

**LIGA DE ENSINO DO RIO GRANDE DO NORTE  
FACULDADE NATALENSE PARA O DESENVOLVIMENTO  
DO RIO GRANDE DO NORTE  
CURSO DE BACHARELADO EM SISTEMA DE INFORMAÇÃO**

**DESENVOLVIMENTO DE UM WEB SERVICE APLICADO AO  
MONITORAMENTO REMOTO DE AMBIENTES HETEROGÊNEOS**

**DESENVOLVIMENTO DE UM WEB SERVICE APLICADO AO  
MONITORAMENTO REMOTO DE AMBIENTES HETEROGÊNEOS**

Monografia apresentada ao Curso de  
Bacharelado em Sistema de Informação  
da Faculdade Natalense para o  
Desenvolvimento do Rio Grande do  
Norte como requisito parcial para a  
obtenção do título de Bacharel em  
Sistema de Informação.

Orientador:

**Lenilson Antonio de Medeiros**

Aluno: Fernando da Costa Lima

**NATAL/RN  
2004**

**NATAL/RN  
2004**

LIC. **LENILSON ANTONIO DE MEDEIROS**  
FACULDADE NATALENSE PARA O DESENVOLVIMENTO  
DO RIO GRANDE DO NORTE  
CURSO DE BACHARELADO EM SISTEMA DE INFORMAÇÃO

DESENVOLVIMENTO DE UM WEB SERVICE APLICADO AO  
MONITORAMENTO REMOTO DE AMBIENTES HETEROGÊNEOS

## **DESENVOLVIMENTO DE UM WEB SERVICE APLICADO AO MONITORAMENTO REMOTO DE AMBIENTES HETEROGÊNEOS**

Monografia apresentada e aprovada em \_\_\_\_\_ de \_\_\_\_\_ de 2004, pela  
banca examinadora composta pelos seguintes membros:

Monografia apresentada ao Curso de  
Bacharelado em Sistema de Informação  
da Faculdade Natalense para o  
Desenvolvimento do Rio Grande do  
Norte como requisito parcial para a  
obtenção do título em Bacharel em  
Sistema de Informação.

**Orientador:**

Aluízio Ferreira da Rocha Neto

**NATAL/RN  
2004**

DESENVOLVIMENTO DE UM WEB SERVICE APLICADO AO  
MONITORAMENTO REMOTO DE AMBIENTES HETEROGÊNEOS

Monografia apresentada ao Curso de  
Bacharelado em Sistemas de Informação  
da Faculdade Natalense para o  
desenvolvimento do Rio Grande do  
Norte em parcial para a  
obtenção do título em Bacharel em  
Sistemas de Informação.

Catálogo na Publicação – Biblioteca da Farn  
Setor de Processos Técnicos

Medeiros, Lenilson Antonio de

Desenvolvimento de um web service aplicado ao monitoramento  
remoto de ambientes heterogêneos/ Lenilson Antônio de Medeiros.  
Natal: Farn, 2004.

64p.

Orientador: Aluizio Ferreira da Rocha Neto.

Monografia (Graduação em Sistemas de Informação) – Faculdade  
Natalense Para o Desenvolvimento do Rio Grande do Norte.

1. Internet – Web Services – monografia.
2. Monitoramento Remoto – monografia.
3. Plataforma .Net – monografia I.Título.

RN/FARN/BC

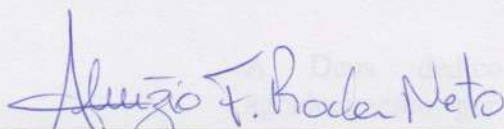
CDN 004.738.5

**LIGA DE ENSINO DO RIO GRANDE DO NORTE  
FACULDADE NATALENSE PARA O DESENVOLVIMENTO  
DO RIO GRANDE DO NORTE  
CURSO DE BACHARELADO EM SISTEMA DE INFORMAÇÃO**

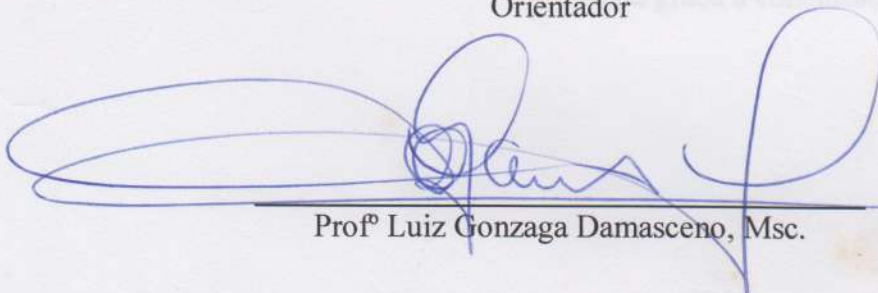
**DESENVOLVIMENTO DE UM WEB SERVICE APLICADO AO  
MONITORAMENTO REMOTO DE AMBIENTES HETEROGÊNEOS**

**Lenilson Antonio de Medeiros**

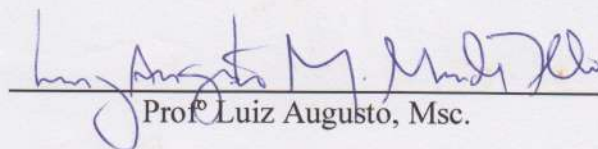
Monografia apresentada e aprovada em 11 de DEZEMBRO de 2004, pela  
banca examinadora composta pelos seguintes membros.



Profº Aluizio Ferreira da Rocha Neto, Msc.  
Orientador

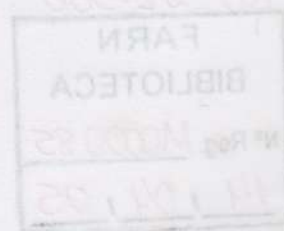


Profº Luiz Gonzaga Damasceno, Msc.



Profº Luiz Augusto, Msc.

Natal, 11 de DEZEMBRO de 2004.



LIGA DE ENSINO DO RIO GRANDE DO NORTE  
FACULDADE NATALENSE PARA O DESENVOLVIMENTO  
DO RIO GRANDE DO NORTE  
CURSO DE BACHARELADO EM SISTEMA DE INFORMAÇÃO

DESENVOLVIMENTO DE UM WEB SERVICE APLICADO AO  
MONITORAMENTO REMOTO DE AMBIENTES HETEROGÊNEOS

Lenilson Antonio de Medeiros

Monografia apresentada e aprovada em 11 de dezembro de 2004, pela  
banca examinadora composta pelos seguintes membros:

Prof. Aluizio Pereira da Rocha Neto, Msc.  
Orientador

Prof. Luiz Gonzaga Damasceno, Msc.

Prof. Luiz Augusto, Msc.



M  
004.738.5  
M438d  
2004

Natal, 11 de dezembro de 2004

## AGRADECIMENTOS

Este trabalho é resultado de quatro anos de estudos, dedicação e de grandes realizações. Sem o apoio dos professores, da família e dos grandes amigos este trabalho não seria concluído. Por este motivo é momento de agradecer aqueles que em meio a suas atividades foram solidários com a causa e incondicionalmente demonstraram serem realmente grandes incentivadores, além de amigos e companheiros.

Seria assim agradeço em primeiro lugar a Deus e a Virgem Maria a qual tenho uma grande devoção e admiração, pois, sem a graça de Deus e a intercessão da Virgem Maria este trabalho não seria concluído.

A minha família, em especial a minha mãe que sempre foi a figura mais presente em minha vida. Que soube orientar meus caminhos de descanço, me apoiar e me incentivar neste projeto.

A os grandes amigos Fernando Santos, Heliano Padilha, Ivanildo David, Walter I. Leite, Ricardo Pinheiro, Ottoniel Junior, Márcio Souza, Yuri Vale, Wend Junior, João Ubirajara, Lucilei Alves e todos aqueles outros que nas horas de grandes dificuldades estiveram sempre presentes e de forma incondicional me apoiaram e ajudaram na conclusão deste projeto.

A todos os professores que durante estes anos nos transmitiram seus conhecimentos e sua experiência de vida. Agradeço pelo estímulo nos dando para enfrentar os desafios que a vida acadêmica e a vida profissional. Um especial agradecimento ao professor Adriano Ferreira de Rocha Neto que além de professor tornou-se um grande amigo e que soube com grande sabedoria e paciência orientar este projeto.

A todos os meus companheiros de curso que a o longo destes quatro anos, dividimos juntos grandes alegrias e momentos de intensa dificuldade. Todos estes contribuíram muito na realização deste projeto.

A Faculdade Mariluz para o Curso de Engenharia de Materiais que me proporcionou a oportunidade de concluir este projeto.

E a todos os demais e de não citados.

A Deus dedico este trabalho como agradecimento por ele ter me congratulado com o sopro da vida e ter propiciado com sua infinita graça a conclusão deste projeto.

## AGRADECIMENTOS

Este trabalho é resultado de quatro anos de estudos, dedicação e de grandes renúncias. Sem o apoio dos professores, da família e dos grandes amigos esta tarefa não seria concluída. Por este motivo é momento de agradecer aqueles que em meio a suas atividades foram solidários com a causa e incondicionalmente demonstraram serem realmente grandes incentivadores, além de amigos e companheiros.

Sendo assim agradeço em primeiro lugar a Deus, a Virgem Maria a qual tenho uma grande devoção e admiração, pois, sem a graça de Deus e a intercessão da Virgem Maria este trabalho não seria concluído.

A minha família, em especial a minha mãe que sempre foi à figura mais presente em minha vida. Que soube renunciar seus momentos de descanso pra me apoiar e me incentivar neste projeto.

A os grandes amigos Fernando Santos, Ticiano Padilha, Ivanildo David, Walter Lúcio, Ricardo Pinheiro, Otoniel Junior, Márcio Souza, Yuri Vale, Wend Jason, João Ubirajara, Lucila Alves e todos aqueles outros que nas horas de grandes dificuldades estiveram sempre presentes e de forma incondicional me apoiaram e ajudaram na conclusão deste projeto.

A todos os professores que durante estes anos nos transmitiram seus ensinamentos e suas experiências de vida. Agradeço pelo estímulo nos dado para enfrentar os desafios que o meio acadêmico e a vida nos oferece. Em especial agradeço a o professor Aluizio Ferreira da Rocha Neto que além de professor tornou-se um grande amigo e que soube com grande sabedoria e paciência orientar este projeto.

A todos os meus companheiros de curso que a o longo desses quatro anos, dividimos juntos grandes alegrias e momentos de intensa dificuldade. Todos estes contribuíram direta ou indiretamente para a realização deste projeto.

À Faculdade Natalense para o Desenvolvimento do Rio Grande do Norte, por ter propiciado a oportunidade produtiva do conhecimento.

E a todos citados e os não citados, o meu Muito Obrigado.

## RESUMO

Este trabalho descreve uma tecnologia nova e em ascensão, destacando seus principais conceitos e a sua aplicabilidade ao monitoramento remoto. Esta tecnologia apresenta-se como precursora de uma revolução na comunicação entre aplicações, uma vez que esta tecnologia alavancou o paradigma da comunicação dos Sistemas Distribuídos: os *Web Services*. Alguns pontos relevantes desta tecnologia nos despertam muita atenção porque o seu processamento é via Internet (*Web*), sendo assim a grande maioria das empresas hoje em dia está conectada a Internet (*Web*), viabilizando o seu funcionamento. Outro ponto relevante que é destacado é que os *Web Services* não são propriedades de nenhuma empresa de Tecnologia, como por exemplo: Sun Micro Systems, Microsoft, IBM e Oracle etc. Essas empresas (Sun Micro Systems, Microsoft, IBM e Oracle) concordaram em apoiar um conjunto de padrões que definem como sistemas diferentes devem interagir entre si, num nível até então improvável de cooperação entre concorrentes. Além disso, a abordagem de *Web Services* não torna necessariamente as tecnologias de integração anteriores obsoletas, mas viabiliza tipos de integração anteriormente muito complexos.

Palavras-chave : Web Services, Internet, Web, .NET, XML.

## ABSTRACT

This work describes a new and in ascension technology, emphasizing your main concepts and your applicability to the remote monitoring. This technology comes as precursory of a revolution in the communication among applications, once this technology improved the paradigm of the communication of the Distributed Systems: the *Web Services*. Some important points of this technology has a lot of attention because your processing is through Internet (Web), so that the great majority of the companies is connected Internet (Web), making possible your operation. Another important point that can highlight it is that the *Web Services* is not properties of any company of Technology, as for instance: Sun Micro Systems, Microsoft, IBM and Oracle etc. Those companies (Sun Micro Systems, Microsoft, IBM and Oracle) agreed in supporting a group of patterns that you/they define as different systems they should interact to each other, in a level until then unlikely of cooperation among competitive. Besides, *Web Services's* approach doesn't necessarily turn the Technologies of previous integration obsolete, but it makes possible integration types previously very complex.

Keywords: Web Services, Internet, Web, .NET, XML

## SUMÁRIO

|          |                                                                  |           |
|----------|------------------------------------------------------------------|-----------|
| <b>1</b> | <b>INTRODUÇÃO .....</b>                                          | <b>11</b> |
| 1.1      | JUSTIFICATIVA .....                                              | 12        |
| 1.2      | OBJETIVOS.....                                                   | 13        |
| 1.3      | ORGANIZAÇÃO DO TRABALHO .....                                    | 13        |
| <b>2</b> | <b>METODOLOGIA.....</b>                                          | <b>15</b> |
| <b>3</b> | <b>REVISÃO BIBLIOGRÁFICA .....</b>                               | <b>16</b> |
| 3.1      | LINGUAGEM DE MODELAGEM UNIFICADA - UML .....                     | 16        |
| 3.2      | PLATAFORMA .NET .....                                            | 17        |
| 3.3      | XML.....                                                         | 20        |
| 3.4      | VB.NET .....                                                     | 23        |
| 3.5      | ASP.NET.....                                                     | 24        |
| 3.6      | WEB SERVICES.....                                                | 25        |
| 3.6.1    | Características e Divisão .....                                  | 26        |
| 3.6.2    | Especificações Técnicas.....                                     | 27        |
| 3.6.3    | Arquitetura da Web Com XML .....                                 | 27        |
| 3.6.4    | SOAP e Web Services.....                                         | 29        |
| 3.6.5    | SOAP e RPC .....                                                 | 31        |
| <b>4</b> | <b>DESENVOLVIMENTO .....</b>                                     | <b>33</b> |
| 4.1      | DESENVOLVIMENTO DO SISTEMA .....                                 | 33        |
| 4.1.1    | Especificações do sistema .....                                  | 33        |
| 4.1.2    | Etapas do desenvolvimento da solução .....                       | 34        |
| 4.1.2.1  | Etapa 1 - Análise e modelagem do Sistema de forma macro. ....    | 35        |
| 4.1.2.2  | Etapa 2 – Modelagem da aplicação Servidora .....                 | 35        |
| 4.1.2.3  | Etapa 3 - Modelagem da aplicação cliente .....                   | 37        |
| 4.1.2.4  | Etapa 4 - Exposição do código da aplicação.....                  | 39        |
| 4.1.2.5  | Exposição do código da Aplicação Cliente ( <i>Usuário</i> )..... | 58        |
| <b>5</b> | <b>CONSIDERAÇÕES FINAIS.....</b>                                 | <b>61</b> |
| 5.1      | DIFICULDADES ENCONTRADAS .....                                   | 61        |
| <b>6</b> | <b>CONCLUSÃO.....</b>                                            | <b>63</b> |
|          | <b>REFERÊNCIAS BIBLIOGRÁFICAS.....</b>                           | <b>64</b> |

## LISTA DE FIGURAS

|                                                                               |    |
|-------------------------------------------------------------------------------|----|
| FIGURA 3.1: ESTRUTURA DE UM DOCUMENTO XML.....                                | 22 |
| FIGURA 3.2: UM DTD - <i>DOCUMENT TYPE DEFINITION</i> .....                    | 22 |
| FIGURA 3.3: ESTRUTURA DE UM DOCUMENTO SOAP.....                               | 31 |
| FIGURA 3.4: SOLICITAÇÃO CLIENTE SOAP.....                                     | 32 |
| FIGURA 4.1: USUÁRIO MONITORANDO REMOTAMENTE UM AMBIENTE.....                  | 34 |
| FIGURA 4.2: DIAGRAMA DE CASO DE USO DE TODA A APLICAÇÃO.....                  | 35 |
| FIGURA 4.3 : DIAGRAMA DE CASO DE USO DA APLICAÇÃO SERVIDORA.....              | 36 |
| FIGURA 4.4 : DIAGRAMA DE CASO DE USO DA APLICAÇÃO CLIENTE.....                | 37 |
| FIGURA 4.5 : FUNCIONAMENTO DE UMA REQUISIÇÃO CLIENTE.....                     | 38 |
| FIGURA 4.6 : TELA DE CRIAÇÃO DO <i>WEB SERVICE</i> .....                      | 39 |
| FIGURA 4.7: TELA DO WEBMETHOD LOGINWEBCAM.....                                | 41 |
| FIGURA 4.8 : SOLICITAÇÃO SOAP FEITA PELO MÉTODO LOGINWEBCAM.....              | 42 |
| FIGURA 4.9 : RESPOSTA SOAP FEITA PELO MÉTODO LOGINWEBCAM.....                 | 42 |
| FIGURA 4.10: SOLICITAÇÃO HTTP POST SOAP FEITA PELO MÉTODO LOGINWEBCAM.....    | 43 |
| FIGURA 4.11: RESPOSTA HTTP POST SOAP FEITA PELO MÉTODO LOGINWEBCAM.....       | 43 |
| FIGURA 4.12 : TELA DO WEBMETHOD LOGOFFWEBCAM.....                             | 44 |
| FIGURA 4.13 : SOLICITAÇÃO SOAP FEITA PELO MÉTODO LOGOFFWEBCAM.....            | 44 |
| FIGURA 4.14: RESPOSTA SOAP FEITA PELO MÉTODO LOGOFFWEBCAM.....                | 45 |
| FIGURA 4.15 : SOLICITAÇÃO HTTP POST SOAP FEITA PELO MÉTODO LOGOFFWEBCAM.....  | 45 |
| FIGURA 4.16 : RESPOSTA HTTP POST SOAP FEITA PELO MÉTODO LOGOFFWEBCAM.....     | 45 |
| FIGURA 4.17: TELA DO WEBMETHOD RESETARWEBCAM.....                             | 46 |
| FIGURA 4.18: SOLICITAÇÃO SOAP FEITA PELO MÉTODO CAMRESETCAMERA.....           | 47 |
| FIGURA 4.19 : RESPOSTA SOAP FEITA PELO MÉTODO CAMRESETCAMERA.....             | 47 |
| FIGURA 4.20 : HTTP POST SOAP FEITA PELO MÉTODO CAMRESETCAMERA.....            | 48 |
| FIGURA 4.21: RESPOSTA HTTP POST SOAP FEITA PELO MÉTODO CAMRESETCAMERA.....    | 48 |
| FIGURA 4.22: TELA DO WEBMETHOD MOVERPCIMA.....                                | 49 |
| FIGURA 4.23: SOLICITAÇÃO SOAP FEITA PELO MÉTODO MOVERPCIMA.....               | 49 |
| FIGURA 4.24: RESPOSTA SOAP FEITA PELO MÉTODO MOVERPCIMA.....                  | 50 |
| FIGURA 4.25: SOLICITAÇÃO HTTP POST SOAP FEITA PELO MÉTODO MOVERPCIMA.....     | 50 |
| FIGURA 4.26: RESPOSTA HTTP POST SOAP FEITA PELO MÉTODO MOVERPCIMA.....        | 50 |
| FIGURA 4.27: TELA DO WEBMETHOD MOVERPBAIXO.....                               | 51 |
| FIGURA 4.28: SOLICITAÇÃO SOAP FEITA PELO MÉTODO MOVERPBAIXO.....              | 52 |
| FIGURA 4.29: RESPOSTA SOAP FEITA PELO MÉTODO MOVERPBAIXO.....                 | 52 |
| FIGURA 4.30: SOLICITAÇÃO HTTP POST SOAP FEITA PELO MÉTODO MOVERPBAIXO.....    | 53 |
| FIGURA 4.31: RESPOSTA HTTP POST SOAP FEITA PELO MÉTODO MOVERPBAIXO.....       | 53 |
| FIGURA 4.32: TELA DO WEBMETHOD MOVERPDIREITA.....                             | 54 |
| FIGURA 4.33: SOLICITAÇÃO SOAP FEITA PELO MÉTODO MOVERPDIREITA.....            | 54 |
| FIGURA 4.34: RESPOSTA SOAP FEITA PELO MÉTODO MOVERPDIREITA.....               | 55 |
| FIGURA 4.35 : SOLICITAÇÃO HTTP POST SOAP FEITA PELO MÉTODO MOVERPDIREITA..... | 55 |
| FIGURA 4.36: RESPOSTA HTTP POST SOAP FEITA PELO MÉTODO MOVERPDIREITA.....     | 55 |
| FIGURA 4.37: TELA DO WEBMETHOD MOVERPESQUERDA.....                            | 56 |
| FIGURA 4.38: SOLICITAÇÃO SOAP FEITA PELO MÉTODO MOVERPESQUERDA.....           | 57 |
| FIGURA 4.39: RESPOSTA SOAP FEITA PELO MÉTODO MOVERPESQUERDA.....              | 57 |
| FIGURA 4.40: SOLICITAÇÃO HTTP POST SOAP FEITA PELO MÉTODO MOVERPESQUERDA..... | 58 |
| FIGURA 4.41: RESPOSTA HTTP POST SOAP FEITA PELO MÉTODO MOVERPESQUERDA.....    | 58 |
| FIGURA 4.42: TELA PRINCIPAL DA APLICAÇÃO CLIENTE.....                         | 59 |
| FIGURA 4.43: TELA COM AS OPÇÕES DO SISTEMA PARA O CLIENTE.....                | 59 |
| FIGURA 4.44: TELA COM AS OPÇÕES DE MOVIMENTAÇÃO DO HARDWARE.....              | 60 |

## LISTA DE QUADROS

### I. INTRODUÇÃO

|                                                 |    |
|-------------------------------------------------|----|
| QUADRO 1: CÓDIGO DO WEBMETHOD LOGINWEBCAM ..... | 40 |
| QUADRO 2: WEBMETHOD LOGOFFWEBCAM.....           | 43 |
| QUADRO 3: WEBMETHOD RESETARWEBCAM.....          | 46 |
| QUADRO 4: WEBMETHOD MOVERPCIMA.....             | 48 |
| QUADRO 5: WEBMETHOD MOVERPBAIXO .....           | 51 |
| QUADRO 6: WEBMETHOD MOVERPDIREITA.....          | 53 |
| QUADRO 7: WEBMETHOD MOVERPESQUERDA.....         | 56 |

# 1 INTRODUÇÃO

A presente monografia aborda uma solução desenvolvida para a Internet (*Web*) que soluciona o problema da portabilidade enfrentado pelos sistemas de informação que apresentam características de arquiteturas específicas da computação.

O monitoramento remoto é uma técnica muito utilizada hoje em dia, para se observar à longa e a curta distância o que acontece em um determinado ambiente, seja ele qual for. A forma convencional de monitoramento remoto é através de câmeras de vídeo analógicas dispostas em ambientes distantes, e uma sala de reprodução destas imagens com um pessoal observando o que acontece. Com o avanço da tecnologia, principalmente àquela relacionada com a microinformática e Internet, já é possível monitorar ambientes através da Internet (*Web*), utilizando dispositivos de baixo custo, como *Webcams*, uma espécie de câmera para vídeo conferência na Internet.

Há um crescimento bastante expressivo de empresas ou mesmo de residências que querem observar seus ambientes a distância, tendo em vista, uma maior segurança de seus bens. O monitoramento remoto funciona como uma solução eficaz para este tipo de problema.

O objetivo principal dos sistemas de vigilância remota é monitorar e salva-guardar seus ambientes, sejam eles: bancos, residências, casas-lotéricas, escolas, etc. Esta técnica permite um acompanhamento em tempo real de todas as ações que se passam no ambiente.

Os sistemas Médico-Hospitalares têm aderido e utilizado o monitoramento remoto a fim de usufruírem seus inúmeros benefícios, entre eles pode-se destacar: o acompanhamento remoto de cirurgias, exames minuciosos, exames periódicos, partos, etc. Com o auxílio do monitoramento remoto, é possível estar em qualquer lugar e observar o que acontece em qualquer outro, economizando tempo e dinheiro com locomoção.

Os sistemas desenvolvidos atualmente e que se baseiam na técnica de monitoramento remoto, fazem uso de componentes de hardware e software específicos e proprietários. Dessa forma, a maioria dos sistemas não é portátil, isto é, não podem ser utilizados em qualquer ambiente de computação.

Com o intuito de abstrair os detalhes de implementação de serviços de computação disponibilizados para a Internet, a união dos grandes fabricantes da informática, como IBM, Microsoft, Sun Microsystems, Oracle, entre outros, levou ao desenvolvimento e especificação de uma tecnologia chamada *Web Service*. A idéia básica desta tecnologia é utilizar a Internet

como meio de comunicação para a produção e consumo de serviços de computação, independentes de hardware e softwares utilizados para prover tais serviços. Ela se baseia em linguagens e protocolos abertos largamente adotados pelo mercado.

Os sistemas que se baseiam na técnica de monitoramento remoto, sejam eles sistemas Médico-Hospitalares ou de Segurança remota, fazem uso de componentes e plataforma proprietária (Sistema Operacional), tornando-os dependentes de uma arquitetura específica de computação, ou seja, não há uma grande vantagem hoje em dia sobre isso, pois, as arquiteturas de computação (Sistemas Operacional) podem ser mudadas facilmente, levando em consideração o alto custo das mesmas. Ao observar a não portabilidade destes sistemas atuais pelo aspecto econômico enfrentado pela sociedade hoje em dia cheguei a conclusão que não se torna viável a aquisição de um sistema de monitoramento remoto com estas características.

## 1.1 JUSTIFICATIVA

O surpreendente crescimento e disseminação do uso do monitoramento remoto na sociedade hoje em dia, assim como a tecnologia envolvida no desenvolvimento de *Web Service* que ainda são assuntos novos e em grande ascensão, leva-me hoje a desenvolver esta pesquisa objetivando um maior aprofundamento e esclarecimento nestes dois tópicos. Sendo assim esta pesquisa tende a tornar-se, no meio acadêmico, como uma grande fonte de pesquisa e de esclarecimentos.

Inúmeras soluções na Internet (*Web*) atualmente são implementadas usando a tecnologia baseada em *Web Service*. Algumas empresas conhecidas aderiram ao uso desta tecnologia. Podemos citar os Correios, Banco Bradesco e por fim a empresa de transportes de mercadoria Fedex.

Os Correios atualmente disponibiliza via Internet (*Web*) o cálculo de fretes e a consulta à sua base de dados de CEP (Código de Endereçamento Postal). Com esta tecnologia, é possível ter informações sempre atualizadas, uma vez que as informações estão sempre no Servidor da empresa que as possui e as pesquisas são executadas pelo cliente. Com isso é solucionado o problema da informação de CEP desatualizado, pois, era comum a criação de novos endereços e por inúmeras vezes era necessário ir até a empresa (Correios) para se obter o CEP atualizado referente ao endereço.

O Bradesco também faz uso desta tecnologia com a geração de boletos bancários e pagamentos on-line, facilitando seus processos e melhorando o atendimento a os seus clientes.

A empresa de entrega de mercadoria Fedex também faz uso da tecnologia baseada em *Web Service*, disponibilizando alguns serviços como cálculo de fretes, *tracking* e restrições alfandegárias, via Internet (*Web*).

Esta monografia aborda o desenvolvimento de uma solução via Internet (*Web*), solução esta que não estará atrelada a uma arquitetura específica de computação ou proprietária (Sistema Operacional) para funcionar. Este desenvolvimento se baseará nos conceitos de *Web Service* e nos conceitos relacionados ao monitoramento remoto via Internet (*Web*). As funções que irão servir para prover o monitoramento remoto estarão inclusas no *Web Service*. Desde funções de manipulação da Câmera, giro, movimentação da esquerda pra direita, etc., até o envio de notificações ao cliente.

## 1.2 OBJETIVOS

O objetivo geral desta monografia é desenvolver uma solução baseada na Internet (*Web*) para monitoramento remoto que seja portátil, ou seja, que não dependa de nenhuma arquitetura de computação (Sistema Operacional) ou componente proprietário para ser executada.

Podemos destacar alguns objetivos específicos a serem alcançados neste trabalho:

- Desenvolver o referencial bibliográfico para torná-lo como fonte de pesquisa.
- Desenvolver uma aplicação prática dividida entre cliente (usuário) e servidor (*Web Service*) destacando suas principais funções e características.
- Tornar este projeto de pesquisa como fonte de informação e de pesquisa para o meio acadêmico, uma vez que o mesmo aborda temas complexos e atuais, sendo assim de muita relevância para o meio acadêmico.

## 1.3 ORGANIZAÇÃO DO TRABALHO

O presente trabalho monográfico é dividido em 5 capítulos, os quais são previamente descritos a seguir.

**Capítulo 1** – Introdução da monografia, descrevendo os pontos principais que serão abordados no conteúdo do trabalho, descrevendo os objetivos do trabalho e as atividades envolvidas no mesmo.

**Capítulo 2** – Esta etapa aborda a metodologia envolvida no desenvolvimento desta monografia.

**Capítulo 3** – Esta etapa dedica-se à revisão bibliográfica do trabalho, descrevendo as principais características dos pontos abordados no assunto do trabalho, ou seja, *Web Service* e monitoramento remoto.

**Capítulo 4** – Esta etapa envolve o desenvolvimento prático do trabalho monográfico, ou seja, o desenvolvimento da aplicação cliente e servidora, descrevendo todas as etapas do desenvolvimento prático.

**Capítulo 5** – Esta etapa refere-se à discussão e conclusão de todos os resultados obtidos no processo de desenvolvimento do sistema, relatando, também, todos os problemas encontrados no desenvolvimento prático.

**Capítulo 6** – Neste capítulo são feitas as considerações finais sobre o assunto abordado: *Web Service* e monitoramento remoto.

## 2 METODOLOGIA GRÁFICA

Esta monografia é dividida em pontos que fundamentam sua pesquisa: o desenvolvimento do referencial teórico metodológico da pesquisa, que utiliza o objeto de estudo da pesquisa e o outro ponto é a criação de uma situação prática que se fundamenta em um estudo de caso, que fornecerá subsídios para a implementação de um sistema. Sendo assim podemos classificar o tipo da pesquisa proposta por este projeto como: Exploratória, aplicada, e bibliográfica.

Esta pesquisa abordará como procedimentos e técnicas de pesquisa o uso da metodologia aplicada, pois, em seu desenvolvimento, buscará aplicar conhecimentos que são frutos de estudos para a solução do problema proposto. Em seqüência será desenvolvida uma análise utilizando a linguagem gráfica de modelagem UML e a modelagem do banco de dados através das entidades e relacionamentos encontrados na análise.

A codificação do projeto será feita em camadas, obedecendo rigorosamente o conceito de n camadas, sendo a primeira a camada de apresentação, a segunda sendo a camada de regras de negócio e a terceira a camada de persistência dos dados. A primeira camada será a feita a interação com o usuário, a segunda camada concentrará a lógica de negócios e finalmente a terceira camada que é responsável pela comunicação com os dados.

Esta monografia também fará uso da pesquisa bibliográfica, pois se buscará em materiais já publicados como livros, material de Internet (artigos, textos etc.) e artigos de periódicos, conhecimentos a respeito de novas tecnologias que poderão ser abordadas e utilizadas no seu desenvolvimento.

A UML é chamada de linguagem de modelagem, não é um modelo. A maioria dos autores consiste, pelo menos em princípio, de uma linguagem de modelagem e de um processo. A linguagem de modelagem é a notação (principalmente gráfica) utilizada por engenheiros para expressar projetos. O processo é a sugestão de quais passos a serem seguidos na elaboração do projeto.

Para Marcos Junior (2003), a UML é uma linguagem para especificação, documentação, visualização e desenvolvimento de sistemas orientados a objetos. Sua estrutura

### 3 REVISÃO BIBLIOGRÁFICA

Para a compreensão dos conceitos envolvidos nesta pesquisa, será feita uma contextualização das principais ferramentas utilizadas.

#### 3.1 LINGUAGEM DE MODELAGEM UNIFICADA - UML

As linguagens de modelagem orientadas a objetos surgiram entre a metade da década de 70 e o final da década de 80, à medida que o pessoal envolvido com metodologia, diante de um novo gênero de linguagens de programação orientadas a objetos e de aplicações cada vez mais complexas, começou a experimentar métodos alternativos de análise e projeto.

A quantidade de métodos orientados a objetos e de aplicações cada vez mais complexas, começou a experimentar métodos alternativos de análise de projeto. A quantidade de métodos orientados a objeto aumentou um pouco mais de 10 para mais de 50 durante o período de 1989 a 1994. Muitos usuários desses métodos tiveram dificuldade para encontrar uma linguagem de modelagem capaz de atender inteiramente suas necessidades, alimentando, assim, a chamada guerra de métodos. Aprendendo com a experiência, novas gerações desses métodos começaram a surgir, com um claro destaque de alguns como o Booch, o OOSE (*Object-Oriented Software Engineering*) de Jacobson, e o OMT (*Object Modeling Technique*) de Rumbaugh. Entre outros métodos importantes, encontravam-se Fusion, Shlaer-Mellor e Coad-Yourdon. Todos eram métodos completos, apesar de cada um conter pontos fortes e fracos.

A UML (*Unified Modeling Language*) é a sucessora da onda de métodos de análise e projeto orientado a objetos (OOA & D) que surgiu no final dos anos oitenta e no início dos anos noventa. Mais especificamente, ela unifica os métodos de Booch, Rumbaugh (OMT) e Jacobson, mas seu alcance é bem maior: UML passou por um processo de padronização pela OMG (*Object Management Group*) e é agora um padrão OMG.

A UML é chamada de linguagem de modelagem; não é um método. A maioria dos métodos consiste, pelo menos em princípio, de uma linguagem de modelagem e de um processo. A **linguagem de modelagem** é a notação (principalmente gráfica) utilizada por métodos para expressar projetos. O processo é a sugestão de quais passos a serem seguidos na elaboração do projeto.

Para Marcos Junior (2003), a UML é uma linguagem para especificação, documentação, visualização e desenvolvimento de sistemas orientados a objetos. Sintetiza os

principais métodos existentes, sendo considerada uma das linguagens mais expressivas para modelagem de sistemas orientados a objetos. Por meio de seus diagramas é possível representar sistemas de softwares sob diversas perspectivas de visualização. Facilita a comunicação de todas as pessoas envolvidas no processo de desenvolvimento de um sistema - gerentes, coordenadores, analistas, desenvolvedores - por apresentar um vocabulário de fácil entendimento.

A UML tem um grande suporte da indústria. Por ser a especificação suportada pelo OMG com mais de 850 membros, é o padrão da indústria do software de fato para a modelagem visual e o desenvolvimento. O fato de que todas as principais ferramentas para modelar os sistemas com muito software agora suportam a UML torna-se um padrão de fato também.

Tendo em vista o grau de complexidade do desenvolvimento deste projeto, optamos por escolher a UML como ferramenta de modelagem e análise para este desenvolvimento, por oferecer todos os recursos necessários exigidos para o desenvolvimento deste projeto.

Além disso, as ferramentas de modelagem UML geram uma quantidade crescente de código-fonte para a plataforma .NET, tornando o seu uso altamente indispensável e funcional.

### 3.2 PLATAFORMA .NET

Segundo Gutierrez (2002, p. 3), ao longo dos últimos anos, a tecnologia da informação tem dado saltos extraordinários – alguns chegam mesmo a falar numa “revolução”, tal o ritmo vertiginoso das inovações, tal sua capacidade de alterar a vida cotidiana de centenas de milhões de pessoas em todo o mundo.

Atualmente, há praticamente um computador pessoal em qualquer casa de classe média, mesmo em um país pobre como esse em que vivemos algo que se tornou comum, pelo fato de os computadores deixarem de ser assunto meramente militar. Hoje em nosso país que acabou de sair da “idade da pedra” das tele-comunicações - mais de onze milhões de pessoas acessam a Internet regularmente. Em menos de três anos, foram vendidos cerca de 30 milhões de telefones celulares no Brasil, uma verdadeira febre de consumo, conforme nos diz Gutierrez (2002, p. 3).

Nestes poucos anos, a computação deixou as salas blindadas reservadas ao uso exclusivo de alguns militares de primeiro mundo (os de terceiro mundo tinham de se contentar com calcular trajetórias balísticas com o auxílio do dedo levantado para obter a direção e a velocidade do vento), passou pelas salas igualmente blindadas das grandes

corporações, alcançou as mesas de escritório de qualquer pequena empresa e bancadas domésticas e está se dirigindo aceleradamente para os eletrodomésticos e dispositivos de uso pessoal. Se nós pensarmos que a memória disponível em um aparelho celular simplesmente para armazenar um catálogo de telefones pessoais já foi reservada para o processamento da folha de pagamentos de uma grande empresa, então teremos uma vaga noção da velocidade e do significado da tecnologia na vida contemporânea.

Com o uso mais abrangente dos computadores, se tornou possível no decorrer do tempo à comunicação entre os mesmos através de uma rede, a princípio essa conexão ficou restrita apenas as Universidades e as grandes corporações por ter um custo muito elevado e ser de grande complexidade. Às pequenas empresas e usuários domésticos bastava a computação isolada para cumprir a maioria das tarefas para os quais havia demanda. As redes de computadores estavam restritas às grandes corporações e ao mundo acadêmico, baseadas em computadores de médio e grande porte. Neste tempo (já longínquo para muitos), o máximo de tecnologia de comunicações pessoais atingia a maioria das residências eram os aparelhos de fax, então símbolo de status. Dos já poucos usuários de computadores pessoais muito poucos sequer haviam ouvido falar de modem e placas de rede.

Segundo Gutierrez (2002, p. 5), graças ao sucesso da *Word Wide Web* (WWW), o cenário se modificou abruptamente. Praticamente todos os usuários de computador pessoal acessam regularmente a Internet. Chega-se mesmo a comprar um computador para, em primeiro lugar, acessar a Internet – não para processar texto ou simplesmente jogar, notem bem, mas para se comunicar com o resto do mundo. Essa máquina está se tornando, antes de tudo, num dispositivo de comunicação e não de computação. Que o digam os milhões de usuários de salas de Chat e programas de comunicação.

No mundo corporativo, por outro lado, o fenômeno é igualmente devastador. Rumar para Internet se tornou a palavra de ordem. A menor empresa dotada de um computador tem também um modem para se conectar a Internet. Além disso, a competitividade acirrada no ramo dos negócios, faz com que os empreendedores mais atentos invistam em Tecnologia de Informação, mas precisamente em conexão. Estar conectado hoje na Internet para o ramo de negócios, não significa apenas ter um Site informativo e estático no ar, mas de programar serviços on-line de maneira dinâmica para consumidores e parceiros.

Evidentemente, alguns setores são mais dinâmicos que outros em invadir a Internet. O setor bancário, no Brasil, é um desses segmentos de ponta. O primeiro banco a disponibilizar um site de serviços na Internet era brasileiro. Isso talvez se deva ao fato de a melhor

tecnologia bancária ser a nossa. É possível que isso tenha sido fruto de anos de convivência com uma inflação crônica, que tornou os negócios financeiros o melhor meio de ganhar dinheiro, mas é um fato.

De acordo com Gutierrez (2002, p. 6), o Setor bancário é talvez um exemplo externo. Para todos os demais, porém, essa realidade está se tornando cada vez mais presente em todos os ramos de negócios. É certo que a Internet pode ser - e será - utilizada largamente no mundo dos negócios. O crescimento do consumo através da rede tem sido constante, a cada dia que passa mais e mais consumidor (s) adquirem a confiança necessária à realização de transações de compra e venda pela Internet. Mas não é só esse meio de explorarmos o mundo *business-to-consumer*, talvez nem mesmo seja o mais universal e melhor. O exemplo do setor bancário, a Internet tem sido utilizada com sucesso como meio de interação entre a empresa e seus clientes no acesso a serviços os mais diversos.

Segundo Fridschtein (2003, p. 1) a Internet está penetrando em todos os campos da atuação humana e proporcionando o combustível da globalização. Em poucos anos, quem não utilizar direta ou indiretamente a Rede, vai sentir que ficou de fora. Aí poderá ser difícil recuperar o tempo perdido.

Com este avanço da Internet (*Web*) nos demais ramos de negócios, a Microsoft desenvolveu uma estratégia chamada de .NET, que visa fornecer software e serviços que tornarão esse mundo conectado em um mundo voltado para a realidade e não apenas para uma visão publicitária. Porque acredita que este será o mundo de um futuro próximo, a Microsoft está construindo parte deste mundo e concomitantemente, fornecendo meios para que os usuários dos seus produtos e serviços partilhem igualmente dos benefícios desse "mundo conectado".

Segundo McDowell (2003, p. 1) vice-presidente de relações com o mercado corporativo da Microsoft Corporation, afirmou:

"Estamos tornando a experiência na Internet mais fácil, personalizada e produtiva. São novas possibilidades que surgem a um público muito maior, permitindo a troca de informações por meio de qualquer dispositivo, a qualquer hora e a partir de qualquer lugar".

McDowell (2003, p. 1) afirma ainda que a plataforma .NET é a base da nova geração de software e serviços que a Microsoft está apresentando ao mercado. Dentre as principais novidades, destaca-se a melhor experiência na Internet, facilidade de uso e integração de negócios e oportunidades. A família de produtos e tecnologias baseados na estratégia - novas

gerações de sistemas operacionais, servidores Windows DNA, Microsoft Office, rede de serviços MSN e a ferramenta de desenvolvimento Visual Studio - permite aos desenvolvedores a criação de inovações e dispositivos para a Web. A Microsoft .NET consiste entre uma nova gama de tecnologias para construir a experiência dos usuários da próxima geração. A implementação do novo modelo de programação, que provê a infra-estrutura da plataforma, é baseado no padrão XML e ajuda os desenvolvedores a integrar, operar e agrupar serviços da Web. O Visual Studio, nova versão do pacote de ferramentas utilizado por 50% dos profissionais da área em todo o mundo, dará apoio abrangente para o desenvolvimento de serviços de rede, também baseados em XML.

Segundo Gutierrez (2004, p. 12) afirma que:

“A Microsoft está construindo parte desse mundo, e concomitantemente, fornecendo meios para que os usuários e, concomitantemente, fornecendo meios para que os usuários dos seus produtos e serviços partilhem igualmente dos benefícios desse “Admirável mundo conectado”.

### 3.3 XML

Segundo Macoratti (2000, p.1) a XML é uma linguagem de marcadores e foi desenhada para descrever dados, a sua grande vantagem é que ela é extensível, ou seja, você não está limitado a certo número de *tags*, e pode criar as suas próprias *tags*, assim sendo ela é uma linguagem auto definível. Os dados de um arquivo XML são descritos usando o **DTD** ou **Document Type Definition**. Com tais características é possível estabelecer os objetos, métodos, parâmetros, dados e seus respectivos tipos que serão interpretados pela aplicação destino.

Segundo Macoratti (2000, p.1) a XML não é uma linguagem que veio substituir a HTML pois a XML foi criada com um objetivo diferente da HTML. Enquanto HTML foi criada para exibir dados, a XML foi criada para descrever dados e ela se preocupa com o que os dados são. HTML está relacionada com exibir dados enquanto que XML está relacionada em descrever dados. As *tags* usadas em documentos HTML são predefinidas e o autor de um documento HTML pode usar somente *tags* que estão definidas em HTML padrão. Já a XML permite que o autor defina as suas próprias *tags* e a própria estrutura do seu documento.

Segundo Martin et al (2001, p.33) a marcação XML descreve e oferece estrutura para o conteúdo de um arquivo XML ou pacote de dados, geralmente conhecido como **entidade XML**. A estrutura de elementos XML é basicamente a mesma que no HTML e a marcação

XML tem os mesmos colchetes para delimitar tags que comecem com um símbolo 'menor que' (<) e terminando com um símbolo 'maior que' (>).

Martin et al (2001, p.34) nos diz que o XML está voltado para o uso internacional, os caracteres não se limitam a conjuntos de caracteres ASCII de 7 bits. O XML especifica que caracteres são os números definidos no conjunto de caracteres Unicode 2.1 de 16-bit+.

Um documento XML possuem as seguintes partes:

- Um Prólogo opcional.
- O Corpo do documento, consistindo de um ou mais elementos, na forma de uma árvore hierárquica que também pode conter **dados de caracteres**.
- Um epílogo opcional com "miscelânea" composto de comentários, instruções de processamento e ou espaço em branco que segue o final da árvore de elemento.

Um exemplo de documento XML é visto na Figura 3.1.

|                |                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Prólogo</b> | <pre>&lt;?xml version "1.0" ?&gt; &lt;!-- comments and processing instructions allowes here--&gt; &lt;!DOCTYPE books SYSTEM http://myserver.com/books.dtd &gt; &lt;!-- comments and processing instructions allowes here--&gt;</pre>                                                                                                                                                                |
| <b>Corpo</b>   | <pre>&lt;books&gt;   &lt;book category = "reference"&gt;     &lt;author&gt; Nigel Rees &lt;/author&gt;     &lt;title&gt; Sayings of the Century &lt;/title&gt;     &lt;price&gt;8.95 &lt;/price&gt;   &lt;book category = "fiction"&gt;     &lt;author&gt; Evelyn Waugh &lt;/author&gt;     &lt;title&gt; Sword of Honour &lt;/title&gt;     &lt;price&gt;12.99 &lt;/price&gt; &lt;/books&gt;</pre> |
| <b>Epílogo</b> | <pre>&lt;!-- comments and processing instructions allowes here--&gt;</pre>                                                                                                                                                                                                                                                                                                                          |

Figura 3.1:

|                |                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Prólogo</b> | <pre>&lt;?xml version "1.0" ?&gt; &lt;!-- comments and processing instructions allowes here--&gt; &lt;!DOCTYPE books SYSTEM http://myserver.com/books.dtd &gt; &lt;!-- comments and processing instructions allowes here--&gt;</pre>                                                                                                                                                                |
| <b>Corpo</b>   | <pre>&lt;books&gt;   &lt;book category = "reference"&gt;     &lt;author&gt; Nigel Rees &lt;/author&gt;     &lt;title&gt; Sayings of the Century &lt;/title&gt;     &lt;price&gt;8.95 &lt;/price&gt;   &lt;book category = "fiction"&gt;     &lt;author&gt; Evelyn Waugh &lt;/author&gt;     &lt;title&gt; Sword of Honour &lt;/title&gt;     &lt;price&gt;12.99 &lt;/price&gt; &lt;/books&gt;</pre> |
| <b>Epílogo</b> | <pre>&lt;!-- comments and processing instructions allowes here--&gt;</pre>                                                                                                                                                                                                                                                                                                                          |

**Figura 3.1: Estrutura de um documento XML.**

Para Macoratti (2000, p.3) para que um documento XML seja válido temos que usar o que chamamos de definição do tipo de documento ou originalmente de DTD – *Document Type Definition*. O propósito da DTD é definir uma construção válida para um documento XML, e ela define a estrutura do documento usando uma lista de elementos válidos. A DTD pode ser declarada dentro de um documento XML ou em um arquivo a parte. O DTD permite descrever cada marca e fornecer regras para interpretar cada informação usada em um arquivo XML. Quando usada em um arquivo XML a DTD aparece entre colchetes [ e termina com um colchete ], seguido de um sinal de maior (>), como pode ser observado na Figura 3.2.

```
<?xml version = "1.0"?>

<!DOCTYPE note [
  <!ELEMENT aviso (para, de, cabecalho, corpo)>
  <!ELEMENT para (#PCDATA)>
  <!ELEMENT de (#PCDATA)>
  <!ELEMENT cabecalho (#PCDATA)>
  <!ELEMENT corpo (#PCDATA)>
]>

<aviso>
  <para> janice data="01/04/2000" </para>
  <de> Jefferson</de>
  <cabecalho> Lembre-se</cabecalho>
  <corpo> Amanha você tem prova de matemática </corpo>
</aviso>
```

**Figura 3.2: Um DTD - Document Type Definition**

Interpretando o código segundo Macoratti (2000, p.3) temos:

- **!DOCTYPE note** - Declara uma seção do documento com um DTD com o nome de nota.
- **!ELEMENT aviso** - Define o elemento “aviso” como tendo 4 elementos (para, de, cabeçalho e corpo).
- **!ELEMENT para** - Define o elemento “para” como sendo do tipo “CDATA”.
- **!ELEMENT de** - Define o elemento de como sendo do tipo “CDATA”.
- **!ELEMENT cabeçalho** - Define o elemento “cabeçalho” como sendo do tipo “CDATA”.
- **!ELEMENT corpo** - Define a elemento “corpo” como sendo do tipo “CDATA”.

Segundo Martin et al (2001, p.11) nos diz que em 1996 o *World Wide Consortium* ou W3C começou o projeto de uma abrangente linguagem de marcação que combinaria flexibilidade e capacidade do SGML com ampla aceitação ao HTML<sup>1</sup>. A criação do XML foi motivada pela complexidade que envolvia a linguagem de marcação SGML<sup>2</sup>, Martin et al (2001, p.11) afirma que a idéia de que o XML deveria ser mais simples que o SGML foi motivada por considerações sobre a facilidade de uso.

### 3.4 VB.NET

O Visual Basic .NET é uma das mais novas ferramentas de desenvolvimento da arquitetura .NET e foi a ferramenta escolhida para o desenvolvimento do *Web Service* deste projeto de pesquisa, pois, oferece os requisitos técnicos necessários e exigidos para o desenvolvimento deste.

Com algumas mudanças em relação a sua versão anterior (Visual Basic 6.0), foi incluído uma nova API<sup>3</sup> para programação, que tornou o Visual Basic .NET totalmente orientado a objetos, diferenciando-o de sua versão anterior que tinha estilo procedural que por sua vez era orientado a eventos.

Um outro aspecto que devemos considerar nessas mudanças é a profunda alteração sofrida nos diversos serviços de suporte fornecidos pelo sistema operacional e pelos diferentes

<sup>1</sup> *Hyper Text Markup Language* – Linguagem de Marcação de Texto Hipermédia, usada na publicação de conteúdo para a Web.

<sup>2</sup> *Standard Generalized Markup Language* – Linguagem Generalizada de Marcação Padrão, a primeira linguagem de marcação moderna (1986).

<sup>3</sup> *Application Program Interface* – Interface de Programa de Aplicação, conjunto de especificações de funções e procedimentos desenvolvidas para programação de aplicações.

produtos de suporte às aplicações, como os servidores Web, monitores de transação, serviços de suporte a distribuição de objetos, etc. Nesse aspecto, toda a arquitetura do ambiente Windows sofreu profundas alterações, pelo menos no que se refere àquela camada dessa arquitetura que é exposta ao programador.

Segundo Macoratti (2004, p. 1) Com o advento da plataforma .NET, programar ficou muito mais fácil para quem usava o Visual Basic (versão anterior ao VB .Net) . O VB .NET incorpora um conjunto de classes com poderosos recursos que facilitam a codificação e poupam o tempo de desenvolvimento.

### 3.5 ASP.NET

O Visual ASP.NET faz parte do pacote de ferramentas de desenvolvimento da plataforma .Net e foi escolhida para o desenvolvimento da aplicação cliente (usuária) que através de métodos remotos fará chamadas ao *Web Service*. O ASP.Net oferece todos os requisitos técnicos necessários e exigidos para o desenvolvimento desta aplicação.

Nos primórdios da *World Wide Web*, o *browser*<sup>4</sup> era apenas um aplicativo que recebia e interpretava arquivos escritos em HTML.

Segundo Damasceno Junior (2001, p. 4) uma das primeiras tentativas de usar a Web como plataforma para aplicações de rede foi o CGI<sup>5</sup> que provia um mecanismo pela qual a execução de um programa era solicitada pelo *browser* junto com o envio de eventuais dados. Esse programa processava os dados e enviava de volta dinamicamente linhas para construção de nova página.

Em 1995, foi lançado o Java com a idéia de uma *Java Virtual Machine* que, embutida no *browser*, recebia, interpretava e processava aplicações, agora no cliente. Segundo Damasceno Junior (2001, p.4) isso seria a revolução das *applets*, que acabou não acontecendo. A grande questão era a arquitetura Java que trazia um modelo de trabalho com componentes que até então eram conflitantes com a arquitetura COM<sup>6</sup> da Microsoft. Quando esta por sua vez descobriu onde estava entrando, apoiando entusiasticamente o Java, recuou; congelou a versão do Java suportada pelo Internet Explorer e lançou outra tecnologia: as *Active Server Pages* (ASP).

<sup>4</sup> Software utilizado como navegador da web.

<sup>5</sup> *Common Gateway Interface* – Interface comum para *Gateway* de Internet, que era o servidor de páginas provendo algum mecanismo de controle do conteúdo destas páginas.

<sup>6</sup> *Component Object Model* – modelo para aplicações distribuídas na Internet. Diferente do CGI que processava todo o seu conteúdo de uma vez, o objeto COM dividia o processamento entre cliente e servidor.

O ASP.NET roda sobre a CLR da plataforma .Net. Segundo Marco Antônio Gutierrez (2002, p.253) nos diz que ainda que tenha sido mantido toda compatibilidade com todo o código ASP já produzido o *framework* .Net introduz profundas alterações no modelo de desenvolvimento de aplicações web destinadas ao MS *Internet Information Server*. Todo o *framework* .Net está, desde já, à disposição das aplicações Web, que podem agora ser escritas em qualquer linguagem que atenda às especificações da CLR, além de tradicionais linguagens de script hoje suportadas.

As novas aplicações ASP.Net incorporam todos os recursos do *framework* .Net, a começar pela CLR até a compilação de código nativo de todos os programas, feito pelo mesmo *JIT* disponível para aplicações *desktop*. Além disso, todo um esforço de otimização foi feito pela Microsoft, visando dotar suas aplicações do máximo de desempenho possível. Igualmente simples de administrar, uma aplicação ASP.Net passa a ser configurada por arquivos XML.

Haddad (2004) profissional certificado da Microsoft e palestrantes sobre a tecnologia .Net conclui que:

“O ASP.NET é atualmente o meio mais produtivo e rápido de se criar aplicações para Internet, além, é claro, de oferecer melhor performance e facilidade de manutenção.”

### 3.6 WEB SERVICES

Há alguns anos atrás quando você precisava incluir determinada funcionalidade em seu sistema, você dizia “-Eu preciso de uma rotina”. Então você pedia uma rotina para outro programador ou fazia sua própria. Depois das rotinas vieram as classes, e tudo ficou mais fácil. Mais tarde você disse: “-Eu preciso de um componente”, e por horas procurava nos sites por aquele componente. Depois os componentes foram distribuídos por uma rede e surgiu a computação de objetos distribuídos. E finalmente, hoje e daqui por diante quando você precisar realizar uma determinada tarefa, você vai dizer “-Eu preciso de um *Web Service*”.

Segundo Gutierrez (2002, p. 253) os *Web Services* são construídos em torno de conceitos tais como que seja possível construí-los em qualquer plataforma de hardware e software, o que pode torná-los largamente utilizados. Trata-se apenas de um padrão de interoperabilidade entre componentes de software na Internet (Web).

Os *Web Services* permitem que empresas e corporações compartilhem lógica de negócio (B2B – Negócio pra negócio), de forma que você pode utilizar um determinado serviço de outra empresa pra incluir funcionalidade ao seu software.

O *Web Service* é provavelmente a tecnologia mais interessante e mais eficaz para troca de informações pela Internet (Web). Eles permitem que diferentes ramos de negócios, mesmo utilizando distintas tecnologias e plataformas, conectem-se de maneira padrão, executem procedimentos remotos sobre o protocolo HTTP<sup>7</sup>, o que possibilita seu uso por meio de *firewalls*, sem necessidade de abrir brechas na segurança corporativa.

Um grande ponto positivo desta tecnologia (*Web Service*) é que a criação de servidores e clientes independem da linguagem de programação e do sistema operacional que são implementados.

### 3.6.1 Características e Divisão

Um *Web Service* é uma aplicação publicada, localizada e chamada através da Internet. Sua função é de encapsular, contratar funções e objetos remotos oferecidos via um protocolo padrão e conhecido, conforme nos explica Camelo (2002, p.5).

Camelo (2002, p.5) nos explica ainda que entre uma aplicação distribuída e um *Web Service* não existem muitas diferenças, contudo a forma como seus dados são manipulados e como as camadas de comunicação se comportam se dá uma forma bastante peculiar. Desde que a arquitetura de *Web Service* representa um novo paradigma para aplicações distribuídas, esta consiste em três componentes básicos, conforme o quadro abaixo:

- Um ***Service Broker*** que atua na localização entre o provedor do serviço e o requisitor.
- Um ***Service Provider*** que publica seus serviços para o *Service Broker*.
- E finalmente um requisitor de serviço que atua solicitando o serviço para um ***Service Broker*** que encontra um provedor disponível e repasse o pedido para o provedor.

Este tipo de *middleware*<sup>8</sup> faz uso da porta HTTP para comunicação. Como os *firewalls* convencionais e *proxies* não bloqueiam esta porta não existem grandes restrições para o uso deste tipo de aplicação em redes de longas distâncias.

<sup>7</sup> *Hyper Text Transport Protocol* – Protocolo de Transporte de Hiper Texto, padrão largamente utilizado na transferência de conteúdo na Web.

<sup>8</sup> Camada de software intermediária entre aplicações, sejam elas aplicações *Web* ou *Desktop*.

A comunicação entre aplicações de *Web Services* faz uso de quatro camadas que empacotam a requisição e a resposta entre um servidor e um cliente. As Camadas utilizadas são as seguintes:

- XML (Extensible Markup Language)
- SOAP (Simple Object Access Protocol)
- WSDL (*Web Services Definition Language*)
- UDDI (Universal Discovery Description Integration)

### 3.6.2 Especificações Técnicas

Segundo Macoratti (2000, p.1) um *Web Service* é armazenado em um arquivo texto com extensão .ASMIX que não necessita ser compilado, ou seja, você pode usar o ambiente do Visual Studio .Net ou até mesmo o bloco de notas para gerar este arquivo. Depois de gerado, basta disponibilizá-lo na Web e acessá-lo via uma URL<sup>9</sup> no *browser* como acessamos um arquivo .ASPX, ou com outra extensão.

A primeira instrução do programa é uma diretiva do tipo “<%@ **WebService Language="VB" Class="Nome da Classe" %>”** que informa que o código da página deve ser disponibilizado como um *Web Service* e que a classe deve ser instanciada para atender às requisições direcionadas para o serviço. Estas diretivas em outras palavras executam duas funções:

1. Indica ao .NET *Framework* para processar o arquivo como um *Web Service*.
2. Indica a linguagem a ser usada e o nome da classe declarada no arquivo.

Segundo Gutierrez (2002, p. 266) nos diz que um *Web Service* pode herdar da classe *System.Web.Service*. Essa classe dispõe de algumas funções que normalmente podem ser de interesse. A primeira delas é a propriedade *Context*, que fornece acesso ao contexto HTTP corrente, um objeto da classe *HttpContext*. A Segunda é a propriedade *Session.HttpSessionState*, contendo informações sobre o estado da sessão corrente.

Algo relevante na construção de um *Web Service* é que devemos assinalar através de um atributo que o método deve ser exposto como um método para acesso remoto.

### 3.6.3 Arquitetura da Web Com XML

<sup>9</sup> Uniform Resource Locator, forma padronizada de endereçamento de conteúdo na Web.

Para poder comentar sobre a arquitetura da Internet (Web) com os arquivos XML é necessário levantar as características da arquitetura clássica da Web.

Na arquitetura clássica da Web, a aplicação cliente é um browser Web agindo em nome de um agente humano. O browser transmite um pedido de uma página a um servidor HTTP. O pedido é acompanhado por uma série de nomes e valores de parâmetros. Estes podem ser anexados ao URL da página (HTTP GET) ou enviados separadamente (HTTP POST). Os parâmetros e seus nomes são determinados pela aplicação e devem ser divulgados ao cliente, incluindo-os na página que os pede. Conseqüentemente, o desenvolvedor da aplicação do servidor deve criar também a página do cliente ou reverter à engenharia da estrutura do pedido da página ao cliente. Tal reversão de engenharia está sujeita a falhas sempre que o servidor modificar a estrutura do pedido, conforme nos explica Didier Martin et al (2001, p.14).

Esta arquitetura funciona bem o suficiente, é claro, apesar de ter algumas limitações bastante conhecidas:

- Estamos restritos a cliente browser.
- Ninguém constrói páginas de servidor para se comunicar com um agente de programação ou com outra aplicação.
- Todo conteúdo é enviado como HTML. Isto limita a habilidade do cliente de efetuar qualquer pós-processamento, e o usuário está limitado à visualização determinada pela aplicação do servidor no momento da entrega.
- A aplicação é frágil; o cliente e o servidor precisam estar em sincronia. A menor disfunção na geração de um pedido normalmente irá derrubar a aplicação do servidor.

Conforme Martin et al (2001, p.15) um cliente – browser ou programa – envia um documento XML como um pedido a um servidor. Ele contém parâmetros identificados do modo bem parecido com o que fazem as aplicações Web. Ao contrário de um cliente tradicional, porém, a estrutura do pedido pode ser oficialmente especificada usando um mecanismo padronizado obtido no tempo de funcionamento de um servidor. Este mecanismo documenta a estrutura esperada pelo servidor e também permite ao cliente verificar a correção do pedido antes da transmissão. Uma vez que o pedido chegue o servidor pode processar do mesmo modo que aplicações tradicionais da Web, ou pode trazer alguma ajuda. O XML não trata de comunicação servidor a servidor em si, mas as partes inovadoras dentro da comunidade XML usaram esta tecnologia para formalizar tais comunicações. Uma vez que os

dados que um servidor receberia em tal troca também é XML, é bastante fácil para o primeiro servidor fundir vários documentos ou transformar um documento em outro formato para satisfazer um pedido.

Portanto o documento XML é a base para comunicação entre os *Web Services*.

### 3.6.4 SOAP e Web Services

Na visão de Cunha (2002, p.1) os *Web Services* são identificados por um URI (*Unique Resource Identifier*), e são descritos e definidos usando XML. Um dos motivos que tornam *Web Services* atrativos é o fato de este modelo ser baseado em tecnologias *Standards*, em particular XML e HTTP. *Web Services* São utilizados para disponibilizar serviços interativos via Web, podendo ser acessados por outras aplicações. O protocolo de troca de informações SOAP utilizado na comunicação entre aplicações e *Web Services*, está se tornando padrão para troca de mensagens, já que é uma tecnologia construída com base em XML e HTTP.

O protocolo SOAP é um protocolo utilizado para estabelecer comunicação e invocar métodos em um ambiente heterogêneo de funções. Davi Cunha (2002, p.1) também afirma que:

“SOAP é um protocolo utilizado para invocar aplicações remotas, através da RPC (*Remote Procedure Calls*) ou trocas de mensagens, em um ambiente independente de plataformas e linguagem de programação.”

Camelo (2002, p.6) afirma que:

“O SOAP é um protocolo que estabelece a comunicação em um ambiente heterogêneo de funções de forma independente. Sua função na cadeia é basicamente definir mecanismos que expressem o modelo semântico das aplicações, e o modelo em que os dados são decodificados.”

SOAP é, portanto, na visão de Cunha (2002, p.1) um padrão normalmente aceito para utilizar-se com *Web Services*. Dessa forma, pretende-se garantir a interoperabilidade e intercomunicação entre diferentes sistemas, através da utilização de uma linguagem (XML) e mecanismo de transporte (HTTP).

Segundo Santana (2004, p.1) o protocolo SOAP tem diversas vantagens sobre outras maneiras de chamar funções remotamente como DCOM, CORBA ou diretamente no TCP/IP.

- É simples de implementar, testar e usar.
- É um padrão criado por um consórcio W3C e por várias empresas.

- Usa os mesmos padrões Web para quase tudo: a comunicação é feita via HTTP com pacotes virtualmente idênticos; os protocolos de autenticação e encriptação são os mesmos; a manutenção de estado é feita da mesma forma; é normalmente implementado pelo próprio servidor Web.
- Atravessa “*firewalls*” e roteadores, que pensam que é uma comunicação HTTP.
- É independente do sistema operacional e CPU.
- Pode ser usado tanto na forma anônima como com autenticação (nome/senha).

Na visão de Cunha (2002, p.1) as características do protocolo SOAP têm a seguinte formatação ver Figura 3.0:

**Envelope:** Toda mensagem SOAP deve contê-lo. É o elemento raiz do documento XML. O envelope deve conter declarações de namespaces e também atributos adicionais como o que define o estilo de codificação (encoding style). Um encoding style define como os dados são representados no documento XML.

- **Header:** É um cabeçalho opcional. Ele carrega informações adicionais como, por exemplo, se a mensagem deve ser processada por um determinado nó intermediário. Quando utilizado, o *Header* deve ser o primeiro elemento do nó intermediário.
- **Body:** Este elemento é obrigatório e contém o *payload*, ou a informação a ser transportada para o seu destino final. O elemento *Body* pode conter um elemento opcional *fault*, usado para carregar mensagens de status e erros retornados pelos “nós” a o processarem a mensagem.

A chamada de procedimento remoto ou RPC, segundo Paulo Roberto Rizzone (2000, p.26) é um mecanismo de comunicação fundamental. Permite que clientes façam chamadas de procedimentos em servidores remotos como se fossem chamadas de procedimento local.

Cunha (2002, p.2) nos diz que as chamadas antes de serem enviadas pela rede, as chamadas de RPC são encapsuladas (ou serializadas) segundo o padrão SOAP. O serviço cliente, ao receber a mensagem faz o processo contrário, desencapsulando-a e extraiendo as chamadas do método. A aplicação servidor então processa esta chamada, e envia uma

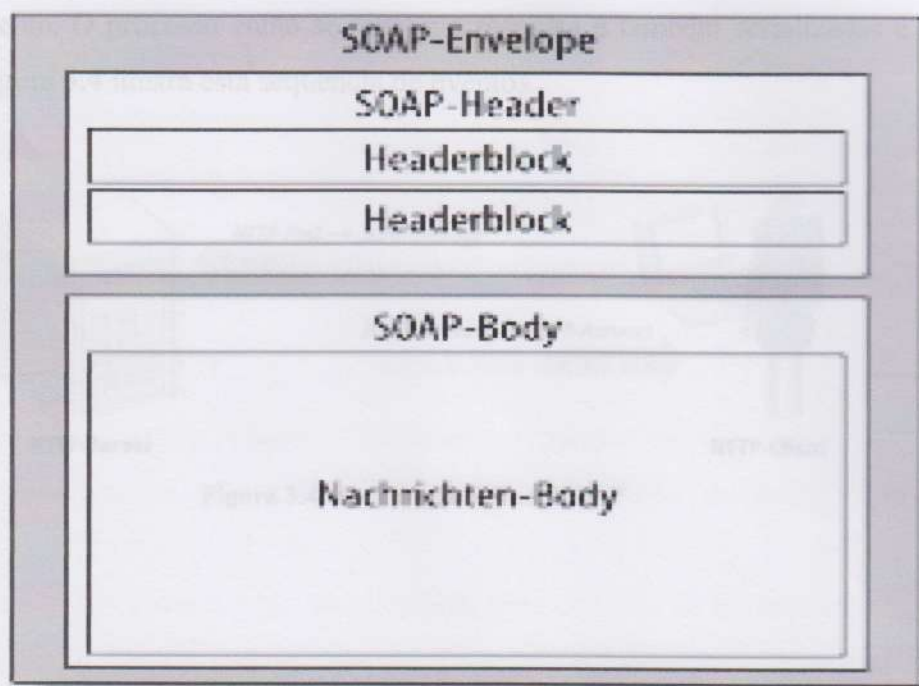


Figura 3.3: Estrutura de um documento SOAP

Os pedidos SOAP segundo Mauro Santana (2004, p.1) podem ser feitos em três padrões: **GET**, **POST** e **SOAP**. Os padrões **GET** e **POST** são idênticos aos pedidos feitos por navegadores Internet. O **SOAP** é um padrão semelhante ao **POST**, mas os pedidos são feitos em XML e permitem recursos mais sofisticados como passar estruturas e *arrays*.

Independente de como seja feito o pedido, as respostas são sempre em XML. O XML descreve perfeitamente os dados em tempo de execução e evita problemas causados por inadvertidas mudanças nas funções, já que os objetos chamados têm a possibilidade de sempre validar os argumentos das funções, tornando o protocolo muito robusto.

### 3.6.5 SOAP e RPC

A chamada de procedimento remoto ou RPC, segundo Paulo Roberto Riccione (2000, p.26) é um mecanismo de comunicação fundamental. Permite que clientes façam chamadas de procedimentos em servidores remotos como se fossem chamadas de procedimento local.

Cunha (2002, p.2) nos diz que as chamadas antes de serem enviadas pela rede, as chamadas de RPC são encapsuladas (ou serializadas) segundo o padrão SOAP. O serviço remoto, ao receber a mensagem faz o processo contrário, desencapsulando-a e extraindo as chamadas do método. A aplicação servidora então processa esta chamada, e envia uma

resposta ao cliente. O processo então se repete: a resposta é também serializada e enviada pela rede. A Figura 3.4 ilustra esta sequência de eventos.

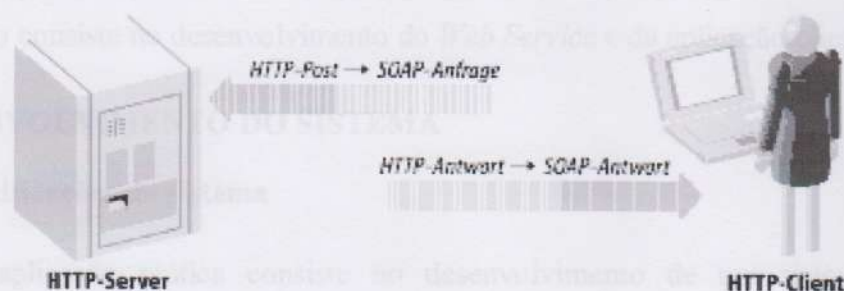


Figura 3.4: Solicitação Cliente SOAP

## 4 DESENVOLVIMENTO

Este capítulo detém-se ao desenvolvimento da aplicação prática outrora mencionada. Esta aplicação consiste no desenvolvimento do *Web Service* e da aplicação cliente.

### 4.1 DESENVOLVIMENTO DO SISTEMA

#### 4.1.1 Especificações do sistema

Esta aplicação prática consiste no desenvolvimento de um sistema através da linguagem de programação Visual Basic .Net e da plataforma de desenvolvimento da Microsoft .Net.

O Sistema a ser desenvolvido se baseia em uma situação de monitoramento remoto. Esta situação consiste que um cliente (usuário) que esteja em um determinado ambiente seja ele qual for possa enxergar (monitorar) outro local (ambiente), seja, este local (ambiente) próximo ou distante. O Hardware (*WebCam*) estará disposto em algum local de uma sala, por exemplo, e o cliente (usuário) deve estar fazendo acesso através do browser de sua máquina, daí a definição da necessidade da criação de uma aplicação cliente (usuário) e de uma aplicação servidora.

A Figura 4.1 ilustra claramente uma situação de monitoramento remoto referida neste projeto.

Para o desenvolvimento desta aplicação darei ênfase à comunicação entre o cliente (usuário) com o *Web Service* (servidor) que por sua vez fará comunicação com o Hardware (*WebCam*) através de seus *Webmethod's* que iram fazer acesso a DLL do Hardware (*WebCam*).

O Cliente (usuário) através do sistema fará uso das funções contidas no mesmo sistema para fazer acesso a o *Web Service* (servidor).

Para o desenvolvimento completo desta solução (desenvolvimento da aplicação cliente e servidor) deverão ser seguidas cinco etapas que descreverão todo o caminho passo a passo para conclusão da solução.

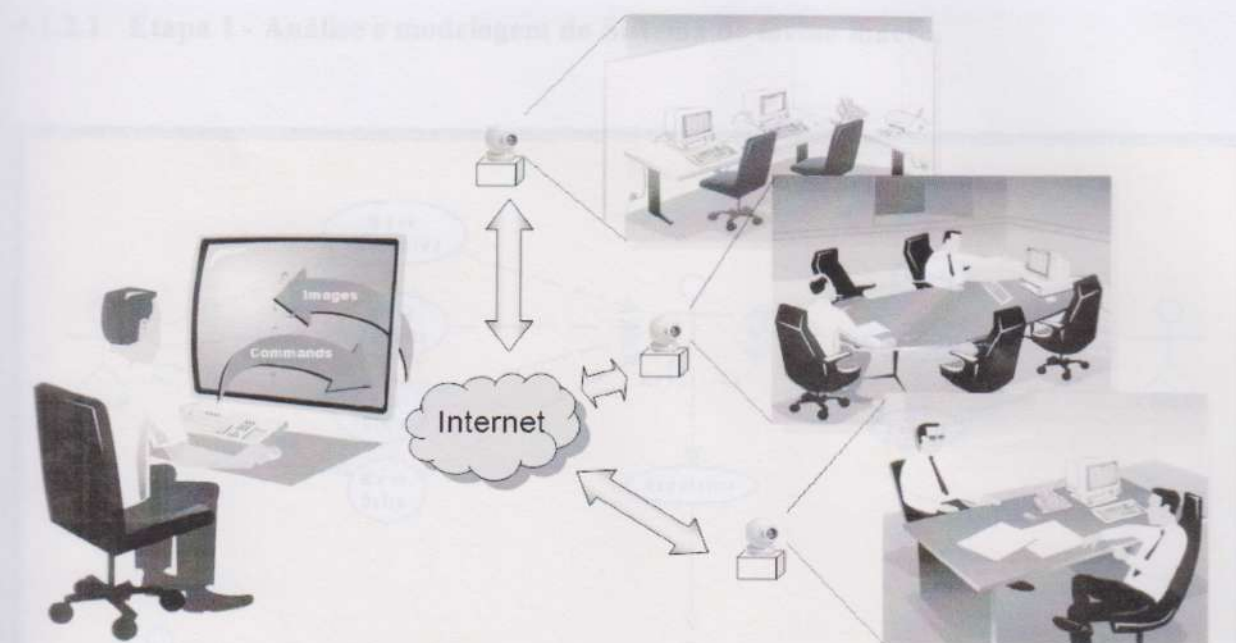


Figura 4.1: Usuário monitorando remotamente um ambiente

#### 4.1.2 Etapas do desenvolvimento da solução

As etapas para o desenvolvimento da solução são as seguintes: Análise e modelagem do Sistema de forma macro, modelagem da aplicação servidora, modelagem da aplicação cliente, por fim a exposição do código da aplicação. Serão utilizados em algumas etapas de desenvolvimento os modelos da linguagem UML (*Unified Modeling Language*) como os modelos Caso de Uso (*Use case*).

- **Etapa 1 - Análise e modelagem do Sistema de forma macro:** Nesta etapa o sistema será modelado de forma macro, não levando em consideração os detalhes de sua implementação, em outra etapa discutiremos tais detalhes.
- **Etapa 2 - Modelagem da aplicação servidora:** Nesta etapa do desenvolvimento do sistema modelaremos a aplicação servidora que fará acesso a o Hardware (*Web Can*).
- **Etapa 3 - Modelagem da aplicação cliente:** Nesta etapa do desenvolvimento do sistema modelaremos a aplicação cliente que fará acesso a o nosso *Web Service* (Servidor).
- **Etapa 4 - Exposição do código da aplicação:** Nesta etapa do desenvolvimento do sistema será exposto o código da aplicação com seus devidos detalhes de implementação.

#### 4.1.2.1 Etapa 1 - Análise e modelagem do Sistema de forma macro.

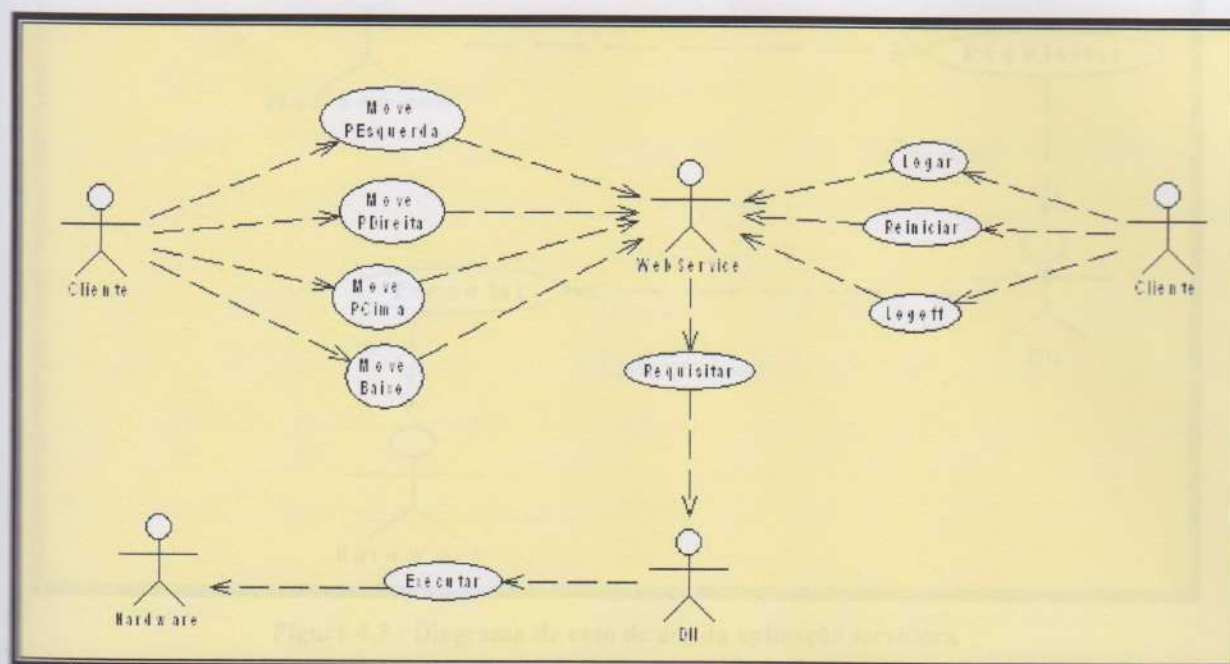


Figura 4.2: Diagrama de caso de uso de toda a aplicação

Este diagrama (caso de uso) demonstra de forma global o funcionamento do Sistema.

Os processos que envolvem o sistema são os seguintes:

- O ator **Cliente** comunica-se através dos seus métodos (conectar, girar esquerda e girar direita) com o ator **Web Service** (servidor).
- O ator **Web Service** (servidor) comunica-se através do método (requisitar) com o ator **DLL**.
- O ator **DLL** através de o método executar fará comunicação com o ator **Hardware** (*WebCam*).

#### 4.1.2.2 Etapa 2 – Modelagem da aplicação Servidora

- Caso de uso da modelagem do *Web Service* do Sistema.

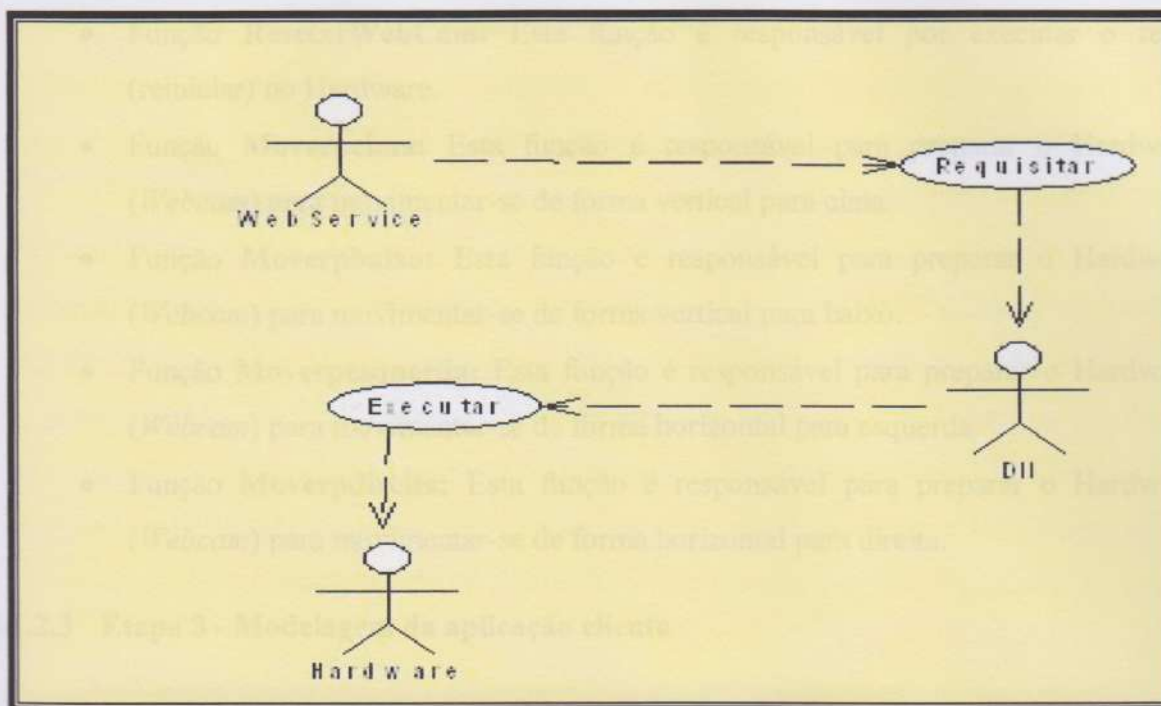


Figura 4.3 : Diagrama de caso de uso da aplicação servidora

Este caso de uso expõe o modelo de funcionamento do *Web Service* (servidor).

É importante observar e nunca é demais lembrar que qualquer linguagem de programação pode criar um *Web Service* capaz de ser executado em qualquer servidor web rodando sob qualquer SO (Sistema Operacional) e em qualquer plataforma de Hardware.

O *Web Service* nada mais é do que um método que será invocado remotamente a partir de nosso cliente (aplicação). O cliente fará a invocação do(s) método(s) do *Web Service* utilizando o mecanismo de invocação baseado em SOAP cujo transporte será feito pelo protocolo HTTP.

Na arquitetura do *Web Service* os métodos são expostos e acessados via Internet (Web) graças à declaração no Visual Basic .Net do `<Webmethod ()>`. Sem esta declaração não conseguiríamos enxergar as funções a serem executadas.

Os *Webmethod's* que irão constar em nossa aplicação servidora são os seguintes:

- Função **LoginWebCam**: Esta função é responsável por efetuar-mos o login no Hardware. Esta função é de grande relevância, pois, será a partir dela que as outras funções serão executadas.
- Função **LogoffWebCam**: Esta função é responsável por efetuar-mos o logoff no Hardware.

- Função **ResetarWebCam**: Esta função é responsável por executar o reset (reiniciar) no Hardware.
- Função **Moverpcima**: Esta função é responsável para preparar o Hardware (*Webcam*) para movimentar-se de forma vertical para cima.
- Função **Moverpbaixo**: Esta função é responsável para preparar o Hardware (*Webcam*) para movimentar-se de forma vertical para baixo.
- Função **Moverpesquerda**: Esta função é responsável para preparar o Hardware (*Webcam*) para movimentar-se de forma horizontal para esquerda.
- Função **Moverpdireita**: Esta função é responsável para preparar o Hardware (*Webcam*) para movimentar-se de forma horizontal para direita.

#### 4.1.2.3 Etapa 3 - Modelagem da aplicação cliente

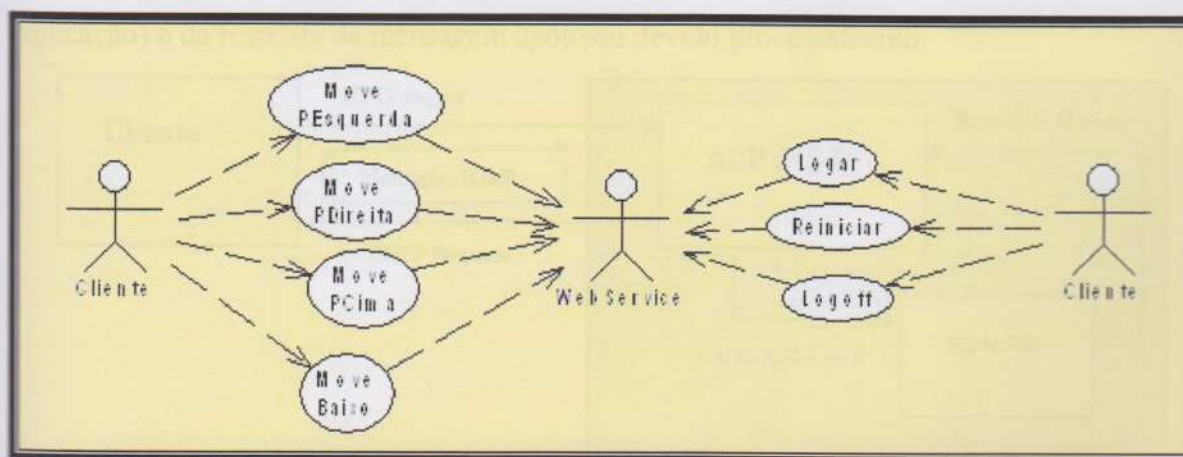


Figura 4.4 : Diagrama de caso de uso da aplicação cliente

Este caso de uso expõe o modelo de funcionamento da aplicação cliente do sistema (usuário).

É importante observar que a aplicação cliente a ser desenvolvida compartilha de alguns recursos da aplicação servidora no que diz respeito a o desenvolvimento da mesma em plataforma heterogênea, pois, a aplicação cliente como a servidora não depende exclusivamente de uma arquitetura específica de desenvolvimento e proprietária.

Esta aplicação Poderia ser desenvolvida utilizando como ferramenta de programação o Java da Sun Micro Systems ou até mesmo a ferramenta Delphi da Borland como também em Visual Basic .Net. Escolhi para o desenvolvimento deste sistema como outrora mencionado o Visual Basic .Net para a criação do *Web Service* e para aplicação cliente utilizaremos o ASP .Net também uma ferramenta pertencente a o pacote de desenvolvimento da plataforma .Net.

A aplicação cliente (usuário) fará vários pedidos a o *Web Service* para solicitar os seus serviços que serão utilizados para a manipulação da *WebCam*. Os pedidos (*request's*) serão transportados pelo protocolo HTTP, estes pedidos transportarão uma mensagem SOAP.

A infra-estrutura ASP .Net se encarrega de acessar a descrição padronizada do *Web Service* e gerar classes *proxies*, que serão chamadas localmente pela aplicação e que se encarregarão de despachar a requisição. A invocação remota seja RPC (*Remote Procedure Calls*) ou RMI (*Remote Method Invocation*) da tecnologia Java. Quando a aplicação cliente – que pode ser tanto uma aplicação *desktop* quanto uma aplicação web – invoca o método remoto, o *proxie* é chamado e converte a chamada numa mensagem SOAP, despachando-a através de um POST HTTP para o *Web Service* remoto. Quando o *response* HTTP é recebido, o *Proxy* converte a mensagem SOAP num retorno de método e o devolve para a aplicação cliente.

A Figura 4.5 demonstra o funcionamento da solicitação (*request*) feita pelo cliente (aplicação) e da resposta da mensagem após seu devido processamento.

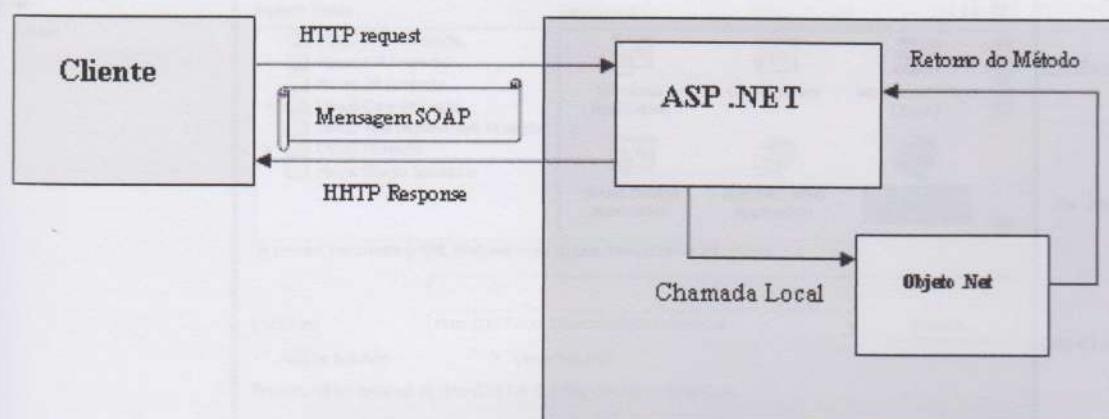


Figura 4.5 : Funcionamento de uma requisição cliente

O *Proxy* é uma classe herdada de `System.Web.Services.Protocols.SoapHttpClientProtocol` que programa todo código necessário para o acesso a o método remoto. Sua propriedade URL aponta para o URL onde está localizado o serviço. Quando a aplicação cliente invoca o método remoto utilizando o método `Getprice` do *Proxy*, este redireciona a chamada utilizando seu método herdado `Invoke`. É este método que converte a chamada numa mensagem SOAP e a despacha para o destino utilizando um HTTP POST. Quando a mensagem SOAP é devolvida, o método a devolve já convertida para o tipo apropriado.

#### 4.1.2.4 Etapa 4 - Exposição do código da aplicação.

Nesta etapa irei expor todo o código do desenvolvimento da aplicação servidora (*Web service*) e cliente (usuário), com os devidos comentários de implementação de cada aplicação.

Para tal dividi em duas partes para melhor visualização, entendimento e compreensão do código.

##### 4.1.2.4.1 Exposição do código da Aplicação Servidora (*Web Service*).

Para demonstrar mais claramente a criação do *Web Service* daremos ênfase neste tópico a todo o código e as telas utilizadas na criação do *Web Service* e da aplicação cliente.

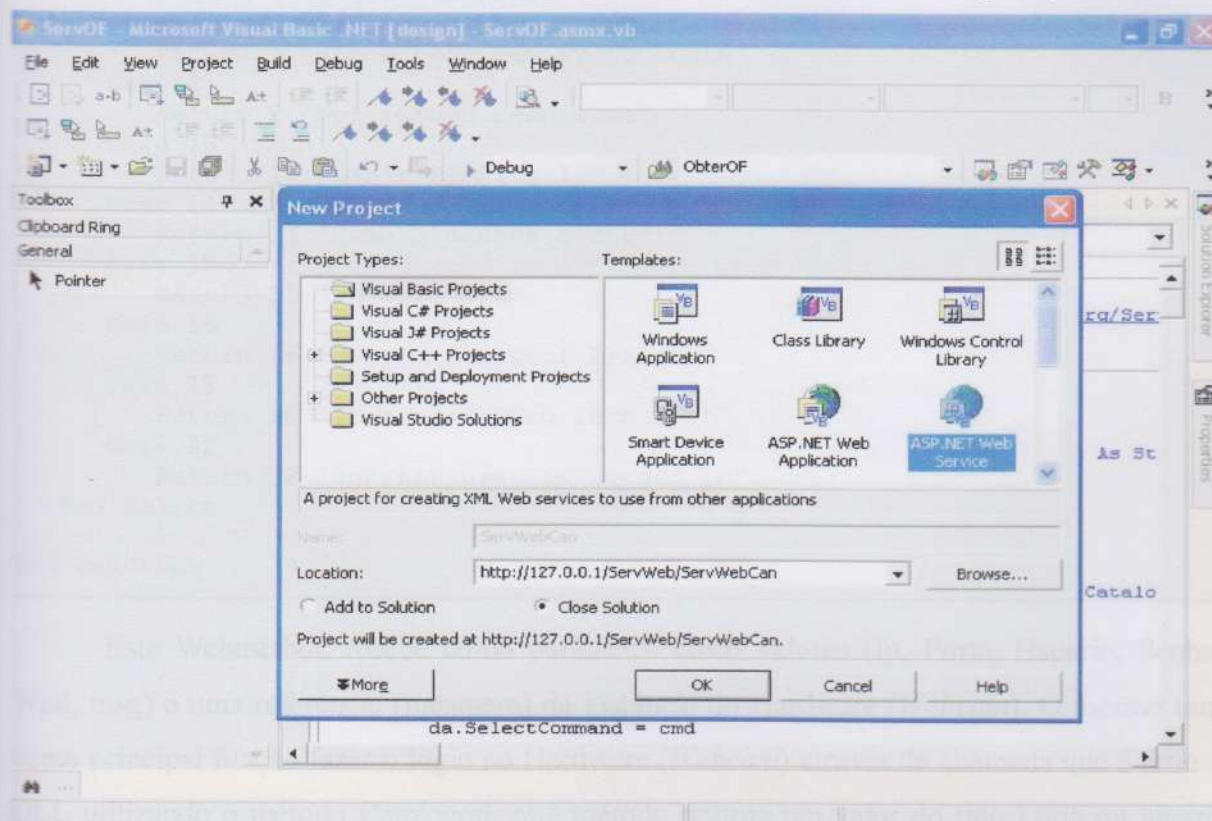


Figura 4.6 : Tela de Criação do *Web Service*

O desenvolvimento do *Web Service* utilizando o Visual Basic .Net foi dividido em *Webmethod's*, ou seja, métodos publicados que irão ser acessados remotamente por uma aplicação cliente (usuário). Seja esta aplicação Desktop ou Web (browser).

Os *Webmethod's* que constam dentro de nosso *Web Service* são os seguintes: **LoginWebcam**, **LogoffWebcam**, **ResetarWebcam**, **Moverpcima**, **Moverpbaixo**, **Moverpesquerda**, **Moverpdireita**. Cada *Webmethod* fará uma chamada a Dll que por sua vez executará as funções no Hardware:

Quadro 1: Código do webmethod LoginWebcam

```

<WebMethod()>
Public Function LoginWebcam(ByVal Ip As String, ByVal Porta As Integer,
ByVal Usuario As String, ByVal Senha As String, ByVal Wnd As Long, ByVal
msg As Integer) As Integer

    Dim Login As Long

    'Enchendo a Variável LpCamera Lenilson
    Login = Camlogon (Ip, Porta, Usuario, Senha, Wnd, msg, lpCamera)

    'Chamando a função da Dll Através do método do Webservice
    'Fazendo as Devidas Verificações Lenilson
    Select Case Login
        Case 0
            Return 0 'Login Succesful
        Case 1
            Return 1 'Not enough memory to login"
        Case 2
            Return 2 'Not enough resources"
        Case 13
            Return 13 'Can't Resolve IP"
        Case 14
            Return 14 'Camera cannot connect"
        Case 15
            Return 15 'Network Error"
        Case 16
            Return 16 'Camera Protocol Error"
        Case 18
            Return 18 'ID or Password Incorrect"
        Case 22
            Return 22 'Another user is logged in"
    End Select
End Function

```

Este Webmethod recebe como parâmetro cinco valores (Ip, Porta, Usuario, Senha, Wnd, msg) e uma referência (lpcamera) da instância do Hardware (*Webcam*). O mesmo tem como principal função fazer o login no Hardware (*Webcam*) através da chamada que é feito à DLL utilizando o método Camlogon, este método retorna um valor do tipo Long ou inteiro que pode ser manipulado para programar a mensagem que será dada ao cliente (usuário).

Na Figura 4.7, pode-se observar execução do código acima produzindo a tela do Webmethod.

Service1 Web Service - Microsoft Internet Explorer

Arquivo Editar Exibir Favoritos Ferramentas Ajuda

Endereço <http://127.0.0.1/ServWeb/ServWebCam/service1.aspx?op=LoginWebCam> Ir Links

msn! Busca Marcador Opções Pop-ups Bloqueados (91) Hotmail

## Service1

Clique [aqui](#) para obter uma lista completa das operações.

### LoginWebCam

**Teste**

Para testar a operação usando o protocolo HTTP POST, clique no botão 'Invocar'.

| Parâmetro | Valor                |
|-----------|----------------------|
| Ip:       | <input type="text"/> |
| Porta:    | <input type="text"/> |
| Usuario:  | <input type="text"/> |
| Senha:    | <input type="text"/> |
| Wnd:      | <input type="text"/> |
| msg:      | <input type="text"/> |

Chamar

SOAP

**Figura 4.7: Tela do Webmethod Loginwebcam**

A seguir é apresentada uma amostra de solicitação e de resposta SOAP. Os espaços reservados mostrados precisam ser substituídos por valores reais.

A comunicação é feita através de um arquivo Xml. Neste arquivo contém informações relevantes da solicitação feita, como: nome do método e parâmetros a serem preenchidos.

- Nome do Método: <LoginWebcam: xmlns = "http://tempuri.org/ServWebcam/Service1">
- Os valores a serem preenchidos são os seguintes: <Ip> string</Ip>, <Porta> string</Porta>, <Usuario>String</Usuario>, <Senha> String </Senha>, <Wnd> Long </Wnd>, <msg>int</msg>.

```

POST /ServWeb/ServWebCan/service1.smx HTTP/1.1
Host: 127.0.0.1
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tanpuri.org/ServWebCan/Service1/LoginWebCam"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <LoginWebCam xmlns="http://tanpuri.org/ServWebCan/Service1">
      <Ip>string</Ip>
      <Porta>int</Porta>
      <Usuario>string</Usuario>
      <Senha>string</Senha>
      <Web>long</Web>
      <neg>int</neg>
    </LoginWebCam>
  </soap:Body>
</soap:Envelope>

```

Figura 4.8 : Solicitação SOAP feita pelo método LoginWebcam.

A resposta SOAP contém a informação do retorno do processamento da solicitação. O campo <LoginWebcamResult> int </LoginWebcamResult> conterá o valor da resposta.

```

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <LoginWebCamResponse
xmlns="http://tanpuri.org/ServWebCan/Service1">
      <LoginWebCamResult>int</LoginWebCamResult>
    </LoginWebCamResponse>
  </soap:Body>
</soap:Envelope>

```

Figura 4.9 : Resposta SOAP feita pelo método LoginWebcam.

A seguir é apresentada uma amostra de solicitação e resposta HTTP POST. Os espaços reservados mostrados precisam ser substituídos por valores reais. Os espaços a serem

substituídos na solicitação HTTP POST por valores reais são os seguintes: Ip, Porta, Usuario, Senha, Wnd e msg.

```
POST /ServWeb/ServWebCam/service1.saw/LoginWebCam HTTP/1.1
Host: 127.0.0.1
Content-Type: application/x-www-form-urlencoded
Content-Length: length

Ip=stringPort=stringUsuario=stringSenha=stringWnd=stringmsg
```

Figura 4.10: Solicitação HTTP POST SOAP feita pelo método LoginWebcam.

A resposta HTTP POST SOAP contém o valor do retorno do processamento da solicitação HTTP POST SOAP.

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8">
<int xmlns="http://tempuri.org/ServWebCam/Service1">int</int>
```

Figura 4.11: Resposta HTTP POST SOAP feita pelo método LoginWebcam.

#### Quadro 2: Webmethod LogoffWebcam

```
Public Function LogoffWebcam (ByVal lpCam As Long) As Long
    'Chamando o método para Resetar a câmera
    CamLogoff(lpCam)
End Function
```

Este Webmethod recebe como parâmetro um valor (Lpcam) Long ou inteiro e tem como principal função fazer o logoff no Hardware (*Webcam*) através da chamada que é feita a Dll através do método Camlogoff.

A Figura 4.12 mostra a execução deste *Webmethod*.

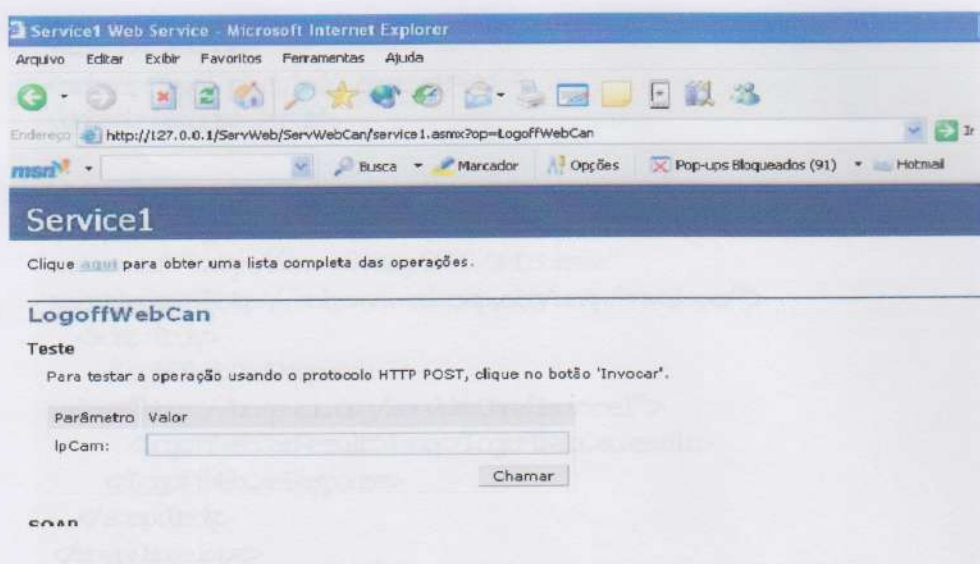


Figura 4.14: Resposta SOAP feita pelo método LogoffWebcam

Figura 4.12 : Tela do Webmethod LogoffWebcam

A seguir é apresentada uma amostra de solicitação e de resposta SOAP. Os espaços reservados mostrados precisam ser substituídos por valores reais. Os valores reais a serem substituídos são os seguintes: `<lpCam> Long </lpCam>`.

```
POST /ServWeb/ServWebCan/service1.aspx HTTP/1.1
Host: 127.0.0.1
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/ServWebCan/Service1/LogoffWebCam"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <LogoffWebCam xmlns="http://tempuri.org/ServWebCan/Service1">
      <lpCam>Long</lpCam>
    </LogoffWebCam>
  </soap:Body>
</soap:Envelope>
```

Figura 4.13 : Solicitação SOAP feita pelo método LogoffWebcam

A resposta SOAP contém a informação do retorno do processamento da solicitação. O campo `<LogoffWebcamresult> int </LogoffWebcamresult>` conterá o valor da resposta.

```

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <LogoffWebCamResponse
xmlns="http://bepurc.org/ServiceCam/Service1">
      <LogoffWebCamResult>long</LogoffWebCamResult>
    </LogoffWebCamResponse>
  </soap:Body>
</soap:Envelope>

```

Figura 4.14: Resposta SOAP feita pelo método LogoffWebcam

A seguir é apresentada uma amostra de solicitação e resposta HTTP POST. Os espaços reservados mostrados precisam ser substituídos por valores reais. Os espaços a serem substituídos na solicitação HTTP POST por valores reais são os seguintes: lpCam.

```

POST /Service/ServiceCam/service1.smx/LogoffWebCam HTTP/1.1
Host: 127.0.0.1
Content-Type: application/x-www-form-urlencoded
Content-Length: length

lpCamstring

```

Figura 4.15 : Solicitação HTTP POST SOAP feita pelo método LogoffWebcam

A resposta HTTP POST SOAP contém o valor do retorno do processamento da solicitação HTTP POST SOAP.

```

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<long xmlns="http://bepurc.org/ServiceCam/Service1">long</long>

```

Figura 4.16 : Resposta HTTP POST SOAP feita pelo método LogoffWebcam

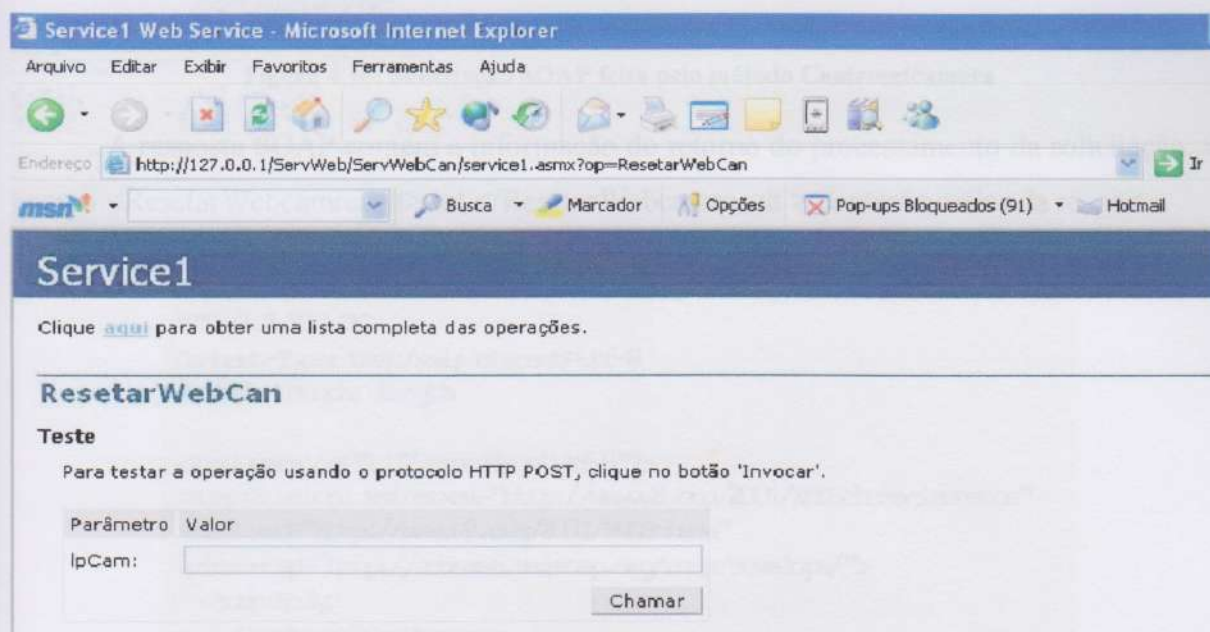
**Quadro 3: Webmethod ResetarWebcam**

```
<Web Method ()> _  
Public Function ResetarWebcam (ByVal lpCam As Long) As Long  
    'Chamando o método para Resetar a câmera  
    Camresetcamera (lpCam)  
End Function
```

Este Webmethod recebe como parâmetro um valor (lpCam) Long ou inteiro e tem como principal função fazer o reset no Hardware (*Webcam*) através da chamada que é feita a Dll através do método Camresetcamera.

Este Webmethod Camresetcamera assemelha-se ao Webmethod LogoffWebcam, pois ambos recebem apenas um valor como parâmetro.

A tela deste **Webmethod** pode ser vista na Figura 4.17.



**Figura 4.17: Tela do Webmethod ResetarWebcam**

A seguir é apresentada uma amostra de solicitação e de resposta SOAP. Os espaços reservados mostrados precisam ser substituídos por valores reais. Os valores reais a serem substituídos são os seguintes: <lpCam> Long </lpCam>.

```

POST /ServWeb/ServWebCam/service1.smx HTTP/1.1
Host: 127.0.0.1
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://taipuci.org/ServWebCam/Service1/ResetWebCam"

<?xml version="1.0" encoding="utf-8">
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ResetWebCam xmlns="http://taipuci.org/ServWebCam/Service1">
      <lpCam>long</lpCam>
    </ResetWebCam>
  </soap:Body>
</soap:Envelope>

```

Figura 4.18: Solicitação SOAP feita pelo método Camresetcamera

A resposta SOAP contém a informação do retorno do processamento da solicitação. O campo <ResetWebCamresult> int </ResetWebCamresult> conterá o valor da resposta.

```

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8">
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ResetWebCamResponse
xmlns="http://taipuci.org/ServWebCam/Service1">
      <ResetWebCamResult>long</ResetWebCamResult>
    </ResetWebCamResponse>
  </soap:Body>
</soap:Envelope>

```

Figura 4.19 : Resposta SOAP feita pelo método Camresetcamera

A seguir é apresentada uma amostra de solicitação e resposta HTTP POST. Os espaços reservados mostrados precisam ser substituídos por valores reais. Os espaços a serem substituídos na solicitação HTTP POST por valores reais são os seguintes: lpCam.

```
POST /ServWeb/ServWebCam/service1.saw/ResetWebCam HTTP/1.1
Host: 127.0.0.1
Content-Type: application/x-www-form-urlencoded
Content-Length: length
```

```
lpCam#string
```

Figura 4.20 : HTTP POST SOAP feita pelo método Camresetcamera

A resposta HTTP POST SOAP contém o valor do retorno do processamento da solicitação HTTP POST SOAP.

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8">
<long xmlns="http://tempuri.org/ServWebCam/Service1">length</long>
```

Figura 4.21: Resposta HTTP POST SOAP feita pelo método Camresetcamera

#### Quadro 4: Webmethod MoverPCima

```
<WebMethod()>
Public Function Moverpcima (ByVal Direcao As Integer, ByVal Posicao As
Integer) As Integer
    If Posicao = 1 Then
        MoverPCima = CamPosition(lpCamera, 2)
    Else
        MoverPCima = CamPosition(lpCamera, 3)
    End If
End Function
```

Este *Webmethod* recebe dois valores do tipo inteiro como parâmetro e tem como principal função movimentar o Hardware (*Webcam*) verticalmente para cima. Este *Webmethod* faz a chamada a DLL através do método *Composition*, passando dois valores como parâmetro para este método.

Os métodos de giro do Hardware (*Webcam*) são similares, pois recebem como parâmetro dois valores do tipo inteiro.

A tela deste **Webmethod** pode ser vista na Figura 4.22.

Figura 4.22: Solicitação SOAP feita pelo método Moverpcima

A resposta SOAP contém a informação do retorno do processamento da solicitação. O campo *<Moverpcimaresult>* ou *<Moverpcimaout>* contém o valor da resposta.

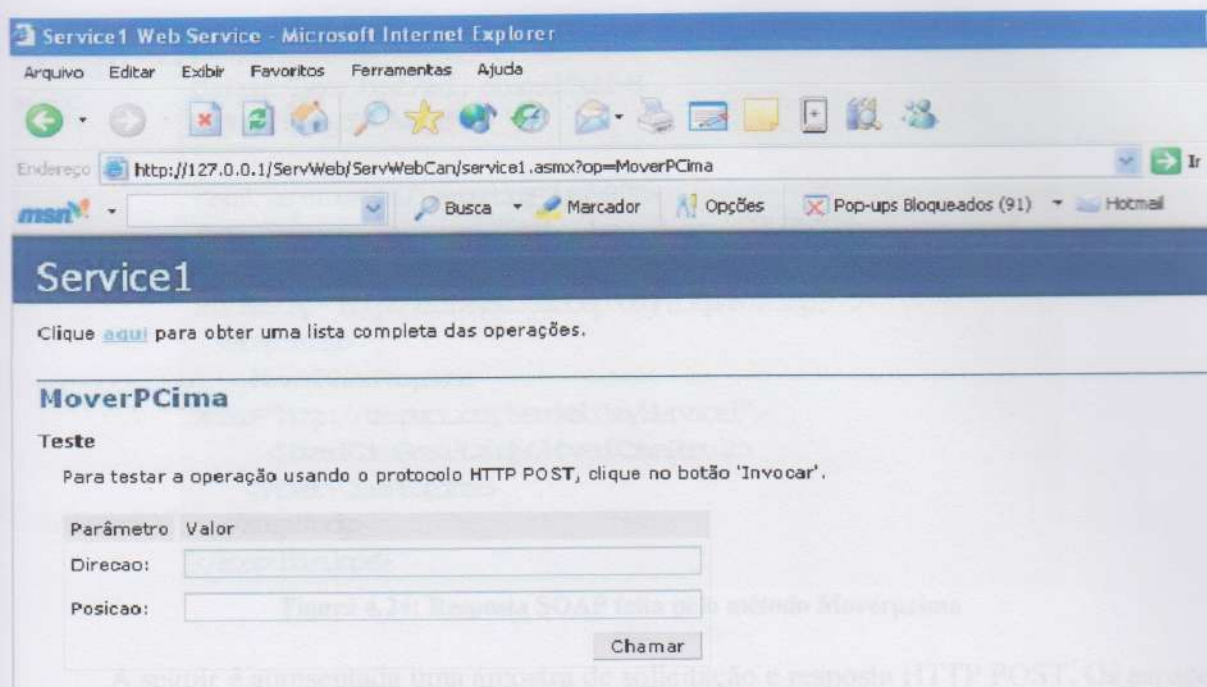


Figura 4.22: Tela do Webmethod Moverpcima

A seguir é apresentada uma amostra de solicitação e de resposta SOAP. Os espaços reservados mostrados precisam ser substituídos por valores reais. Os valores reais a serem substituídos são os seguintes: `<Direcao> int </Direcao>` e `<Posicao> int </Posicao>`.

```
POST /ServWeb/ServWebCan/service1.asmx HTTP/1.1
Host: 127.0.0.1
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/ServWebCan/Service1/MoverPCima"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <MoverPCima xmlns="http://tempuri.org/ServWebCan/Service1">
      <Direcao>int</Direcao>
      <Posicao>int</Posicao>
    </MoverPCima>
  </soap:Body>
</soap:Envelope>
```

Figura 4.23: Solicitação SOAP feita pelo método Moverpcima

A resposta SOAP contém a informação do retorno do processamento da solicitação. O campo `<Moverpcimaresult> int </Moverpcimaresult>` conterá o valor da resposta.

```

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8">
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <MoverPCimaResponse
xmlns="http://tempuri.org/ServiceCan/Service1">
      <MoverPCimaResult>int</MoverPCimaResult>
    </MoverPCimaResponse>
  </soap:Body>
</soap:Envelope>

```

Figura 4.24: Resposta SOAP feita pelo método Moverpcima

A seguir é apresentada uma amostra de solicitação e resposta HTTP POST. Os espaços reservados mostrados precisam ser substituídos por valores reais. Os espaços a serem substituídos na solicitação HTTP POST por valores reais são os seguintes: Direcao e Posicao.

```

POST /Service/ServiceCan/service1.smx/ResetarWebCan HTTP/1.1
Host: 127.0.0.1
Content-Type: application/x-www-form-urlencoded
Content-Length: length

lpCan=string

```

Figura 4.25: Solicitação HTTP POST SOAP feita pelo método Moverpcima

A resposta HTTP POST SOAP contém o valor do retorno do processamento da solicitação HTTP POST SOAP.

```

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8">
<int xmlns="http://tempuri.org/ServiceCan/Service1">int</int>

```

Figura 4.26: Resposta HTTP POST SOAP feita pelo método Moverpcima

**Quadro 5: Webmethod Moverpbaixo**

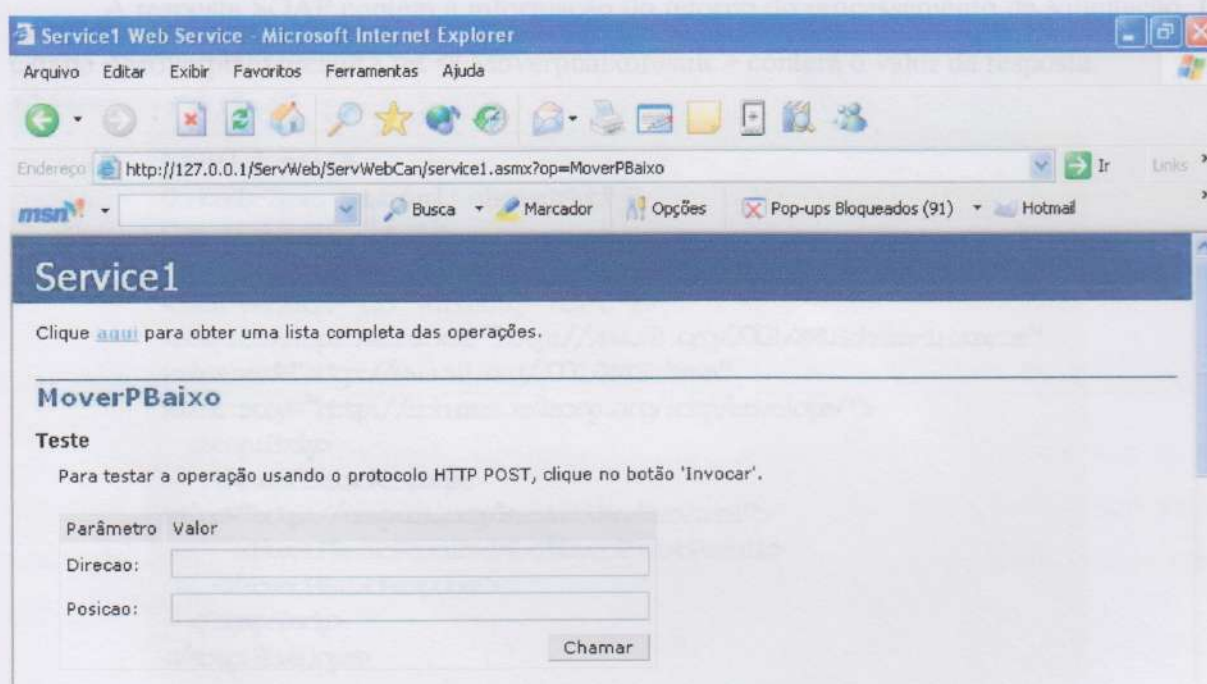
```

<Webmethod ()>
Public Function Moverpbaixo (ByVal Direcao As Integer, ByVal Posicao As Integer) As Integer
    If Posicao = 0 Then
        Moverpbaixo = CamPosition(lpCamera, 0)
    Else
        Moverpbaixo = CamPosition(lpCamera, 1)
    End If
End Function

```

Este **Webmethod** recebe dois valores do tipo inteiro como parâmetro e tem como principal função movimentar o Hardware (*Webcam*) verticalmente para baixo. Este **Webmethod** faz a chamada a Dll através do método *Camposition*, passando dois valores como parâmetro para este método.

A Figura 4.27 apresenta a tela deste **Webmethod**.



**Figura 4.27: Tela do Webmethod Moverpbaixo**

A seguir é apresentada uma amostra de solicitação e de resposta SOAP. Os espaços reservados mostrados precisam ser substituídos por valores reais. Os valores reais a serem substituídos são os seguintes: <Direcao> int </Direcao> e <Posicao> int </Posicao>.

```

POST /ServWeb/ServWebCan/service1.asmx HTTP/1.1
Host: 127.0.0.1
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://taipuci.org/ServWebCan/Service1/MoverPBaixo"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <MoverPBaixo xmlns="http://taipuci.org/ServWebCan/Service1">
      <Direcao>int</Direcao>
      <Posicao>int</Posicao>
    </MoverPBaixo>
  </soap:Body>
</soap:Envelope>

```

Figura 4.28: Solicitação SOAP feita pelo método Moverpbaixo

A resposta SOAP contém a informação do retorno do processamento da solicitação. O campo <Moverpbaixoresult> int </ Moverpbaixoresult > conterá o valor da resposta.

```

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <MoverPBaixoResponse
xmlns="http://taipuci.org/ServWebCan/Service1">
      <MoverPBaixoResult>int</MoverPBaixoResult>
    </MoverPBaixoResponse>
  </soap:Body>
</soap:Envelope>

```

Figura 4.29: Resposta SOAP feita pelo método Moverpbaixo

A seguir é apresentada uma amostra de solicitação e resposta HTTP POST. Os espaços reservados mostrados precisam ser substituídos por valores reais. Os espaços a serem substituídos na solicitação HTTP POST por valores reais são os seguintes: Direcao e Posicao.

```
POST /ServWeb/ServWebCam/service1.smx/MoverPCima HTTP/1.1
Host: 127.0.0.1
Content-Type: application/x-www-form-urlencoded
Content-Length: length
```

```
Direcao=string;Posicao=string
```

Figura 4.30: Solicitação HTTP POST SOAP feita pelo método Moverpbaixo

A resposta HTTP POST SOAP contém o valor do retorno do processamento da solicitação HTTP POST SOAP.

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<int xmlns="http://tempuri.org/ServWebCam/Service1">int</int>
```

Figura 4.31: Resposta HTTP POST SOAP feita pelo método Moverpbaixo

#### Quadro 6: Webmethod Moverpdireita

```
<WebMethod()> _
Public Function Moverpdireita (ByVal Direcao As Integer, ByVal Posicao As
Integer) As Integer
    If Posicao = 3 Then
        Moverpdireita = CamPosition(lpCamera, 6)
    Else
        Moverpdireita = CamPosition(lpCamera, 7)
    End If
End Function
```

Este **Webmethod** recebe dois valores do tipo inteiro como parâmetro e tem como principal função movimentar o Hardware (*Webcam*) para direita. Este **Webmethod** faz a chamada a Dll através do método *Camposition*, passando dois valores como parâmetro para este método.

A tela deste **Webmethod** pode ser vista na Figura 4.32.

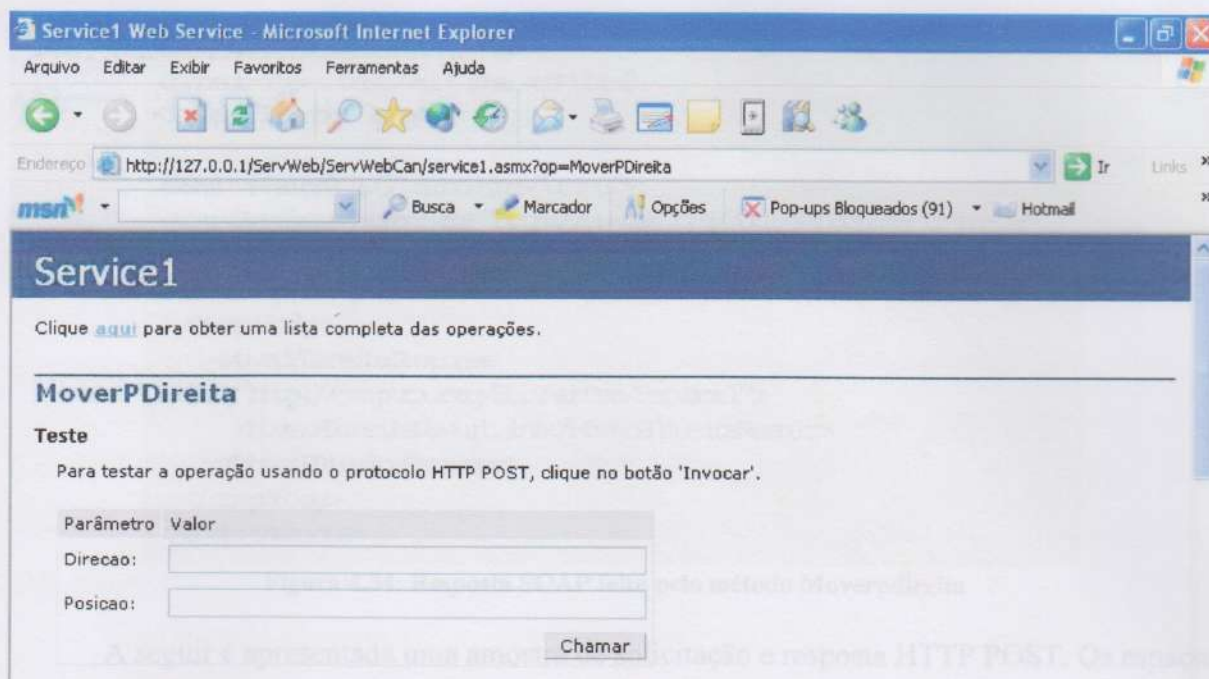


Figura 4.32: Tela do Webmethod Moverpdireita

A seguir é apresentada uma amostra de solicitação e de resposta SOAP. Os espaços reservados mostrados precisam ser substituídos por valores reais. Os valores reais a serem substituídos são os seguintes: <Direcao> int </Direcao> e <Posicao> int </Posicao>.

```
POST /ServWeb/ServWebCan/service1.asmx HTTP/1.1
Host: 127.0.0.1
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://bampuri.org/ServWebCan/Service1/MoverPDireita"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <MoverPDireita xmlns="http://bampuri.org/ServWebCan/Service1">
      <Direcao>int</Direcao>
      <Posicao>int</Posicao>
    </MoverPDireita>
  </soap:Body>
</soap:Envelope>
```

Figura 4.33: Solicitação SOAP feita pelo método Moverpdireita

A resposta SOAP contém a informação do retorno do processamento da solicitação. O campo <Moverpdireitaresult> int </Moverpdireitaresult> conterá o valor da resposta.

```

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <MoverPDireitaResponse
xmlns="http://tempuri.org/ServiceCan/Service1">
      <MoverPDireitaResult>int</MoverPDireitaResult>
    </MoverPDireitaResponse>
  </soap:Body>
</soap:Envelope>

```

Figura 4.34: Resposta SOAP feita pelo método Moverpdireita

A seguir é apresentada uma amostra de solicitação e resposta HTTP POST. Os espaços reservados mostrados precisam ser substituídos por valores reais. Os espaços a serem substituídos na solicitação HTTP POST por valores reais são os seguintes: Direcao e Posicao.

```

POST /Service/ServiceCan/service1.asmx/MoverPDireita HTTP/1.1
Host: 127.0.0.1
Content-Type: application/x-www-form-urlencoded
Content-Length: length

Direcao=string;Posicao=string

```

Figura 4.35 : Solicitação HTTP POST SOAP feita pelo método Moverpdireita

A resposta HTTP POST SOAP contém o valor do retorno do processamento da solicitação HTTP POST SOAP.

```

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<int xmlns="http://tempuri.org/ServiceCan/Service1">int</int>

```

Figura 4.36: Resposta HTTP POST SOAP feita pelo método Moverpdireita

**Quadro 7: Webmethod Moverpesquerda**

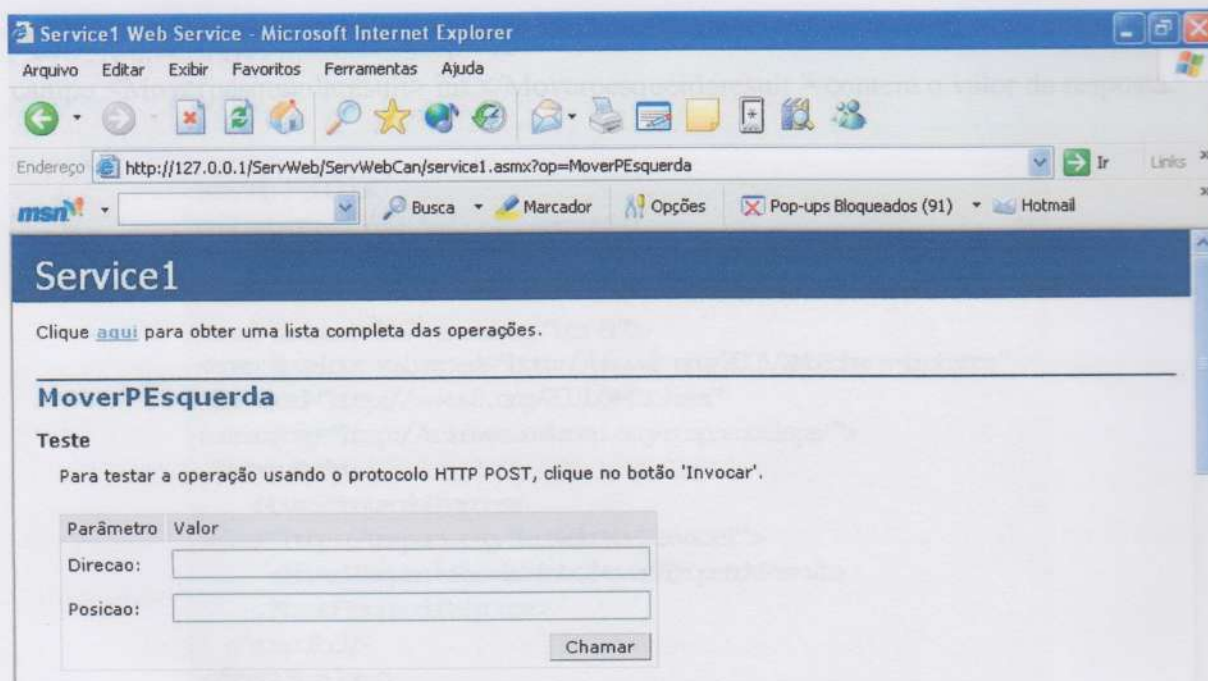
```

<WebMethod()> _
Public Function Moverpesquerda (ByVal Direcao As Integer, ByVal Posicao As
Integer) As Integer
    If Posicao = 2 Then
        Moverpesquerda = CamPosition(lpCamera, 4)
    Else
        Moverpesquerda = CamPosition(lpCamera, 5)
    End If
End Function

```

Este **Webmethod** recebe dois valores do tipo inteiro como parâmetro e tem como principal função movimentar o Hardware (*Webcam*) para esquerda. Este **Webmethod** faz a chamada a Dll através do método *Camposition*, passando dois valores como parâmetro para este método.

Tela deste **Webmethod**:



**Figura 4.37: Tela do Webmethod Moverpesquerda**

A seguir é apresentada uma amostra de solicitação e de resposta SOAP. Os espaços reservados mostrados precisam ser substituídos por valores reais. Os valores reais a serem substituídos são os seguintes: <Direcao> int </Direcao> e <Posicao> int </Posicao>.

```

POST /ServWeb/ServWebCan/service1.ssnk HTTP/1.1
Host: 127.0.0.1
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/ServWebCan/Service1/MoverPesquerda"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <MoverPesquerda xmlns="http://tempuri.org/ServWebCan/Service1">
      <Direcao>int</Direcao>
      <Posicao>int</Posicao>
    </MoverPesquerda>
  </soap:Body>
</soap:Envelope>

```

Figura 4.38: Solicitação SOAP feita pelo método Moverpesquerda

A resposta SOAP contém a informação do retorno do processamento da solicitação. O campo <Moverpesquerdaresult> int </Moverpesquerdaresult > conterá o valor da resposta.

```

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <MoverPesquerdaResponse
xmlns="http://tempuri.org/ServWebCan/Service1">
      <MoverPesquerdaResult>int</MoverPesquerdaResult>
    </MoverPesquerdaResponse>
  </soap:Body>
</soap:Envelope>

```

Figura 4.39: Resposta SOAP feita pelo método Moverpesquerda

A seguir é apresentada uma amostra de solicitação e resposta HTTP POST. Os espaços reservados mostrados precisam ser substituídos por valores reais. Os espaços a serem substituídos na solicitação HTTP POST por valores reais são os seguintes: Direcao e Posicao.

```
POST /ServWeb/ServWebCan/service1.saw/MoverPesquerda HTTP/1.1
Host: 127.0.0.1
Content-Type: application/x-www-form-urlencoded
Content-Length: length
```

```
Direcao=string;Posicao=string
```

Figura 4.40: Solicitação HTTP POST SOAP feita pelo método Moverpesquerda

A resposta HTTP POST SOAP contém o valor do retorno do processamento da solicitação HTTP POST SOAP.

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<xml version="1.0" encoding="utf-8">
<int xmlns="http://tempuri.org/ServWebCan/Service1">int</int>
```

Figura 4.41: Resposta HTTP POST SOAP feita pelo método Moverpesquerda

#### 4.1.2.5 Exposição do código da Aplicação Cliente (Usuário).

O desenvolvimento da aplicação cliente foi feito utilizando como ferramenta de criação o ASP .Net ferramenta pertencente ao pacote ferramentas do *framework .Net* e outrora mencionado neste projeto.

O Cliente acessará os métodos publicados no *Web Service* para executar as funções de acesso ao Hardware (*Webcam*), **LoginWebcam**, **LogoffWebcam**, **ResetarWebcam**, **Moverpcima**, **Moverpbaixo**, **Moverpesquerda** e **Moverpdireita**.

A tela principal da aplicação cliente pode ser vista na Figura 4.42.

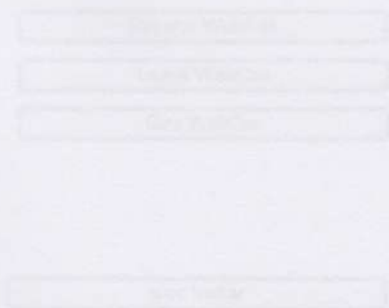


Figura 4.42: Tela com as opções da aplicação para o cliente

WebForm1 - Microsoft Internet Explorer

Arquivo Editar Exibir Favoritos Ferramentas Ajuda

Endereço: <http://127.0.0.1/ServWeb/ClienteWebServ/WebForm1.aspx>

msn Busca Marcador Opções Pop-ups Bloqueados (91) Hotmail

Usuário:

Senha:

IP:     Porta:

Figura 4.42: Tela principal da aplicação cliente

Nesta tela o cliente (usuário) deverá preencher os campos de **usuário**, **senha**, o **endereço IP** e a **porta**. Estes campos são requeridos e obrigatórios o seu preenchimento, pois, é a partir do Login que o cliente (usuário) poderá acessar os outros métodos.

Após o Login a página de Menu aparecerá disponibilizando as opções de acesso que são: Resetar, Logoff e Giro.

Menu - Microsoft Internet Explorer

Arquivo Editar Exibir Favoritos Ferramentas Ajuda

Endereço: <http://127.0.0.1/ServWeb/ClienteWebServ/Menu.aspx>

msn Busca Marcador Opções Pop-ups Bloqueados (91) Hotmail

Figura 4.43: Tela com as opções do Sistema para o cliente

A opção de Giro ao ser selecionado aparecerá uma nova página onde o usuário deverá informar a posição na qual deseja que o Hardware (*Webcam*) se movimente.

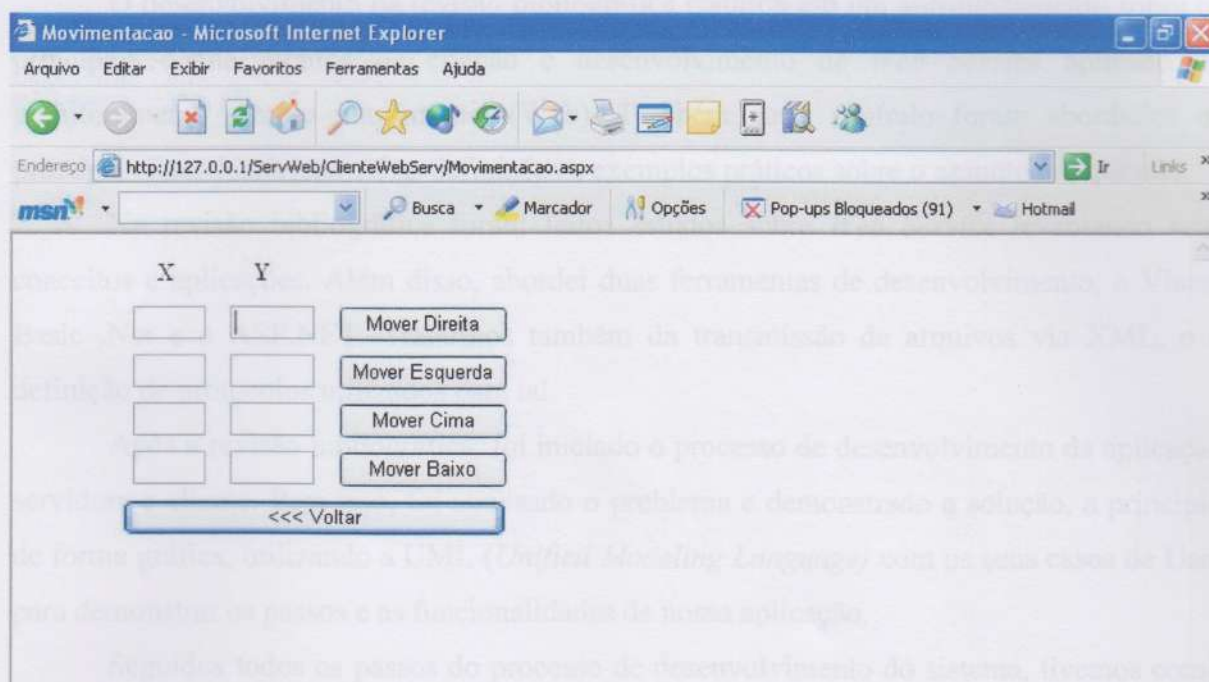


Figura 4.44: Tela com as opções de Movimentação do Hardware

### 3.1 DIFICULDADES ENCONTRADAS

O desenvolvimento da revista bibliográfica não apresentou grandes dificuldades, pois, existe uma grande variedade de conteúdos bibliográficos em forma de livro e artigos científicos que permitem o acesso ao Web Service.

As dificuldades começaram a surgir a partir do desenvolvimento do sistema. Uma dificuldade encontrada foi saber como o Web Service fazia chamadas à DLL, para executar as funções do Hardware, pois, a forma de programação do Visual Basic ainda mudou e muitas coisas ainda eram novas e deviam ser descobertas. Outro problema encontrado foi a DLL, que seria para aplicações Desktop e não aplicações Web.

## 5 CONSIDERAÇÕES FINAIS

O desenvolvimento da revisão bibliográfica resultou em um aprofundamento sobre os principais conhecimentos de criação e desenvolvimento de *Web Service* aplicado ao monitoramento remoto via Internet (Web). Também neste capítulo foram abordados os principais fatos históricos até as definições e exemplos práticos sobre o assunto em questão.

Na revisão bibliográfica foram feitos estudos sobre *Web Service* levantando seus conceitos e aplicações. Além disso, abordei duas ferramentas de desenvolvimento, o Visual Basic .Net e o ASP.NET. Tratarmos também da transmissão de arquivos via XML, e a definição de protocolos utilizados para tal.

Após a revisão bibliográfica, foi iniciado o processo de desenvolvimento da aplicação servidora e cliente. Para isso, foi analisado o problema e demonstrado a solução, a princípio de forma gráfica, utilizando a UML (*Unified Modeling Language*) com os seus casos de Uso, para demonstrar os passos e as funcionalidades de nossa aplicação.

Seguidos todos os passos do processo de desenvolvimento do sistema, tivemos como resultado uma aplicação cliente (usuário) e uma aplicação servidora (*Web Service*), que trazem as funcionalidades exigidas na especificação deste trabalho monográfico. A aplicação cliente traz as funcionalidades de acesso ao *Web Service* que por sua vez executará os métodos no Hardware (*Webcam*), a aplicação cliente poderá manipular o Hardware (*Webcam*) de qualquer local ou ambiente. Sendo assim é concluída a parte prática do projeto monográfico.

### 5.1 DIFICULDADES ENCONTRADAS

O desenvolvimento da revisão bibliográfica não apresentou grandes dificuldades, pois, existia uma grande variedade de conteúdo bibliográfico em forma de livro e artigos eletrônicos que tratavam do assunto de *Web Service*.

As dificuldades começaram a surgir a partir do desenvolvimento do sistema. Uma dificuldade encontrada foi saber como o *Web Service* faria chamadas à DLL para executar as funções do Hardware, pois, a forma de programação do Visual Basic tinha mudado e muitas coisas ainda eram novas e deveras desconhecidas. Outrora as chamadas às DLL's eram feitas por aplicações *Desktop* e não aplicações *Web*.

Tivemos outra dificuldade quando fomos mapear todas as funções do Hardware. Deveríamos colocar todas as funções, entretanto, não colocamos as funções de vídeo para serem carregadas na aplicação cliente, pois, tais funções eram muito complexas e levaria muito tempo para serem mapeadas e implementadas. Por isso, nos detivemos às funções de giro e movimentação da *Webcam*.

Outra dificuldade foi a nova forma que o Visual Basic .Net definiu seus métodos. Tivemos que reescrever todos os tipos (*types*) e estruturas (*structures*) que estavam em VB 6.0 para VB.NET, pois a API do Visual Basic .Net foi totalmente reescrita e adaptada para ser orientada a objetos. Esta migração foi devido ao fato de somente termos acessado a DLL do hardware usando VB 6.0, a qual é a versão anterior ao VB.NET.

Outra dificuldade que nos deparamos foi a escassez de recursos, no que diz respeito à infra-estrutura do laboratório, pois, o Serviço IIS estava desabilitado no Servidor do laboratório, e este ambiente quase nunca estava disponível para realizarmos os devidos testes.

## 6 CONCLUSÃO BIBLIOGRÁFICAS

O desenvolvimento deste trabalho monográfico resultou em um aprofundamento dos conhecimentos sobre *Web Service* e monitoramento remoto tanto em seus aspectos teóricos quanto nos práticos. Através deste aprofundamento o presente trabalho atingiu seus objetivos iniciais.

Tais conhecimentos aprofundados são deveras relevante para o meio acadêmico, pois, trará mais informações e inúmeros esclarecimentos sobre os assuntos abordados.

Este trabalho não se encerra com a conclusão desta monografia, pois ele pode ser considerado como um início de uma longa caminhada. Acreditamos fortemente nas tecnologias aqui abordadas, pois permitem que vários outros projetos sejam desenvolvidos para a construção de um novo mundo tecnológico.

- FOWLER, MARTIN, SCOTT, KENDALL. UML - Essencial: Um breve guia para a linguagem padrão de modelagem de objetos. Bookman, 2000.
- FRIDSCHEIM, R. A Internet como Ferramenta de Comunicação. Disponível em: <http://www.linhabocollins.com.br/debemvivemos/WEB5A.asp> Acesso em: 05. Abr. 2004.
- GUTIERREZ, M.A. Estratégias .NET: Por dentro parada: Economia Digital. Books, 2000.
- HADJIO, R. Treinamento em ASP.NET. Disponível em: <http://www.machobrasil.com.br/tecnologia/Tecnologia/Default.aspx?Access=09.Nov.2004>.
- HOWARD, R. Cinco passos para iniciar com o ASP.NET. Disponível em: <http://www.machobrasil.com.br/tecnologia/Tecnologia/Default.aspx?Access=09.Nov.2004>.
- JUNIOR, D. Aprendendo ASP.NET com C#. Price, 2001.
- L.J., F.L.G. Web Service na vida real. Disponível em: [http://www.machobrasil.com.br/tecnologia/web\\_services\\_10.asp](http://www.machobrasil.com.br/tecnologia/web_services_10.asp) Acesso em: 19. Maio. 2004.
- LEE, W.M. Web Services: Comunique-se em Diferentes Plataformas. Disponível em: <http://www.linhabocollins.com.br/debemvivemos/ServiceWeb.asp> Acesso em: 18. Maio. 2004.

## REFERÊNCIAS BIBLIOGRÁFICAS

- B.J, J.M.B. A Nova Geração de Aplicações Web. Disponível em:  
<<http://www.vertisnet.com.br>>  
Acesso em: 01. Maio. 2004.
- BOOCH, GRADY; RUMBAUGH, JAMES; JACOBSON, IVAR. **UML Guia do Usuário**: O mais avançado tutorial sobre UML. Campos, 2000.
- CAMELO, D. *Web Services e Objetos Distribuídos*. Disponível em:  
<<http://www.cqgp.sp.gov.br/downloads/WebServices.pdf>>  
Acesso em: 20. Abr.2004.
- CUNHA, DAVI. *Web Services, SOAP e Aplicações Web*. Disponível em:  
<[http://devedge.netscape.com/viewsource/2002/soap\\_overview/index\\_pt\\_br.html](http://devedge.netscape.com/viewsource/2002/soap_overview/index_pt_br.html)>.  
Acesso em: 20. Abr.2004.
- FOWLWR, MARTIN; SCOTT, KENDALL. **UML Essencial**: Um breve guia para a linguagem-padrão de modelagem de objetos. Bookman, 2000.
- FRIDSCHTEIN, R. A Internet como Ferramenta de Comunicação. Disponível em:  
<<http://www.promofoor.com.br/meio/artigos/0305A.asp>>  
Acesso em: 06. Abr.2004.
- GUTIERREZ, M.A. **Estratégia .NET**: Próxima parada: Economia Digital. Books, 2000.
- HADDAD, R. Treinamento em ASP.NET: Disponível em:  
<<http://www.microsoft.com/brasil/msdn/Tecnologias/Default.msp>>  
Acesso: 09. Nov. 2004.
- HOWARD, R. Cinco passos para iniciar com o ASP.NET: Disponível em:  
<<http://www.microsoft.com/brasil/msdn/Tecnologias/Default.msp>>  
Acesso: 09. Nov. 2004.
- JUNIOR, D. **Aprendendo ASP .NET com C#**: Érica, 2001.
- L.J, J.L.G. *Web Service na vida real*. Disponível em:  
<[http://www.msdnbrasil.com.br/tecnologias/web\\_services\\_10.aspx](http://www.msdnbrasil.com.br/tecnologias/web_services_10.aspx)>  
Acesso em: 19. Maio. 2004.
- LEE, W.M. *Web Services: Comunique-se em Diferentes Plataformas*. Disponível em:  
<<http://www.linhadecodigo.com.br/desenvolvimento/webservices.asp>>  
Acesso em: 18. Maio. 2004.

- MACORATTI, J.C. Criando *Web Services*: Disponível em:  
<[http://www.macoratti.net/vbn\\_wbs.htm](http://www.macoratti.net/vbn_wbs.htm)>  
Acesso: 09. Nov. 2004.
- MACORATTI, J.C. VB .Net: Disponível em:  
<[http://www.macoratti.net/vbnet\\_1.htm](http://www.macoratti.net/vbnet_1.htm)>  
Acesso: 18. Maio. 2004.
- MACORATTI, J.C. XML - Introdução: Disponível em:  
<<http://www.macoratti.net/xml.htm>>  
Acesso: 02. Set. 2004.
- MCDOWELL, R. Estratégia .NET. Disponível em:  
<<http://www.microsoft.com/brasil/pr/comdex2000.htm>>  
Acesso em: 16. Abr.2004.
- RICCONI, P.R. **Introdução a Objetos Distribuídos com Corba**. Books, 2000.
- SANTANA, M. SOAP e *Web Services*: Disponível em:  
<[http://www.linhadecodigo.com.br/artigos.asp?id\\_ac=38&pag1](http://www.linhadecodigo.com.br/artigos.asp?id_ac=38&pag1)>  
Acesso: 03. Set. 2004.