# INTEGRAÇÃO CENTRAL DE ALARME COM REDE DE TELEFONIA MÓVEL

Aluno: Pedro Henrique Neves Maciel de Oliveira – RA: 2021858/5

Orientador: M.C Maria Marony S.F. Nascimento

Brasília – DF, julho de 2009

INTEGRAÇÃO CENTRAL DE ALARME COM REDE DE TELEFONIA MÓVEL

por

Pedro Henrique Neves Maciel de Oliveira

Trabalho apresentado à Banca examinadora do curso de Engenharia da Computação da FATECS – Faculdade de Tecnologia e Ciências Sociais Aplicadas – Centro Universitário de Brasília como requisito parcial para obtenção do título de Engenheiro da Computação

Brasília – DF, julho de 2009

**Banca Examinadora**

_____
Prof. Msc. Maria Marony S.F. Nascimento
Orientador

_____
Prof. Msc. Francisco Javier
Examinador

# AGRADECIMENTOS

Em primeiro lugar a Deus pela saúde, pelas oportunidades e pelos desafios que me colocou na vida, até hoje, para que conseguisse com muita dedicação e sabedoria, alcançar mais um objetivo, mais uma vitória.

Aos meus Pais, João Maciel de Oliveira e Elizabeth Neves de Oliveira, pelo carinho e preocupação que sempre tiveram. Pela educação que me deram, e, claro, pela vontade e dedicação incansáveis de me guiar e orientar a seguir, sempre, o melhor caminho.

A minha querida irmã, Jaqueline, sempre presente em todos os momentos de minha vida, fazendo o possível e impossível para me ajudar.

A minha amada Jeiza Jerônimo, por estar diretamente ligada a conclusão deste trabalho, pois sem suas ameaças possivelmente este projeto não estaria sendo concluído neste momento.

A todos os colegas e principalmente, os amigos, que me deram todo o apoio e incentivo para a realização deste projeto. Em especial, aos que se envolveram em momentos críticos para me ajudar no desenvolvimento e na interminável batalha de conclusão desde projeto.

E, por fim, para todos aqueles que direta ou indiretamente, estiveram presentes e dispostos a ajudar de alguma forma na conclusão deste projeto.

# SUMÁRIO

# LISTA DE FIGURAS

# INDICE DE SIGLAS E ABREVIATURAS

| | |
|---|---|
| .NET | Microsoft Framework |
| AMPS | Advanced Mobile Phone System |
| CDMA | Code division multiple access |
| CI | Circuito Integrado |
| EDGE | Enhanced Data rates for GSM Evolution |
| EIA | Electronic Industries Alliance |
| EVDO | Evolution-Data Optimized |
| GPRS | General Packet Radio Service |
| GSM | Global System for Mobile Communications |
| HSCSD | High Speed Circuit-Switched Data |
| HSDPA | High-Speed Downlink Packet Access |
| HSPA | High Speed Packet Access |
| HSUPA | High-Speed Uplink Packet Access |
| IETF | Internet Engineering Task Force |
| NMT | Networked Media Tank |
| NRZ | Non-Return-to-Zero |
| OASIS | Organization for the Advancement of Structured Information Standards |
| RS | Recommended Standard |
| SMS | Short Message Service |
| SOA | Service oriented architecture |
| SOAP | Simple Object Access Protocol |
| TDMA | Time Division Multiple Access |
| W3C | World Wide Web Consortium |
| WSDL | Web Services Description Language |
| WS-I | Web Services Interoperability Organization |
| XML | eXtensible Markup Language |

# RESUMO

Neste trabalho é apresentado o desenvolvimento de uma solução de integração de centrais de alarmes com a rede de telefonia móvel, utilizando o auxílio da tecnologia de SMS. Essa integração permite que usuários de centrais de alarmes possam ser imediatamente informados da ocorrência de um evento, possibilitando assim, que providências sejam tomadas.

A solução desenvolvida é composta por 2 módulos: Uma aplicação de gerência em ASP NET – FrameWork 2.0, responsável pelo cadastro dos usuários os quais receberão os alertas, esse módulo é um website, que pode ser publicado localmente ou na internet, para que a gerência do sistema possa ser feita de qualquer local. O outro módulo é uma interface microcontrolada que recebe um sinal proveniente da Central de Alarmes JFL 251B, que é transmitido através da porta serial de um PC, enviando alertas aos usuários do sistema através de uma conexão com a internet e um provedor de envio de mensagens SMS.

O envio das informações aos usuários, pela a internet, é possível através de chamadas SOAP(WEBSERVICES), este serviço é disponibilizado pela operadora de SMS contratada.

Os dados do módulo de gerência são armazenados em um arquivo do tipo XML- que contém o nome e o telefone dos usuários. Este mesmo arquivo é consumido no momento do envio das mensagens SMS.

Palavras-chave: SMS, APS NET – FrameWork, JFL 251B, bits, microcontrolador PCI 16F628SA, PC, provedor, XML, banco de dados.

# CAPÍTULO 1 – INTRODUÇÃO

A imagem 1.1 ilustra uma visão geral do projeto em sua fase final, onde todos os componentes da solução se encontram interligados. O componente 1 é o servidor, onde estão instalados o Módulo Gestor, responsável por gerenciar os usuários cadastrados no sistema e o Serviço Windows, responsável por monitorar o sinal emitido pela interface de comunicação, componente identificado com o número 3. O componente identificado com o número 2 é a central de alarme, a qual emite o sinal analógico que será enviado para a interface de comunicação e convertido em um sinal digital. O componente identificado pelo número 4 são os sensores sem fio utilizados pela central de alarme.

Nos capítulos seguintes serão detalhados todos os elementos existentes nesta imagem, assim como as tecnologias utilizadas.


**Imagem 1.1 – Topologia do Projeto**

## 1.1 **Motivação**

A falta de segurança em nosso país é um grave problema. Desta forma, as centrais de alarme residenciais acabam se tornando um artifício bastante utilizado, principalmente por moradores de casas, onde a segurança não é tão efetiva quanto em apartamentos.

Dados da Abese (Associação Brasileira das Empresas de Segurança Eletrônica) mostram que a utilização de sistemas de segurança são eficientes para o consumidor.[10]

As estatísticas defendem que a cada 100 tentativas de furtos em estabelecimentos com sistemas de alarme, sejam unidades comerciais ou residenciais, em 94% dos casos essas tentativas são fracassadas. [10]

O brasileiro, no entanto, ainda não se acostumou com o uso constante desses equipamentos. A entidade diz que 95% dos alarmes são acionados pelos próprios clientes, sem que estejam em situações de perigo, o que demonstra a necessidade de cuidados no uso dos sistemas. [10]

"A quantidade de bens roubados em estabelecimentos sem alarmes é dez vezes maior que as lojas que dispõem de segurança", conta o diretor de comunicação da ABESE, Oswaldo Oggiam Júnior. [10]

Mais de 3 milhões de imóveis comerciais e residenciais em todo o Brasil já contam com dispositivos básicos de segurança eletrônica. Para a Abese, este número é baixo, visto que o Brasil tem 49,1 milhões de imóveis, nas contas do IBGE (Instituto Brasileiro de Geografia e Estatística). [10]

A Abese estima que ao menos a metade dos condomínios existentes na cidade de São Paulo conta com algum tipo de sistema de proteção eletrônica instalado e que, na última década, houve um crescimento no setor de 12% ao ano. [10]

Dessa crescente utilização de centrais de alarme e monitoração de segurança residencial, surgiu a necessidade de um serviço que atualmente não é oferecido pelas empresas: manter os moradores sempre atualizados com relação ao status do alarme de sua residência.

Tal necessidade pode ser exemplificada na situação em que um morador se encontra ausente e outro morador ativa a central e se ausenta da residência. O morador inicialmente ausente, não sabe se o alarme se encontra ativado ou não, ocasionando eventuais disparos da central quando da sua chegada.

A internet por sua vez, viabilizou a implementação deste projeto, tendo em vista que tornou-se um dos mais importantes meios de comunicação no mundo, possibilitando a obtenção e a troca de informações dos mais diversos tipos. Com a mesma velocidade de crescimento, a rede de telefonia móvel tornou-se também outro meio de comunicação que se moderniza a cada dia que passa.

O mercado brasileiro e mundial de telefonia celular continua em franco crescimento, sem perspectivas de estagnação ou baixa no curto e médio prazo. Os números indicativos destes mercados são explosivos e extraordinários. Pesquisa divulgada no Brasil, no último dia 26 de setembro, e realizada pela União Internacional de Telecomunicações (UIT), órgão das Nações Unidas (ONU), revela um crescimento anual de 24% do mercado global nos últimos oito anos. O mundo deverá encerrar o ano de 2008 com 4 bilhões de matrículas de telefone móvel.[16]

Em 2006,o escritor norte-americano Bruce Sterling declarou em uma entrevista para a revista *The Economist*, que no futuro os telefones móveis serão chaves de casa, controle remoto, lanternas, bússolas, mapas, memória flash, monitores de sinais vitais, microfones, gravadores, apontadores de laser, passaportes, alarme anti-roubo, dentre outros. Muitos dos itens dessa lista já estão hoje disponíveis no mercado. Atualmente, os celulares já são o meio preferido para alguns serviços, como por exemplo: consulta ao horário de filmes nos cinemas, verificação de cotações de bolsas de valores, compra de refrigerante em máquina, acesso a documentos em situações emergenciais, envio de arquivos etc.[16]

Juntamente com a Internet, outro fator determinante para a viabilização deste projeto foi a utilização da rede de telefonia móvel, a qual atinge boa parte

da população mundial, possibilitando que o projeto seja feito aproveitando os recursos disponibilizados por esta tecnologia.

## 1.2 **Objetivos**

O projeto proposto tem como objetivo manter os usuários de sistemas de segurança residenciais ou comerciais, sempre informados da situação atual dos locais onde os mesmos são usuários. Viabilizando desta forma, um maior controle da movimentação de pessoas dentro dos imóveis.

Para atender os objetivos desse projeto, foi necessário produzir uma interface que receba o sinal analógico(+/- 5V) proveniente da central de alarmes e converta esse sinal em um sinal digital(bits) para ser transmitido através da porta serial de um PC, para que sejam enviados os alertas, aos usuários do sistema através de uma conexão com a internet e um provedor de envio de mensagens SMS.

Os objetivos específicos do trabalho são:

a) Conectar a central de alarme JFL a um computador, via porta serial.

b) Obter dados da central de alarme, como:
    a. Momento de Desativação de Alarme;
    b. Ativação de Alarme;
    c. Disparos de Alarme;

c) Os dados recebidos da central são interpretados pelo Serviço de envio de emails e enviados para os aparelhos celulares cadastrados através da Internet.

## 1.3 **Estrutura da Monografia**

Este projeto constitui-se de quatro capítulos, incluindo a introdução. Segue uma breve descrição:

- Capítulo 2 – Referencial Tecnológico – aborda os principais conceitos e definições envolvidas neste projeto final, como Microcontroladores, ASP.NET/C#, Telefonia Móvel, dentre outros.
- Capítulo 3 – Desenvolvimento – descrição detalhada do desenvolvimento propriamente dito, deste projeto. Citando-se todas as ferramentas utilizadas, descrições detalhadas de código desenvolvido nas aplicações, detalhes da implementação, dos testes e dos resultados obtidos.
- Capítulo 4 – Conclusões – descreve as principais conclusões obtidas neste projeto.

# CAPÍTULO 2 – REFERENCIAL TECNOLÓGICO

Neste capítulo, faz-se referência teórica de todos os assuntos abordados no desenvolvimento deste projeto.

O projeto foi dividido basicamente em 2 partes, uma responsável pela interface entre a central de alarme e o computador, essa comunicação é feita através de um circuito microcontrolado, responsável por fazer a conversão do sinal analógico da central em sinal digital, transmitido para o computador através da porta serial, utilizando-se comunicação serial. A construção deste modulo envolve hardware e software. A programação deste modulo foi feita utilizando-se a linguagem C.

O segundo módulo é composto de dois aplicativos, o primeiro responsável por fazer a gerência dos dados de usuários, esse modulo é composto exclusivamente de software. O módulo de gerência alimenta um arquivo em formato XML, mantendo assim o cadastro de usuários. O segundo módulo é responsável por monitorar a porta serial, aguardando o recebimento de sinais provenientes da central de alarme. Para os dois aplicativos foi utilizado a Linguagem de ASP.Net/C#(Microsoft .Net Framework).

## 2.1. Hardware

### 2.1.1. Canais de Comunicação

Um canal de comunicação é um caminho sobre o qual a informação pode trafegar. Ela pode ser definida por uma linha física (fio) que conecta dispositivos de comunicação, ou por um rádio, laser, ou outra fonte de energia radiante.[11]

Em comunicação digital, a informação é representada por bits de dados individuais, que podem ser encapsulados em mensagens de vários bits. Um byte (conjunto de 8 bits) é um exemplo de uma unidade de mensagem que

pode trafegar através de um canal digital de comunicações. Uma coleção de bytes pode ser agrupada em um “frame” ou outra unidade de mensagem de maior nível. Esses múltiplos níveis de encapsulamento facilitam o reconhecimento de mensagens e interconexões de dados complexos. [11]

Um canal no qual a direção de transmissão é inalterada é referida como ***canal simplex***. Por exemplo, uma estação de rádio é um canal simplex porque ela sempre transmite o sinal para os ouvintes e nunca é permitido a transmissão inversa. [11]

Um ***canal half-duplex*** é um canal físico simples no qual a direção pode ser revertida. As mensagens podem fluir nas duas direções, mas nunca ao mesmo tempo. Em uma chamada telefônica, uma parte fala enquanto a outra escuta. Depois de uma pausa, a outra parte fala e a primeira escuta. Falar simultaneamente resulta em sons que não podem ser compreendidos. [11]

Um ***canal full-duplex*** permite que mensagens sejam trocadas simultaneamente em ambas as direções. Ele pode ser visto como dois canais simplex, um canal direto e um canal reverso, conectados nos mesmos pontos. [11]

#### 2.1.1.1. **Comunicação Serial**

A maioria das mensagens digitais são mais longas que alguns poucos bits. Por não ser prático nem econômico transferir todos os bits de uma mensagem simultaneamente, a mensagem é quebrada em partes menores e transmitida seqüencialmente. A transmissão bit-serial converte a mensagem em um bit por vez através de um canal. Cada bit representa uma parte da mensagem. Os bits individuais são então rearranjados no destino para compor

a mensagem original. Em geral, um canal irá passar apenas um bit por vez. A transmissão bit-serial é normalmente chamada de transmissão serial, e é o método de comunicação escolhido por diversos periféricos de computadores.

A transmissão byte-serial converte 8 bits por vez através de 8 canais paralelos. Embora a taxa de transferência seja 8 vezes mais rápida que na transmissão bit-serial, são necessários 8 canais, e o custo poderá ser maior do que 8 vezes para transmitir a mensagem. Quando as distâncias são curtas, é factível e econômico usar canais paralelos como justificativa para as altas taxas de transmissão. [11]

#### 2.1.1.2. **Transmissão Assíncrona x Transmissão Síncrona**

Geralmente, dados serializados não são enviados de maneira uniforme através de um canal. Ao invés disso, pacotes com informação regulares são enviados seguidos de uma pausa. Os pacotes de dados binários são enviados dessa maneira, possivelmente com comprimentos de pausa variável entre pacotes, até que a mensagem tenha sido totalmente transmitida. O circuito receptor dos dados deve saber o momento apropriado para ler os bits individuais desse canal, saber exatamente quando um pacote começa e quanto tempo decorre entre bits. Quando essa temporização for conhecida, o receptor é dito estar sincronizado com o transmissor, e a transferência de dados precisa torna-se possível. Falhas na manutenção do sincronismo durante a transmissão irão causar a corrupção ou perda de dados.[12]

Duas técnicas básicas são empregadas para garantir a sincronização correta. Em sistemas síncronos, canais separados são usados para transmitir dados e informação de tempo. O canal de temporização transmite pulsos de clock para

o receptor. Através da recepção de um pulso de clock, o receptor lê o canal de dado e armazena o valor do bit encontrado naquele momento. O canal de dados não é lido novamente até que o próximo pulso de clock chegue. Como o transmissor é responsável pelos pulsos de dados e de temporização, o receptor irá ler o canal de dados apenas quando comandado pelo transmissor, e portanto a sincronização é garantida. [12]

Existem técnicas que compõem o sinal de clock e de dados em um único canal. Isso é usual quando transmissões síncronas são enviadas através de um modem. Dois métodos no qual os sinais de dados contém informação de tempo são: codificação NRZ (Non-Return-to-Zero) e a codificação Manchester. [12]

Em sistemas assíncronos, a informação trafega por um canal único. O transmissor e o receptor devem ser configurados antecipadamente para que a comunicação se estabeleça a contento. Um oscilador preciso no receptor irá gerar um sinal de clock interno que é igual (ou muito próximo) ao do transmissor. Para o protocolo serial mais comum, os dados são enviados em pequenos pacotes de 10 ou 11 bits, dos quais 8 constituem a mensagem. Quando o canal está em repouso, o sinal correspondente no canal tem um nível lógico ‘1’. Um pacote de dados sempre começa com um nível lógico ‘0’ (start bit) para sinalizar ao receptor que um transmissão foi iniciada. O “start bit” inicializa um temporizador interno no receptor avisando que a transmissão começou e que serão necessários pulsos de clocks. Seguido do start bit, 8 bits de dados de mensagem são enviados na taxa de transmissão especificada. O pacote é concluído com os bits de paridade e de parada (“stop bit”). [12]

O comprimento do pacote de dados é pequeno em sistemas assíncronos para minimizar o risco do oscilador do transmissor e do receptor variar. Quando

osciladores a cristal são utilizados, a sincronização pode ser garantida sobre os 11 bits de período. A cada novo pacote enviado, o “start bit” reseta a sincronização, portanto a pausa entre pacotes pode ser longa. [12]

#### 2.1.1.3. **Interface Serial RS232 (EIA232)**

RS é uma abreviação de “Recommended Standard”. Ela relata uma padronização de uma interface comum para comunicação de dados entre equipamentos, criada no início dos anos 60, por um comitê conhecido atualmente como “Electronic Industries Association” (EIA). Naquele tempo, a comunicação de dados compreendia a troca de dados digitais entre um computador central (mainframe) e terminais de computador remotos, ou entre dois terminais sem o envolvimento do computador. Estes dispositivos poderiam ser conectados através de linha telefônica, e consequentemente necessitavam um modem em cada lado para fazer a decodificação dos sinais.

Dessas idéias nasceu o padrão RS232. Ele especifica as tensões, temporizações e funções dos sinais, um protocolo para troca de informações, e as conexões mecânicas.

A mais de 30 anos desde que essa padronização foi desenvolvida, a EIA publicou três modificações.[13]

A mais recente, EIA232E, foi introduzida em 1991. Ao lado da mudança de nome de RS232 para EIA232, algumas linhas de sinais foram renomeadas e várias linhas novas foram definidas. [13]

Embora tenha sofrido poucas alterações, muitos fabricantes adotaram diversas soluções mais simplificadas que tornaram impossível a simplificação

da padronização proposta. As maiores dificuldades encontradas pelos usuários na utilização da interface RS232 incluem pelo menos um dos seguintes fatores:

- A ausência ou conexão errada de sinais de controle, resultam em estouro do buffer

(“overflow”) ou travamento da comunicação.

- Função incorreta de comunicação para o cabo em uso, resultam em inversão das linhas de Transmissão e Recepção, bem como a inversão de uma ou mais linhas de controle (“handshaking”).

Felizmente, os drivers utilizados são bastante tolerantes aos abusos cometidos, e os CIs normalmente sobrevivem. [13]

### 2.1.2. Telefonia Celular

Um telefone celular (português brasileiro) ou telemóvel (português europeu) é um aparelho de comunicação por ondas electromagnéticas que permite a transmissão bidireccional de voz e dados utilizáveis em uma área geográfica que se encontra dividida em células (de onde provém a nomenclatura celular), cada uma delas servida por um transmissor/receptor. A invenção do telefone celular ocorreu em 1947 pelo laboratório Bell, nos EUA[14].

Há diferentes tecnologias para a difusão das ondas eletromagnéticas nos telefones móveis, baseadas na compressão das informações ou na sua

distribuição: na primeira geração (1G) (a analógica, desenvolvida no início dos anos 80), com os sistemas NMT e AMPS; na segunda geração (2G) (digital, desenvolvida no final dos anos 80 e início dos anos 90): GSM, CDMA e TDMA; na segunda geração e meia (2,5G) (uma evolução à 2G, com melhorias significativas em capacidade de transmissão de dados e na adoção da tecnologia de pacotes e não mais comutação de circuitos), presente nas tecnologias GPRS, EDGE, HSCSD, EVDO e 1xRTT; na terceira geração (3G) (digital, com mais recursos, em desenvolvimento desde o final dos anos 90), como UMTS; na terceira geração e meia (3,5G), como HSDPA, HSPA e HSUPA. [14]

A indústria classifica os sistemas de telefonia móvel em gerações: a primeira geração (1G), analógica; a segunda geração (2G), digital; a segunda geração e meia (2,5G), com melhorias significativas em capacidade de transmissão de dados e na adoção da tecnologia de pacotes e não mais comutação de circuitos; a terceira geração (3G). E já em desenvolvimento a 4G (quarta geração). [14]

Com isso podemos definir um telefone celular como sendo uma estação móvel que funciona através de um sistema de comunicação sem fio, que possui características específicas, normalmente vinculadas a alguma geração de telefonia celular, como TDMA, CDMA, GSM, GPRS, EDGE, entre outras.

Em sua 1 geração, os telefones celulares eram utilizados exclusivamente para conversas, utilizada somente a banda de Voz. A partir de sua 2 geração tornou-se possível o trafego de dados, como o envio de mensagens SMS. Nas gerações atuais já se é possivel o trafego de dados em alta velocidade, o envio de vídeos, navegação na internet, e etc.

## 2.2. Software

### 2.2.1. **Web Services**

Nos últimos anos, a necessidade de conectar pessoas, informações e processos mudou a forma como o software vem sendo desenvolvido. Sistemas bem-sucedidos de TI exigem cada vez mais interoperabilidade entre

plataformas e serviços flexíveis que possam evoluir facilmente com o tempo. Isso tem levado ao domínio de XML como a linguagem universal para representar e transmitir dados estruturados que sejam independentes de linguagem de programação, plataforma de software e hardware. [6]

Criado sob a ampla aceitação de XML, os Web Services são aplicativos que usam transportes, codificações e protocolos padrão para troca de informações. Com amplo suporte entre fornecedores e empresas, os Web Services permitem que sistemas de computação em qualquer plataforma se comuniquem pelas intranets e extranets da empresa e na Internet com suporte para segurança de ponta a ponta, serviços de mensagens confiáveis, transações distribuídas e muito mais. [6]

Os Web Services baseiam-se em um conjunto central de padrões que descrevem a sintaxe e a semântica da comunicação por software: o XML fornece a sintaxe comum para a representação de dados; o protocolo SOAP (Simple Object Access Protocol) fornece a semântica para a troca de dados; o WSDL (Web Services Description Language) fornece um mecanismo para descrever as capacidades de um serviço da Web. Especificações adicionais, conhecidas de um modo geral como a arquitetura WS-*, definem a funcionalidade de detecção, os sistemas de eventos, os anexos, a segurança, os serviços de mensagens confiáveis, as transações e o gerênciamento dos Web Services. [6]

#### 2.2.1.1. **Padrões**

O grande consenso dos fornecedores com padrões e interoperabilidade comprovada fizeram a diferença dos Web Services, além das tecnologias de integração do passado. [6]

#### 2.2.1.2. **Padrões horizontais de Web Services**

##### 2.2.1.2.1. **Arquitetura WS-***

Quando o mercado de Web Services começou a crescer rapidamente, aumentou a necessidade de padrões avançados para a segurança, a confiabilidade e as transações dos Web Services. Os fornecedores desse serviço responderam a essa necessidade criando uma série de especificações, conhecidas, de um modo geral, como arquitetura WS-*. O objetivo dessas especificações é fornecer um plano gráfico para a funcionalidade avançada, mantendo a simplicidade dos serviços básicos da Web.

O atributo mais importante da arquitetura WS-* é a capacidade de combinação e reutilização de elementos. Essa capacidade de combinação e reutilização de protocolos permite o desenvolvimento incremental de soluções de Web Services apenas quando requisitos individuais (como segurança, serviços de mensagens confiáveis, anexos, detecção, etc.) são necessários. Isoladamente, cada um desses requisitos resolve uma necessidade básica. Como um todo, eles tratam funcionalidades de alto nível exigidas normalmente por aplicativos distribuídos. Assim, as especificações WS-* podem ser usadas de forma independente ou em combinação com uma outra. Isso elimina a complexidade e a sobrecarga associadas a especificações que tentam definir várias capacidades ou que estejam intimamente ligadas a outras especificações. Além disso, os desenvolvedores podem aplicar apenas a funcionalidade específica necessária para resolver a necessidade imediata. À medida que surgem novos requisitos dos aplicativos, novas especificações podem ser criadas, sem comprometer a compatibilidade com versões anteriores. [6]

#### 2.2.1.2.2. **Órgãos de padrões horizontais**

Até o momento, centenas de fornecedores de TI têm participado do processo de padronização de Web Services sob o patrocínio da W3C (World Wide Web Consortium), OASIS (Organization for the Advancement of Structured Information Standards) e WS-I (Web Services Interoperability Organization). [6]

- **W3C**

A base dos Web Services foi sedimentada em 1998, quando a W3C lançou o XML 1.0. Desde então, a W3C tem desempenhado um importante papel na padronização de Web Services, lançando especificações como WSDL, SOAP, WS-Addressing e MTOM (Message Transmission Optimization Mechanism). A Microsoft continua a desempenhar um papel ativo na W3C, assumindo posições de presidente nos grupos XQuery, XML Coordination e WSDL Working. A Microsoft também é membro eleito do grupo de arquitetura técnica e membro do grupo de coordenação de Web Services.

- **OASIS**

A OASIS desenvolveu importantes especificações de segurança para Web Services, incluindo WS-Security e SAML. A Microsoft é membro da Diretoria, do Conselho Executivo e do Comitê de Processos e Políticas. Além disso, a Microsoft é co-patrocinadora da criação do Conselho Técnico de UDDI (Universal Description Discovery and Integration), BPEL (Business Process Execution

Language), XrML (eXtensible rights Markup Language), WS-Security e WS-ReliableExchange.

- **WS-I**

Quando as especificações de Web Services começaram a surgir, ficou claro que o agrupamento de especificações em "perfis" era essencial para o aumento da interoperabilidade. Assim sendo, a Microsoft e outras empresas do setor fundaram a WS-I — uma organização aberta destinada a promover a interoperabilidade de Web Services. A WS-I lançou os perfis mais adotados de Web Services, incluindo o WS-I BasicProfile e o WS-I BasicSecurityProfile. Lançou também uma variedade de ferramentas para testes de conformidade.

#### 2.2.1.3. **Interoperabilidade**

A interoperabilidade da arquitetura WS-* é garantida por meio de dois processos: o envio de especificações a órgãos de desenvolvimento de padrões, como OASIS e W3C, e o Web Services Workshop Process. O envio a órgãos de desenvolvimento de padrões garante a análise das especificações por um comitê técnico formado por especialistas da indústria. O Web Services Workshop Process, que fornece um canal para comentários da comunidade e da indústria, baseia-se no princípio da IETF (Internet Engineering Task Force), que exige pelo menos duas implementações interoperacionais de uma especificação antes de ser submetida aos órgãos de desenvolvimento de padrões. A natureza complementar desses processos é fundamental para a

reconciliação de especificações atípicas e para a obtenção de amplo suporte da indústria para protocolos comuns de interoperabilidade. [6]

Atualmente, mais de 70 fornecedores fazem parte do processo WS-*. Com muitas das especificações padronizadas ou submetidas aos órgãos de desenvolvimento de padrões, os fornecedores estão voltando sua atenção à implementação da arquitetura WS-*. Os desenvolvedores podem esperar interoperabilidade nos produtos de fornecedores que implementarem tais especificações, permitindo que as empresas desenvolvam sistemas heterogêneos que estejam interligados. [6]

### 2.2.2. **Linguagem C**

Desenvolvida nos laboratórios Bell na década de 70, a partir da Linguagem B (criada no final dos anos 60 por Ken Thompson), que foi reformulada por Brian Kernighan e Dennis M. Ritchie e posteriormente renomeada para C. [17]

Podendo ser considerada como uma linguagem de médio nível, pois possui instruções que a tornam ora uma linguagem de alto nível e estruturada como o Pascal, se assim se fizer necessário, ora uma linguagem de baixo nível pois possui instruções tão próximas da máquina, que só o Assembler possui. [17]

De fato com a linguagem C podemos construir programas organizados e concisos (como o Pascal), ocupando pouco espaço de memória com alta velocidade de execução (como o Assembler). Infelizmente, dada toda a flexibilidade da linguagem, também poderemos escrever programas desorganizados e difíceis de serem compreendidos (como usualmente são os programas em BASIC). [17]

Devemos lembrar que a linguagem C foi desenvolvida a partir da necessidade de se escrever programas que utilizassem recursos próprios da linguagem de máquina de uma forma mais simples e portável que o assembler.

Uma análise superficial dos programas escritos em C e Clipper, nos permite perceber que a linguagem C supera em muito em dificuldade o programa análogo em Clipper. Ora, então porque não desenvolvermos programas somente em Clipper?

A inúmeras razões para a escolha da linguagem C como a predileta para os desenvolvedores “profissionais”. As características da Linguagem C servirão para mostrar o porquê de sua ampla utilização.

#### 2.2.2.1. **Características da Linguagem C**

- Portabilidade entre máquinas e sistemas operacionais.
- Dados compostos em forma estruturada.
- Programas Estruturados.
- Total interação com o Sistema Operacional.

Código compacto e rápido, quando comparado ao código de outras linguagem de complexidade análoga.

### 2.2.3. **XML (Extensible Markup Language)**

O XML é considerado um bom formato para a criação de documentos com dados organizados de forma hierárquica, como se vê frequentemente em documentos de texto formatados, imagens vetoriais ou bancos de dados.[18]

Pela sua portabilidade, um banco de dados pode através de uma aplicação escrever em um arquivo XML, e um outro banco distinto pode ler então estes mesmos dados. [18]

É uma linguagem recomendada pela W3C para gerar linguagens de marcação para necessidades especiais. [18]

É um subtipo de SGML (acrônimo de Standard Generalized Markup Language, ou Linguagem Padronizada de Marcação Genérica) capaz de descrever diversos tipos de dados. Seu propósito principal é a facilidade de compartilhamento de informações através da Internet. Entre linguagens baseadas em XML incluem-se XHTML (formato para páginas Web), RDF,SDMX ,SMIL, MathML (formato para expressões matemáticas), NCL, XBRL, XSIL e SVG (formato gráfico vetorial). [18]

#### 2.2.3.1. **Características**

Estimulado pela insatisfação com os formatos existentes (padronizados ou não), o World Wide Web Consortium (W3C) começou a trabalhar em meados da década de 1990 em uma linguagem de marcação que combinasse a flexibilidade da SGML com a simplicidade da HTML. O principio do projeto era criar uma linguagem que pudesse ser lida por software, e integrar-se com as demais linguagens. Sua filosofia seria incorporada por vários princípios importantes: [18]

- Separação do conteúdo da formatação
- Simplicidade e Legibilidade, tanto para humanos quanto para computadores
- Possibilidade de criação de tags sem limitação
- Criação de arquivos para validação de estrutura (Chamados DTDs)

- Interligação de bancos de dados distintos
- Concentração na estrutura da informação, e não na sua aparência

### 2.2.4. ASP.NET

O ASP.NET é uma linguagem de programação voltada para a Web e baseada no .NET Framework. Assim, herda todas as características deste Framework, podendo ser escrita em várias linguagens, como C# e Visual Basic .NET.

É através de um componente do IIS (Internet Information Service) que o .NET Framework é interpretado, possibilitando a exibição de paginas Web dinâmicas.

O .NET Framework é uma plataforma única para desenvolvimento e execução de sistemas e aplicações. Todo e qualquer código escrito e gerado pelo .NET poderá ser executado em qualquer dispositivo que possua o framework instalado. [19]

# CAPÍTULO 3 – DESENVOLVIMENTO

Este capítulo trata das especificações técnicas dos dispositivos envolvidos neste projeto e da implementação.

Primeiramente fala-se à respeito da descrição técnica de cada dispositivo, em seguida, será apresentada a topologia do projeto. Com esta breve descrição técnica, será explicado o desenvolvimento da interface que interpreta os dados enviados pela central de alarme, converte os dados e transmite para o serviço Windows desenvolvido, responsável por realizar o envio das mensagens para os celulares cadastrados via Internet.

E, por fim, a interface WEB desenvolvida para a gerência dos usuários, a fim de manter o cadastro dos números de celulares que receberão as mensagens SMS.

## 3.1 Especificações Técnicas

Nesta seção são apresentados todos os dispositivos utilizados e suas especificações técnicas.

### 3.1.1. **Central de alarme**

O modelo escolhido foi a Central de Alarme *JFL 322 voz*, produzido pela empresa brasileira JFL. É um equipamento que possui sensores de movimento e de abertura de portas, todos sem fio, divididos em 2 zonas. O modelo escolhido oferece suporte para integração, pois a mesma possui uma saída auxiliar de sinal e também uma entrada de sinal. A central fica ilustrada na imagem 3.1.


**Imagem 3.1 – Central de Alarme**

### 3.1.2. **MicroControlador**

O microcontrolador selecionado foi o PIC 16F628A, para sua utilização foi adquirido um kit de desenvolvimento da marca LabTools, esse kit dispõe também de uma interface serial, RS232, responsável por fazer a comunicação através de porta serial. O kit é composto também por uma gravadora flash, necessária para fazer a programação do microcontrolador. A imagem 3.2 ilustra o kit de desenvolvimento.

**Imagem 3.2 – Kit Desenvolvimento LabTools**

### 3.1.3. **Telefonia Celular**

Foi utilizado o sistema de telefonia celular convencional, aproveitando toda a infraestrutura disponível.

## 1.1. Topologia do Projeto

A topologia a ser utilizada no projeto pode ser melhor visualizada na Imagem1.1.

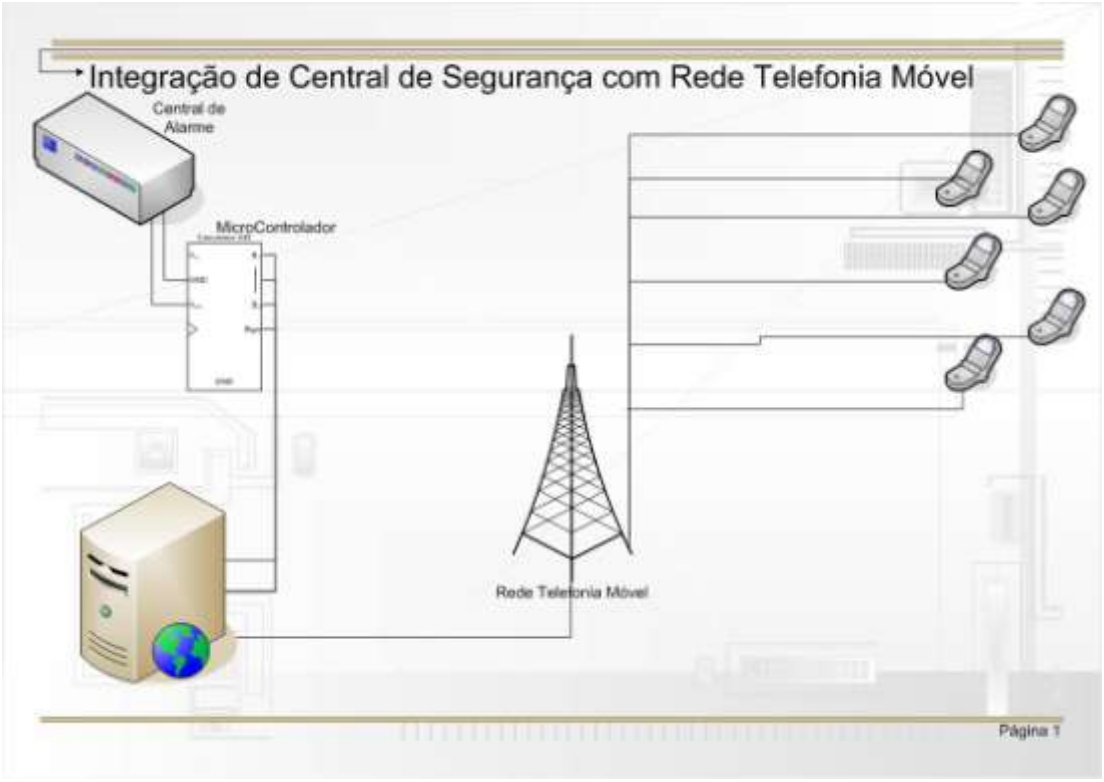


**Imagem 3.3 – Topologia do Projeto**

Conforme mostrado na imagem 3.3, o sinal analógico é emitido pela central de alarme, através de pulsos elétricos de +/- 5V, esses pulsos são recebidos pelo microcontrolador e convertidos em sinal digital (bits) e encaminhados para o serviço de monitoração, responsável por interpretar tal sinal e emitir os alertas para os usuários cadastrados através de uma chamada WebService via internet.

## 3.3. Protótipo

O protótipo desenvolvido compreende a integração dos seguintes componentes:

- Central de Alarmes JFL 322voz;
- Rede Telefonia Móvel Celular – Mensagens SMS;
- Kit de desenvolvimento composto por:

1. Microcontrolador PIC 16F628A,
2. Saída padrão RS232;

- Aplicação de Gerência em ASP.NET – FrameWork 2.0;
- Serviço Windows, Leitor do Microcontrolador – em .NET – FrameWork 2.0.

### 3.3.1. **Características do Protótipo**

Na Imagem 3.3 é mostrado o diagrama de funcionamento dos diversos componentes integrados. O microcontrolador é um dos componentes fundamentais, pois é nele onde os dados enviados pela central de alarme serão interpretados e enviados para o servidor, onde o envio das mensagens SMS serão realizados:

- Recebimento do sinal analógico (+/- 5v) , enviado pela central;
- Conversão do sinal analógico recebi em sinal digital, BITs;
- Envio do sinal, através de interface RS232(Porta Serial) ao servidor;
- Rotinas de tratamento e recepção dos dados transmitidos pela central;
- Execução das rotinas de envio de mensagens SMS para os celulares cadastrados na aplicação de gerência do sistema;

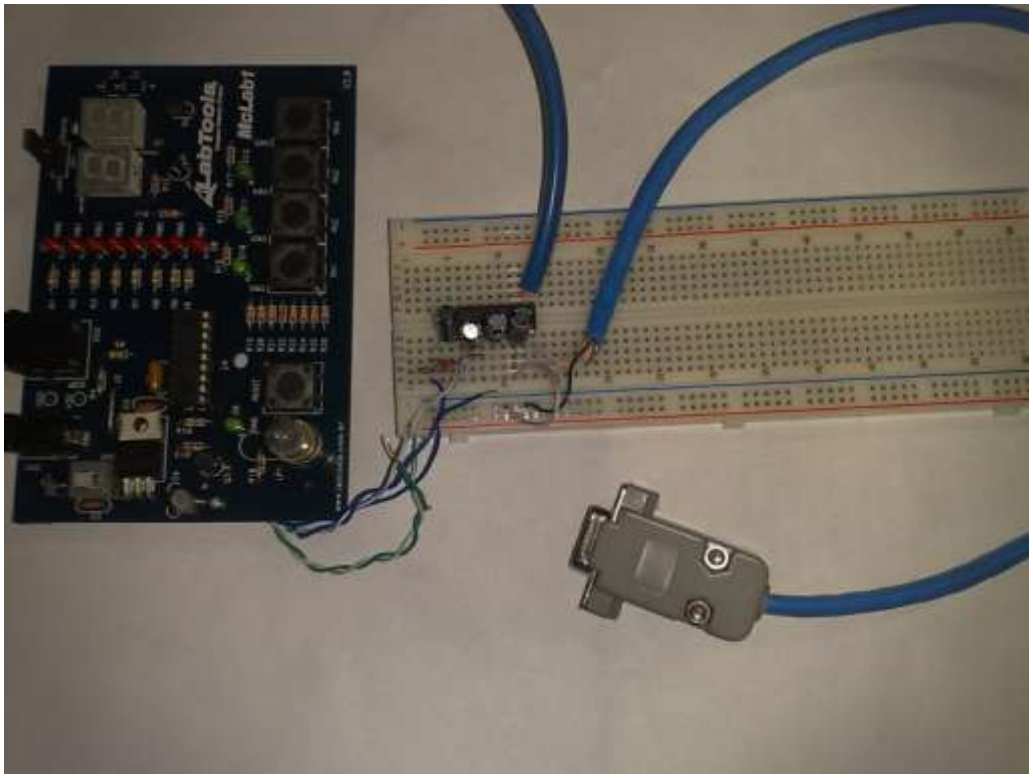


**Imagem 3.4 – Fotografia do projeto em fase de finalização**

Conectada à porta serial do servidor, onde está a base de dados com as informações dos usuários cadastrados para o recebimento das mensagens. No servidor temos, rodando, a aplicação à espera dos dados enviados pelo módulo conversor (microcontrolador), para que sejam feitas as consultas e envios das mensagens. Desta forma, havendo atividade de ativação, desativação ou disparo do alarme, a aplicação procederá no envio das mensagens.

O fluxo de dados é iniciado com o cadastramento de um usuário no módulo de gerência do sistema. Ele deve ser cadastrado no sistema, juntamente com o numero de seu telefone. Após o cadastro, o usuário está habilitado a receber as mensagens quando houverem alterações no status da central de alarme. O detalhamento para o cadastro de usuários se dará no item 3.4.2(Módulo de gerência).

Nesse módulo de gerência, alem de se cadastrar usuários, é possível consultar os usuários cadastrados e remove-los, se isso for desejado.

Os dados dos usuários são armazenados em arquivo XML, onde esse arquivo é alimentado com os cadastros de usuários.

## 3.4. **Software**

### 3.4.1. **Banco de Dados**

Depois de uma analise, foi definido que não seria necessária a utilização de um banco de dados para a persistência das informações do projeto, que são basicamente Nome do usuário e telefone o qual deve ser enviada as mensagens do sistema. Desta forma, as informações serão armazenas em arquivo do tipo XML. Cujo formato utilizado segue no modelo de código abaixo:

```
<?xml version="1.0" standalone="yes"?>
<MonitorAlarme>
  <Responsavel Nome="Cabide" Numero="556181124090" />
  <Responsavel Nome="fesd" Numero="556181893245" />
  <Responsavel Nome="João Maciel" Numero="556181291913" />
</MonitorAlarme
```

### 3.4.2. **Módulo Gestor**

O módulo gestor é responsável manter as informações referentes aos usuários cadastrados no sistema.

Este módulo foi desenvolvido com a utilização da tecnologia .NET Framework 2.0 e a linguagem utilizada foi o C#.

O módulo foi desenvolvido para ser utilizado via internet, WEBSITE, pois desta forma, os responsáveis por manter o sistema, poderão faze-lo via internet, facilitando a utilização e alteração dos usuários cadastrados no sistema.

O WebSite é composto por uma única pagina, na qual são exibidos todos os usuários cadastrados e dois botões botões para cada usuário,

responsáveis por alterar os dados de um usuário e remover o usuário do sistema. Essa tela também é composta por um campo, onde novos usuários poderão ser cadastrados.

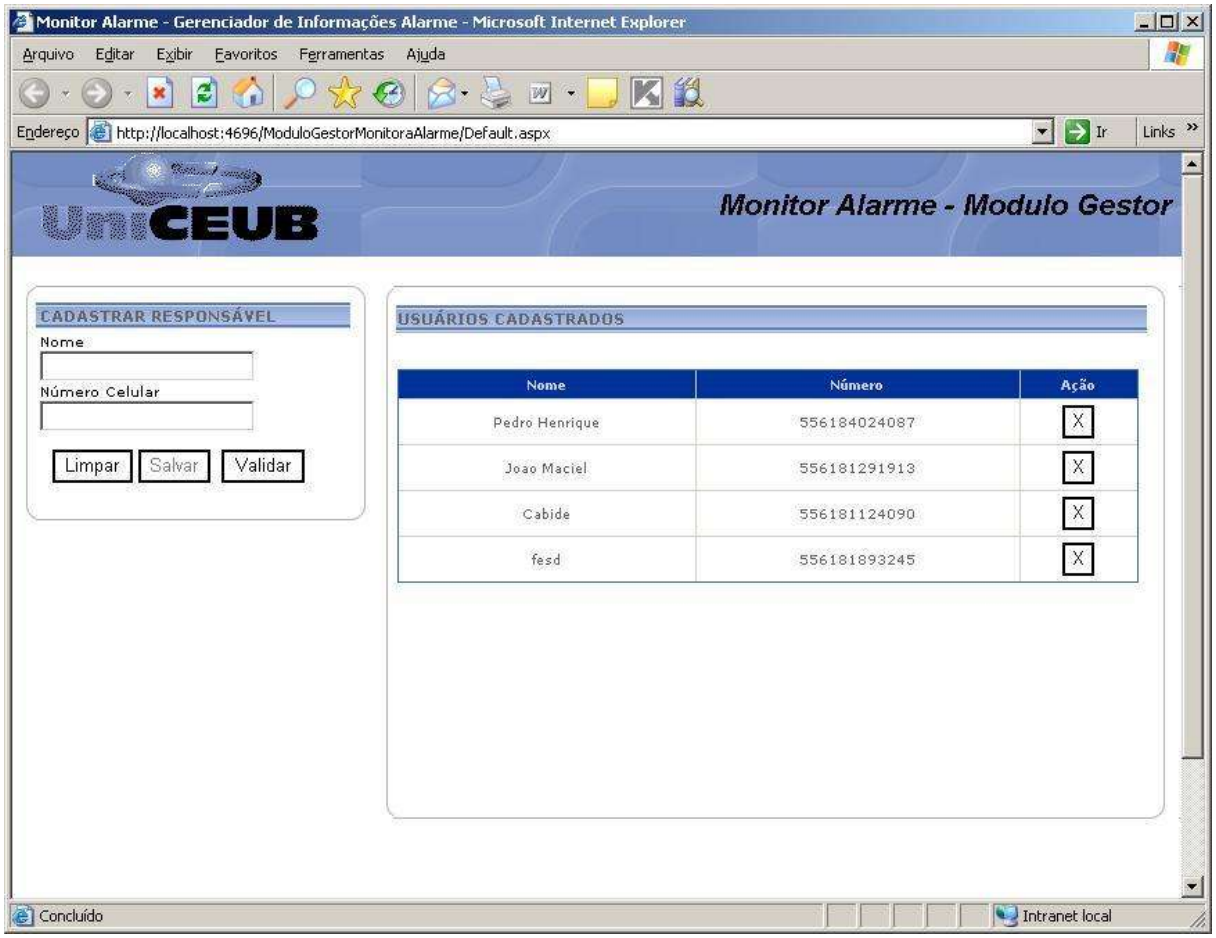


**Imagem 3.5 – Tela Módulo Gestor**

3.4.2.1. **Regras**

Para esse módulo a única regra que deve ser respeitada é no que diz respeito a inserção de novos usuários. Apenas números validos podem ser cadastrados no sistema. Desta forma, apenas após a validação do numero inserido que o usuário poderá ser incluído no sistema. Essa validação é feita através de uma chamada a um WebService, o qual valida se o numero informado é valido ou não e retorna o resultado através de uma variável String, quando a mesma e retornada vazia significa que a validação foi realizada com sucesso, caso contrario, o

conteúdo dessa variável conterá o resultado de erro retornado da validação.

O usuário apenas será cadastrado se o numero informado for validado com sucesso, caso contrario o usuário deverá informar um numero correto ou não poderá cadastrar o usuário.

Para essa validação, se utiliza o seguinte código(C#):

```
private string ValidaNumeroCelular(string SentId, string User, string
Pass, string CelNro)
    {
        br.com.cgi2sms.webservice.VolaSDKService websend = new
VolaSDKService();
        int hresult = websend.sendMessage(User, Pass, true, "Monitor
Alarme", CelNro, DateTime.Now.ToShortDateString() + " - Central de
Alarme Disparada. Chame a Policia", SentId, "");
        switch (hresult)
        {
            case 00:
                return "";
            case 15:
                return "Adquira Novos Créditos na Operadora de SMS";
            case 12:
                return "Número Inválido";
            default:
                return "Erro Inesperado, entre em contato com o
adminitrador do Sistema.";
        }
   }
```

### 3.4.3. **Serviço Windows**

O serviço windows é responsável por interpretar os dados enviados pelo microcontrolador e disparar as mensagem para os usuários.

Esse serviço ficará sendo executado em um servidor, que estará ligado ao microcontrolador, através de conexão serial.

Este módulo foi desenvolvido com a utilização da tecnologia .NET Framework 2.0 e a linguagem utilizada foi o C#.

A imagem 3.6 mostra o serviço Windows instalado no sistema operacional e sendo executado.

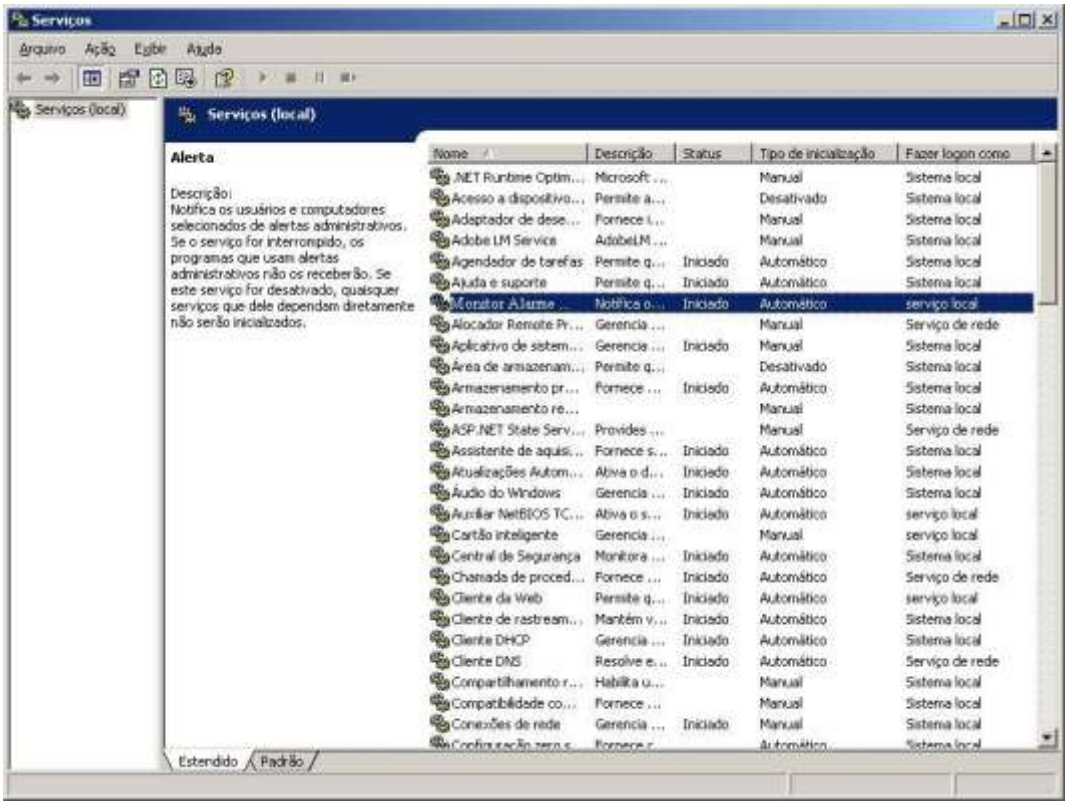


**Imagem 3.6 – Serviço Windows**

Para a comunicação serial da aplicação, foi usada a classe System.IO.Ports, disponibilizada pelo Microsoft .NET Framework 2.0.

O código utilizado para a leitura da porta serial segue abaixo:

```
Public Class PortChat
    Shared _continue As Boolean
    Shared _serialPort As SerialPort

    Public Shared Sub Main()
        Dim name As String
        Dim message As String
        Dim sComparer As StringComparer =
StringComparer.OrdinalIgnoreCase
        Dim readThread As Thread = New Thread(AddressOf Read)

        ' Create a new SerialPort object with default settings.
        _serialPort = New SerialPort()

        ' Allow the user to set the appropriate properties.
        _serialPort.PortName = SetPortName(_serialPort.PortName)
        _serialPort.BaudRate = SetPortBaudRate(_serialPort.BaudRate)
```

```
        _serialPort.Parity = SetPortParity(_serialPort.Parity)
        _serialPort.DataBits = SetPortDataBits(_serialPort.DataBits)
        _serialPort.StopBits = SetPortStopBits(_serialPort.StopBits)
        _serialPort.Handshake =
SetPortHandshake(_serialPort.Handshake)
End Sub
```

#### 3.4.3.1. **Regras**

As mensagens serão enviadas para todos os usuários cadastrados no sistema, de acordo com o quadro abaixo:

| Tipo de Evento | Estado Anterior | Estado Posterior | Mensagem |
|---|---|---|---|
| 1 | Central Desativada | Central Ativada | Central de Alarme Ativada. |
| 2 | Central Ativada | Central Disparada | Central de Alarme Disparada, possível invasão de pessoas não autorizadas, entrar em contato urgente com a Policia através do telefone 190. |
| 3 | Central Ativada | Central Desativada | Central de Alarme Desativada. |

**Quadro demonstrativo das regras do Sistema**

## 3.5. **Demonstração Prática**

Neste tópico será feita uma demonstração pratica do sistema em funcionamento.

Primeiramente será demonstrado o funcionamento do módulo de gerência.

A imagem 3.7 ilustra o cadastramento de um número de celular inexistente.

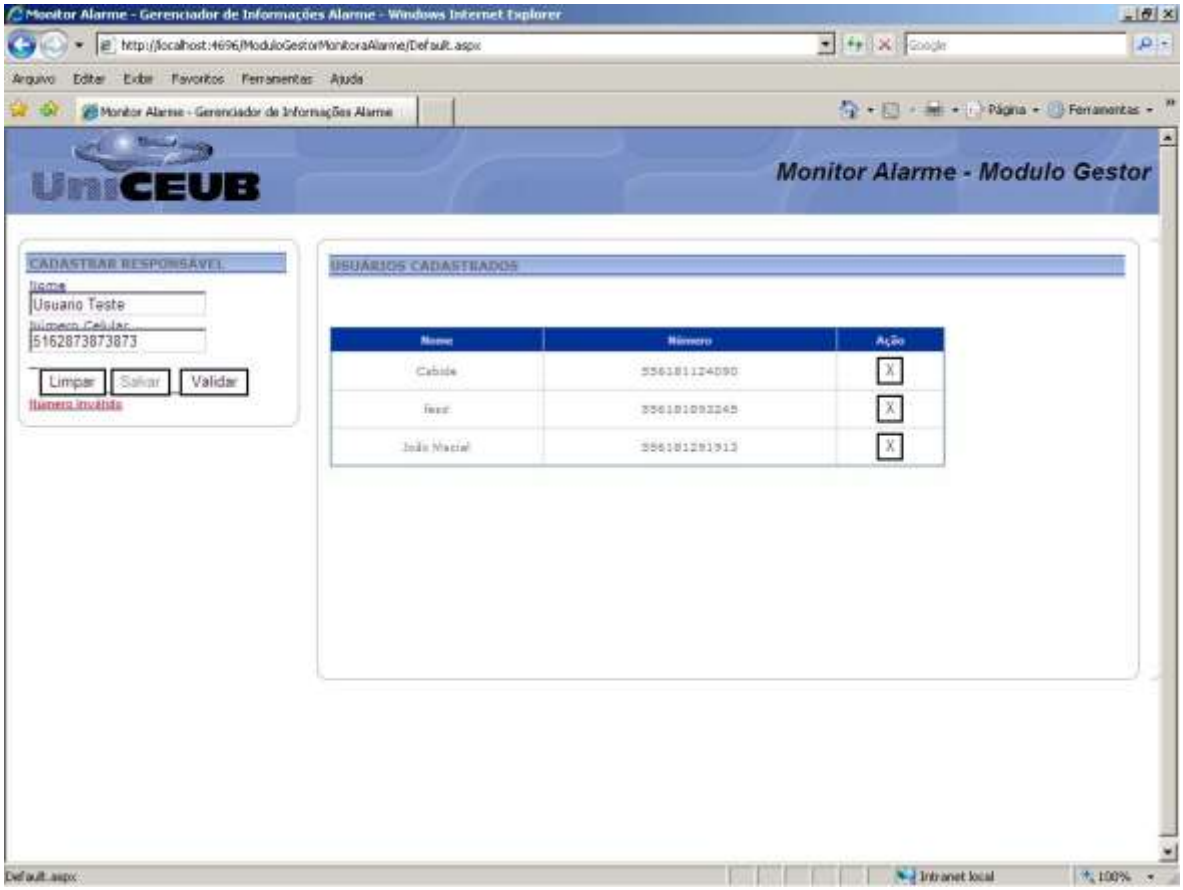


**Imagem 3.7 – Módulo de Gerência, número inválido**

Note que a mensagem “número inválido” é exibida e o usuário não e aceito pelo sistema. Após isso o número foi corrigido para um numero de celular válido. O resultado pode ser visto na imagem 3.8.

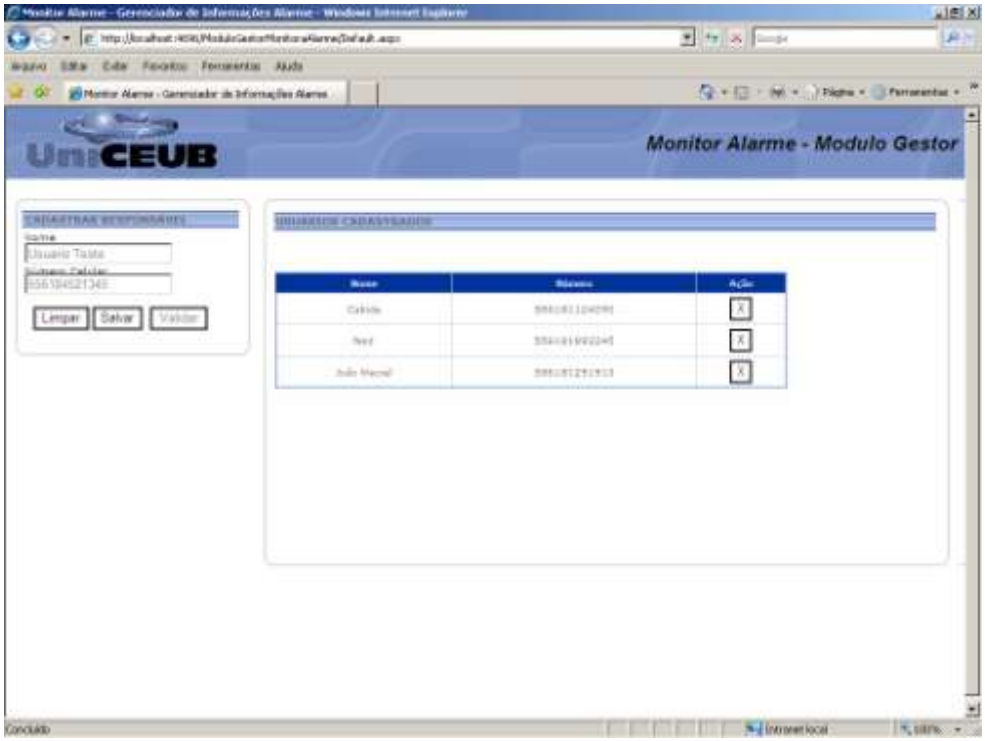


**Imagem 3.8 – Módulo de Gerência, Número valido**

Note que a validação sendo bem sucedida, o botão salvar é habilitado, após clicado o botão salvar o usuário e cadastrado no sistema conforme ilustrado na imagem 3.9.

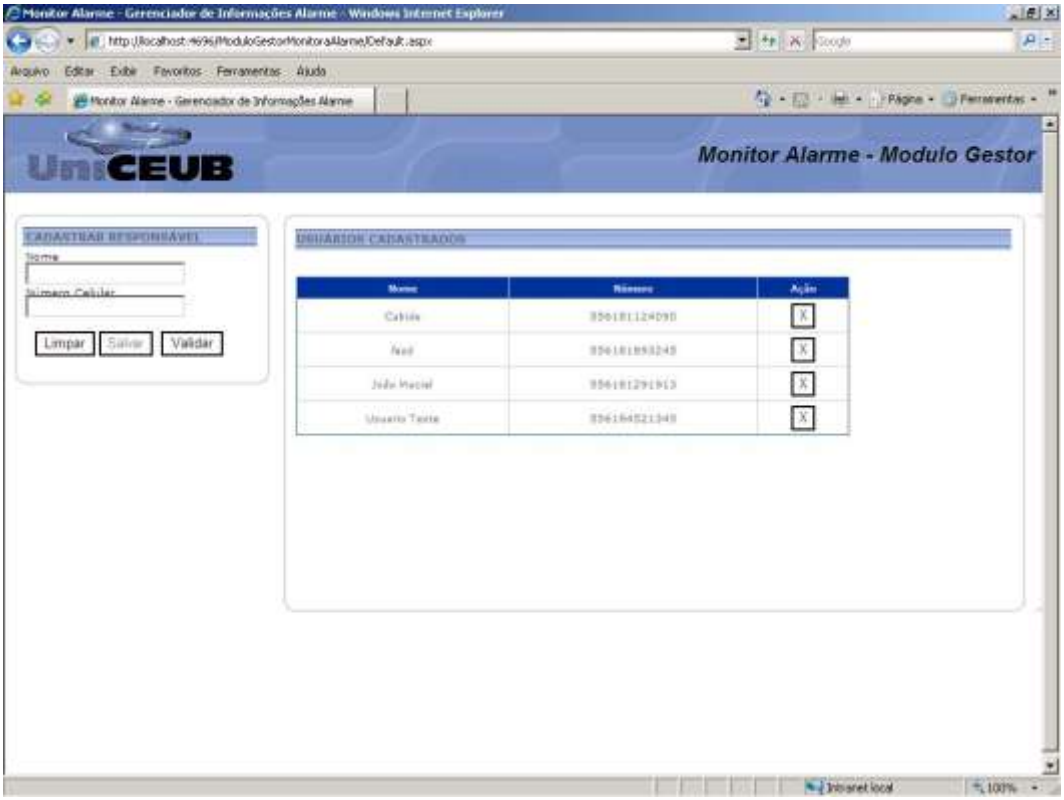


**Imagem 3.9 – Módulo de Gerência, Usuário Cadastrado**

Após a atualização do cadastro de usuários no sistema, através do módulo de gerência, é iniciada a utilização da central de alarmes.

A imagem 3.10 mostra a central de alarme sendo ativada, através de um clique no botão.

**Imagem 3.10 – Botão sendo clicado, Central Ativada**

Neste momento a central emitiu o sinal de ativação e a mensagem foi disparada para os celulares cadastrados. A imagem 3.11 mostra a mensagem recebida no celular.


**Imagem 3.11 – SMS Recebido, Central Ativada**

Após esse momento uma porta foi aberta, conforme ilustrado nas imagens 3.12 e 3.13.

**Imagem 3.12 – Sensor, Porta Fechada**

**Imagem 3.13 – Sensor, Porta Aberta**

Após a porta ter sido aberta, novamente a central emitiu um sinal, porem o sinal foi de violação de perímetro. Nova mensagem foi emitida, conforme imagem 3.14.


**Imagem 3.14 – SMS Recebido, Central Disparada**

A imagem 3.15 mostra agora a central de alarme sendo desativada, através de um clique no botão.

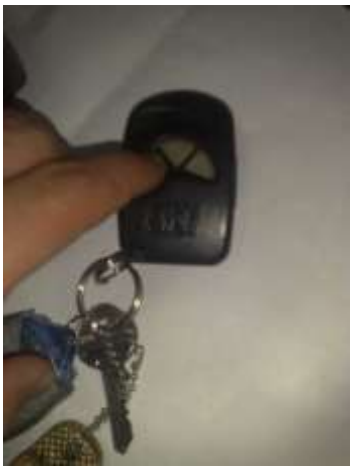

**Imagem 3.15 – Botão sendo clicado, Central Desativada**

Neste momento a central emite novamente um sinal de desativação e a mensagem foi disparada para os celulares cadastrados. Na imagem 3.16 é mostrada a mensagem recebida no celular.


**Imagem 3.16 – SMS Recebido, Central Desativada**

## 3.6. Problemas encontrados

Descrevem-se abaixo os problemas encontrados e a solução para resolvê-los durante o projeto.

1) Queima de componentes:O microcontrolador queimou quando o projeto estava em fase de testes,ocasionando atrasados na execução da parte de testes. Novo microcontrolador teve que ser adquirido.

2) Dificuldades na comunicação serial:A montagem do circuito RS232 teve uma complexidade muito elevada, foi necessária a montagem de um circuito complementar no protoboard.

# CAPÍTULO 4 – CONCLUSÕES

Este projeto teve como finalidade o desenvolvimento de uma Interface composta por um microcontrolador, responsável por converter dados analógicos transmitidos por uma central de alarme, converte-los para um sinal digital e transmiti-los para um serviço windows (utilizando linguagem de programação C#, tecnologia Windows Services) e através deste serviço, estabelecer a conexão com um provedor de mensagens SMS, através de um WebService. A utilização deste provedor se fez necessária pois ele fornece convenio para envio de mensagens SMS para todas as operadoras de telefonia móvel do Brasil, exceto VIVO. Essa conexão com o provedor de mensagens é feita através de uma chamada SOAP (WebService) através da Internet.

Paralelamente ao desenvolvimento desta Interface, desenvolveu-se a interface Web, denominado de Módulo de Gerência que seria responsável pela manutenção e armazenamento dos dados dos celulares que devem receber as mensagens do sistema. Portanto, foi desenvolvida uma solução completa de recepção, conversão de sinal, interpretação dos dados e envio de mensagens SMS. Para toda a Interface WEB e Serviço Windows foi utilizada a tecnologia Microsoft.NET – Framework 2.0 e ASP.NET e para a interface de conversão de sinal foi utilizada a linguagem C.

De acordo com o estudo e os testes realizados neste projeto, constatou-se que se pode fornecer uma solução de comunicação em tempo real de eventos em um local monitorado através de uma central de alarme de baixo custo, utilizando redes e transmissões já existentes(Internet e Rede Telefonia Móvel), que tornam a solução deste projeto, uma solução de alta disponibilidade.

Nenhum tipo de problema ou indisponibilidade dos serviços de telefonia móvel durante toda a fase de implementação, desenvolvimento e testes foram detectados.

Também não se detectou nenhuma falha ou indisponibilidade nos serviços de hospedagem da Interface Web, tendo em vista que o mesmo foi hospedado localmente, possibilitam assim a manutenção e visualização do cadastro de usuários cadastrados para o recebimento de mensagens SMS.

Podemos assim, concluir que o objetivo deste projeto foi atingido com sucesso.

Como proposta de trabalhos futuros, sugere-se a melhoria da interface Web e do Microcontrolador para que seja possível alterar o staus da central de alarme através do envio de mensagens SMS. Isso possibilitaria a ativação ou desativação da central de alarmes remotamente

Além disso, seria interessante possibilitar uma forma de registro de Logs para toda a comunicação SMS, assim como das alterações feitas no sistema, como por exemplo, a exclusão ou cadastro de usuários.

# REFERÊNCIAS BIBLIOGRÁFICAS


[1] DEITEL H. M. 2003 – DEITEL, **C# Como programar.** 1. Ed. Pearson Education, 2003.

[2] AHMED, Ashfaq. **Eletrônica de Potência.** São Paulo: Prentice Hall, 2000.

[3] MALVINO, Albert Paul. **Eletrônica: volume 1.** 4. ed. São Paulo: Makron Books, 1995.

[4] BASIURA, Russ. **Professional ASP.NET Web Services**. 4 ed. São Paulo: Makron Books, 2003

[5] DEITEL, H. M. , DEITEL, P. J. **C# for Proframmers.** 2. ed. São Paulo: Prentice-Hall, 2005.

[6] MSDN Microsoft **[Home Page].** 2007. Disponível em: <http://www.msdn.com.br>. Acesso em: 15 de Março de 2006.

[7] PEREIRA, Fabio **MicroControladores PIC: Programação em C.** 1. ed. São Paulo: Érica, 2003.

[8] Microchip Datasheets – Fabricante PIC **[Home Page].** 2007. Disponível em: <http://www.microchip.com>. Acesso em: 25 de Abril de 2007.

[9] MSDN Microsoft Development Library - http://msdn.microsoft.com/pt-br/library/aa932479.aspx.

[10] Site http://economia.dgabc.com.br/default.asp?pt=secao&pg=detalhe&c=3&id=24689 Acessado no dia 22/05/2009.

[11] Site http://www.coinfo.cefetpb.edu.br/professor/leonidas/irc/apostilas/comun_serial.pdf. Acessado no dia 03/04/2007.

[12] Site www.dee.ufcg.edu.br/~aco/Lab_Arquitetura/MODULO%2010.doc Acessado no dia 04/04/2007.

[13] Site www.cerne-tec.com.br/Comunica%E7%E3o232485.pdf Acessado no dia 09/05/2007.

[14] Site http://pt.wikipedia.org/wiki/Telefone_celular Acessado no dia 03/06/2009.

[15]. TANENBAUM 2003 - TANENBAUM, A. S. *Redes de Computadores*. 4. ed. Rio de Janeiro, Brasil: Ed. Campus, 2003.

[16] Site http://www.brasilwiki.com.br/noticia.php?id_noticia=7222 Acessado no dia 22/05/2009.

[17] Site http://www.unimep.br/~vmdzilio/turboc/historic.htm. Acessado no dia 15/03/2008.

[18] Site http://www.tecnologiacursos.com.br/tecnologia/principal/conteudo.asp?id=4395. Acessado no dia 15/03/2008

[19] Site www.webartigos.com/articles/17989/1/microsoft-visual-studio-team-system-e-projeto-rosario/pagina1.html. Acessado no dia 20/10/2008.

[20] VASCONCELOS, Laércio. **Hardware Total.** São Paulo: Makron Books, 2002.

# ANEXO I

Transcreve-se abaixo parte do manual de utilização do WebService disponibilizado pelo provedor de Mensagens SMS.

## 4. Classes

Os métodos descritos no item 5 se utilizam de algumas classes básicas disponibilizadas pelo webservice e
descritas em detalhes nos itens abaixo.

### 4.1 Message

Classe utilizada para armazenar os dados relativos a uma mensagem a ser enviada a um destinatário. A
classe possui as seguintes propriedades:

**Propriedade Tipo Descrição**

ID string Identificador da mensagem com até 20 caracteres. É utilizado para referenciar uma mensagem enviada no momento da verificação de seu status de entrega

body string Conteúdo da mensagem. 150 caracteres para todas as operadoras exceto para a Nextel que é de 500 caracteres

schedule string Data/Hora para agendamento de envio da mensagem (a mensagem é armazenada em nosso servidor e enviada para a operadora no momento especificado). O
formato é AAAA-MM-DD hh:mm:ss

sender string Remetente da mensagem com até 10 caracteres. O texto definido neste campo é concatenado ao início da mensagem diminuindo o conteúdo máximo da propriedade
"body" em até 10 caracteres. Caso seja deixado vazio (em branco), nada é
concatenado ao início da mensagem

target string Destinatário da mensagem. É necessário que o número esteja no formato internacional sem o sinal de "+", ou seja, para enviar uma mensagem para um
telefone de São Paulo, por exemplo, deve-se colocar na forma 5511NNNNNNNN
onde o 55 é o código do Brasil e 11 o de SP.

### 4.2 StatusPartResult

Após o disparo da verificação de status de uma ou mais mensagens, o sistema retorna um array de
StatusPartResult contendo, cada um, o status de uma das mensagens verificadas.

**Propriedade Tipo Descrição**

ID string Identificador da mensagem com até 20 caracteres

explanation string Descrição textual do status de entrega da mensagem

status int Código do status relacionado à mensagem (ver Apêndice B)

### 4.3 MessageStatusResult

Armazena a resposta à uma verificação de status de mensagens enviadas.

**Propriedade Tipo Descrição**

result string Código de erro geral relacionado com a verificação de status (ver Apêndice A)

status array Array de StatusPartResult



### 4.4 TargetPartResult

Armazena o resultado individual de cada mensagem no envio através dos métodos SendBatch e SendSeveral

**Propriedade Tipo Descrição**

result int Código de erro relativo ao destinatário (ver Apêndice A)

target string Destinatário

### 4.5 ID

Armazena os identificadores de mensagens enviadas a serem passados para o método MessageReceive de
forma a verificar seu status de entrega.

**Propriedade Tipo Descrição**

id array Array de strings sendo, cada uma, um identificador de mensageme enviada

### 4.6 ReceivedPartResult

Armazena os dados de uma mensagem M.O. enviada de um aparelho celular para determinado número de
nosso sistema.

**Propriedade Tipo Descrição**

dateTime string Data/Hora em que a mensagem chegou ao nosso servidor
message string Conteúdo da mensagem
sender string Número do aparelho que originou a mensagem (número do celular do remetente)

### 4.7 MessageReceiveResult

Resposta a uma verificação por mensagens recebidas.

**Propriedade Tipo Descrição**

result int Código de erro geral relacionado com a verificação de status (ver Apêndice A)
messageCount int Quantidade de mensagens recebidas na requisição
messages array Array de ReceivedPartResult

### 4.8 SendResult

Resposta a uma requisição de envio de mensagens através dos métodos SendBatch e SendSeveral.

**Propriedade Tipo Descrição**

result int Código de erro geral relacionado com a verificação de status (ver Apêndice A)
messages array Array de TargetPartResult

### 4.9 CreditResult

Resposta a uma requisição de verificação de créditos disponíveis.

**Propriedade Tipo Descrição**

result int Código de erro geral relacionado com a verificação de status (ver Apêndice A)
credit float Quantidade de créditos restantes



### 4.10 Target

Armazena os dados de destinatário para envio de mensagens através do método SendBatch.

**Propriedade Tipo Descrição**

ID string Identificador da mensagem com até 20 caracteres
phone string Destinatário da mensagem. É necessário que o número esteja no formato internacional sem o sinal de "+", ou seja, para enviar uma mensagem para um
telefone de São Paulo, por exemplo, deve-se colocar na forma 5511NNNNNNNN
onde o 55 é o código do Brasil e 11 o de SP.



## 5. Métodos

O webservice disponibiliza uma série de métodos, listados abaixo e explorados em mais detalhes nos
seguintes itens:

- **GetCredit** : busca informações de créditos do usuário
- **SendMessage** : envia uma mensagem para um destinatário
- **SendBatch** : envia uma única mensagem para vários destinatários
- **SendSeveral** : envia várias mensagens diferentes (uma mensagem por destinatário)
- **GetMessageStatus** : busca o status de entrega das mensagens
- **MessageReceive** : recebe mensagens MO

### 5.1 GetCredit

Busca os créditos restantes na conta do usuário.

**Parâmetros Nome Tipo Descrição**

Entrada user string Usuário do sistema
password string Senha
Saída CreditResult Ver item 4.9

### 5.2 SendMessage

Envia uma mensagem para apenas um destinatário.

**Parâmetros Nome Tipo Descrição**

Entrada username string Usuário do sistema

password string Senha
testMode boolean Ativa/Desativa modo teste (ver Apêndice D)
sender string Remetente da mensagem. Alfanumérico com até 10 dígitos podendo ser vazio
target string Destinatário da mensagem. Número de telefone contendo o código do país (55 no caso do Brasil). Ex: para enviar uma mensagem para São Paulo usar 5511NNNNNNNN
body string Conteúdo da mensagem (ver Apêndice C para tamanho)
ID string Identificador da mensagem enviada com até 20 caracteres alfanuméricos. É usado para posterior verificação do status da mensagem
schedule string Caso se deseje agendar a mensagem para posterior envio para o destinatário deve-se usar este campo, colocando data/hora no formato AAAA-MM-DD hh:mm:ss
Retorno result integer Código de erro resultante da chamada do método


## 5.3 SendBatch

Envia uma mensagem para vários destinatários. Caso for enviada uma mensagem para 10 destinatários, o
sistema irá expandir para 10 mensagens iguais enviadas a 10 destinatários, sendo tarifadas como tal.

**Parâmetros Nome Tipo Descrição**
Entrada user string Usuário do sistema
password string Senha
testMode boolean Ativa/Desativa modo teste (ver Apêndice C)
sender string Remetente da mensagem. Alfanumérico com até 10 dígitos podendo ser vazio
target Array de Target
Array contendo vários destinatário, cada um em um objetod da classe Target (ver item 4.10)
mensagem string Conteúdo da mensagem (ver Apêndice B para tamanho)
schedule string Caso se deseje agendar a mensagem para posterior envio para o destinatário deve-se usar este campo, colocando data/hora no formato AAAA-MM-DD hh:mm:ss
Retorno SendResult Ver item 4.8

## 5.3 SendSeveral

Envia várias mensagens podendo, cada uma delas, ter conteúdos e destinatários diferentes

**Parâmetros Nome Tipo Descrição**
Entrada user string Usuário do sistema
password string Senha
testMode boolean Ativa/Desativa modo teste (ver Apêndice C)
message Array de Message
Array contendo as mensagens a serem enviadas, cada uma em um objeto da classe Message (ver item 4.1)
Retorno SendResult Ver item 4.8

## 5.4 GetMessageStatus

Verifica o status de entrega de mensagens enviadas previamente.

**Parâmetros Nome Tipo Descrição**
Entrada user string Usuário do sistema
password string Senha
ids Array de string Array de strings contendo os identificadores das mensagens para verificação do status. Cada ID deve ocupar uma posição do array.
Retorno MessageStatusResult Ver item 4.3


## 5.5 MessageReceive

Verifica se o servidor está armazenando mensagens enviadas a partir de um aparelho celular e as retorna
em caso positivo

**Parâmetros Nome Tipo Descrição**
Entrada user string Usuário do sistema

password string Senha
Retorno MessageReceiveResult Ver item 4.7


## Apêndice A – Códigos de Erro

Os códigos de erro retornados pelo sistema são os seguintes:
00 – Operação realizada com sucesso
01 – Erro de conexão com o servidor
02 – Erro no retorno dos parâmetros do servidor para o WebService
03 – Erro na verificação do resultado do envio
04 – Usuário / senha vazios
06 – Mensagem idêntica já está sendo enviada nesta requisição
07 – Data/hora de agendamento inválida
08 – Erro de acesso a banco de dados
10 – Username e/ou Senha inválido(s)
11 – Parâmetro(s) inválido(s) ou faltando
12 – Número de telefone inválido ou não coberto pelo sistema
13 – Operadora desativada para envio de mensagens
14 – Usuário não pode enviar mensagens para esta operadora
15 – Créditos insuficientes
16 – Tempo mínimo entre duas requisições em andamento
17 – Permissão negada para a utilização do CGI/Produtos
18 – Operadora Offline
19 – Envio negado a partir do IP de origem

## Apêndice B – Códigos de Status de Mensagens

Os códigos de status para mensagens postadas no servidor do CGI2Sms são:
0 – aguardando envio da mensagem para a operadora
1 – cancelada antes do seu envio para a operadora
2 – expirada sem ter sido enviada para a operadora
3 – enviada para a operadora
4 – mensagem na operadora aguardando envio para aparelho celular
5 – entregue ao aparelho celular
6 – rejeitada pela operadora
7 – expirada dentro da operadora (aparelho desligado ou fora de área por muito tempo)
8 – removida da operadora por algum motivo

# APÊNDICE I

Código do Aplicativo desenvolvido para a Interface Web.

**Default.aspx**

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default" EnableEventValidation="false"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<HTML>
  <HEAD
id=topo_Head1>
                    <title>Monitor Alarme - Gerenciador de Informações Alarme </title>
                    <link href="estilo/estilo.css" rel="stylesheet" type="text/css">
                              <script language="javascript">
      function openPagina(theURL) { //v2.0
                    var winName
                    var features
                    winName = "FD";
                    features = "width=600,height=380";
          //,winName,features
                    window.open(theURL,winName,features);
      }
                              </script>
</HEAD>
          <body style="MARGIN:0px">
```

```
<form name="form1" method="post" action="Default.aspx" id="Form2" runat="server">
<table width="100%" height="100%" border="0" cellpadding="0" cellspacing="0">
<tr>
<td colspan="2" height="85" valign="top">
<!-- estilo inicio -->
<!-- estilo fim -->
<!-- topo inicio -->
<table bgcolor="#f0f0f0" width="100%" border="0" cellpadding="3" cellspacing="0" height="64">
<tr>
<td background="Imagens/topo_Logo.gif" style="width: 246px; height: 71px; text-align: center">
</td>
<td background="Imagens/topo.gif" style="height: 71px; text-align: right">
 <asp:Label ID="lblTitulo" runat="server" Font-Bold="True" Font-Italic="True"
Font-Names="Arial" Font-Size="16pt" Text="Monitor Alarme - Modulo Gestor"></asp:Label></td>
</tr>
</table>
<!-- topo fim -->
</td>
</tr>
<tr valign="top">
<td width="230" style="height: 559px">
<!-- menu inicio -->
```

```
                                        <table border="0" cellpadding="0"
cellspacing="0" style="BORDER-COLLAPSE: collapse" bordercolor="#000099"
                                                id="AutoNumber1" height="65"
width="260" class="BancoobFont">
                                                <tr>
                                                    <td
background="imagens/supesq.gif" height="23" style="width: 27px"><font
size="1"> </font></td>
                                                    <td
background="imagens/horizontal1.gif" width="245" height="23"><font
size="1"> </font></td>
                                                    <td
background="imagens/supdir.gif" width="26" height="23"><font size="1"> </font></td>
                                                </tr>
                                                <tr>
                                                    <td height="100%"
background="imagens/vertical1.gif" style="width: 27px"> </td>
                                                    <td valign="top" width="245">
                                                        <table
bgcolor="#f0f0f0" width="100%" border="0" cellpadding="3" cellspacing="0" align="center">
                                                            <tr>
                                                                <td
background="Imagens/barraDegrade.gif" width="208" style="height: 8px">
                                                                    <span style="font-size: 8pt"><strong>
                                                                    CADASTRAR RESPONSÁVEL</strong></span></td>
                                                            </tr>
                                                            <tr>
                                                                <td
class="TxtLinkHome" bgcolor="#ffffff" width="208" height="18" style="text-align: left">
                                                                    <table border="0" cellpadding="0" cellspacing="0" width="200">
```

```
                                    <tr>

                                        <td width="350">


        <asp:Label id="lblCoopCentral" runat="server" Width="104px" Height="11px">Nome</asp:Label><br />

                                    <asp:TextBox ID="tbxNome" runat="server"></asp:TextBox><br />


        <asp:Label id="lblCooperativa" runat="server" Width="120px" Height="8px">Número Celular</asp:Label><br />

                                    <asp:TextBox ID="tbxNumero" runat="server"></asp:TextBox><br />

                                          

                                    </td>

                                </tr>

                                <tr>

                                    <td style="text-align: center" width="350">


        <asp:Button ID="btnLimpar" runat="server" Text="Limpar" BackColor="White" BorderColor="Black" OnClick="btnLimpar_Click" TabIndex="1" UseSubmitBehavior="False" />

                                        <asp:Button ID="btnSalvar" runat="server" Text="Salvar" OnClick="btnSalvar_Click" BackColor="White" BorderColor="Black" Enabled="False" TabIndex="2" UseSubmitBehavior="False" />

                                     <asp:Button ID="btnValidarTelefone" runat="server" Text="Validar" OnClick="btnValidarTelefone_Click" BackColor="White" BorderColor="Black" /></td>

                                </tr>

                                <tr>

                                    <td>

                                    <asp:Label ID="lblMensagem" runat="server" Font-Bold="True" Font-Names="Arial" Font-Size="X-Small"
```

```
                                ForeColor="Red"                                Height="100%"
Width="90%"></asp:Label></td>

                                    </tr>

                                </table>

                                                                                    </td>

                                                                            </tr>

                                                                        </table>

                                                                    </td>

                                                                    <td
background="imagens/vertical1.gif" width="26"> </td>

                                                                </tr>

                                                                <tr>

                                                                    <td
background="imagens/infesq.gif"          height="23"          style="width:          27px"><font
size="1"> </font></td>

                                                                    <td                    height="10"
background="imagens/horizontal1.gif" width="245"><font size="1"> </font></td>

                                                                    <td
background="imagens/infdir.gif" height="23" width="26"><font size="1"> </font></td>

                                                                </tr>

                                                            </table>

                                                            <div class="BancoobFont">

                                                                   </div>

                                                        </td>

                                                        <td width="100%" style="height: 559px">

                                                            <table          border="0"          cellpadding="0"
cellspacing="0" style="WIDTH: 104%; BORDER-COLLAPSE: collapse;"

                                                                bordercolor="#000099"       id="Table3"
class="BancoobFont">
```

```
                                                    <tr>

                                                        <td
background="imagens/supesq.gif" style="width: 7px; height: 23px"></td>

                                                        <td
background="imagens/horizontal1.gif" style="height: 23px"></td>

                                                        <td
background="imagens/supdir.gif" width="21" style="height: 23px"></td>

                                                    </tr>

                                                    <tr>

                                                        <td
background="imagens/vertical1.gif" style="width: 7px; height: 360px;"> </td>

                                                        <td valign="top" width="90%"
align="center" style="height: 360px">

                            <table width="100%">

                                <tr>

                                    <td background="Imagens/barraDegrade.gif" style="width: 100px;
height: 18px; text-align: left">

                                        <strong><span style="font-size: 8pt">

                                            <asp:Label ID="Label1" runat="server" Font-Bold="True"
Text="USUÁRIOS CADASTRADOS"

                                                Width="293px"></asp:Label></span></strong></td>

                                </tr>

                                <tr>

                                    <td align="center" style="width: 100px; height: 201px">

                                                                <asp:DataGrid
id="dgResponsaveis" runat="server" Width="535px" Height="21px" BackColor="White"
BorderColor="#3366CC"


        BorderStyle="Solid" BorderWidth="1px" CellPadding="4" Font-Names="Verdana" Font-
Size="XX-Small"                                                 AutoGenerateColumns="False"
```

```
OnItemDataBound="dgResponsaveis_ItemDataBound"
OnItemCommand="dgResponsaveis_ItemCommand">

<FooterStyle ForeColor="#003399" BackColor="#99CCCC">

</FooterStyle>



<HeaderStyle Font-Bold="True" ForeColor="#CCCCFF" BackColor="#003399">

</HeaderStyle>

                        <Columns>

                            <asp:BoundColumn                                DataField="Nome"
HeaderText="Nome"></asp:BoundColumn>

                            <asp:BoundColumn                               DataField="Numero"
HeaderText="N&#250;mero"></asp:BoundColumn>

                            <asp:ButtonColumn                          ButtonType="PushButton"
CommandName="Select" Text=" X " HeaderText="A&#231;&#227;o"></asp:ButtonColumn>

                        </Columns>

                                                                    </asp:DataGrid></td>

                        </tr>

                      </table>

                        

                    </td>

                                                        <td
background="imagens/vertical1.gif" style="height: 360px"> </td>

                                                    </tr>

                                                    <tr>

                                                        <td
background="imagens/infesq.gif"          style="width:          7px;          height:          19px"><font
size="1"> </font></td>
```

```
                                                    <td
background="imagens/horizontal1.gif"        width="638"        style="height:        19px"><font
size="1"> </font></td>

                                                    <td                    width="21"
background="imagens/infdir.gif" style="height: 19px"><font size="1"> </font></td>

                                                </tr>

                                            </table>

                                        </td>

                                    </tr>

                                </table>

                            </form>

                        </body>

</HTML>
```

**Default.aspx.cs**

```
using System;

using System.Data;

using System.Configuration;

using System.Web;

using System.Web.Security;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

using System.Web.UI.HtmlControls;

using System.Xml;

using br.com.cgi2sms.webservice;



public partial class _Default : System.Web.UI.Page
```

```
{


    private string user =
System.Configuration.ConfigurationManager.AppSettings.Get("UserComunika");

    private string pass =
System.Configuration.ConfigurationManager.AppSettings.Get("PassComunika");


    protected void Page_Load(object sender, EventArgs e)

    {

        lblMensagem.Text = "";

        if (!IsPostBack)

        {

            string retorno = CarregarResponsaveisCadastrados();

            if (retorno.Length > 0)

            {

                lblMensagem.Text = retorno;

            }

        }


    }


    private string CarregarResponsaveisCadastrados()

    {

        try

        {

            DataSet ds = new DataSet();

            ds.ReadXml(HttpContext.Current.Server.MapPath("./base.xml"));


            Session["_Responsaveis"] = ds;


            dgResponsaveis.DataSource = ds.Tables[0];
```

```
            dgResponsaveis.DataBind();
            return "";
        }
        catch (Exception ex)
        {
            return ex.Message;
        }
    }
    protected void btnSalvar_Click(object sender, EventArgs e)
    {
        //string SentId = DateTime.Now.ToFileTime().ToString();
        string celNro = tbxNumero.Text;
        string celNome = tbxNome.Text;

        string retorno = InsereUsuarioResponsavel(celNome, celNro);
        if (retorno.Length > 0)
        {
            lblMensagem.Text = retorno;
        }

        this.CarregarResponsaveisCadastrados();

        this.PreparaNovaInsercao();
    }

    private string InsereUsuarioResponsavel(string CelNome, string CelNro)
    {
        //br.com.cgi2sms.webservice.VolaSDKService websend = new VolaSDKService();
        //int hresult = websend.sendMessage(user, pass, false, "Comunica", "556181291913",
DateTime.Now.ToLongDateString() + " - Central de Alarme Disparada. Chame a Policia", "001",
"");
```

```
//string[] id = new string[1];
//id[0] = "001";
//MessageStatusResult result = websend.getMessageStatus("phmaciel", "RQK6S5", id);

try
{
    XmlDocument xmlDoc = new XmlDocument();
    //xmlDoc.Load(HttpContext.Current.Server.MapPath("./base.xml"));
    DataSet ds = new DataSet();
    ds.ReadXml(HttpContext.Current.Server.MapPath("./base.xml"));

    DataTable dt = new DataTable();
    dt = ds.Tables[0];

    DataRow dr = dt.NewRow();
    dr.BeginEdit();

    dr["Nome"] = CelNome;
    dr["Numero"] = CelNro;

    dr.EndEdit();

    ds.Tables[0].Rows.Add(dr);

    ds.WriteXml(HttpContext.Current.Server.MapPath("./base.xml"));

    return "";
}
catch (Exception ex)
```

```
        {
            return ex.Message;
        }


    }


    protected void dgResponsaveis_ItemDataBound(object sender, DataGridItemEventArgs e)
    {
        if (e.Item.ItemType == ListItemType.AlternatingItem || e.Item.ItemType ==
ListItemType.Item || e.Item.ItemType == ListItemType.SelectedItem)
        {
            Button btnSelect = (Button)e.Item.Cells[2].Controls[0];
            btnSelect.ToolTip = "Excluir";
            btnSelect.BorderColor = System.Drawing.Color.Black;
            btnSelect.BackColor = System.Drawing.Color.White;
        }
    }
    protected void btnValidarTelefone_Click(object sender, EventArgs e)
    {
        string SentId = DateTime.Now.ToFileTime().ToString();
        string celNro = tbxNumero.Text;
        //string sad =



        //System.Threading.Thread.Sleep(5000);
        //string[] id = new string[1];
        //id[0] = SentId;
        //MessageStatusResult result = websend.getMessageStatus(user, pass, id);
        //while (result.ToString() != "0")
        //{
        //    result = websend.getMessageStatus(user, pass, id);
```

```
        //}


        string retorno = ValidaNumeroCelular(SentId, user, pass,celNro);
        if (retorno.Length > 0)
        {
            lblMensagem.Text = retorno;
        }
        else
        {
            btnSalvar.Enabled = true;
            btnValidarTelefone.Enabled = false;
            tbxNumero.Enabled = false;
            tbxNome.Enabled = false;


        }



    }
    private string ValidaNumeroCelular(string SentId, string User, string Pass, string CelNro)
    {
        br.com.cgi2sms.webservice.VolaSDKService websend = new VolaSDKService();
        int hresult = websend.sendMessage(User, Pass, true, "Monitor Alarme", CelNro,
DateTime.Now.ToShortDateString() + " - Central de Alarme Disparada. Chame a Policia",
SentId, "");
        switch (hresult)
        {
            case 00:
                return "";
            case 15:
                return "Adquira Novos Créditos na Operadora de SMS";
            case 12:
```

```
            return "Número Inválido";
        default:
            return "Erro Inesperado, entre em contato com o adminitrador do Sistema.";
    }

}

protected void btnLimpar_Click(object sender, EventArgs e)
{
    this.PreparaNovaInsercao();

    this.CarregarResponsaveisCadastrados();

}

private void PreparaNovaInsercao()
{
    tbxNome.Text = "";
    tbxNumero.Text = "";
    lblMensagem.Text = "";
    btnValidarTelefone.Enabled = true;
    btnSalvar.Enabled = false;
    tbxNome.Enabled = true;
    tbxNumero.Enabled = true;
}

protected void dgResponsaveis_ItemCommand(object source, DataGridCommandEventArgs
e)
{
    if (e.CommandName == "Select")
```

```
        {
            DataSet ds = (DataSet)Session["_Responsaveis"];
            ds.Tables[0].Rows.RemoveAt(e.Item.ItemIndex);
```

```
            ds.WriteXml(HttpContext.Current.Server.MapPath("./base.xml"));
        }
        this.CarregarResponsaveisCadastrados();
    }
}
```

### Web.Config

```
<?xml version="1.0"?>
<!--
    Note: As an alternative to hand editing this file you can use the
    web admin tool to configure settings for your application. Use
    the Website->Asp.Net Configuration option in Visual Studio.
    A full list of settings and comments can be found in
    machine.config.comments usually located in
    \Windows\Microsoft.Net\Framework\v2.x\Config
-->
<configuration>
        <appSettings>
                <add key="br.com.cgi2sms.webservice.VolaSDK"
value="http://webservice.cgi2sms.com.br/axis/services/VolaSDK"/>
                <add key="UserComunika" value="*"/>
                <add key="PassComunika" value="*"/>
        </appSettings>
        <connectionStrings/>
        <system.web>
                <!--
```

```
        Set compilation debug="true" to insert debugging
        symbols into the compiled page. Because this
        affects performance, set this value to true only
        during development.
    -->
                <compilation debug="true">


                </compilation>
                <!--
        The <authentication> section enables configuration
        of the security authentication mode used by
        ASP.NET to identify an incoming user.
    -->
                <authentication mode="Windows"/>
                <!--
        The <customErrors> section enables configuration
        of what to do if/when an unhandled error occurs
        during the execution of a request. Specifically,
        it enables developers to configure html error pages
        to be displayed in place of a error stack trace.
   -->
    <customErrors mode="On" defaultRedirect="defaultError.aspx">
        <!--error statusCode="403" redirect="NoAccess.htm" />
        <error statusCode="404" redirect="FileNotFound.htm" /-->
    </customErrors>


      </system.web>
</configuration>
```

# APÊNDICE II

Código do Aplicativo desenvolvido para o Serviço Windows.

**MonitorAlarme.cs**

```
using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Diagnostics;

using System.ServiceProcess;

using System.Text;

using System.Threading;

using System.Timers;


namespace WindowsService1

{

    public partial class Service1 : ServiceBase

    {

        public Service1()

        {

            InitializeComponent();

        }


        protected override void OnStart(string[] args)

        {

            // TODO: Add code here to start your service.

            EscreveLog("Servico Iniciado Com Sucesso!", "Comunica Alarme Service", EventLogEntryType.Information, "Comunica Alarme");


            try
```

```
        {
            Thread threadVerificaSerial = new Thread(IniciaThreadVerificaSerial);
            threadVerificaSerial.Start();
        }
        catch (Exception ex)
        {
            EscreveLog("Erro: " + ex.Message, "Comunica Alarme Service", EventLogEntryType.Error, "Comunica Alarme");

        }
    }

    protected override void OnStop()
    {
        // TODO: Add code here to perform any tear-down necessary to stop your service.
        EscreveLog("Servico Parado Com Sucesso!", "Comunica Alarme Service", EventLogEntryType.Information, "Comunica Alarme");

    }

    private void EscreveLog(string Entry, String AppName , EventLogEntryType evtType, string LogName)
    {
        EventLog evtLog = new EventLog();
        if (!EventLog.SourceExists(AppName))
        {
            EventLog.CreateEventSource(AppName, LogName);
```

```
        }

        evtLog.Source = AppName;


        evtLog.WriteEntry(Entry, evtType);


    }


    private void IniciaThreadVerificaSerial()

    {

        System.Timers.Timer TimerVerificaSerial = new System.Timers.Timer(10.0);


        //EventHandler.CreateDelegate(TimerVerificaSerial.Elapsed,
OnTimedEventVerificaSerial);

        //AddHandler


        TimerVerificaSerial.AutoReset = true;

        TimerVerificaSerial.Enabled = true;

    }


    private void OnTimedEventVerificaSerial()

    {

        EscreveLog("Recebido Sinal da Porta Serial", "Comunica Alarme Service",
EventLogEntryType.Information, "Comunica Alarme");

    }

  }

}
```