

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

**PROTÓTIPO DE UM SISTEMA PARA CONTROLE E**  
**MONITORAÇÃO RESIDENCIAL ATRAVÉS DE**  
**DISPOSITIVOS MÓVEIS UTILIZANDO A PLATAFORMA**  
**.NET**

**ELI VENTURI**

**BLUMENAU**  
**2005**

**2005/1-13**

**ELI VENTURI**

**PROTÓTIPO DE UM SISTEMA PARA CONTROLE E  
MONITORAÇÃO RESIDENCIAL ATRAVÉS DE  
DISPOSITIVOS MÓVEIS UTILIZANDO A PLATAFORMA  
.NET**

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

Prof. Francisco Adell Péricas – Orientador

**BLUMENAU  
2005**

**2005/1-13**

**PROTÓTIPO DE UM SISTEMA PARA CONTROLE E  
MONITORAÇÃO RESIDENCIAL ATRAVÉS DE  
DISPOSITIVOS MÓVEIS UTILIZANDO A PLATAFORMA  
.NET**

Por

**ELI VENTURI**

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

Presidente: \_\_\_\_\_  
Prof. Francisco Adell Péricas – Orientador, FURB

Membro: \_\_\_\_\_  
Prof. Paulo Fernando da Silva

Membro: \_\_\_\_\_  
Prof. Miguel Alexandre Wisintainer

Blumenau, 30 de junho de 2005

Dedico este trabalho a minha esposa Sueli, ao meu filho João Vitor, a minha família e meus amigos que de alguma forma me ajudaram diretamente na realização deste.

## **AGRADECIMENTOS**

À Deus, por ter me acompanhado nesta trajetória e me concedido a graça de terminar este trabalho.

À minha esposa Sueli e ao meu filho João Vitor, duas pessoas que amo muito, mas que não pude dar a atenção que eles mereciam durante a realização do trabalho, mas que sempre estiveram ao meu lado quando precisei.

À minha família, principalmente aos meus pais pelo incentivo a entrar nessa empreitada e seguir em frente até a conclusão do curso.

Ao meu orientador, Francisco Adell Péricas, pela ajuda, pelo conhecimento e pelo incentivo para a conclusão deste trabalho.

A meus amigos e principalmente ao Fábio, pela ajuda e pelas dúvidas tiradas.

No que diz respeito ao empenho, ao compromisso, ao esforço, à dedicação, não existe meio termo. Ou você faz uma coisa bem feita, ou não faz.

Airton Senna da Silva

## RESUMO

Este trabalho apresenta a especificação e implementação de um protótipo de *software* para controle de automação residencial à distância, através de dispositivos móveis. Apresenta um estudo sobre as funções da domótica, sobre a plataforma .NET, *web service* e dispositivos móveis. Na implementação do protótipo foram desenvolvidos: o aplicativo de configurações do ambiente residencial, o aplicativo remoto que é executado no dispositivo móvel, um serviço Windows para monitoramento dos objetos e o canal de comunicação entre o servidor e o dispositivo que é através de *web service*.

Palavras-chave: Automação residencial. Plataforma .NET. .NET *framework*. .NET *compact framework*. Dispositivos móveis. *Web services*.

## **ABSTRACT**

This work presents the specification and implementation of an archetype of software for control of residential automation at a distance, through mobile devices. It presents a study about the functions of the intelligent building, the .NET platform, web service and mobile devices. In the implementation of the archetype they had been developed: the applicatory of configurations of the residential environment, the applicatory remote that is executed in the mobile device, a Windows service for control of objects and the communication channel between the server and the device that is through web service.

**Key-Words:** Residential automation. .NET platform. .NET framework. .NET compact framework. Mobile devices.

## LISTA DE ILUSTRAÇÕES

Figura 1 – Estrutura do .NET <i>Framework</i> .....	28
Figura 2 – Processo de compilação de aplicações .NET .....	29
Figura 3 - Processo de compilação do código fonte para a MSIL.....	30
Figura 4 – <i>Layout</i> do arquivo que acomoda a MSIL e o PE. ....	31
Figura 5 – Processo de compilação da MSIL para código nativo .....	31
Figura 6 – Representação de um envelope SOAP.....	36
Figura 7 – Dispositivos móveis mais recentes.....	39
Figura 8 – Casos de uso do aplicativo de configurações .....	44
Figura 9 – Casos de uso do aplicativo remoto.....	45
Figura 10 – Casos de uso do serviço Windows .....	45
Figura 11 – Diagrama de classes do protótipo .....	46
Figura 12 – Diagrama de atividades .....	48
Figura 13 – Diagrama de seqüência “Alterar estado dos objetos” .....	49
Figura 13 – Diagrama de seqüência “Monitorar Eventos” .....	50
Figura 15 – <i>Visual Studio</i> .NET 2003.....	52
Figura 16 – Tela de cadastro dos objetos .....	53
Figura 17 – Tela do aplicativo remoto.....	54
Figura 18 – Diagrama de fluxo de dados.....	57
Figura 19 – Circuito do simulador de ambiente residencial (atuadores).....	58
Figura 20 – Circuito do simulador de ambiente residencial (sensores).....	58
Figura 21 – Tela de cadastro de objetos .....	60
Figura 22 – Tela de monitoramento do aplicativo remoto .....	61
Figura 23 – Simulador de estado dos objetos .....	62

## LISTA DE QUADROS E TABELAS

Quadro 1 – Código do método “Excluir” da classe objeto.....	54
Quadro 2 – Métodos do <i>web service</i> que são chamados pelo aplicativo remoto. ....	55
Quadro 3 – Código dos métodos GetBit e SetBit da classe Evento. ....	56
Quadro 4 – Código da classe PortaParalela e os métodos EscreverPorta e LerPortar. ....	56
Tabela 1 – Comparação entre os trabalhos correlatos .....	63

## LISTA DE SIGLAS

CLR - *Common Language Runtime*

CLS - *Common Language Specification*

DLL – *Dynamic Link Library*

ECMA - *European Computer Manufacturers Association*

FCL - *Framework Class Library*

HTTP - *HyperText Transfer Protocol*

MSIL - *Microsoft Intermediate Language*

PDA - *Personal Digital Assistants*

RDSI – *Rede Digital de Serviços Integrados*

SGML - *Standard Generalized Markup Language*

SOAP - *Simple Object Access Protocol*

UML – *Unified Modeling Language*

W3C - *World Wide Web Consortium*

WWW - *World Wide Web*

XML - *eXtensible Markup Language*

# SUMÁRIO

<b>1 INTRODUÇÃO.....</b>	<b>13</b>
1.1 OBJETIVOS DO TRABALHO .....	15
1.2 ESTRUTURA DO TRABALHO .....	15
<b>2 DOMÓTICA.....</b>	<b>17</b>
2.1 FUNÇÕES DOMÓTICAS .....	19
2.1.1 Função de Gestão .....	19
2.1.1.1 Gestão da iluminação.....	19
2.1.1.2 Gestão da calefação, ventilação e ar condicionado .....	20
2.1.1.3 Gestão da qualidade do ar.....	20
2.1.1.4 Gestão da funcionalidade dos espaços.....	20
2.1.2 Função de Controle .....	21
2.1.2.1 Controle técnico.....	22
2.1.2.2 Segurança e tele-transmissão .....	23
2.1.2.3 Assistência – saúde .....	23
2.1.3 Função de Comunicação .....	23
2.1.3.1 Comunicação – controle .....	24
2.1.3.2 Comunicação – espaçamento.....	24
2.1.3.3 Comunicação – serviços .....	25
2.2 REDES DOMÓTICAS.....	25
<b>3 ESTRATÉGIA .NET.....</b>	<b>26</b>
3.1 PLATAFORMA .NET .....	27
3.1.1 .NET <i>Framework</i> .....	28
3.1.2 Common Language Runtime .....	29
3.1.3 Microsoft Intermediate Language.....	30
3.1.4 Compilação <i>Just-In-Time</i> .....	31
3.1.5 .NET <i>Compact Framework</i> .....	32
3.2 <i>WEB SERVICES</i> .....	33
3.2.1 XML.....	34
3.2.2 SOAP .....	35
3.3 VANTAGENS DO .NET <i>FRAMEWORK</i> .....	36
<b>4 DISPOSITIVOS MÓVEIS.....</b>	<b>38</b>

<b>5 DESENVOLVIMENTO DO TRABALHO</b> .....	<b>41</b>
5.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO.....	41
5.2 ESPECIFICAÇÃO .....	43
5.2.1 Diagrama de Casos de Usos.....	43
5.2.2 Diagramas de Classes.....	46
5.2.3 Diagrama de Atividades.....	47
5.2.4 Diagramas de Seqüência .....	48
5.2.4.1 Alterar o Estado dos Objetos .....	49
5.2.4.2 Monitorar eventos .....	50
5.3 IMPLEMENTAÇÃO .....	51
5.3.1 Técnicas e ferramentas utilizadas.....	51
5.3.2 Implementação do protótipo .....	52
5.3.3 Operacionalidade da implementação .....	59
5.3.3.1 Cadastro de um objeto .....	59
5.3.3.2 Monitoramento pelo aplicativo remoto .....	60
5.4 RESULTADOS E DISCUSSÃO .....	62
<b>6 CONCLUSÕES</b> .....	<b>64</b>
6.1 EXTENSÕES .....	65
<b>REFERÊNCIAS BIBLIOGRÁFICAS</b> .....	<b>66</b>

## 1 INTRODUÇÃO

Primeiramente veio a automação industrial, ligada ao controle e à supervisão das linhas de produção, e posteriormente veio a automação de edifícios comerciais, direcionadas às áreas patrimonial e institucional. Chega-se agora à automação residencial, um mercado emergente que já é realidade no Brasil, com soluções interessantes e diferenciadas voltadas aos serviços para o usuário.

A automação residencial inicialmente é percebida pelo cliente como um símbolo de *status* e modernidade. No momento seguinte, o conforto e a conveniência passam a ser decisivos. E por fim, tornar-se-á uma necessidade vital e um fator de economia.

Juntando a automação residencial com a necessidade de poder controlar e monitorar à distância, chega-se aos dispositivos móveis.

Os dispositivos móveis oferecem uma conectividade que outros dispositivos não possuem. A tendência é que o desenvolvimento de aplicações para os dispositivos móveis aumente significativamente em poucos anos (DEPINÉ, 2002, p. 1).

Besen (1996) apresentou uma solução para controle de um ambiente residencial através de um microcomputador, controlando variáveis como: iluminação, temperatura e supervisão de janelas e portas. O sistema possui dois módulos: o módulo “Controlador” que foi desenvolvido utilizando o microcontrolador 8031, responsável pelo controle das variáveis; e pelo módulo “Supervisor” que opera em um microcomputador e tem como objetivo interagir com o usuário e com o “Controlador”.

Depiné (2002) e Schaefer (2004) apresentaram soluções que trabalhavam com dispositivos móveis. O primeiro apresentou uma solução para cálculo de planilha de *rally* de regularidade em um celular utilizando a plataforma Java 2 Micro *Edition* (J2ME), onde o

usuário entra com os dados do percurso e a solução retorna o tempo ideal para percorrê-lo. Já o segundo mostrou uma solução para coletar e transmitir informações através de dispositivos móveis e alimentar uma base de dados remota para controle de gastos em viagem, aplicado a uma empresa transportadora. Estes gastos são informados pelos motoristas dos caminhões que estão em viagem e enviados via e-mail. A base alimentada remotamente serve para geração de gráficos, consultas e relatórios das despesas, podendo ser gerados por período e/ou número do celular pela qual as despesas foram informadas.

Krüger (2002) apresentou uma solução de baixo custo para monitoramento de segurança predial utilizando recursos da internet. Desenvolveu um módulo para o monitoramento de portas, janelas e de presença. Esses módulos comunicavam-se com um microcomputador através da porta paralela. As principais funções do sistema instalado no microcomputador são enviar mensagens eletrônicas ao usuário e gerar um arquivo de *log* com as atividades detectadas pelo monitoramento.

Galvin (2004) apresentou a especificação e o desenvolvimento de um protótipo de *software* para integração e troca de dados com um aplicativo cliente/servidor de uma empresa através de dispositivos móveis. A aplicação visa auxiliar o trabalho de relacionamento entre empresa e clientes diretamente no campo, disponibilizando informações importantes para o usuário sobre seus clientes e também fazendo a parte de força de vendas no *front*.

O desenvolvimento desse trabalho prevê a implementação de um protótipo de controle e monitoramento de automação residencial a distância, onde serão utilizados dispositivos móveis que suportem a plataforma .NET. Para o ambiente residencial prevê-se o desenvolvimento de um simulador. No desenvolvimento do software será utilizada a linguagem de programação C#. Segundo Microsoft Corporation (2002), a plataforma .NET conecta informações, sistemas, pessoas e dispositivos, conectando uma grande variedade de tecnologias de uso pessoal e de negócios, de telefones celulares a servidores corporativos,

permitindo o acesso a informações, onde e sempre que forem necessárias.

## 1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é desenvolver um protótipo de sistema para controle e monitoração residencial através de dispositivos móveis, utilizando *web services* para comunicação dos dados via internet.

Os objetivos específicos do trabalho são:

- a) controlar e monitorar os objetos de uma residência à distância, tais como: iluminação, abertura de portas, abertura de janelas, abertura de cortinas e irrigação do jardim;
- b) desenvolver o canal de comunicação entre o dispositivo móvel e o servidor;
- c) centralizar no servidor os controles dos objetos propostos;
- d) desenvolver um simulador para o ambiente residencial.

## 1.2 ESTRUTURA DO TRABALHO

A apresentação deste trabalho está disposta em 6 capítulos divididos nos temas domótica, estratégia .NET, dispositivos móveis, desenvolvimento do trabalho e a conclusão final.

O primeiro capítulo apresenta a introdução e os trabalhos correlatos.

O segundo capítulo apresenta uma visão geral sobre o que é a domótica e o que cada uma das suas funções se propõem a desempenhar.

A estratégia .NET está descrita no terceiro capítulo, aonde são apresentados os fundamentos e pontos importantes que levam a escolha da plataforma .NET para o

desenvolvimento da solução apresentada.

No quarto capítulo fala-se sobre os dispositivos móveis e as vantagens de utilizá-los para obter informações aonde quer que a pessoa esteja.

O desenvolvimento do trabalho, a especificação, as ferramentas utilizadas e a metodologia de implementação são apresentados no quinto capítulo.

E por último são apresentadas as conclusões a qual chegou-se com o desenvolvimento do trabalho, com propostas de extensões do mesmo.

## 2 DOMÓTICA

O termo domótica, vem da fusão da palavra latina domus (casa) e da palavra robótica. Foi adotado na Europa para designar o campo de aplicação tecnológica que visa a integração do espaço arquitetônico, da informática e das telecomunicações (ANGEL, 1993, p. 13). Já nos Estados Unidos e no Japão adotou-se a expressão "*intelligent building*".

A domótica pode ser definida como um conjunto de tecnologias que ajudam na gestão e execução de tarefas domésticas cotidianas. A sua utilização tem por objetivo proporcionar um maior nível de conforto, comodidade e segurança além de um menor e mais racional consumo de energia. (CASA PLUS, 2004).

Conforme Breternitz (2001), as primeiras aplicações domóticas utilizavam sensores e atuadores (dispositivos que alteravam os parâmetros em função de informações captadas pelos sensores), que numa arquitetura centralizada eram ligados a um controlador onde estava a inteligência necessária. Quase sempre eram sistemas proprietários, pouco flexíveis, e principalmente caros.

Mas atualmente, graças à internet, observa-se o surgimento de mais fabricantes e provedores de serviços, desenvolvendo produtos e serviços que conjugam o melhor da internet, com tecnologia padrão de redes, e com isso passa-se a acreditar que a domótica atingirá um novo patamar de utilização e popularidade, já havendo quem diga que em função dessas novidades pretendem adotar a expressão "Teledomótica", por conta da sinergia que se está produzindo pelo uso conjunto da internet, da telefonia móvel e da domótica propriamente dita (BRETERNITZ, 2001).

Segundo Breternitz (2001), o acesso à internet por banda larga está tendo um papel importante para que o mercado de Teledomótica cresça. Além de garantir a recepção de comandos dados pelos usuários quando estão fora de casa, tornará viáveis operações como alarmes médicos e cuidados à pessoas incapacitadas, entre outros.

Segundo Angel (1993, p. 43), a domótica é um novo domínio de aplicação tecnológica,

tendo como objetivo básico melhorar a qualidade de vida, reduzindo o trabalho doméstico, aumentando o bem estar e a segurança de seus habitantes e visando também uma utilização racional e planejada dos diversos meios de consumo. A domótica procura uma melhor integração através da automatização nas áreas de segurança, de comunicação e de controle e gestão de fluídos. Esta proposta integradora busca dar resposta às necessidades do homem, que podem ser agrupadas em três grupos:

a) as necessidades de segurança: estão relacionadas com:

- a qualidade do ar,
- a prevenção de acidentes físicos e materiais,
- a assistência à saúde,
- a segurança contra intrusos;

b) as necessidades de conforto ambiental: implicam na criação de um meio ambiente agradável:

- conforto térmico,
- conforto acústico,
- conforto visual,
- conforto olfativo,
- conforto espacial;

c) as necessidades de conforto de atividades: vêm facilitar os hábitos cotidianos:

- para dormir,
- para alimentar-se,
- para cuidar-se,
- para manutenção (dos locais e dos materiais),
- para comunicar-se,
- para divertir-se,

- para trabalhar.

## 2.1 FUNÇÕES DOMÓTICAS

Segundo Angel (1993, p. 46), as funções domóticas nos permitem satisfazer a um número considerável das necessidades anteriormente discutidas. Define-se então, três grandes classes de funções segundo o tipo de "serviço" a que elas se dirigem, as quais são divididas em sub-funções elementares, que podem ser mais facilmente analisadas.

### 2.1.1 Função de Gestão

Essa função tem áreas comuns com a função de controle. A função de gestão tem por objetivo automatizar um certo número de ações sistemáticas. As automatizações se realizam segundo uma programação, um controle dos consumos e uma manutenção. As ações sistemáticas dessa função se relacionam principalmente com o conforto (ANGEL, 1993, p. 48).

#### 2.1.1.1 Gestão da iluminação

Segundo Angel (1993, p. 48), a gestão de iluminação fornece um dos primeiros elementos voltados ao conforto, adequando o ambiente segundo a necessidade de cada usuário de acordo com a idade, capacidades físico-motoras, uso dos espaços ao longo do dia, repercussão sobre a ocupação dos espaços. A otimização do uso e economia de eletricidade é outro aspecto importante desta sub-função, sem deixar de lado o conforto dos usuários. Como serviços auxiliares pode-se citar a temporização, a variação de intensidade, o acendimento e

apagamento automático programado, comandado à distância ou por comando de voz.

#### 2.1.1.2 Gestão da calefação, ventilação e ar condicionado

A gestão de calefação, ventilação e ar condicionado visa permitir ao usuário medir e controlar sua calefação, as cargas elétricas e seu próprio conforto. Teve um grande impulso com a evolução dos sensores e com a necessidade de racionalização de energia. Entre os confortos gerados por esta gestão, pode-se citar a otimização em relação ao meio externo; a auto-adaptação em relação aos equipamentos; a gestão, ambiente por ambiente. Sob o aspecto dos requisitos possíveis, podemos citar o controle a distância, passagem automática do regime conforto para o regime redução no caso da ausência de indivíduos, dentre outras (ANGEL, 1993, p. 51-52).

#### 2.1.1.3 Gestão da qualidade do ar

Através da gestão da qualidade do ar, a domótica pode controlar totalmente o ar do ambiente, não somente da temperatura e umidade, mas também verificando a existência de gases tóxicos como gás de cozinha (ANGEL, 1993, p. 53).

#### 2.1.1.4 Gestão da funcionalidade dos espaços

A gestão da funcionalidade dos espaços tem como objetivo flexibilizar o ambiente quando houver modificações nos grupos familiares, novos modos de vida e adaptação a novas necessidades. Tal flexibilidade visa permitir futuras atualizações nos sistemas atuais, instalação de novos sistemas, ou interconexão dos mesmos, e como isso permitir a evolução

das necessidades dos usuários (ANGEL, 1993, p. 54).

Segundo Angel (1993, p. 54-55), a flexibilidade dos espaços, as instalações e os equipamentos são de vital importância para a domótica. Para tanto, o projeto deve garantir as adaptações a novos equipamentos no futuro, levando em conta a estrutura da residência, pisos e tetos flexíveis, etc.

As exigências citadas vêm do fato de que uma edificação é um conjunto flexível que integra distintas tecnologias, abrangendo sistemas de gestão, de controle energético, de vigilância e manutenção, de segurança, de comunicação interna e externa, como também sistemas mecânicos, elétricos e eletrônicos que devem conectar-se entre si (ANGEL, 1996, p. 54-55).

Os requisitos básicos que a gestão de funcionalidade de espaços deve atender são:

- a) aumentar a produtividade e a segurança;
- b) empregar todos os recursos de forma mais eficiente possível;
- c) ter flexibilidade para novas e eventuais necessidades de uso.

### 2.1.2 Função de Controle

Conforme Angel (1993, p. 55), a função de controle dá ao usuário, por um lado, informações sobre o estado de funcionamento dos equipamentos e das instalações que os integram; e por outro lado, criam um registro dos diversos parâmetros e eventualmente, induzem a comandos corretivos. Para tanto ele conta com controles instantâneos e memorizados. Essa função tem por objetivo atuar sobre os dispositivos de regulação das instalações, com a finalidade de que as tarefas programadas sejam respeitadas. As funções de controle associadas com um algoritmo ou com uma unidade de tratamento da informação conduzirão às funções de comando.

### 2.1.2.1 Controle técnico

Segundo Angel (1993, p. 56), o controle técnico visa auxiliar o usuário a fazer o uso dos equipamentos, dispositivos e instalações mais confiáveis e também prover autodiagnóstico dos mesmos, o que permite, entre outros temas, programar os gastos.

Esta sub-função é responsável por controlar os diferentes equipamentos e eletrodomésticos, as diferentes redes de alimentação, os diferentes fluídos utilizados na casa, a presença de intrusos e os parâmetros fundamentais para verificação do estado de saúde dos membros da família, etc.

Os valores apresentados devem ser extremamente confiáveis, para que possam ser utilizados como assistências ao usuário, como também ser ergonômica para atender todas as idades dos usuários (ANGEL, 1993, p. 56).

Como exemplos de controle técnico pode-se citar:

- a) recepção de mensagens de mau funcionamento de equipamentos e instalações em um monitor de TV ou outro indicador de controle;
- b) centralização do estado dos sistemas em pequenos ou grandes painéis que indicam portas e persianas abertas, luzes acessas, etc;
- c) desligamento seletivo de cargas para evitar sobrecargas nos sistemas;
- d) informações de consumo de água, gás e eletricidade e os custos dos mesmos;
- e) comandos únicos que atuam sobre diversos equipamentos. Por exemplo uma saída de férias: cortar o fornecimento de água e gás, desligar as luzes, ativar o sistema de alarme, fechar as cortinas, etc.

### 2.1.2.2 Segurança e tele-transmissão

Segundo Angel (1993, p. 58), a domótica tem como preocupação prioritária a segurança, pois está associada aos bens materiais, incluindo a prevenção de intrusos, incêndios e acidentes domésticos. O sistema tem que ser confiável, evitando com isso falsos alarmes e ser de fácil manuseio para todos os membros da família.

Pode-se citar de responsabilidade desta sub-função:

- a) controle de acesso;
- b) detecção de incêndio,
- c) detecção de fuga de gás e água;
- d) detecção de intrusos;
- e) tele-vigilância;
- f) tele-assistência.

### 2.1.2.3 Assistência – saúde

Esta sub-função permite ao usuário a conexão através de um computador pessoal com centros de assistências médicas, que asseguram o controle e acompanhamento da evolução de casos graves de doença ou por motivo de acidentes (ANGEL, 1993, p. 60).

### 2.1.3 Função de Comunicação

Uma característica possibilitada pela comunicação é a interatividade. Através desta é permitido o telecomando e a programação para que os sistemas envolvidos obedeçam a uma certa padronização (ANGEL, 1993, p. 62).

Conforme Angel (1993, p. 62), a função de comunicação pode ser classificada por dois ângulos: a comunicação sem significados, por exemplo sinais de áudio e vídeo, onde se busca a maior fidelidade possível; e a comunicação onde visa-se o intercâmbio de informações de controle de ambiente.

#### 2.1.3.1 Comunicação – controle

Com a utilização de Rede Digital de Serviços Integrados (RDSI), que utiliza uma técnica comum para serviços de voz, texto, dados e imagens, a função de controle pode realizar sua função principal, interligar os vários dispositivos entre si e com o operador do sistema. Além dos serviços já citados, esta rede disponibiliza também a troca de comando entre vários equipamentos e o operador (ANGEL, 1993, p. 64).

#### 2.1.3.2 Comunicação – espaçamento

Segundo Angel (1993, p. 65-66), dentre os serviços oferecidos por esta sub-função pode-se citar a possibilidade do relacionamento da família com o ambiente externo e os serviços coletivos dos imóveis. Para realização destes serviços coletivos tem-se a necessidade de interconectar os aparelhos de áudio e vídeo-comunicação da casa, fazendo-os comunicarem entre si, permitindo assim um melhor serviço de som e imagem, aumentando o conforto.

### 2.1.3.3 Comunicação – serviços

Conforme Angel (1993, p. 67), esta sub-função visa a conexão da rede interna de áudio e vídeo com uma rede exterior, podendo esta ser pública ou privada. Nos objetivos deste serviço estão incluídos: o tele-trabalho, a tele-educação, tele-mantimento, a tele-vigilância-assistência.

## 2.2 REDES DOMÓTICAS

Segundo Angel (1993, p. 112), a rede domótica é o elemento principal de todo o sistema domótico. A rede domótica, ou em outros termos, o cabeamento é o que permite realizar uma comunicação entre os diferentes aparatos conectados à rede e é indubitavelmente o instrumento essencial em que se baseia a domótica. As redes destinadas aos edifícios inteligentes se baseiam em aplicações, onde uma rede separada e independente é utilizada para cada função. É assim que existem redes destinadas à segurança, à detecção de incêndios, ao controle de acessos, à climatização, à informática, etc.

As redes domóticas são, em termos gerais, redes polivalentes que permitem realizar diferentes funções a fim de simplificar a complexidade da instalação da rede. A mesma rede domótica assegura, por exemplo, as funções de segurança, conforto e gestão técnica. A rede pode estar constituída de um ou vários suportes de comunicação de acordo com as funções que esse sistema domótico realiza (ANGEL, 1993, p. 112).

### 3 ESTRATÉGIA .NET

Em meados de 2000, a Microsoft anunciava a iniciativa .NET, dando uma nova visão para abranger a internet e a *World Wide Web* (WWW) no desenvolvimento, engenharia e uso de *software*. A independência de linguagem de programação ou de uma plataforma específica é um aspecto muito importante na estratégia .NET. Os aplicativos .NET podem ser criados com qualquer linguagem de programação compatível com essa tecnologia, possibilitando aos programadores escolher a linguagem que mais for conveniente, permitindo que um projeto de *software* possa ser desenvolvido em várias linguagens diferentes (DEITEL et al., 2003, p. 13).

A estratégia .NET amplia em muito a idéia de reutilização de software, com o conceito de *web services*, que são serviços que podem ser acessados através da internet. Segundo Deitel et al. (2003, p. 13), as empresas podem economizar tempo e energia no desenvolvimento de seus produtos, comprando *web services* de terceiros, e com isso focar os esforços em seus produtos. Os programadores podem criar aplicativos que usam *web services* para banco de dados, segurança, autenticação e tradução de idiomas, sem ter o conhecimento sobre os detalhes internos desses componentes (DEITEL et al, 2003, p.14).

Segundo Deitel et al (2003, p.14), o *Simple Object Access Protocol* (SOAP), que é um protocolo e a *eXtensible Markup Language* (XML), que dá significado aos dados, são as chaves de comunicação dos *web services*.

Outro conceito fundamental da estratégia .NET é o acesso a dados universal, ou seja, os dados podem residir em um repositório central e com isso qualquer dispositivo conectado a internet pode ter acesso a esses dados, os quais seriam formatados adequadamente para uso ou exibição no dispositivo que o solicitou. Assim um documento poder ser visto e editado em computador de mesa, em um *Personal Digital Assistants* (PDA), em celular ou outro dispositivo.

Segundo Deitel et al. (2003, p. 13), outra iniciativa da Microsoft compreende a tecnologia ASP.NET que permite aos programadores criar aplicativos para a *Web*.

### 3.1 PLATAFORMA .NET

A plataforma .NET pode ser definida basicamente como um modelo de desenvolvimento, criado pela Microsoft, que visa a implementação de *software* independente de linguagem, plataforma e dispositivo. Um dos principais objetivos desse modelo é permitir a integração entre aplicações através da troca de informações pela internet (BURÉGIO, 2003, p. 9).

A plataforma .NET é considerada o coração da estratégia .NET. Esta estrutura gerencia e executa aplicativos e *web services*, contém uma biblioteca de classes denominada *Framework Class Library* (FCL), garante a segurança e fornece muitos outros recursos de programação. A FCL contém uma variedade de componentes reutilizáveis, evitando o problema de criar novos componentes por parte dos programadores (DEITEL et al, 2003, p. 14, 16).

As especificações da plataforma .NET são encontradas na *Common Language Specification* (CLS), que contém informações sobre o armazenamento de tipos de dados, objetos, etc.. A CLS foi submetida para padronização à *European Computer Manufacturers Association* (ECMA), tornando com isso mais fácil a portabilidade para outros ambientes (DEITEL et al, 2003, p. 14).

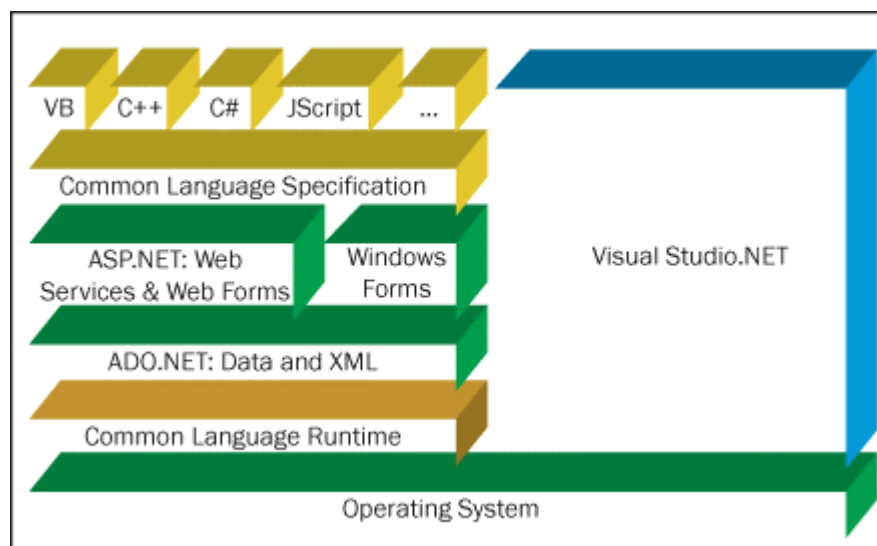
Segundo Deitel et al (2003, p.15), a *Common Language Runtime* (CLR) é uma parte central da plataforma .NET, pois é ela que é responsável pela execução dos programas. Os programas são compilados em código de máquina em duas etapas. A primeira etapa é compilar o programa em *Microsoft Intermediate Language* (MSIL), definindo as instruções para o CLR. Já a segunda etapa compila o MSIL para o código de máquina, criando um

aplicativo único (DEITEL et al, 2003, p. 15).

### 3.1.1 .NET Framework

O .NET Framework é a parte principal da plataforma .NET, sendo responsável pelo gerenciamento dos códigos executados dentro dela. Segundo Barroso (2004), ela é composta por duas partes principais: a CLR, que é responsável pela independência de linguagem de programação e a FCL, que fornece os principais recursos para o desenvolvimento de aplicação. Segundo Microsoft Corporation (2001, p. 108), o .NET Framework é usado para criação, instalação e execução de *web services* e outros aplicativos.

A figura 1 apresenta uma visão da estrutura do .NET Framework.



Fonte: Microsoft apud Finkelstein (2003)

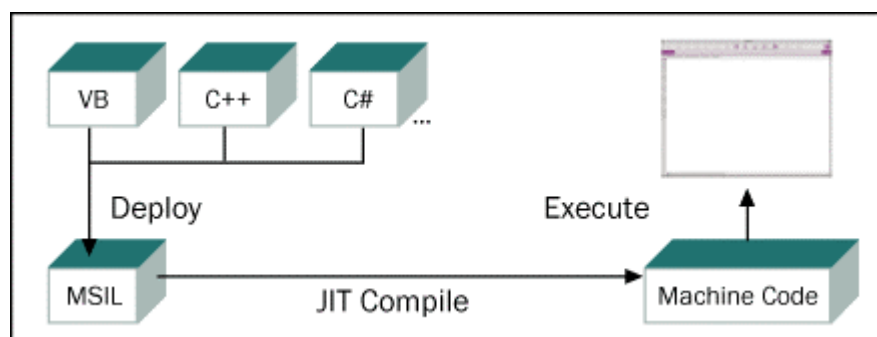
Figura 1 – Estrutura do .NET Framework

Segundo Burégio (2003, p. 14), todas as linguagens compatíveis com .NET possuem praticamente o mesmo poder e fazem uso dos mesmos componentes disponibilizados pelo Framework. A escolha da linguagem de programação poderá ser determinada simplesmente pelo grau de conhecimento ou familiaridade do programador.

### 3.1.2 Common Language Runtime

Segundo Lippman (2003, p. 281), a CLR fornece um ambiente de execução que gerencia a execução do código e fornece serviços como tratamento de erros, segurança, coleta de lixo e controle de versão. Estes serviços estão disponíveis em qualquer linguagem criada para a CLR. Isto quer dizer que a CLR pode servir a uma variedade de linguagens, e pode oferecer um conjunto comum de ferramentas para estas linguagens.

A CLR não “conhece” qual linguagem foi utilizada na escrita do código fonte, pois todas as linguagens compatíveis com a plataforma .NET são compiladas para um código intermediário denominado Microsoft *Intermediate Language* (MSIL) (BURÉGIO, 2003, p. 14). A figura 2 apresenta uma visão do processo de compilação de aplicações .NET.



Fonte: Microsoft apud Finkelstein (2003)

Figura 2 – Processo de compilação de aplicações .NET

Segundo Fundão (2002), a CLR é descrita como o "motor de execução" do .NET. Ela fornece o ambiente dentro do qual os programas executam.

As suas maiores características são:

- a) compilação da MSIL para código nativo da plataforma onde está sendo executado;
- b) gerenciamento de memória, incluindo *garbage collection*;
- c) verificação e reforço de restrições de segurança no código em execução;
- d) carregamento e execução de programas, com controle de versão e outras características.

O código de programação escrito para ser executado exclusivamente sob o controle do CLR é chamado código gerenciado e os objetos que são gerenciados pela CLR são chamados de dados gerenciados (ROMAN et al, 2002, p. 84).

### 3.1.3 Microsoft Intermediate Language

O MSIL é o conjunto de instruções independentes de processador e de sistema operacional em que os programas *.NET Framework* são compilados. Ela contém instruções de carga, armazenagem, inicialização e métodos de chamadas a objetos. É ela que possibilita a verdadeira integração entre várias linguagens. Quando o código fonte de uma linguagem compatível com a plataforma *.NET* é compilado, o compilador o converte para uma linguagem MSIL que é um conjunto de instruções em um tipo de “linguagem de máquina”, mas independentemente de sistema operacional (BRAGAGNOLO, 2004). A figura 3 apresenta o processo de compilação do código fonte para a MSIL.

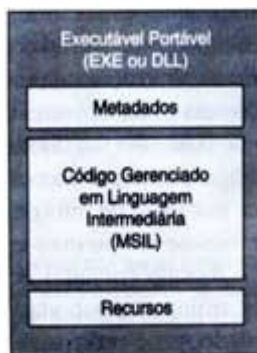


Fonte: Bragagnolo (2004)

Figura 3 - Processo de compilação do código fonte para a MSIL

Segundo Bragagnolo (2004), quando um compilador gera uma MSIL, ele também gera metadados que são informações que descrevem os tipos de dados e suas dependências, objetos e seus membros, referências e outros dados do código que são usados em tempo de execução.

A MSIL e os metadados ficam dentro de um executável portátil (PE), conforme demonstra a figura 4.



Fonte: Bragagnolo (2004)

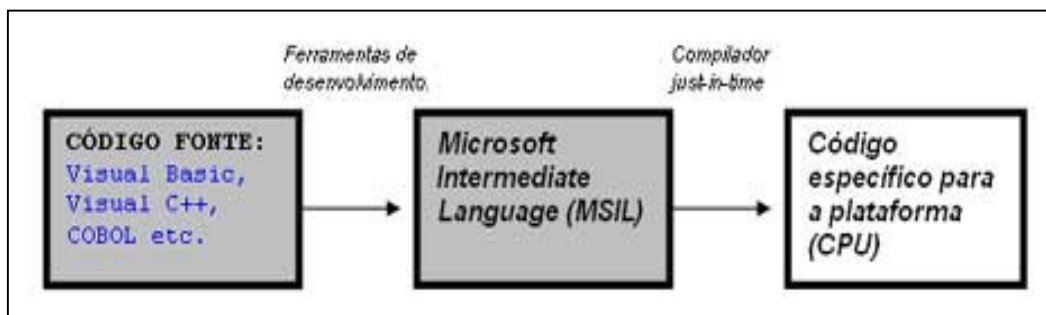
Figura 4 – *Layout* do arquivo que acomoda a MSIL e o PE.

Segundo Roman et al (2002, p. 85), os metadados de um componente de *software* compilado tornam o componente autodescritivo.

### 3.1.4 Compilação *Just-In-Time*

Segundo Roman et al (2002, p. 85), quando o código é executado pela primeira vez, a MSIL é compilada em código nativo (código específico para o processador e sistema operacional no qual o mesmo está sendo executado) pelo compilador JIT.

A figura 5 mostra o processo de compilação da MSIL para código nativo.



Fonte: Bragagnolo (2004)

Figura 5 – Processo de compilação da MSIL para código nativo

Segundo Bragagnolo (2004), como o código nativo é gerado em tempo de execução, uma certa independência de plataforma é proporcionada pelo .NET, desde que cada plataforma tenha seu próprio compilador JIT.

### 3.1.5 .NET Compact Framework

Segundo Haddad (2004), o *.NET Compact Framework* é um sub-conjunto do *.NET Framework*, desenvolvido especialmente para implementação de aplicações cliente em dispositivos móveis. O *.NET Compact Framework* trás para o mundo dos dispositivos móveis o código gerenciado e *web services*, habilitando a execução com segurança em dispositivos como *Personal Digital Assistants* (PDAs), telefones celulares e outros, obtendo com isso uma maior confiabilidade no código, podendo reduzir drasticamente os erros de *software*.

O *.NET Compact Framework*, possui uma nova implementação da CLR que foi modificada para suportar, de maneira mais eficiente, a execução de aplicações no contexto de pequenos dispositivos (BURÉGIO, 2003, p. 24).

A FCL do *.NET Compact Framework* possui menos da metade das classes da versão completa do *.NET Framework*, mas nem por isso as funcionalidades do *.NET Compact Framework* são limitadas se comparadas com a versão completa. A FCL do *.NET Compact Framework* possui as classes básicas da versão completa, que em termos práticos, é o suficiente para o desenvolvimento da maioria das aplicações (BURÉGIO, 2003, p. 25).

Com o *.NET Compact Framework* os programadores podem facilmente reutilizar grande parte do conhecimento e conceitos que foram adquiridos no desenvolvimento de aplicações *desktop* (BURÉGIO, 2003, p. 24).

Com o crescimento do mercado de dispositivos móveis e a grande demanda dos últimos anos, o desenvolvimento de aplicações para estes dispositivos passam a ser só mais um processo de criação de *software*, aumentando a eficiência no desenvolvimento e a um custo de produção baixo (GALVIN, 2004, p. 40).

Segundo Galvin (2004, p. 41), outro ponto que pode ser destacado é a alta performance do *.NET Compact Framework*, pois foi projetada para trabalhar com recursos limitados, normalmente encontrados em dispositivos móveis. A eficiência se deve ao aproveitamento dos recursos sem desperdiçá-los.

O *.NET Compact Framework* visa os dispositivos móveis, tais como os celulares, *smart phones*, PDA's, *Pocket PC's* e outros aparelhos, como eletrodomésticos. Atualmente só permite desenvolver aplicações para aparelhos que executam o sistema operacional *Windows CE* (GALVIN, 2004, p. 42).

### 3.2 WEB SERVICES

Conforme Santos (2003, p. 15), a plataforma *.NET* oferece uma vasta lista de funcionalidades para realização de tarefas diversas, porém, centradas na orientação a objetos e em grande integração com a internet. Dentre os recursos oferecidos, estão os *web services*, que também são oferecidos por outras plataformas, como Java, por exemplo.

Os *web services* são componentes que disponibilizam métodos para utilização remota, através da internet, por outras aplicações, permitindo que aplicações de diferentes plataformas troquem informações (BURÉGIO, 2003, p. 54; SANTOS, 2003, p. 13).

Os *web services* possibilitam que aplicativos em qualquer tipo de computadores ou dispositivos móveis se conectem e interoperem entre si, seja dentro de uma mesma empresa, com parceiros externos ou com clientes. Através de um meio de conexão padronizado, a integração se torna mais rápida, fácil e flexível e pode ser executada entre diferentes sistemas operacionais e linguagens de programação.

Segundo Boaro (2004), no coração da visão dos *web services* está o conceito de operabilidade conjunta, ou seja, a capacidade de sistemas diferentes se comunicarem e compartilhar dados, sem estarem ligados entre si. Um *web service* é uma aplicação lógica, programável, acessível, que usa os protocolos padrão da internet, para que se torne possível a comunicação transparente de máquina-para-máquina e aplicação-para-aplicação.

As chamadas aos métodos utilizam o protocolo *Simple Object Access Protocol* (SOAP), um protocolo que usa *eXtensible Markup Language* (XML) para fazer chamadas a procedimentos remotos por meio de *HyperText Transfer Protocol* (HTTP), permitindo a comunicação entre diferentes plataformas. O HTTP foi escolhido para transmitir as mensagens SOAP, por se tratar de um protocolo padrão para envio de informações pela internet. O uso de XML e HTTP permite que diferentes sistemas operacionais enviem e recebam mensagens SOAP. Outra vantagem do HTTP é o fato de poder trafegar nas redes que possuem *firewalls*, sem que seja barrado (DEITEL et al., 2003, p. 897-898).

### 3.2.1 XML

Conforme Deitel et al (2003, p. 724), a XML foi desenvolvida em 1996, pelo XML *Working Group* do *World Wide Web Consortium* (W3C). A XML é uma tecnologia aberta, portátil e amplamente suportada, para descrever dados.

A XML combina o poder e a capacidade de extensão de sua linguagem mãe (HTML) e o *Standard Generalized Markup Language* (SGML), com a simplicidade exigida pela comunidade da *web* (DEITEL et al (2003, p. 12).

Segundo *World Wide Web Consortium* (2003), XML é um padrão para representação de dados. Possui um formato simples e muito útil para o intercâmbio de dados, o que é conseguido através da característica de marcação da linguagem.

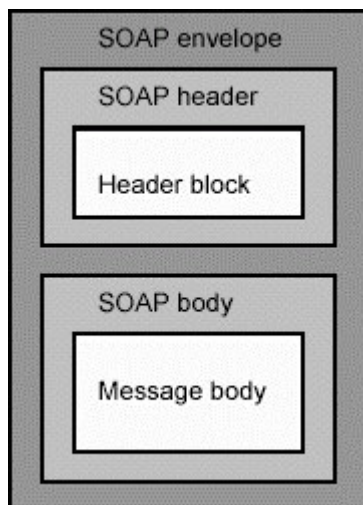
A possibilidade de definição de novas linguagens de marcação é uma característica marcante na XML. Isto é possível pelo fato de as *tags* de marcação poderem ser definidas pelo usuário. Portanto pode-se dizer que dados semi-estruturados são bem representados pela XML (SANTOS, 2003, p.17-18).

### 3.2.2 SOAP

Segundo Burégio (2003, p. 55), o *Simple Object Access Protocol* é um mecanismo simples e leve para promover a troca de informações estruturadas (XML) em um ambiente descentralizado e distribuído.

Na transmissão de mensagens SOAP, a mensagem original (documento XML) é empacotada por um envelope SOAP, que é constituído de duas partes: o cabeçalho (*header*) e o corpo (*body*).

A figura 6 apresenta a estrutura do envelope SOAP.



Fonte: MSDN apud Santos (2003)

Figura 6 – Representação de um envelope SOAP.

Segundo Burégio (2003, p. 56), o *header* de uma mensagem SOAP é opcional e geralmente contém informações referente a processos transacionais e autenticações e o *body* é um elemento obrigatório e contém informações referentes aos dados da requisição do cliente ou os dados relativos a resposta retornada pelo *web service*.

### 3.3 VANTAGENS DO .NET FRAMEWORK

Os principais benefícios ao utilizar o .NET *Framework* são:

- a) independência de linguagem sendo que a plataforma .NET suporta várias linguagens de programação que podem ser utilizadas no desenvolvimento de aplicações;
- b) sistema de tipos único – o .NET *Framework* possui um sistema de tipos único que pode ser utilizado por todas as linguagens compatíveis com .NET. Todos os elementos do sistema de tipos são tratados como objetos o que permite a sua utilização em qualquer linguagem que suporte o modelo;
- c) modelo de aplicação unificado – as funcionalidades do .NET *Framework* estão

disponíveis para qualquer linguagem compatível com o .NET. Essa característica permite, por exemplo, que um mesmo trecho de código possa ser utilizado por aplicações *desktop*, *web* ou até *web services*;

- d) suporte aos principais padrões *web* – o desenvolvimento de aplicações *web* com .NET é facilitado pelo grande suporte que o mesmo possui aos padrões tecnológicos utilizados atualmente na internet. Como exemplos pode-se citar HTML, XML, SOAP, *web services* entre outros;
- e) interação entre linguagens – os objetos criados em linguagens diferentes podem comunicar-se entre si e até mesmo usar uma classe criada em outra linguagem. Como exemplo pode-se definir uma classe no Visual Basic .NET e usar no C#, derivar uma classe original ou chamar um método na classe criada.

## 4 DISPOSITIVOS MÓVEIS

Segundo Pekus (2004a), os dispositivos móveis frequentemente utilizados em processos de computação móvel tornaram-se muito mais do que agendas eletrônicas ou assistentes pessoais e mesmos celulares: tornaram-se pequenos computadores que facilmente leva-se a qualquer lugar. Para aqueles que consomem grande parte do seu tempo trabalhando remotamente, estes equipamentos são versáteis, dedicados, multifuncionais e de uso genérico. Eles são ótimos geradores de informações, podendo ser utilizados desde a automação de processos até a coleta de informações estratégicas.

A tecnologia *wireless* vem ampliar ainda mais a mobilidade já fornecida pelos dispositivos móveis, possibilitando ao usuário coletar informações a qualquer momento e em qualquer lugar. Sem dúvida alguma caminha-se para um mundo sem fronteiras e onde os dispositivos estarão cada vez mais presentes (PEKUS, 2004b).

A abordagem de dispositivos móveis, nos remete a equipamentos que estão presentes no cotidiano das pessoas e tornando-se formas eficazes na busca de comunicação segura e de preferência *on-line*. Eles permitirão ao usuário deslocar-se junto com seu ambiente computacional e ter um acesso constante a fontes de informações (DALFOVO et al, 2003).

Um aspecto que auxilia no crescimento do setor de dispositivos móveis é que as pessoas estão cada vez mais dependentes de informações que estão disponíveis na internet. No contexto da computação moderna, elas estão mudando a maneira pela qual acessam a rede mundial, ou seja, não somente de seus computadores pessoais (DORNAN, 2001, p. 2).

A Microsoft lançou seu primeiro sistema operacional para dispositivos móveis em 1996, o *Windows CE (Compact Edition)*. Porém, as primeiras versões não tiveram muito

sucesso, pois os dispositivos existentes na época não suportavam adequadamente a interface gráfica proposto pelo sistema. Em 2000, com o lançamento do *Pocket PC 2000*, foi lançado juntamente o Windows CE versão 3.0, com a interface gráfica mais bem elaborada e preparada para trabalhar com dispositivos móveis (BURÉGIO, 2003, p. 19).

O dispositivo *Pocket PC* que se tornou rapidamente o maior concorrente do Palm foi o Compaq iPad, e foi através dele que a Microsoft firmou presença no mundo dos dispositivos móveis (GALVIN, 2004, p. 43). A figura 7 apresenta os dispositivos móveis mais recentes.



Fonte: Burégio (2003, p. 20)

Figura 7 – Dispositivos móveis mais recentes

Nos últimos tempos, dispositivos móveis desde *notebooks* a *Pocket PCs*, foram disponibilizados para auxiliar no desenvolvimento de trabalhos que exigem o deslocamento, como representantes de vendas, executivos em viagem entre outros. Esses dispositivos não apenas ajudam no gerenciamento de compromissos e contatos como também representam uma ferramenta para substituição dos processos feitos em papel por aplicativos baseados em formulários (GALVIN, 2004, p. 14-15).

Segundo Galvin (2004, p. 15), os dispositivos móveis estão chegando num cenário que antes era dominado por *desktops* e *notebooks*. Isso em função do surgimento de novos aplicativos exclusivos para esse ambiente. Junto a isso, os usuários estão usufruindo as

facilidades do mundo interligado por redes sem fio e com isso obtendo informações a qualquer hora e em qualquer lugar, bastando para isso estar conectado à internet.

Segundo Pekus (2004a), os dispositivos móveis representam vantagens em relação a outros computadores, como:

- a) dimensões: além de mais leves e simples de manusear, podem ser transportados em qualquer espaço;
- b) consumo de energia: por serem dispositivos mais compactos e econômicos, o consumo de energia e tempo de recarga são menores e a autonomia em campo é maior;
- c) ganho de tempo e eficiência: o tempo de carga de aplicações embutidas nestes dispositivos é inferior quando comparados a outros computadores;
- d) custos operacionais e expansão programada: por serem mais compactos e voltados para atividades específicas, estes dispositivos não contam com vários circuitos e periféricos internos, como por exemplo disco rígido e discos flexíveis, diminuindo de forma evidente os custos com manutenção ou programas desnecessários.

## 5 DESENVOLVIMENTO DO TRABALHO

O protótipo proposto é formado por dois aplicativos, um serviço do Windows e um *web service*. No *desktop* será executado o aplicativo de configurações, onde serão cadastrados os objetos que serão controlados, usuários que controlarão os objetos e as localizações em que os objetos estão distribuídos, o serviço que é responsável por alterar o estado dos objetos que foram acessados remotamente pelo aplicativo remoto e o *web service* que tem a finalidade de fazer a comunicação entre o aplicativo remoto e a banco de dados. O dispositivo móvel executará o aplicativo remoto, por onde o usuário controlará os objetos cadastrados.

### 5.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Devido ao protótipo ser dividido em dois aplicativos, um serviço Windows e um *web service*, os requisitos funcionais e os requisitos não funcionais serão apresentados separados pelas partes envolvidas.

Aplicativo responsável pela configuração:

- a) RF1 – manter objetos: permitir a inclusão, alteração e exclusão de objetos que serão controlados pelo usuário. Deverá conter dados como: descrição, localização, tipo de objeto, atuador, sensor e estado atual;
- b) RF2 – manter usuários: permitir a inclusão, alteração e exclusão de usuários que poderão utilizar o sistema. Deve conter dados como: nome do usuário, senha e tipo de usuário onde poderá ser Administrador ou Usuário;
- c) RF3 – manter localizações: permitir a inclusão, alteração e exclusão de locais em que poderá ter objetos a serem cadastrados. Deve conter os dados como: descrição

do local;

- d) RF4 – consultar o arquivo de *log*: permitir a consulta ao arquivo de *log* dos objetos cujos estados foram alterados, onde deverá conter os campos: data e hora da alteração, objeto alterado, novo estado e o usuário que fez a alteração;
- e) RNF1 - executar no sistema operacional Windows.

Aplicativo remoto responsável pelo controle dos objetos:

- a) RF1 – consultar e alterar estados: permitir a consulta e a alteração do estado dos objetos que serão controlados;
- b) RF2 – autenticar o usuário: solicitar o nome e senha do usuário que consultará e/ou alterará os estados dos objetos e posteriormente solicitar ao *web service* a autenticação do usuário, passando para isso o nome e senha digitada;
- c) RF3 – enviar os dados alterados: enviar para o servidor os objetos que sofreram alteração de estado;
- d) RNF1 – executar em dispositivos móveis que suportem a plataforma .NET.

Serviço responsável pela alteração dos estados dos objetos:

- a) RF1 – monitorar os objetos alterados: o serviço deverá monitorar a cada 5 segundos os eventos que foram gerados pelo usuário e alterar o estado dos objetos;
- b) RF2 – gerar arquivo de *log*: o serviço será o responsável por gerar o arquivo de *log* dos objetos alterados;
- c) RNF1 – gravar em arquivo texto: o arquivo de *log* deverá ser gerado em um arquivo texto denominado “arqlog.txt”.

*Web service* responsável pela comunicação entre o servidor e o dispositivo móvel:

- a) RF1 – autenticação do usuário: permitir ao aplicativo remoto solicitar a autenticação do usuário, retornando se o mesmo poderá alterar o estado dos

objetos;

- b) RF2 – lista dos objetos: permitir ao aplicativo remoto solicitar a lista dos objetos cadastrados pelo aplicativo de configuração;
- c) RF3 – gravar os eventos: gravar no banco de dados os eventos que foram gerados pelas alterações feitas pelo usuário no aplicativo remoto.

## 5.2 ESPECIFICAÇÃO

Na especificação do protótipo foi utilizada a linguagem *Unified Modeling Language* (UML) para descrever os diagramas de casos de usos, de classes, de atividade e de seqüência.

### 5.2.1 Diagrama de Casos de Usos

Serão demonstrados os principais casos de uso do aplicativo de configurações, do aplicativo remoto e do serviço Windows.

O aplicativo de configurações do protótipo possui quatro casos de uso (Figura 8):

- a) manter objetos: responsável pela manutenção dos objetos que serão controlados pelo dispositivo móveis, onde serão cadastradas todas as informações necessárias para controle desse objeto;
- b) manter usuários: responsável pela manutenção dos usuários que poderão controlar os estados dos objetos e os que poderão as fazer configurações dos objetos;
- c) manter localização: responsável pela manutenção das localizações, ou seja, locais em que os objetos estão localizados na casa;
- d) consultar *log*: responsável pela consulta ao arquivo de *log*, onde estão

armazenados todos os eventos que foram gerados em que houve alteração do estado dos objetos.

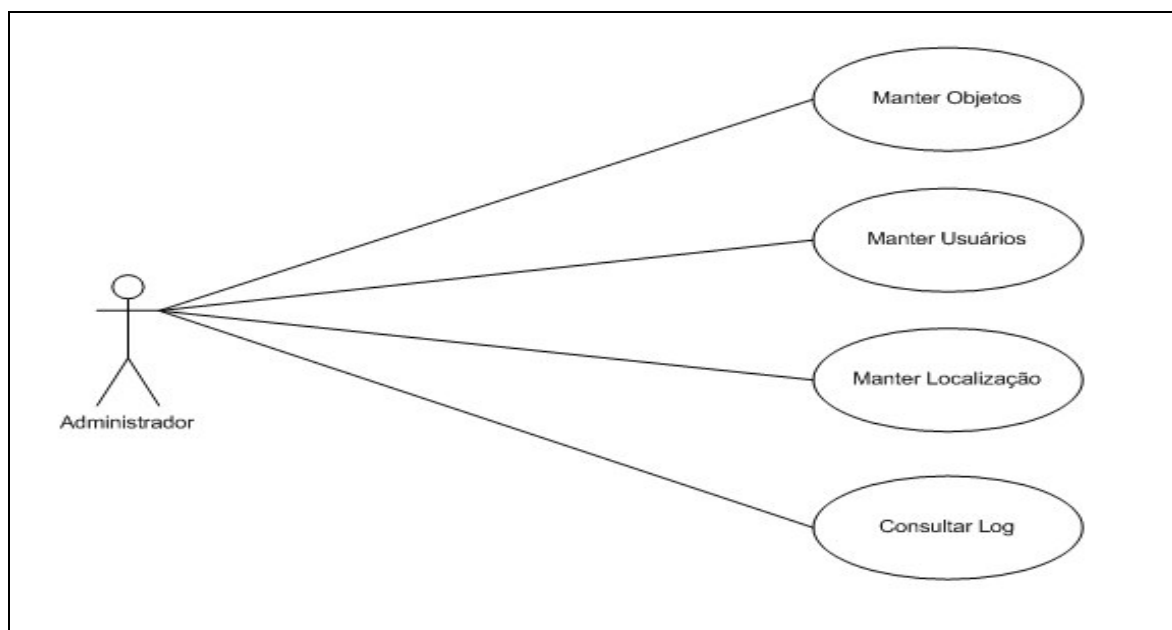


Figura 8 – Casos de uso do aplicativo de configurações

O aplicativo remoto do protótipo possui dois casos de uso (Figura 9):

- a) consultar objetos: responsável por consultar os objetos que podem ser controlados pelo usuário, solicitando através do *web service* a lista de objetos e seus respectivos estados;
- b) alterar estado dos objetos: responsável pela alteração do estado dos objetos no aplicativo remoto e pelo envio dos eventos gerados com a alteração do estado dos objetos.

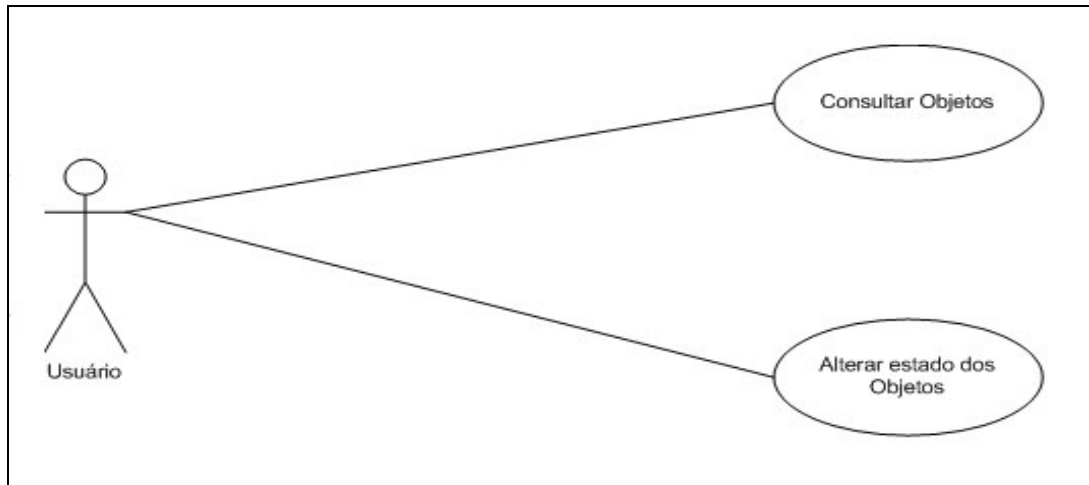


Figura 9 – Casos de uso do aplicativo remoto

O serviço de monitoramento dos objetos possui dois casos de uso (Figura 10):

- a) gravar arquivo *log*: responsável pela gravação do *log* em um arquivo texto o qual deve conter: data e hora de alteração do estado do objeto, o objeto alterado, o novo estado e o usuário que solicitou a alteração;
- b) monitorar eventos: responsável pelo monitoramento dos eventos que foram gerados pela aplicação remota e alteração do estado dos objetos.

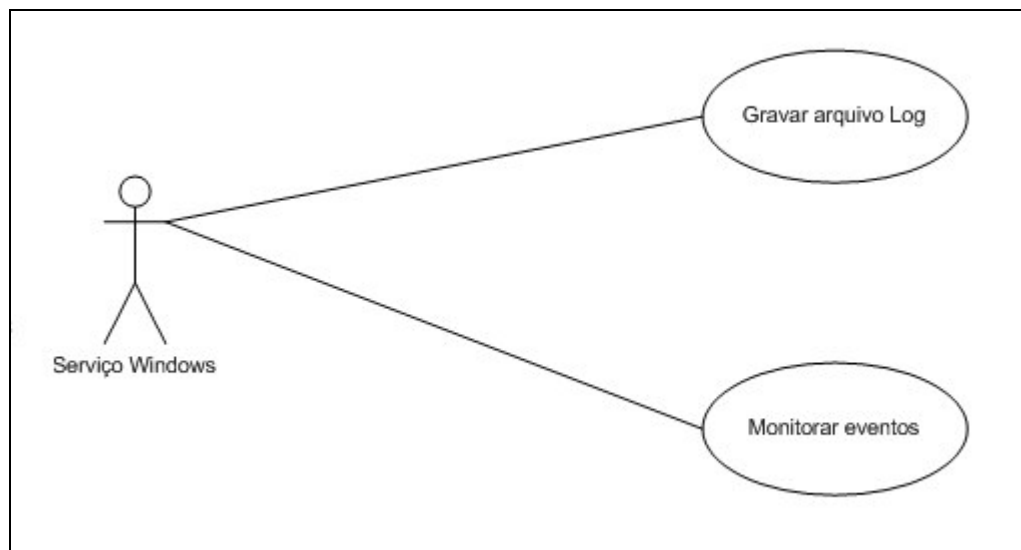


Figura 10 – Casos de uso do serviço Windows

## 5.2.2 Diagramas de Classes

A especificação foi desenvolvida baseada na orientação a objeto, utilizando o diagrama de classes da UML como linguagem de modelagem.

A figura 11 apresenta o modelo de diagrama de classes do protótipo:

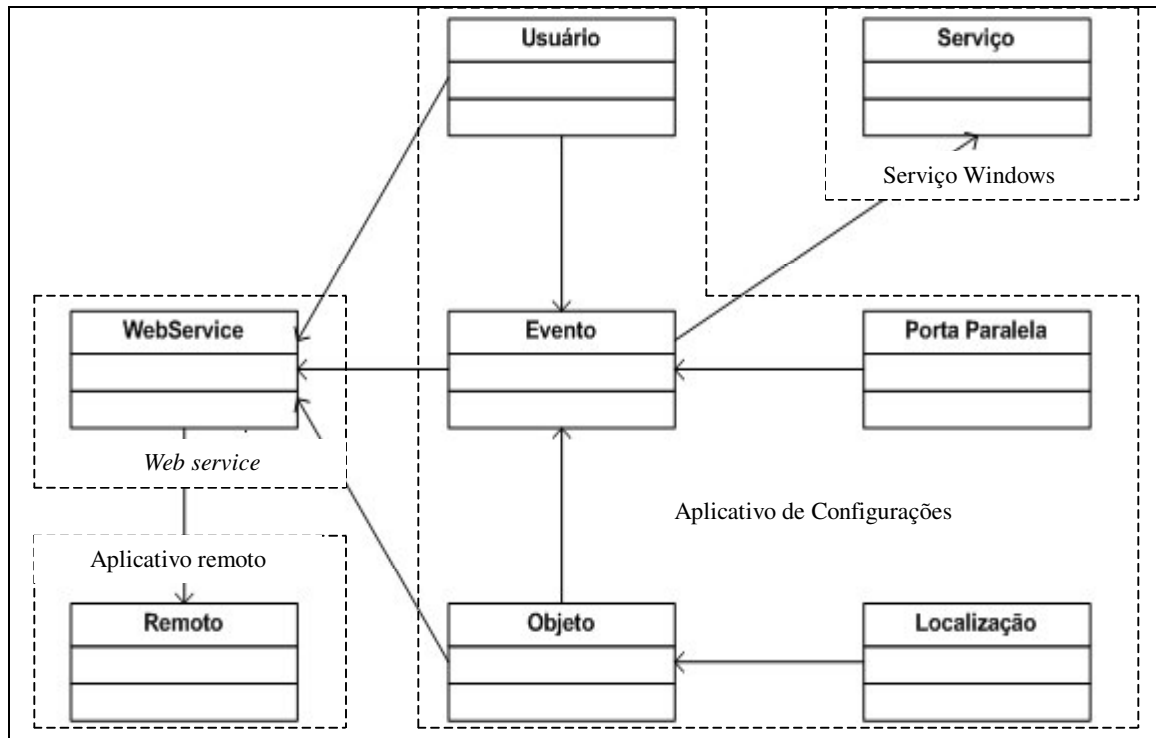


Figura 11 – Diagrama de classes do protótipo

- Usuário**: é uma classe de persistência, sendo responsável pelo acesso aos atributos e métodos dos usuários que poderão utilizar o protótipo, tendo métodos como: incluir, alterar e excluir usuários e o de autenticação de usuário, que valida o usuário para acessar o aplicativo;
- Objeto**: é uma classe de persistência, sendo responsável pelo acesso aos atributos e métodos dos objetos que serão manipulados pelo aplicativo remoto, tendo métodos como: incluir, alterar e excluir objetos e o de consultar objetos;
- Localização**: é uma classe de persistência, a qual é responsável pelo acesso aos

atributos e métodos de localização dos objetos, tendo métodos como: incluir, alterar e excluir localizações e método para carregar as localizações;

- d) Evento: é uma classe de persistência, responsável pela manutenção dos eventos gerados pelo aplicativo remoto e gravação do arquivo de *log*;
- e) Porta Paralela: esta classe é responsável pela comunicação com a porta paralela sendo que possui dois métodos, um para ler e outro para escrever na porta. Esta classe utiliza uma *dynamic link library* (DLL) externa denominada “Inpout32.dll” para fazer o acesso a porta;
- f) Serviço: esta classe é responsável pelo monitoramento a cada segundo para verificar se foram gerados eventos que alterem o estado dos objetos;
- g) Webservice: esta classe é responsável pela comunicação entre o aplicativo remoto e os dados do aplicativo de configurações, sendo ele quem solicita as informações para as classes Usuario e Objeto e passa para a classe Evento os objetos que tiveram o estado alterado pelo usuário;
- h) Remoto: esta classe é responsável pela solicitação de autenticação de usuário, a solicitação da lista de objetos que poderão ser controlados e o envio dos eventos que foram gerados, para a classe Webservice.

### 5.2.3 Diagrama de Atividades

O diagrama de atividades é usado para descrever processos de negócios, ou outros fluxos onde o paralelismo de atividades é importante (BEZERRA, p. 228, 2002).

A figura 12 apresenta o diagrama de atividades entre os módulos do protótipo.

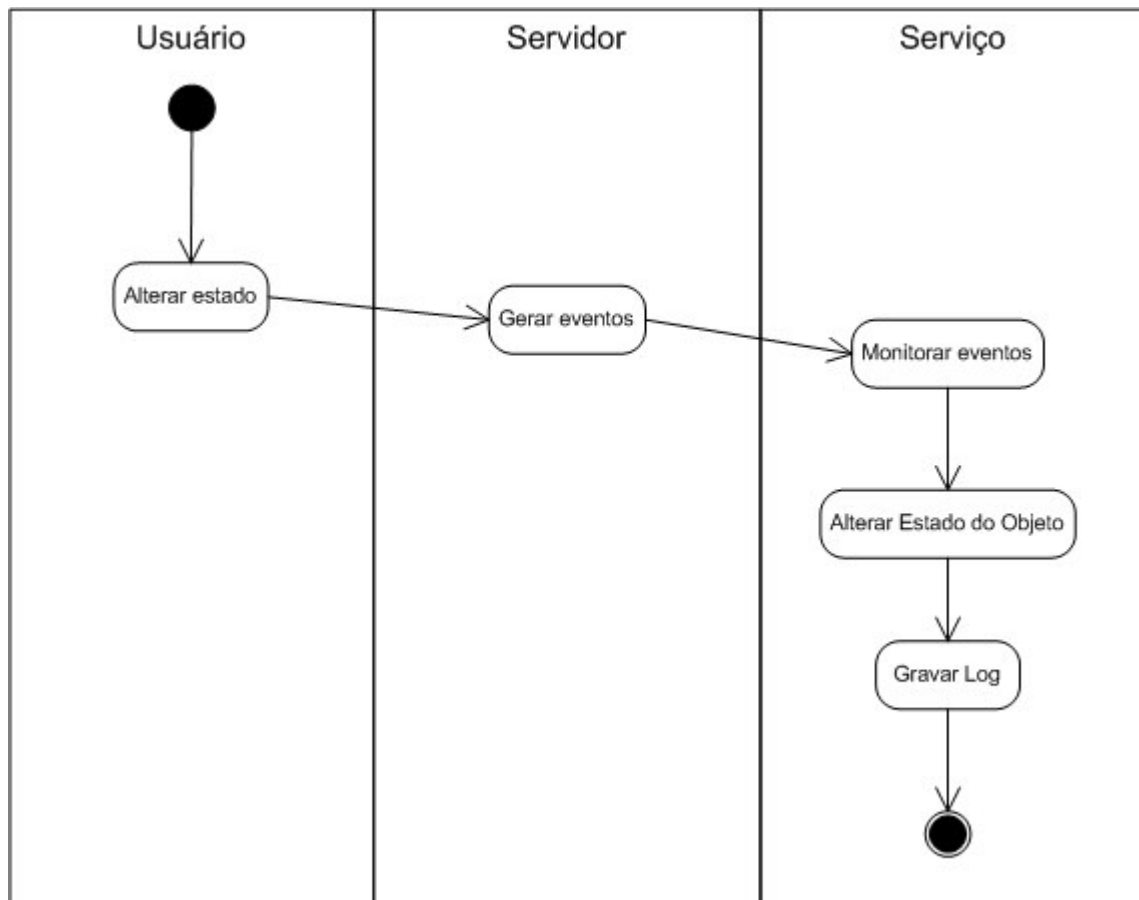


Figura 12 – Diagrama de atividades

No diagrama apresentado o usuário altera o estado no objeto remotamente, envia os dados para o servidor, este por sua vez gera os eventos. Os eventos são lidos pelo serviço Windows responsável por efetuar a alteração de estado dos objetos e gravar no arquivo de *log* as alterações executadas.

#### 5.2.4 Diagramas de Sequência

Os diagramas seqüência demonstram como são executadas as tarefas no protótipo, sendo que serão demonstradas duas das principais tarefas.

### 5.2.4.1 Alterar o Estado dos Objetos

O diagrama de seqüência (figura 13) demonstra a ação do usuário para alterar o estado dos objetos remotamente. Quando o aplicativo remoto é executado, solicita ao usuário o nome e senha para autenticação, após a identificação é chamado o método `AutenticarUsuario` da classe `WebService`, que por sua vez faz uma chamada ao método de mesmo nome para a classe `Usuario`. Caso a resposta da classe for positiva a classe `Remoto` executa seu próprio método denominado `AdicionaObjeto` que por sua vez faz uma chamada ao método `Objeto` da classe `WebService`, que faz a chamada ao método `Consultar` da classe `Objeto`, retornando a lista dos objetos cadastrados. Após a alteração do estado dos objetos pelo usuário, são enviados os eventos dos objetos alterados para a classe `WebService` através do método `GravarEventos` que por sua vez chama o método `Incluir` da classe `Evento`.

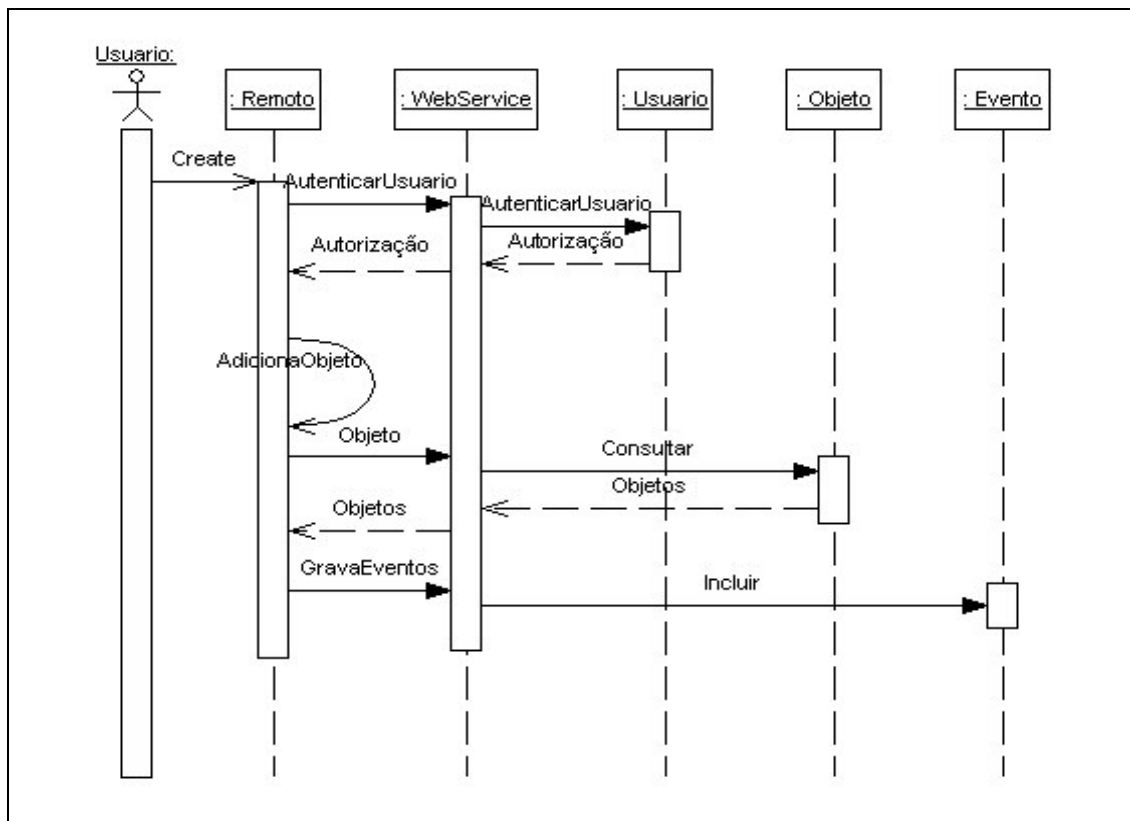


Figura 13 – Diagrama de seqüência “Alterar estado dos objetos”

## 5.2.4.2 Monitorar eventos

O diagrama de seqüência (figura 13) demonstra a ordem e as ações que são executadas a cada segundo pelo serviço Windows para monitoramento dos eventos gerados pelo usuário. De tempo em tempo, o serviço chama o método MonitorarEventos da classe Evento, que por sua vez chama o próprio método CarregarEventos. Caso haja algum evento, é chamado o método LerPorta da classe Porta Paralela retornando o estado da porta atual, altera o bit (que está configurado no objeto) através do método SetBit e escreve na porta paralela chamando o método EscreverPorta da classe Porta Paralela. Após escrever, é alterado o estado do objeto chamando o método AlterarEstado da classe Objeto e gravado o *log* do objeto alterado chamando o método próprio GravarLog.

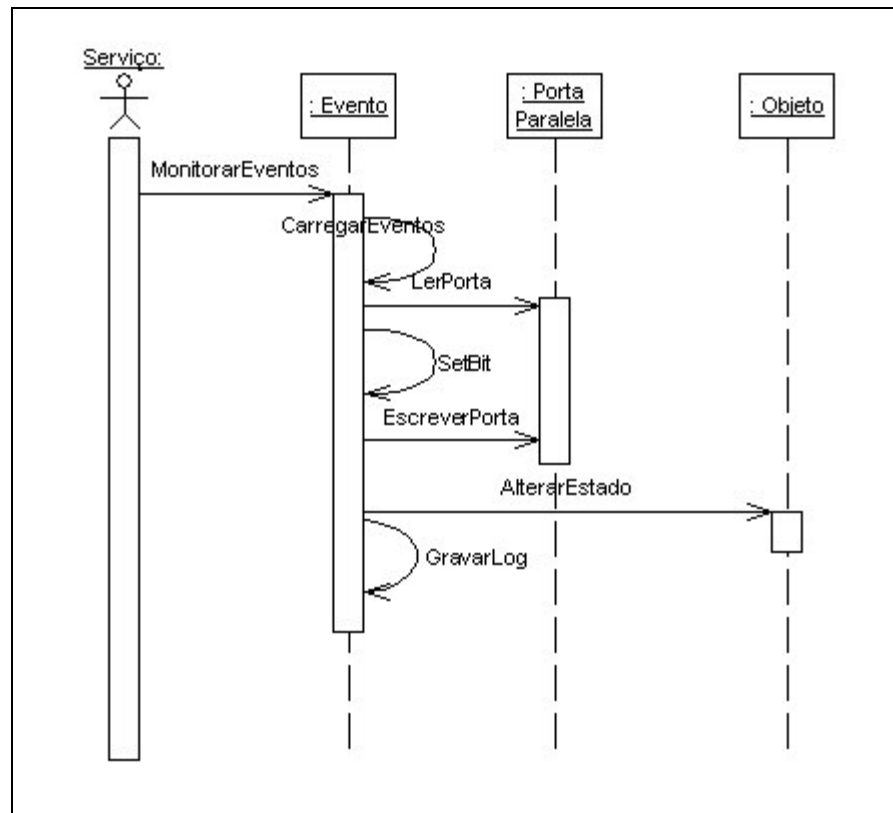


Figura 13 – Diagrama de seqüência “Monitorar Eventos”

## 5.3 IMPLEMENTAÇÃO

Neste capítulo serão apresentados os tópicos referentes à implementação do protótipo desenvolvido neste trabalho.

### 5.3.1 Técnicas e ferramentas utilizadas

No desenvolvimento do protótipo foi utilizado como ambiente de desenvolvimento o *Visual Studio .NET 2003* (figura 15). Segundo MSDN Brasil (2002a), o *Visual Studio .NET 2003* foi desenvolvido para atender as necessidades mais complexas de desenvolvimento do *software*. Com suporte integrado ao *.NET Compact Framework*, o *Visual Studio .NET 2003* pode agora abranger dispositivos móveis e dispositivos integrados na plataforma *.NET*, tais como o *Pocket PC*, além de outros dispositivos que rodam sob o sistema operacional *Microsoft Windows CE .NET*.

A linguagem utilizada para o desenvolvimento do protótipo foi a *Visual C# .NET*, que tem uma certa semelhança com o *C* e o *Java*. Segundo MSDN Brasil (2002b), o *Visual C# .NET 2003* e o *Microsoft .NET Framework* incluem suporte nativo para mais de 200 dispositivos móveis, incluindo telefones celulares, *paggers* e PDAs.

Para armazenamento dos dados foi utilizado o SGBD *SQL Server 2000* da *Microsoft* utilizando *Enterprise Manager* para criação do banco e das tabelas.

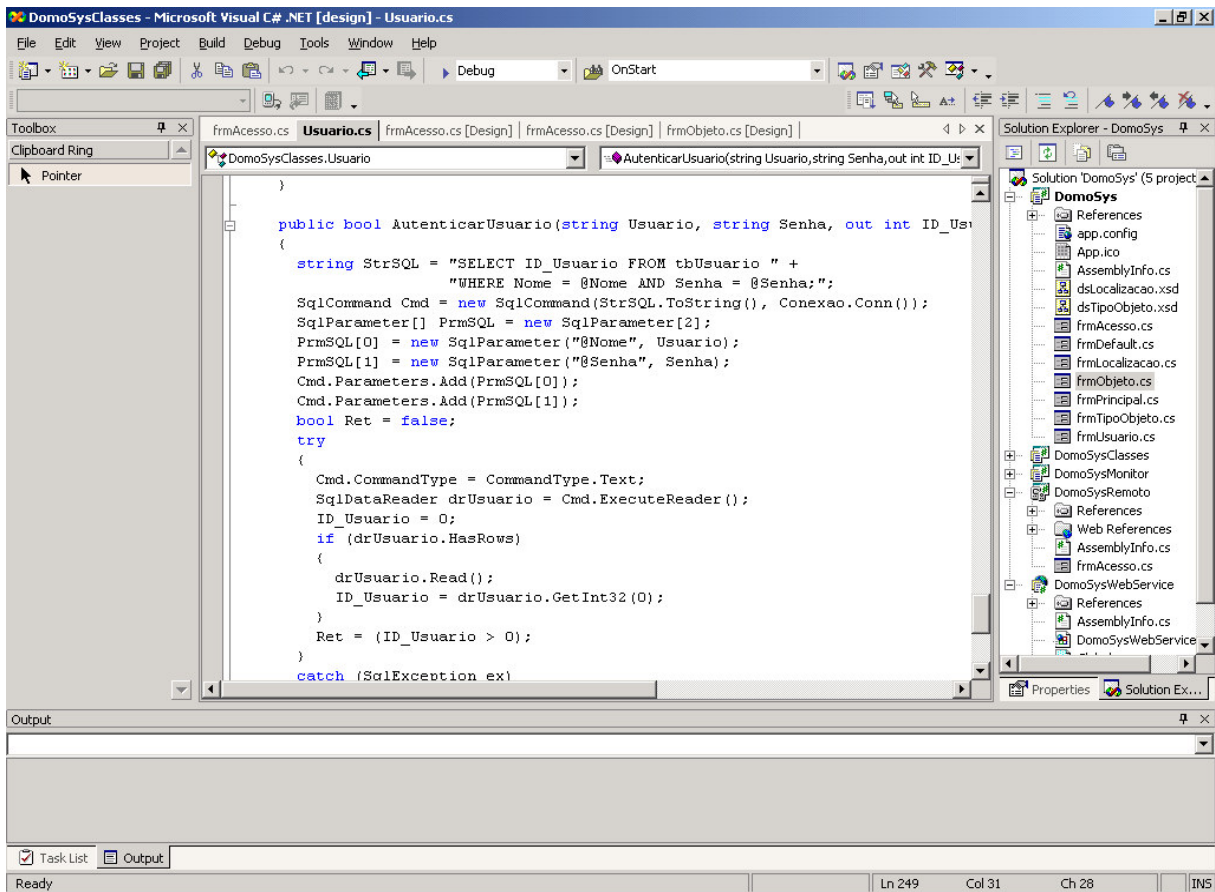


Figura 15 – Visual Studio .NET 2003

### 5.3.2 Implementação do protótipo

A implementação do protótipo foi separada em 4 módulos, sendo eles: o aplicativo de configurações, o aplicativo remoto, o serviço Windows e o *web service*.

A figura 16 mostra a tela de cadastro dos Objetos que faz parte do aplicativo de configurações, sendo que esta utiliza a classe Objeto que tem os métodos para manipulação dos dados.

Inserir Gravar Excluir Cancelar Sair

<< < > >>

Descrição Lampada

Localização Cozinha

Tipo Lampada

Pino Atuador 1

Pino Sensor 0

Estado Desligada

Figura 16 – Tela de cadastro dos objetos

O quadro 1 mostra o método excluir da classe Objeto que é executado quando for pressionado o botão Excluir da tela de cadastro demonstrado na figura 16. Pode-se ver no quadro 1 que a classe de persistência é responsável pela manipulação dos dados pertinentes a ela.

```

public void Excluir()
{
    // comando SQL que será executado para excluir o objeto
    string StrSQL = "DELETE FROM tbObjeto " +
        "WHERE ID_Objeto = @ID_Objeto ";
    SqlCommand Cmd = new SqlCommand(StrSQL.ToString(), Conexao.Conn());
    SqlParameter[] PrmSQL = new SqlParameter[1];
    // Parametro para o comando SQL
    PrmSQL[0] = new SqlParameter("@ID_Objeto", ID_Objeto);
    Cmd.Parameters.Add(PrmSQL[0]);
    try
    {
        Cmd.CommandType = CommandType.Text;
        // Executa o comando SQL sem retornar nada
        Cmd.ExecuteNonQuery();
    }
    // Caso acontecer exceção durante o processo de exclusão do objeto
    catch (SqlException ex)
    {
        throw new Exception("Erro Banco de Dados: " + ex.Message.ToString());
    }
    //Caso tiver outra exceção além da SqlException
    catch (Exception ex)
    {
        throw new Exception("Erro Runtime: " + ex.Message.ToString());
    }
    // Sempre será executado mesmo que acontecer alguma exceção
    finally
    {
        Cmd.Dispose();
    }
}
}

```

Quadro 1 – Código do método “Excluir” da classe objeto

A figura 17 apresenta a tela do aplicativo remoto o qual o usuário utilizará para controlar os objetos remotamente. O aplicativo está sendo executado no emulador do *Pocket PC 2003*.



Figura 17 – Tela do aplicativo remoto

O quadro 2 mostra um trecho de código do *web service* que o aplicativo remoto executa ao solicitar a autenticação de usuário, ao solicitar a lista de objetos e quanto enviar os eventos para gravação.

```
[WebMethod]
public bool AutenticarUsuario(string aNome, string aSenha, out int aID_Usuario)
{
    // Chamada ao método AutenticarUsuario da classe Usuario
    return clUsuario.AutenticarUsuario(aNome, aSenha, out aID_Usuario);
}

[WebMethod]
public DataSet Objetos()
{
    // Chamada ao método Consultar da classe Objeto
    return clObjeto.Consultar();
}

[WebMethod]
public void GravaEventos(int aID_Objeto, byte aEstado, int aID_Usuario)
{
    // Atribuição de valores aos atributos da classe Evento
    clEvento.ID_Objeto = aID_Objeto;
    clEvento.Estado = aEstado;
    clEvento.ID_Usuario = aID_Usuario;
    // Chamada ao método Incluir da classe Evento
    clEvento.Incluir();
}
```

Quadro 2 – Métodos do *web service* que são chamados pelo aplicativo remoto.

O quadro 3 apresenta os métodos GetBit e SetBit da classe Evento. O GetBit retorna Verdadeiro quando o bit da posição passada como parâmetro esteja ligado. Já o SetBit retorna o valor do byte alterado na posição passada como parâmetro.

```

// Retorna True caso o bit da posição passada como parâmetro esteja ligado
private bool GetBit(byte ByteAtual, int Posicao)
{
    return ((ByteAtual & (1 << Posicao - 1)) > 0);
}

// Liga/Desliga o bit da posição passada como parâmetro
private byte SetBit(byte ByteAtual, int Posicao, bool Liga)
{
    if (Liga)
    {
        // Liga o bit da posição passada como parâmetro e retorna o valor do byte alterado
        return (byte)(ByteAtual | (1 << Posicao - 1));
    }
    else
    {
        // Desliga o bit da posição passada como parâmetro e retorna o valor do byte alterado
        return (byte)(ByteAtual & ~(1 << Posicao - 1));
    }
}

```

Quadro 3 – Código dos métodos GetBit e SetBit da classe Evento.

O quadro 4 apresenta a classe PortaParalela e os métodos de leitura e escrita na porta paralela.

```

public class PortaParalela
{
    // Escreve um byte no endereço
    [DllImport("Inpout32.dll", EntryPoint="Out32")]
    private static extern void Escrever(int Endereco, byte Valor);

    // Lê um byte do endereço
    [DllImport("Inpout32.dll", EntryPoint="Inp32")]
    private static extern byte Ler(int Endereco);

    public void EscreverPorta(int Endereco, byte Valor)
    {
        Escrever(Endereco, Valor);
    }

    public byte LerPorta(int Endereco)
    {
        return Ler(Endereco);
    }
}

```

Quadro 4 – Código da classe PortaParalela e os métodos EscreverPorta e LerPorta.

A figura 18 demonstra o fluxo de dados entre o *Pocket PC* e o servidor e entre o servidor e o simulador de estado dos objetos.

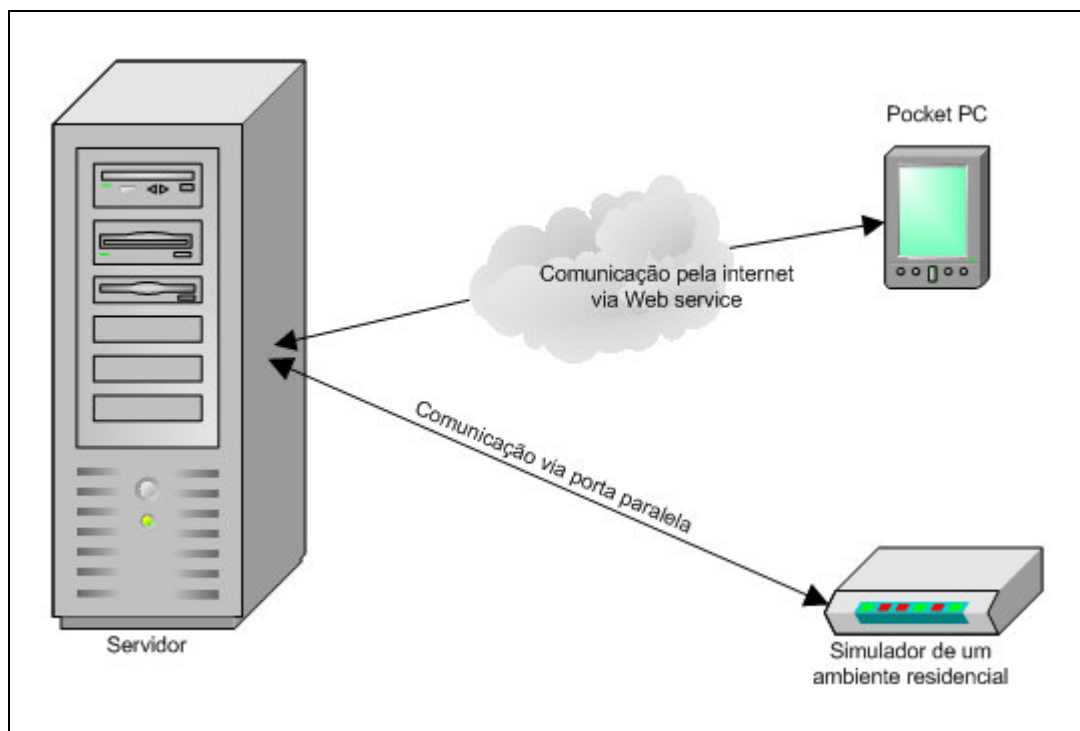


Figura 18 – Diagrama de fluxo de dados

Foi desenvolvido um simulador de ambiente residencial para demonstrar o funcionamento do protótipo. O simulador consiste em mostrar a alteração do estado dos objetos feita através do aplicativo remoto ou quando for alterado fisicamente por um usuário, como a abertura de uma porta.

A figura 19 mostra o circuito eletrônico do simulador de ambiente residencial: parte dos pinos atuadores.

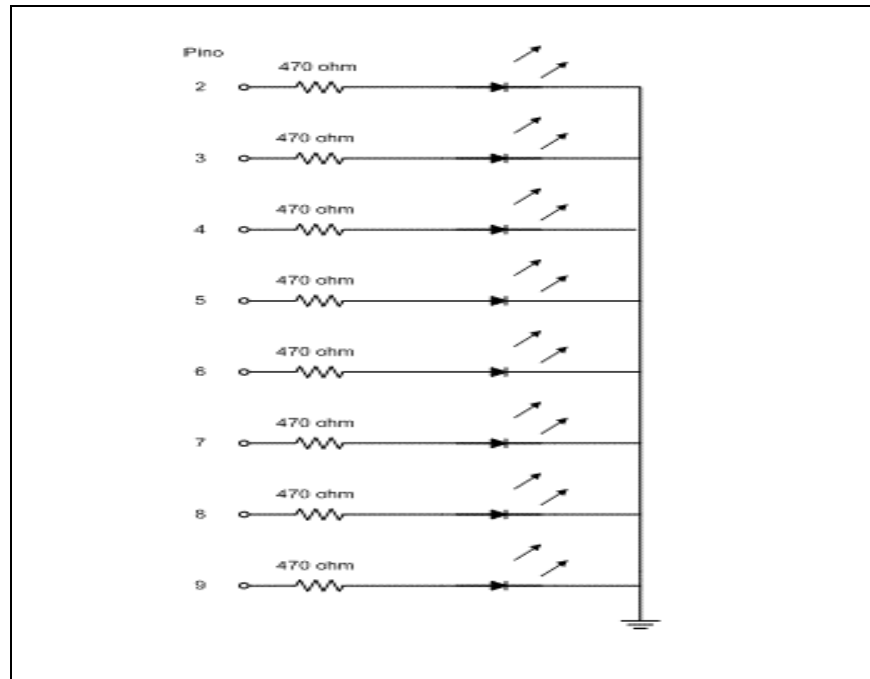


Figura 19 – Circuito do simulador de ambiente residencial (atuadores)

A figura 20 mostra o circuito eletrônico do simulador de ambiente residencial: parte dos pinos sensores.

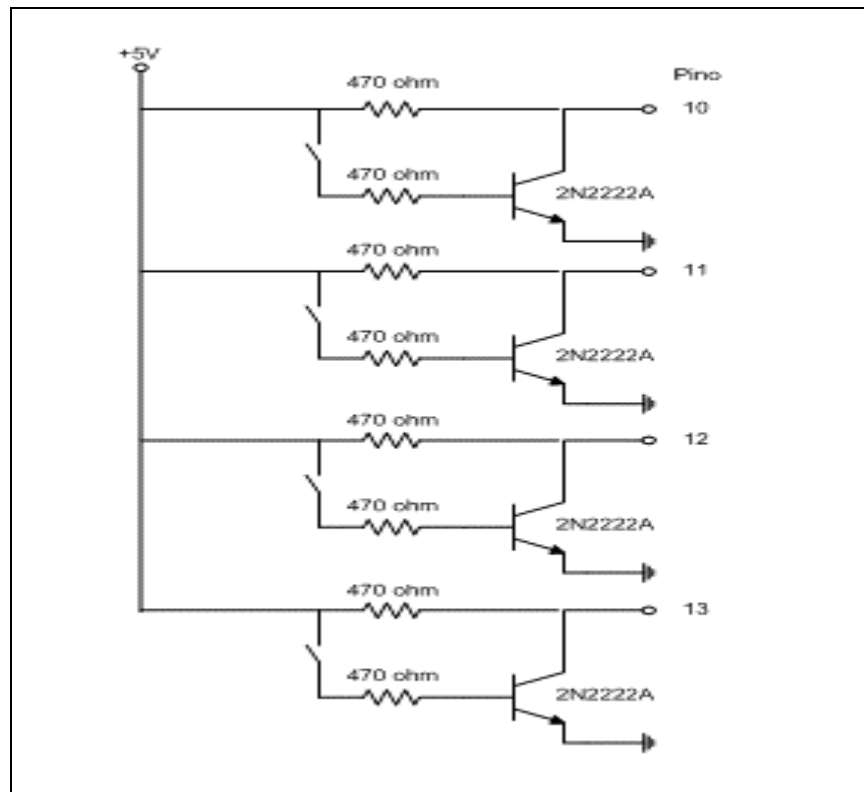


Figura 20 – Circuito do simulador de ambiente residencial (sensores)

### 5.3.3 Operacionalidade da implementação

O protótipo possui duas aplicações, além do serviço Windows e o *web service*, sendo este responsável pela comunicação entre o aplicativo remoto e a base de dados.

Serão exemplificados abaixo os procedimentos para cadastrar um objeto e posteriormente alterar o estado no aplicativo remoto, mostrando os resultados desejados através do simulador.

#### 5.3.3.1 Cadastro de um objeto

Primeiramente para entrar no aplicativo de configurações o usuário terá que ser administrador, sendo que um usuário em que o tipo é Usuário somente poderá acessar o aplicativo remoto.

A figura 21 demonstra a tela de cadastramento de um objeto, que consiste basicamente de informações que posteriormente serão utilizadas para fazer o monitoramento do objeto. Estas informações são:

- a) descrição: descrição do objeto será mostrada na tela do aplicativo remoto, ou seja, deverá ser mais detalhada possível;
- b) localização: local onde o objeto está instalado;
- c) tipo: tipo do objeto que está sendo cadastrado, sendo que poderá ser: Lâmpada, Porta/Portão, Janela, Cortina ou Irrigação;
- d) atuador: informar qual é o pino atuador que será utilizado na porta paralela para alterar o estado do objeto, sendo que os valores possíveis podem ser de 1 a 8;
- e) sensor: informar qual é o pino sensor que será utilizado na porta paralela para

alterar o estado do objeto, sendo que os valores possíveis podem ser de 0 a 4;

- f) estado: este campo está na tela somente como informação, não sendo possível editá-lo.

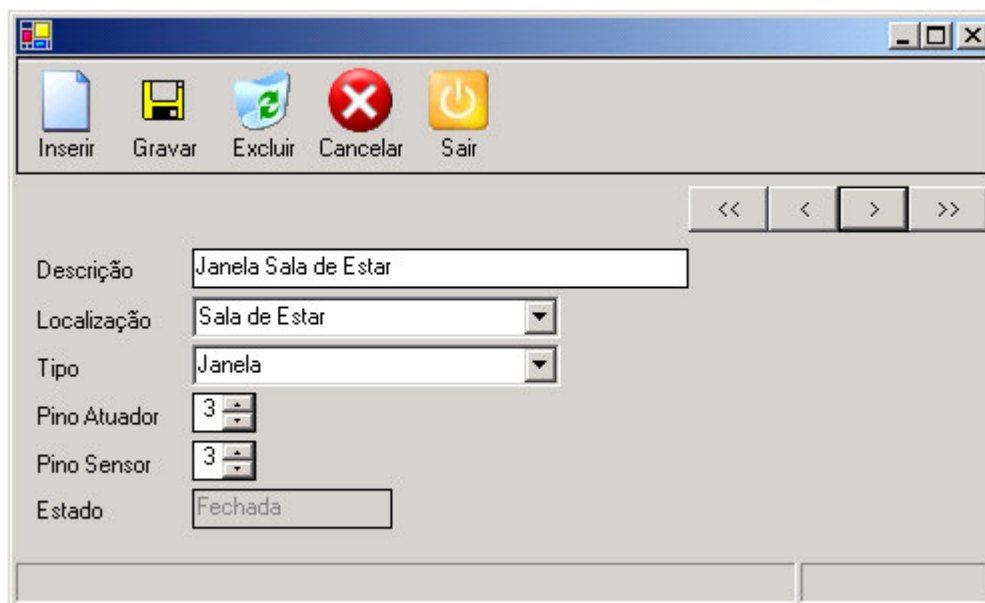


Figura 21 – Tela de cadastro de objetos

#### 5.3.3.2 Monitoramento pelo aplicativo remoto

Para acesso ao aplicativo remoto o usuário deverá digitar o nome e a senha, que serão solicitados na execução do sistema.

A figura 22 demonstra os objetos cadastrados sendo manipulados no aplicativo remoto pelo usuário. O usuário poderá marcar ou desmarcar o objeto, alterando com isso o estado do objeto, assim que ele clicar no botão “Enviar”. Para fazer uma nova consulta e verificar se teve alteração do estado dos objetos, pressiona-se o botão “Ler” que esse atualizará os dados do aplicativo remoto.



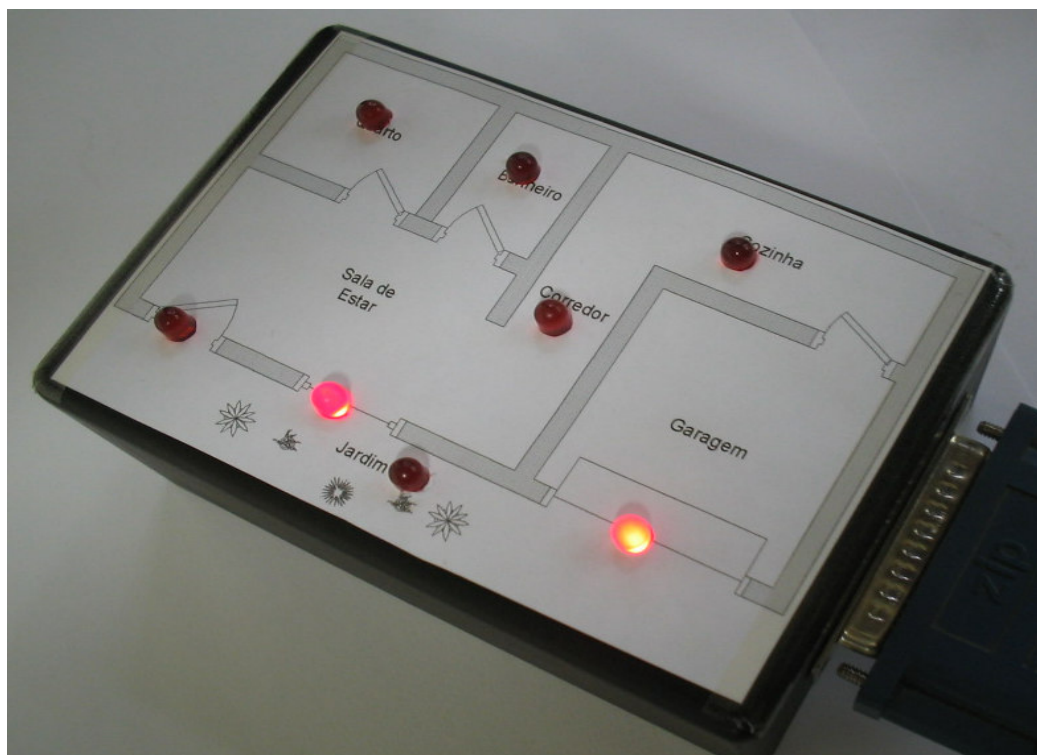


Figura 23 – Simulador de estado dos objetos

#### 5.4 RESULTADOS E DISCUSSÃO

O protótipo apresentou os resultados esperados, já que foram cumpridos todos objetivos. O que mudou em relação à proposta inicial foi o desenvolvimento de um hardware para demonstrar os resultados e não simular através de *software* como previsto. Optou-se por desenvolver somente o simulador em função de existirem trabalhos correlatos que desenvolveram um centralizador: Besen (1996) e Krueger (2002). Apesar deste trabalho ter como objetivo demonstrar uma aplicação para controle residencial à distância, concentrou-se em demonstrar as vantagens da comunicação *on-line*, tanto para envio como para recebimento de informações, mostrando com isso um diferencial em relação aos outros trabalhos sobre automação residencial.

Nos testes realizados o protótipo teve sua performance prejudicada em função do aplicativo remoto estar sendo executado em um emulador.

Percebe-se que o primeiro acesso ao *web service* tem um tempo de resposta maior em relação aos acessos seguintes.

Quanto às interfaces com o usuário, procurou-se simplificar ao máximo para tornar o uso mais fácil, principalmente no aplicativo remoto, já que este tem poucas teclas e dificuldade de se trabalhar com telas complexas.

A comparação entre os trabalhos correlatos é apresentada na tabela 1.

Tabela 1 – Comparação entre os trabalhos correlatos

<b>Acadêmicos</b> <b>Itens</b>	<b>Este trabalho</b>	<b>Depiné (2002)</b>	<b>Schaefer (2004)</b>	<b>Galvin (2004)</b>	<b>Krüger (2002)</b>
Linguagem de desenvolvimento	C# .NET	Java (J2ME)	Java (J2ME)	VB .NET	Object Pascal
Plataforma de desenvolvimento	.NET	Java	Java	.NET	Win32
Dispositivo móvel	<i>Pocket PC</i>	Celular convencional	Celular convencional	<i>Pocket PC</i>	Celular convencional
Acesso a <i>web</i>	Sim	Não	Sim	Sim	Sim
Forma de transmissão de dados	<i>Web service</i>	-	E-mail	<i>Web service</i>	E-mail
Objetivos	Monitorar objetos residenciais através de dispositivos móveis à distância	Calcular o melhor tempo de deslocamento em <i>rally</i> de regularidade	Coletar e transmitir informações através de dispositivos móveis	Disponibilizar informações através de dispositivos móveis para melhorar o atendimento ao cliente	Solução de baixo custo para monitoramento de segurança predial utilizando recursos da internet

O trabalho de Besen (1996), não foi citado na comparação por não ter a maioria dos itens da tabela, sendo que este foi citado como correlato pelo motivo de ter tratado do assunto “Automação Residencial”.

## 6 CONCLUSÕES

Observou-se que a plataforma .NET cumpriu o prometido em relação à facilidade de desenvolvimento de aplicativos para os dispositivos móveis, só que com um número menor de componentes. O que não foi muito satisfatório foi o desempenho do aplicativo remoto, mas isso pode ser explicado em função do aplicativo remoto executar sobre um emulador do *Pocket PC*.

A escolha da plataforma .NET e a utilização do *web service* para a comunicação entre o aplicativo remoto e servidor mostrou-se acertada, mesmo tendo a performance não muito satisfatória, mas com vantagem de trabalhar-se *on-line*.

Teve-se uma certa dificuldade inicial quando à utilização da linguagem Visual C#, já que não tinha-se muito conhecimento da mesma. Mas com consultas a livros e artigos sobre o assunto contornou-se o problema e a insegurança quanto à escolha da mesma.

Apesar de atingir os objetivos propostos, o protótipo tem o limite de somente poder alterar o estado do objeto de forma binário, ou seja, o objeto somente pode estar ligado ou desligado, aberto ou fechado. O administrador somente poderá administrar o sistema através do servidor, não podendo acessar através do aplicativo remoto.

Outra característica deste protótipo é que o mesmo não tem criptográfica e nem garante a integridade dos dados, sendo que sem isso o sistema tornar-se mais vulnerável a interferências externas indesejáveis.

## 6.1 EXTENSÕES

Este trabalho ainda foi pouco explorado, já que o mesmo somente utiliza uma propriedade por objeto. Pode-se utilizar novas propriedades como, por exemplo, controlar a intensidade de uma lâmpada, o percentual de abertura de uma janela, cortina ou porta, ou ainda o controle de temperatura do ar condicionado.

Criptografar e compactar os dados para que o sistema ganhe segurança e performace no tráfego dos mesmos.

Possibilitar a administração do sistema através do aplicativo remoto, não sendo mais necessário o administrador acessar o sistema através do servidor.

Gerar imagens através de câmera e possibilitar o acesso a elas através do aplicativo remoto, onde o usuário escolhe qual local da residência ele quer visualizar no dispositivo móvel.

Criar um protocolo de comunicação entre o serviço de monitoramento e o *hardware* centralizador para que se possa controlar um número maior de objetos na residência.

Avisar o usuário quando houver alteração de estado dos objetos que tiverem sensores.

## REFERÊNCIAS BIBLIOGRÁFICAS

- ANGEL, Patrícia Marta. **Introducción a la domótica**: tomo I. Embalse: EBAI, 1993.
- BARROSO, José A. de Souza. **Arquitetura de aplicações distribuídas utilizando .NET**. [S.l.], [2004?]. Disponível em:  
<[http://www.linhadecodigo.com.br/artigos.asp?id\\_ac=458&pag=1](http://www.linhadecodigo.com.br/artigos.asp?id_ac=458&pag=1)>. Acesso em: 24 set. 2004.
- BESEN, Nelson. **Sistema domótico para automação e controle de um cômodo residencial**. 1996. 69 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.
- BEZERRA, Eduardo. **Princípios de análise e projetos de sistemas com UML**. Rio de Janeiro: Campus, 2002.
- BOARO, Márcio. **Web services - uma visão inicial**. São Paulo, [2004]. Disponível em:  
<[http://www.microsoft.com/brasil/msdn/colunas/webservices/col\\_webservices\\_1.aspx](http://www.microsoft.com/brasil/msdn/colunas/webservices/col_webservices_1.aspx)>. Acesso em: 05 abr. 2005.
- BRAGAGNOLO, Daniel Henrique. **.NET framework - visão geral**. Campinas, [2004]. Disponível em: <<http://www.brdevelopers.net/BrPortal1/Artigos/overviewframework.aspx>>. Acesso em: 06 abr. 2005.
- BRETERNITZ Vivaldo José. **Domótica: as casas inteligentes**. jun. 2001. Disponível em:  
<<http://www.widebiz.com.br/gente/vivaldo/domotica.html>>. Acesso: em 30 mar. 2005.
- BURÉGIO, Vanilson A. A. **Desenvolvimento de aplicações para dispositivos móveis com .NET**. 2003. 69 f. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) – Centro de Informática, Universidade Federal de Pernambuco, Recife.
- CASA PLUS. **Domótica**. Portugal, [2004]. Disponível em:  
<<http://www.casaplus.pt/domotica/domotica.html>>. Acesso em: 30 mar. 2005.
- DALFOVO, Oscar et al. **A tecnologia do futuro Wi-Fi (Wireless Fidelity)**. Blumenau, 2003. Disponível em:  
<[http://campeche.inf.furb.br/siic/siego/docs/Artigo\\_Wireless\\_Uniplac\\_2003.pdf](http://campeche.inf.furb.br/siic/siego/docs/Artigo_Wireless_Uniplac_2003.pdf)>. Acesso em: 09 abr. 2005.
- DEITEL, H. M et al. **C# - como programar**. Tradução João Eduardo Nóbrega Tortello. São Paulo: Pearson Education do Brasil, 2003.

DEPINÉ, Fábio M. **Protótipo de software para dispositivos móveis utilizando Java ME para cálculo de regularidade em rally**. 2002. 55 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

DORNAN, Andy. **Wireless communication** – o guia essencial de comunicação sem fio. Tradução Fábio Freitas. Rio de Janeiro: Campus, 2001.

FINKELSTEIN, Clive. **Microsoft web services support**. Austrália, fev. 2003. Disponível em: <[http://www.dmreview.com/article\\_sub.cfm?articleId=6304](http://www.dmreview.com/article_sub.cfm?articleId=6304)>. Acesso em: 05 abr. 2005.

FUNDÃO DA COMPUTAÇÃO. **Visão geral do .NET**. Campo Grande, out. 2002. Disponível em: <<http://www2.fundao.pro.br/articles.asp?cod=59>>. Acesso em: 05 abr. 2005.

GALVIN, Deleon. **Protótipo de sistema de CRM para dispositivos móveis utilizando a tecnologia .NET**. 2004. 89 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

HADDAD, Renato. **Visão geral do .NET compact framework**. São Paulo, [2004]. Disponível em: <<http://www.microsoft.com/brasil/msdn/Tecnologias/Default.msp>>. Acesso em: 09 set. 2004.

KRÜGER, Erasmo. **Protótipo de sistema de segurança predial através de monitoramento utilizando recursos da internet**. 2002. 61 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

LIPPMAN, Stanley B. **C# - um guia prático**. Tradução Werner Loeffler. Porto Alegre: Bookman, 2003.

MICROSOFT CORPORATION. **Introdução a .NET**. São Paulo, [2002]. Disponível em: <<http://www.microsoft.com/brasil/dotnet/introducao/oquee.asp>>. Acesso em: 30 set. 2004.

\_\_\_\_\_. **Microsoft .NET: framework e aplicativos web**. Tradução Izabel Cristina de Mendonça Santos; Ana Beatriz Tavares. Rio de Janeiro: Campus, 2001.

MSDN Brasil. **Visão geral do Visual Studio .NET 2003**. São Paulo, [2002a]. Disponível em: <<http://www.microsoft.com/brasil/msdn/produtos/VisualStudio/VSNET2003/Default.msp>>. Acesso em: 29 mai. 2005.

\_\_\_\_\_. **As novidades do Visual C# .NET 2003**. São Paulo, [2002b]. Disponível em: <<http://www.microsoft.com/brasil/msdn/produtos/VisualStudio/CSharp2003/Default.msp>>. Acesso em: 29 mai. 2005.

PEKUS. **Dispositivos móveis**. São Paulo, [2004a]. Disponível em: <<http://www.pekus.com.br/palmtops.htm>>. Acesso em: 28 set. 2004.

\_\_\_\_\_. **Wireless**. São Paulo, [2004b]. Disponível em: <<http://www.pekus.com.br/wireless.htm>>. Acesso em: 28 set. 2004.

ROMAN, Steven; PETRUSKA Ron; LOMAX Paul. **Linguagem VB .NET – o guia essencial**. Tradução Kátia Roque. Rio de Janeiro: Campus, 2002.

SANTOS, Michael S. **Utilização de web services na plataforma .NET para a criação de um aplicativo visualizador de notícias para dispositivos móveis**. 2003. 89 f. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) - Centro Universitário Luterano de Palmas, Palmas.

SCHAEFER, Carina. **Protótipo de aplicativo para transmissão de dados a partir de dispositivos móveis aplicado a uma empresa de transportes**. 2004. 52 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

WORLD WIDE WEB CONSORTIUM. **Extensible Markup Language (XML)**. USA, 2003. Disponível em: <<http://www.w3.org/XML>>. Acesso em: 09 abr. 2005.