>
  </node>
- <node id="logicalcpu:1" claimed="true" class="processor" handle="CPU:1.1">
  <description>Logical CPU</description>
  <physid>1.2</physid>
  <width units="bits">64</width>
- <capabilities>
  <capability id="logical">Logical CPU</capability>
  </capabilities>
  </node>
  </node>
- <node id="pci" claimed="true" class="bridge" handle="PCIBUS:0000:00">
  <description>Host bridge</description>
  <product>E7520 Memory Controller Hub</product>
  <vendor>Intel Corporation</vendor>
  <physid>100</physid>
  <businfo>pci@0000:00:00.0</businfo>
  <version>0c</version>
  <width units="bits">32</width>
  <clock units="Hz">33000000</clock>
- <node id="pci:0" claimed="true" class="bridge" handle="PCIBUS:0000:02">
  <description>PCI bridge</description>
  <product>E7525/E7520/E7320 PCI Express Port A</product>
  <vendor>Intel Corporation</vendor>
  <physid>2</physid>
  <businfo>pci@0000:00:02.0</businfo>
  <version>0c</version>
  <width units="bits">32</width>
  <clock units="Hz">33000000</clock>
- <configuration>
  <setting id="driver" value="pcieport-driver" />
  </configuration>
- <capabilities>
  <capability id="pci" />
  <capability id="normal_decode" />
  <capability id="bus_master">bus mastering</capability>
  <capability id="cap_list">PCI capabilities listing</capability>
  </capabilities>
- <node id="pci:0" claimed="true" class="bridge" handle="PCIBUS:0000:03">
  <description>PCI bridge</description>
  <product>6700PXH PCI Express-to-PCI Bridge A</product>
  <vendor>Intel Corporation</vendor>
  <physid>0</physid>
  <businfo>pci@0000:02:00.0</businfo>
  <version>09</version>
  <width units="bits">32</width>
  <clock units="Hz">33000000</clock>
- <capabilities>
  <capability id="pci" />
  <capability id="normal_decode" />
  <capability id="bus_master">bus mastering</capability>
  <capability id="cap_list">PCI capabilities listing</capability>
  </capabilities>
- <node id="scsi:0" claimed="true" class="storage" handle="SCSI:02">
  <description>SCSI storage controller</description>
  <product>53c1030 PCI-X Fusion-MPT Dual Ultra320 SCSI</product>
  <vendor>LSI Logic / Symbios Logic</vendor>
  <physid>3</physid>
  <businfo>pci@0000:03:03.0</businfo>
  <logicalname>scsi2</logicalname>
  <version>07</version>
  <width units="bits">64</width>
  <clock units="Hz">66000000</clock>
- <configuration>
  <setting id="driver" value="mptspi" />
  <setting id="latency" value="72" />
  <setting id="maxlatency" value="18" />
  <setting id="mingnt" value="17" />
  <setting id="module" value="mptspi" />
  </configuration>

```

```

- <capabilities>
- <capability id="scsi" />
- <capability id="bus_master">bus mastering</capability>
- <capability id="cap_list">PCI capabilities listing</capability>
- <capability id="scsi-host">SCSI host adapter</capability>
- </capabilities>
- </node>
- <node id="scsi:1" claimed="true" class="storage" handle="SCSI:03">
- <description>SCSI storage controller</description>
- <product>53c1030 PCI-X Fusion-MPT Dual Ultra320 SCSI</product>
- <vendor>LSI Logic / Symbios Logic</vendor>
- <physid>3.1</physid>
- <businfo>pci@0000:03:03.1</businfo>
- <logicalname>scsi3</logicalname>
- <version>07</version>
- <width units="bits">64</width>
- <clock units="Hz">66000000</clock>
- <configuration>
- <setting id="driver" value="mptspi" />
- <setting id="latency" value="72" />
- <setting id="maxlatency" value="18" />
- <setting id="mingnt" value="17" />
- <setting id="module" value="mptspi" />
- </configuration>
- <capabilities>
- <capability id="scsi" />
- <capability id="bus_master">bus mastering</capability>
- <capability id="cap_list">PCI capabilities listing</capability>
- <capability id="scsi-host">SCSI host adapter</capability>
- </capabilities>
- </node>
- </node>
- <node id="pci:1" claimed="true" class="bridge" handle="PCIBUS:0000:07">
- <description>PCI bridge</description>
- <product>6700PXH PCI Express-to-PCI Bridge B</product>
- <vendor>Intel Corporation</vendor>
- <physid>0.2</physid>
- <businfo>pci@0000:02:00.2</businfo>
- <version>09</version>
- <width units="bits">32</width>
- <clock units="Hz">33000000</clock>
- <capabilities>
- <capability id="pci" />
- <capability id="normal_decode" />
- <capability id="bus_master">bus mastering</capability>
- <capability id="cap_list">PCI capabilities listing</capability>
- </capabilities>
- <node id="network:0" claimed="true" class="network" handle="PCI:0000:07:02.0">
- <description>Ethernet interface</description>
- <product>NetXtreme BCM5703 Gigabit Ethernet</product>
- <vendor>Broadcom Corporation</vendor>
- <physid>2</physid>
- <businfo>pci@0000:07:02.0</businfo>
- <logicalname>eth0</logicalname>
- <version>10</version>
- <serial>00:13:21:ea:00:f1</serial>
- <width units="bits">64</width>
- <clock units="Hz">66000000</clock>
- <configuration>
- <setting id="broadcast" value="yes" />
- <setting id="driver" value="tg3" />
- <setting id="driverversion" value="3.86" />
- <setting id="firmware" value="5703-v2.33" />
- <setting id="ip" value="172.17.9.1" />
- <setting id="latency" value="64" />
- <setting id="mingnt" value="64" />
- <setting id="module" value="tg3" />
- <setting id="multicast" value="yes" />
- </configuration>
- <capabilities>
- <capability id="bus_master">bus mastering</capability>
- <capability id="cap_list">PCI capabilities listing</capability>
- <capability id="ethernet" />
- <capability id="physical">Physical interface</capability>
- </capabilities>
- </node>
- <node id="network:1" claimed="true" class="network" handle="PCI:0000:07:03.0">

```

```

<description>Ethernet interface</description>
<product>NetXtreme BCM5703 Gigabit Ethernet</product>
<vendor>Broadcom Corporation</vendor>
<physid>3</physid>
<businfo>pci@0000:07:03.0</businfo>
<logicalname>eth1</logicalname>
<version>10</version>
<serial>00:14:c2:5b:c3:2f</serial>
<width units="bits">64</width>
<clock units="Hz">66000000</clock>
<configuration>
<setting id="broadcast" value="yes" />
<setting id="driver" value="tg3" />
<setting id="driverversion" value="3.86" />
<setting id="firmware" value="5703-v2.35, ASFIPMic v2.36" />
<setting id="ip" value="172.16.1.6" />
<setting id="latency" value="64" />
<setting id="mingnt" value="64" />
<setting id="module" value="tg3" />
<setting id="multicast" value="yes" />
</configuration>
<capabilities>
<capability id="bus_master">bus mastering</capability>
<capability id="cap_list">PCI capabilities listing</capability>
<capability id="ethernet" />
<capability id="physical">Physical interface</capability>
</capabilities>
</node>
</node>
</node>
<node id="pci:1" claimed="true" class="bridge" handle="PCIBUS:0000:0b">
<description>PCI bridge</description>
<product>E7525/E7520 PCI Express Port B</product>
<vendor>Intel Corporation</vendor>
<physid>4</physid>
<businfo>pci@0000:00:04.0</businfo>
<version>0c</version>
<width units="bits">32</width>
<clock units="Hz">33000000</clock>
<configuration>
<setting id="driver" value="pcieport-driver" />
</configuration>
<capabilities>
<capability id="pci" />
<capability id="normal_decode" />
<capability id="bus_master">bus mastering</capability>
<capability id="cap_list">PCI capabilities listing</capability>
</capabilities>
</node>
<node id="pci:2" claimed="true" class="bridge" handle="PCIBUS:0000:0e">
<description>PCI bridge</description>
<product>E7520 PCI Express Port B1</product>
<vendor>Intel Corporation</vendor>
<physid>5</physid>
<businfo>pci@0000:00:05.0</businfo>
<version>0c</version>
<width units="bits">32</width>
<clock units="Hz">33000000</clock>
<configuration>
<setting id="driver" value="pcieport-driver" />
</configuration>
<capabilities>
<capability id="pci" />
<capability id="normal_decode" />
<capability id="bus_master">bus mastering</capability>
<capability id="cap_list">PCI capabilities listing</capability>
</capabilities>
</node>
<node id="usb:0" claimed="true" class="bus" handle="PCI:0000:00:1d.0">
<description>USB Controller</description>
<product>82801EB/ER (ICH5/ICH5R) USB UHCI Controller #1</product>
<vendor>Intel Corporation</vendor>
<physid>1d</physid>
<businfo>pci@0000:00:1d.0</businfo>
<version>02</version>
<width units="bits">32</width>
<clock units="Hz">33000000</clock>

```

```

- <configuration>
- <setting id="driver" value="uhci_hcd" />
- <setting id="latency" value="0" />
- <setting id="module" value="uhci_hcd" />
- </configuration>
- <capabilities>
- <capability id="uhci">Universal Host Controller Interface (USB1)</capability>
- <capability id="bus_master">bus mastering</capability>
- </capabilities>
- </node>
- <node id="usb:1" claimed="true" class="bus" handle="PCI:0000:00:1d.1">
- <description>USB Controller</description>
- <product>82801EB/ER (ICH5/ICH5R) USB UHCI Controller #2</product>
- <vendor>Intel Corporation</vendor>
- <physid>1d.1</physid>
- <businfo>pci@0000:00:1d.1</businfo>
- <version>02</version>
- <width units="bits">32</width>
- <clock units="Hz">33000000</clock>
- <configuration>
- <setting id="driver" value="uhci_hcd" />
- <setting id="latency" value="0" />
- <setting id="module" value="uhci_hcd" />
- </configuration>
- <capabilities>
- <capability id="uhci">Universal Host Controller Interface (USB1)</capability>
- <capability id="bus_master">bus mastering</capability>
- </capabilities>
- </node>
- <node id="usb:2" claimed="true" class="bus" handle="PCI:0000:00:1d.2">
- <description>USB Controller</description>
- <product>82801EB/ER (ICH5/ICH5R) USB UHCI Controller #3</product>
- <vendor>Intel Corporation</vendor>
- <physid>1d.2</physid>
- <businfo>pci@0000:00:1d.2</businfo>
- <version>02</version>
- <width units="bits">32</width>
- <clock units="Hz">33000000</clock>
- <configuration>
- <setting id="driver" value="uhci_hcd" />
- <setting id="latency" value="0" />
- <setting id="module" value="uhci_hcd" />
- </configuration>
- <capabilities>
- <capability id="uhci">Universal Host Controller Interface (USB1)</capability>
- <capability id="bus_master">bus mastering</capability>
- </capabilities>
- </node>
- <node id="usb:3" claimed="true" class="bus" handle="PCI:0000:00:1d.3">
- <description>USB Controller</description>
- <product>82801EB/ER (ICH5/ICH5R) USB UHCI Controller #4</product>
- <vendor>Intel Corporation</vendor>
- <physid>1d.3</physid>
- <businfo>pci@0000:00:1d.3</businfo>
- <version>02</version>
- <width units="bits">32</width>
- <clock units="Hz">33000000</clock>
- <configuration>
- <setting id="driver" value="uhci_hcd" />
- <setting id="latency" value="0" />
- <setting id="module" value="uhci_hcd" />
- </configuration>
- <capabilities>
- <capability id="uhci">Universal Host Controller Interface (USB1)</capability>
- <capability id="bus_master">bus mastering</capability>
- </capabilities>
- </node>
- <node id="usb:4" claimed="true" class="bus" handle="PCI:0000:00:1d.7">
- <description>USB Controller</description>
- <product>82801EB/ER (ICH5/ICH5R) USB2 EHCI Controller</product>
- <vendor>Intel Corporation</vendor>
- <physid>1d.7</physid>
- <businfo>pci@0000:00:1d.7</businfo>
- <version>02</version>
- <width units="bits">32</width>
- <clock units="Hz">33000000</clock>
- <configuration>

```



```

<setting id="driver" value="ehci_hcd" />
<setting id="latency" value="0" />
<setting id="module" value="ehci_hcd" />
</configuration>
<capabilities>
<capability id="ehci">Enhanced Host Controller Interface (USB2)</capability>
<capability id="bus_master">bus mastering</capability>
<capability id="cap_list">PCI capabilities listing</capability>
</capabilities>
</node>
<node id="pci:3" claimed="true" class="bridge" handle="PCIBUS:0000:01">
<description>PCI bridge</description>
<product>82801 PCI Bridge</product>
<vendor>Intel Corporation</vendor>
<physid>1e</physid>
<businfo>pci@0000:00:1e.0</businfo>
<version>c2</version>
<width units="bits">32</width>
<clock units="Hz">33000000</clock>
<capabilities>
<capability id="pci" />
<capability id="normal_decode" />
<capability id="bus_master">bus mastering</capability>
</capabilities>
<node id="display" class="display" handle="PCI:0000:01:03.0">
<description>VGA compatible controller</description>
<product>Rage XL</product>
<vendor>ATI Technologies Inc</vendor>
<physid>3</physid>
<businfo>pci@0000:01:03.0</businfo>
<version>27</version>
<width units="bits">32</width>
<clock units="Hz">33000000</clock>
<configuration>
<setting id="latency" value="64" />
<setting id="mingnt" value="8" />
</configuration>
<capabilities>
<capability id="vga_controller" />
<capability id="bus_master">bus mastering</capability>
<capability id="cap_list">PCI capabilities listing</capability>
</capabilities>
</node>
<node id="system:0" class="system" handle="PCI:0000:01:04.0">
<description>System peripheral</description>
<product>Integrated Lights Out Controller</product>
<vendor>Compaq Computer Corporation</vendor>
<physid>4</physid>
<businfo>pci@0000:01:04.0</businfo>
<version>01</version>
<width units="bits">32</width>
<clock units="Hz">33000000</clock>
<configuration>
<setting id="latency" value="0" />
</configuration>
<capabilities>
<capability id="cap_list">PCI capabilities listing</capability>
</capabilities>
</node>
<node id="system:1" class="system" handle="PCI:0000:01:04.2">
<description>System peripheral</description>
<product>Integrated Lights Out Processor</product>
<vendor>Compaq Computer Corporation</vendor>
<physid>4.2</physid>
<businfo>pci@0000:01:04.2</businfo>
<version>01</version>
<width units="bits">32</width>
<clock units="Hz">33000000</clock>
<configuration>
<setting id="latency" value="64" />
</configuration>
<capabilities>
<capability id="bus_master">bus mastering</capability>
<capability id="cap_list">PCI capabilities listing</capability>
</capabilities>
</node>
</node>

```

```

- <node id="isa" claimed="true" class="bridge" handle="PCI:0000:00:1f.0">
  <description>ISA bridge</description>
  <product>82801EB/ER (ICH5/ICH5R) LPC Interface Bridge</product>
  <vendor>Intel Corporation</vendor>
  <physid>1f</physid>
  <businfo>pci@0000:00:1f.0</businfo>
  <version>02</version>
  <width units="bits">32</width>
  <clock units="Hz">33000000</clock>
- <configuration>
  <setting id="latency" value="0" />
- </configuration>
- <capabilities>
  <capability id="isa" />
  <capability id="bus_master">bus mastering</capability>
- </capabilities>
- </node>
- <node id="ide" claimed="true" class="storage" handle="PCI:0000:00:1f.1">
  <description>IDE interface</description>
  <product>82801EB/ER (ICH5/ICH5R) IDE Controller</product>
  <vendor>Intel Corporation</vendor>
  <physid>1f.1</physid>
  <businfo>pci@0000:00:1f.1</businfo>
  <logicalname>scsi0</logicalname>
  <version>02</version>
  <width units="bits">32</width>
  <clock units="Hz">33000000</clock>
- <configuration>
  <setting id="driver" value="ata_piix" />
  <setting id="latency" value="0" />
  <setting id="module" value="ata_piix" />
- </configuration>
- <capabilities>
  <capability id="ide" />
  <capability id="bus_master">bus mastering</capability>
  <capability id="emulated">Emulated device</capability>
- </capabilities>
- </node>
- <node id="cdrom" claimed="true" class="disk" handle="SCSI:00:00:00:00">
  <description>SCSI CD-ROM</description>
  <product>CD-ROM GCR-8482B</product>
  <vendor>HL-DT-ST</vendor>
  <physid>0.0.0</physid>
  <businfo>scsi@0:0.0.0</businfo>
  <logicalname>/dev/cdrom</logicalname>
  <logicalname>/dev/scd0</logicalname>
  <logicalname>/dev/sr0</logicalname>
  <dev>11d:0d</dev>
  <version>2.09</version>
  <serial>HL-DT-STCD-ROM GCR-8482B2.09</serial>
- <configuration>
  <setting id="ansiversion" value="5" />
  <setting id="status" value="ready" />
- </configuration>
- <capabilities>
  <capability id="removable">support is removable</capability>
  <capability id="audio">Audio CD playback</capability>
- </capabilities>
- <node id="medium" claimed="true" class="disk" handle="">
  <physid>0</physid>
  <logicalname>/dev/cdrom</logicalname>
  <dev>11d:0d</dev>
- </node>
- </node>
- </node>
- </node>
- </node>
- </node>
- </node>

```

Quadro 9 – Lista XML de hardware.