



FACULDADE FARIAS BRITO
CIÊNCIA DA COMPUTAÇÃO

PATRICK ROMERO FROTA QUINDERÉ

**Casa Inteligente – Um Protótipo de Sistema de Automação
Residencial de Baixo Custo**

Fortaleza

Janeiro/2009

PATRICK ROMERO FROTA QUINDERÉ

**Casa Inteligente - Um Protótipo de Sistema de Automação
Residencial de Baixo Custo**

Monografia apresentada para obtenção dos créditos da disciplina Trabalho de Conclusão do Curso da Faculdade Farias Brito, como parte das exigências para graduação no Curso de Ciência da Computação.

Orientador: MSc. Roberto Façanha.

Fortaleza 2009

Janeiro/2009

Casa Inteligente - Um Protótipo de Sistema de Automação Residencial de Baixo Custo

Patrick Romero Frota Quinderé

PARECER _____

NOTA: **FINAL:**_____

Data: ____/____/____

BANCA EXAMINADORA:

Prof. Roberto de Almeida Façanha, Ms.
(Orientador)

Profa. Wietske Ineke Meyering, Dra.
(Examinador)

Prof. Maikol Magalhães Rodrigues, Ms.

(Examinador)

RESUMO

Domótica é a área do conhecimento e também da engenharia voltada ao desenvolvimento de soluções de automação residencial para dispor, aos seus usuários, maior conforto e segurança. Outra aplicação, que possui forte apelo social, é prover a acessibilidade a atividades e equipamentos aos portadores de necessidades especiais que, anteriormente, dependeriam da intervenção de outras pessoas. A origem etimológica do termo domótica vem da junção das palavras *Domus*, que em latim significa residência; e robótica, área da mecatrônica que utiliza os conceitos de robótica, eletro-eletrônica e programação, para o desenvolvimento de soluções de automação residencial. Tais soluções normalmente são compostas por um *hardware* de controle, responsável pelo monitoramento de sensores e acionamento de dispositivos, e um *software* de gerenciamento do sistema, que dispõe de funcionalidades básicas de cadastramento de dispositivos, monitoramento de eventos e execução de comandos. Este projeto se propõe a desenvolver um protótipo de um sistema de domótica de baixo custo composto por um *hardware* de controle que se comunica com um computador através de interface paralela, um *software* de gerenciamento, com acesso através da *Internet* e com suporte a dispositivos móveis, e uma maquete de demonstração. Este protótipo também tem o objetivo de demonstrar a viabilidade de soluções de baixo custo, democratizando o acesso a automação residencial.

Palavras chave: Domótica, Automação residencial, Soluções de Baixo Custo, Porta Paralela e .NET.

AGRADECIMENTOS

Agradeço a Deus, por estar sempre presente, dando-me forças para continuar.

Agradeço aos meus pais, Luiz Quinderé Ribeiro e Leniza Romero Frota Quinderé, e aos meus irmãos, Maria Thereza da Frota Quinderé Ribeiro, Luiz Quinderé Ribeiro Filho, Maêthe Romero Frota Quinderé e Melchior de Quental Quinderé Ribeiro, que sempre estiveram ao meu lado, dando apoio e servindo de exemplo.

Agradeço aos meus avós maternos, Isolina Romero Da Frota e Perez Urtiaga, pelo apoio incondicional em todos os momentos.

Agradeço aos meus avós paternos, Joaquim Gomes Ribeiro (in memoriam) e Julieta Quinderé Ribeiro (in memoriam), por todo o amor que me foi dedicado.

Agradeço aos meus tios e primos, que sempre estiveram presente na minha vida, em especial à Tiinha e à Tita, que sempre me encorajaram a continuar.

Agradeço à minha namorada Clara Mota Randal Pompeu, pelo exemplo de força de vontade, dedicação, ajuda e por estar sempre ao meu lado, dando-me forças para continuar, em todos os momentos, e dizendo-me que não estou cansado.

Agradeço ao professor Roberto Façanha, pela orientação, confiança, amizade e tempo dedicado ao meu trabalho.

Agradeço aos professores Maikol, Sérgio, Aragão, Josino, Lúcio, Helano, Manoel, Mateus, Murilo, Sales e Ricardo, por todos os ensinamentos.

Agradeço às minhas coordenadoras Fani e Wietske, por todas as chances, apoio, dedicação ao curso e por acreditarem em mim, sempre.

Agradeço ao diretor Miguel, por toda atenção que me foi dada e dedicação ao curso.

Agradeço aos meus amigos, Felipe Oquendo, Raphael Saldanha, Lívia Figueiredo e Rafael Correia, pelo apoio e colaboração.

Agradeço ao meu grande amigo Jorge Albuquerque, por sempre ter me conduzido pelos melhores caminhos, sendo um verdadeiro irmão.

Agradeço aos colegas, professores e funcionários do curso, que contribuíram para a minha formação pessoal e profissional.

Agradeço ao meu amigo Mauro Neres, por toda a ajuda prestada.

Agradeço aos meus amigos Rodrigo e Bernardo, por toda a ajuda com os laboratórios.

Agradeço aos amigos da banda Mr. Jingles, por tudo o que passamos juntos.

Agradeço a todos os amigos da RCN, em especial Ronaldo Cabral, por suas atitudes, que me servirão de exemplo para o resto da vida.

Agradeço a todos aqueles que, de forma direta ou indireta, colaboraram para a conclusão deste trabalho.

“Só termina quando acaba!”

SUMÁRIO

1	INTRODUÇÃO	13
1.1	Estrutura do Trabalho	14
2	JUSTIFICATIVA	15
3	OBJETIVOS.....	15
3.1	Objetivo geral.....	15
3.2	Objetivos específicos	16
4	REVISÃO DE LITERATURA.....	16
4.1	Domótica.....	16
4.2	Plataforma .NET.....	21
4.3	Comunicação.....	24
4.3.1	Porta Serial	25
4.3.2	Porta Paralela	25
4.3.3	Barramento e Interfaces USB	26
4.4	Padrões de Projeto	26
4.4.1	<i>Abstract Factory</i>	27
4.4.2	<i>Facade</i>	28
4.4.3	Padrão Camadas	29
5	Construção do Protótipo	30
5.1	O <i>Software</i>	31
5.1.1	Ferramentas utilizadas.....	31
5.1.2	Requisitos.....	32
5.1.3	Modelagem	33
5.1.3.1	Diagramas de casos de uso	33
5.1.3.2	Diagramas de classes	34
5.1.3.3	As Telas da Aplicação	43
5.2	Metodologia de Testes	46
5.3	O <i>Hardware</i>	47
5.3.1	Placa de Relés	47
5.3.2	Placa de Sensores	49
5.3.3	A Concepção do <i>Hardware</i>	51
5.4	A Maquete da Casa Inteligente	51

6	DIFICULDADES ENFRENTADAS	53
7	TRABALHOS FUTUROS	54
8	CONCLUSÕES.....	54
	REFERÊNCIAS BIBLIOGRÁFICAS.....	57
	APÊNDICES.....	61
	APÊNDICE A – Namespaces, Classes, Interface e Tipos de dados utilizados.....	62
	APÊNDICE B – Descrição dos métodos da camada DAL	64
	APÊNDICE C – Descrição dos métodos da camada <i>Business</i>	67

LISTA DE FIGURAS

Figura 1: Uso da domótica para personalização e controle de ambientes.	14
Figura 2: Arquitetura centralizada dos equipamentos. Fonte: FERREIRA (2008).	17
Figura 3: Representação da infra-estrutura de uma residência automatizada. Fonte: BOLZANI (2004).	19
Figura 4: Esquema de execução na plataforma .NET (FINKELSTEIN, 2003)	23
Figura 5: Resultado da aplicação do padrão <i>Facade</i>	29
Figura 6: Arquitetura em três camadas. Fonte: Maziero (2008).	30
Figura 7: Diagrama de casos de uso do sistema.	34
Figura 8: Relacionamentos entre as camadas do sistema.	35
Figura 9: Diagrama de classes do pacote Common.	36
Figura 10: Diagrama de interfaces da camada DAL.	37
Figura 11: Classes concretas da camada DAL.	39
Figura 12: Diagrama de interfaces da camada Business	40
Figura 13: Classes Concretas da camada <i>Business</i>	42
Figura 14: Tela de consulta de controladores.	43
Figura 15: Tela de consulta de comandos.	43
Figura 16: tela de consulta de equipamentos.	44
Figura 17: tela de consulta dos comandos associados ao equipamento.	45
Figura 18: tela de associação de eventos a comandos.	45
Figura 19: Aplicação sendo acessada através do <i>iPhone</i>	46
Figura 20: Placa de relés.	48
Figura 21: Circuito elétrico da placa de relés.	49
Figura 22: Placa de sensores.	50
Figura 23: Circuito elétrico da placa de sensores.	50
Figura 24: Maquete da Casa Inteligente.	52
Figura 25: Interruptor.	53

LISTA DE QUADROS

Quadro 1: Situação atual da infra-estrutura das residências (BOLZANI, 2004).....	19
Quadro 2: Situação proposta para a infra-estrutura das residências (BOLZANI, 2004).....	19
Quadro 3: Classes do pacote Common.	36
Quadro 4: Interfaces da camada DAL.	38
Quadro 5: Classes concretas da camada DAL.	39
Quadro 6: Interfaces da camada Business.	40
Quadro 7: Classes concretas da camada <i>Business</i>	41
Quadro 8: Legenda de cores da placa de relés.	48
Quadro 9: Legenda de cores da placa de sensores.	50
Quadro 10: Distribuição dos equipamentos por cômodo.	52
Quadro 11: Namespaces utilizados na implementação do projeto (MSDN, 2009a).	62
Quadro 12: Tipos de valores encontrados na FCL utilizados na implementação do projeto (MSDN, 2009b).	63
Quadro 13: Descrição das assinaturas dos métodos da interface IDAL.	64
Quadro 14: Descrição das assinaturas dos métodos da interface IControladorDAO.	64
Quadro 15: Descrição das assinaturas dos métodos da interface IComandoDAO.	64
Quadro 16: Descrição das assinaturas dos métodos da interface IEquipamentosDAO.	65
Quadro 17: Descrição das assinaturas dos métodos da interface IControladorDAO.	65
Quadro 18: Descrição da assinatura do método das interfaces IComandoDoEquipamentoDAO e IEventoDoEquipamentoDAO.	65
Quadro 19: Descrição dos métodos da classe AbstractDAO.	65
Quadro 20: Descrição dos métodos da classe DAOFactory.	66
Quadro 21: Descrição dos métodos da classe AbstractSQLDAO.	66
Quadro 22: Descrição das assinaturas dos métodos da interface IBusiness.	67
Quadro 23: Descrição das assinaturas dos métodos da interface IControladoresBusiness.	67
Quadro 24: Descrição das assinaturas dos métodos da interface IComandoBusiness.	67
Quadro 25: Descrição das assinaturas dos métodos da interface IEquipamentosBusiness.	68
Quadro 26: Descrição das assinaturas dos métodos da interface IExecutarBusinessFacade.	68
Quadro 27: Descrição dos métodos da classe AbstractBusiness.	68
Quadro 28: Descrição dos métodos da classe ExecutarBusinessFacade.	68

LISTA DE ABREVIATURAS

CAD	<i>Computer Aided Design</i>
CLR	<i>Common Language Runtime</i>
CLS	<i>Common Language Specification</i>
CPU	<i>Central Processing Unit</i>
DAL	<i>Data Access Layer</i>
EPP	<i>Enhanced Parallel Port</i>
FCL	<i>Framework Class Library</i>
IDE	<i>Integrated Development Environment</i>
IEEE	<i>Institute of Electrical and Electronics Engineers</i>
JIT	<i>Just-in-Time</i>
LPT	<i>Line Printer Terminal</i>
LED	<i>Light Emitting Diode</i>
MSIL	<i>Microsoft Intermediate Language</i>
PC	<i>Personal Computer</i>
PDA	<i>Personal Device Assistants</i>
PE	<i>Portable Executable</i>
SPP	<i>Standard Parallel Port</i>
UART	<i>Standard Parallel Port</i>
USART	<i>Personal Device Assistants</i>
UML	<i>Unified Modeling Language</i>
USB	<i>Universal Serial Bus</i>

1 INTRODUÇÃO

Ao longo do tempo, a computação vem ajudando pessoas comuns a fazerem seus trabalhos de forma cada vez mais rápida e eficiente. Com o advento da automação, tarefas repetitivas podem ser realizadas por máquinas.

Uma tarefa realizada por máquina tende a ser concluída em uma velocidade notavelmente mais rápida, apresentando uma maior confiabilidade do que se fosse realizada por um humano, além de acarretar em um menor desgaste do usuário, que poderá realizar outras atividades mais complexas, implicando ganho de tempo e produtividade.

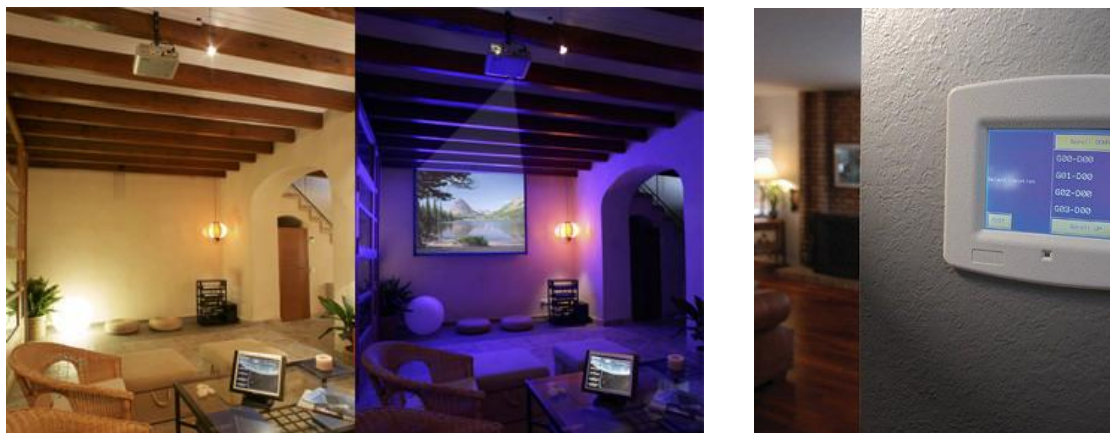
Nos últimos anos, as pessoas têm procurado levar a automação para seu ambiente domiciliar, buscando maior comodidade e mais tempo para descanso. Tendo isso em vista, foi desenvolvida uma nova área da automação, a domótica.

A domótica é uma nova tecnologia que consiste em um sistema integrado capaz de controlar todos os ambientes de uma residência através de um só equipamento, incluindo temperatura, luminosidade, som, segurança, entre outros (BOLZANI, 2004).

A automação já é uma realidade na indústria há muitos anos, dada a necessidade de automatizar atividades como forma de redução de custos, objetivando também a realização de tarefas inadequadas ao ser humano, como, por exemplo, o controle da temperatura de autoforos em siderúrgicas. A novidade aqui consiste na aplicação do mesmo conceito voltado para bem-estar e conforto residenciais, através da automatização da climatização, iluminação e segurança, dentre outros itens possíveis.

Deste ponto em diante, quando o texto fizer menção ao termo casa inteligente, trata-se de uma residência que possui um sistema de automação baseado na tecnologia domótica.

Um bom exemplo de que as pessoas estão buscando certo grau de automação em suas residências é a instalação de soluções proprietárias que permitam a integração de funcionalidades como acionamento de luzes, equipamentos de climatização, televisores e travas de segurança, muitas vezes através de perfis que executam conjuntos de ações de maneira integrada, conforme Figura 1 (a). Estas soluções normalmente são acionadas por um dispositivo central, Figura 1 (b), que pode ser fixo ou móvel.



(a) Adaptação de ambientes
Fonte: Decoesfera (2009)

(b) Acionador fixo.
Fonte: Nicosshop (2009)

Figura 1: Uso da domótica para personalização e controle de ambientes.

Portanto, da mesma forma que ocorreu uma revolução na vida das pessoas com o surgimento dos PCs (*Personal Computers*), é possível que também ocorra uma revolução com o advento da domótica, fazendo com que as casas inteligentes se tornem indispensáveis aos padrões de qualidade de vida atuais.

1.1 Estrutura do Trabalho

O estudo está organizado como segue. O primeiro capítulo refere-se à introdução, enfatizando a contextualização, aplicações em potencial e a estrutura do trabalho. O segundo capítulo aborda a justificativa do trabalho, explicando quais os motivos que levaram a elaboração do mesmo.

O terceiro capítulo refere-se aos objetivos, gerais e específicos, do projeto. O quarto capítulo trata do referencial teórico, enfocando a domótica, a Plataforma .NET, aspectos de sincronia e interfaces de comunicação e os padrões de projeto utilizados.

O quinto capítulo trata da construção do protótipo com as suas especificações e os resultados obtidos. No sexto e sétimo capítulos, analisam-se, respectivamente, as dificuldades enfrentadas diante do projeto e as oportunidades de melhoria e trabalhos futuros.

Por fim, o último capítulo apresenta as conclusões do trabalho.

2 JUSTIFICATIVA

Por ser uma área relativamente nova no Brasil e, portanto, ainda pouco difundida, os custos das soluções proprietárias para automação residencial ainda são bastante elevados. Conseqüentemente, o público alvo deste mercado restringe-se às pessoas de maior poder aquisitivo.

Entretanto, portadores de necessidades especiais existem, indistintamente, em todas as camadas da sociedade. Infelizmente, para as pessoas situadas nas camadas de menor poder aquisitivo, as dificuldades são agravadas por sua condição sócio-econômica, que geralmente fazem com que estas residam em locais com infra-estrutura de saneamento inadequada ou inexistente e / ou com estrutura de transporte limitada. Além disso, muitas vezes estas pessoas encontram-se distantes dos grandes centros – onde as facilidades e serviços de atendimento geralmente encontram-se concentrados.

Quanto ao conforto, comodidade e à segurança providos pela domótica, para uma boa parte da população, não passam de um sonho de consumo inatingível.

No entanto, esta realidade pode ser modificada com soluções que proporcionem tais benefícios, aliando também, um projeto de baixo custo. Dentro deste cenário, este trabalho tem o desafio de apresentar uma solução simples de domótica, que possui baixo custo, com o objetivo de desmistificar a idéia de que a automação residencial não é uma tecnologia palpável.

3 OBJETIVOS

3.1 Objetivo geral

Este trabalho tem como objetivo principal desenvolver um protótipo de sistema de automação residencial, utilizando os conceitos da domótica, capaz de tomar decisões de maneira autônoma, através da leitura de sensores e do controle de cargas elétricas responsáveis pelo acionamento de dispositivos de iluminação e ventilação.

Através deste protótipo, deseja-se mostrar que é possível construir soluções de baixo custo que permitam a um número maior de pessoas se beneficiar das vantagens e conveniências proporcionadas pela domótica.

3.2 Objetivos específicos

Os objetivos específicos deste trabalho são:

- Desenvolver o esquema elétrico do *hardware* para monitoração de sensores,
- Desenvolver o esquema elétrico do *hardware* de acionamento de dispositivos,
- Construir protótipos das placas de circuito impresso responsáveis pelas atividades de monitoramento (de sensores) e acionamento (de dispositivos),
- Implementar as funcionalidades de comunicação entre o computador e o *hardware* através de interface paralela,
- Projetar e implementar o *software* de gerenciamento, e
- Construir uma maquete de demonstração.

4 REVISÃO DE LITERATURA

Neste capítulo serão abordados os conceitos que foram fundamentais para o desenvolvimento do projeto “Casa Inteligente – Um Protótipo de Sistema de Automação Residencial de Baixo Custo”.

4.1 Domótica

O termo domótica é originado da junção das palavras *Domus*, que em latim significa casa, e robótica, que representa uma tecnologia capaz de controlar todos os ambientes de uma residência através de um só equipamento, incluindo temperatura, luminosidade, som, segurança, dentre outros, ou seja, automação residencial (BOLZANI, 2004; FERREIRA, 2008; SGARBI, 2007).

Segundo Brugnera (2008), “a domótica é um recurso utilizado para controle de um ou mais aparelhos eletrônicos por meio de uma central computadorizada”.

Domótica é um processo ou sistema que prioriza a melhoria do estilo de vida (das pessoas), do conforto, da segurança e da economia da residência, através de um controle centralizado das funções desta, como água, luz, telefone e sistemas de segurança, entre outros (ANGEL, 1993 e NUNES, 2002).

Para corresponder as exigências, a domótica faz uso de vários equipamentos distribuídos pela residência de acordo com as necessidades dos moradores. Estes equipamentos podem ser divididos em três principais grupos (TAKIUCHI *et al*, 2004):

- Atuadores: controlam os aparelhos da residência como, por exemplo, luz e ventilador,
- Sensores: capturam informações do ambiente como, por exemplo, luminosidade, umidade e presença,
- Controladores: são responsáveis pela administração dos atuadores e sensores, ou seja, coordenam todos os aparelhos e equipamentos da residência que fazem parte da automação.

A Figura 2 apresenta o esquemático da relação entre os principais grupos de equipamentos.

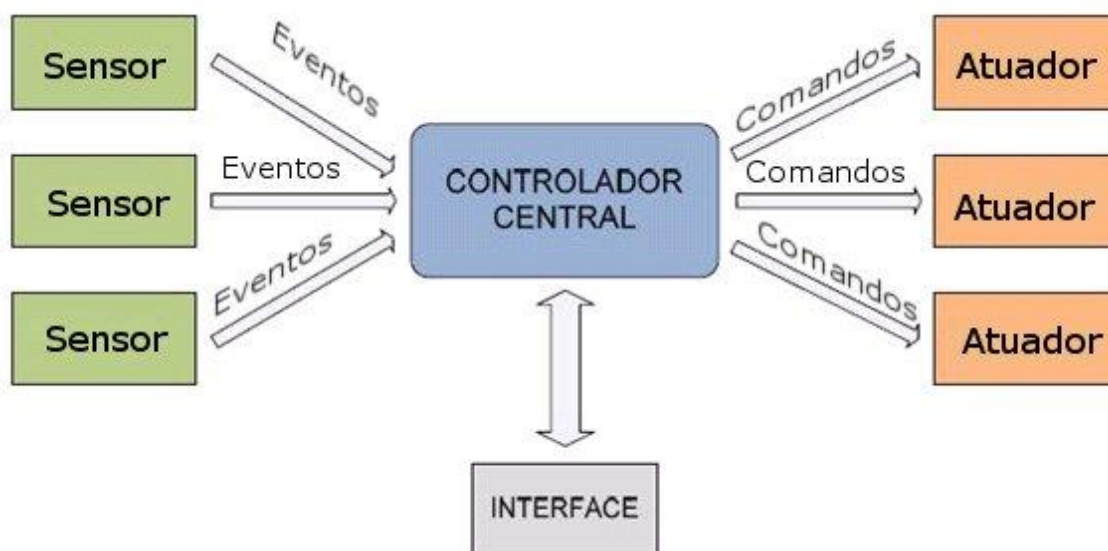


Figura 2: Arquitetura centralizada dos equipamentos. Fonte: FERREIRA (2008).

Segundo Angel (1993), os projetos de domótica podem ser divididos em três tipos de acordo com o nível de integração e complexidade do sistema. Os três tipos são:

- Sistemas autônomos: cada cômodo possui um módulo do sistema que é independente dos demais, tendo o seu controlador no próprio local,
- Sistemas integrados com controle centralizado: existe apenas um controlador para todos os cômodos da residência que estão incluídos na automação,

- Sistemas de automação complexos: nível total de automação na residência, trazendo assim um alto grau de complexidade e a necessidade da residência ser projetada com o intuito de ser totalmente automatizada.

Com a diminuição dos custos de equipamentos como computadores pessoais e componentes eletrônicos utilizados para a fabricação de *hardwares*, bem como com o advento da *Internet* e com o avanço tecnológico utilizado para o desenvolvimento de *softwares*, tornou-se inevitável o surgimento da automação residencial que, inicialmente, foi uma adaptação da automação industrial a residências que, devido às visíveis diferenças entre um ambiente residencial e um industrial, veio a tornar-se uma nova linha de pesquisa e investimentos (BOLZANI, 2004).

A grande guinada da domótica foi após o surgimento e aprimoramento de dispositivos como os microprocessadores, relés e sensores, pois todas as áreas em que a automação estava presente sofreram significativas mudanças quanto à qualidade dos equipamentos, principalmente a área da automação residencial, uma vez que os novos equipamentos não exigiam grandes espaços reservados, passaram a ser capazes de interagir com outros equipamentos e, talvez o mais importante, não precisavam de manutenção constante de técnicos (BOLZANI, 2004).

Atualmente, as pesquisas no setor de automação, incluindo a domótica, tendem para a área da inteligência artificial, visando acrescentar às residências a capacidade de “aprender” com os seus moradores e de se auto-configurar para proporcionar um maior conforto, segurança e praticidade (BOLZANI, 2004).

A automação residencial propõe uma alteração da infra-estrutura da residência para centralizar os diversos tipos de serviços e de dispositivos que executam tarefas em um único equipamento, o integrador. A Figura 3 mostra demonstra como deve ser a estrutura física para que os serviços e equipamentos trabalhem em conjunto (BOLZANI, 2004).

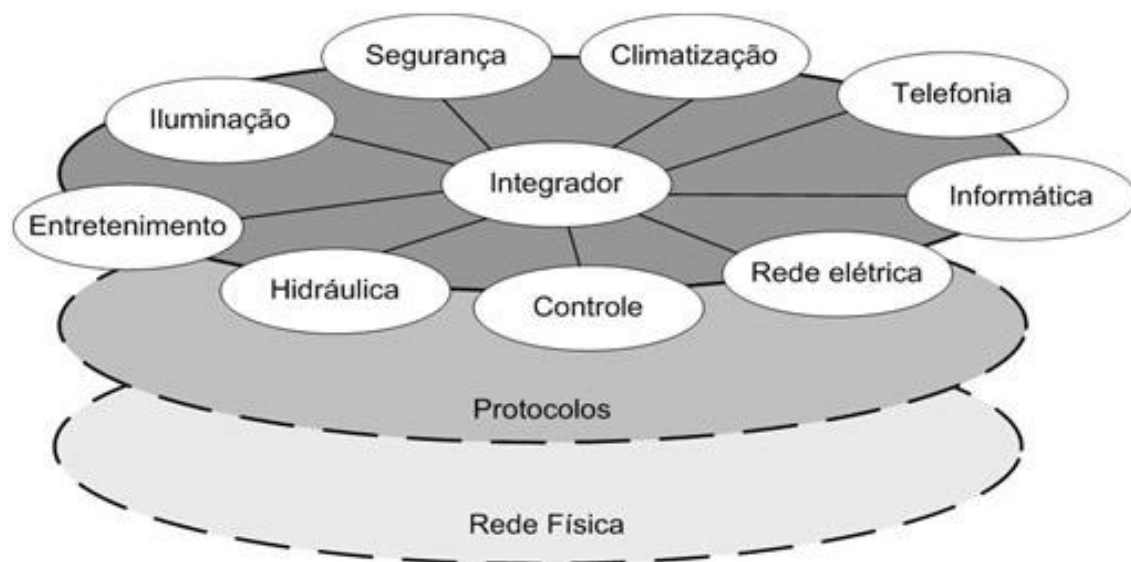


Figura 3: Representação da infra-estrutura de uma residência automatizada. Fonte: BOLZANI (2004).

Essa proposta da infra-estrutura residencial trará vantagens à residência que poderão ser percebidas após uma comparação entre os quadros 01 e 02 que mostram a situação atual e posterior, respectivamente, das residências em relação as mudanças na infra-estrutura.

Quadro 1: Situação atual da infra-estrutura das residências (BOLZANI, 2004).

Situação atual	Conseqüências
Instalações independentes	Multiplicidade de redes e cabos.
Redes não compatíveis	Manutenção cara e complicada, dependência do fornecedor.
Falta de uniformidade	Impossibilidade de automatização global.
Equipamentos limitados	Dificuldade para integrar novos serviços e interligar redes, ampliação do uso de “adaptadores”, obsolescência em curto prazo.

Quadro 2: Situação proposta para a infra-estrutura das residências (BOLZANI, 2004).

Situação proposta	Conseqüências
Automação de residências	Maior conforto e automatização de serviços.
Integração dos serviços	Barateamento de equipamentos e processos.
Centralização de sistemas	Simplificação da rede.
Conexão com redes externas (<i>Internet</i> e dedicadas)	Comando remoto, utilização de conteúdo digital sob demanda.
Monitoramento remoto de pessoas e equipamentos	Facilidade de integração de novos equipamentos e serviços, rapidez no envio de alarmes, <i>homecare</i> .
Eletrodomésticos inteligentes	Acesso à informação de qualquer ponto da casa, diminuição do tempo de procura de avarias, economia de energia.
Auditoria e controle de gastos	Melhoria no funcionamento de sistemas, administração da residência, constante supervisão do conjunto.

Segundo Angel (1993), a domótica oferece uma maior satisfação em relação ao conforto, segurança e outras necessidades já mencionadas anteriormente através das funções domóticas, que podem ser divididas em três principais grupos de acordo com o tipo de serviço. São eles:

- Função de gestão: é responsável pela automação de eventos sistemáticos que são pre-programados pelo usuário,
- Função de controle: é responsável por fornecer ao usuário o poder de atuar sobre os equipamentos e obter informações sobre os mesmos,
- Função de comunicação: é responsável pela interatividade entre o usuário, o sistema e o ambiente.

Assim como toda nova tecnologia, a domótica também encontra dificuldades para ser difundida. A seguir, algumas dificuldades estruturais e conceituais serão analisadas.

Segundo Bolzani (2004), um dos primeiros problemas encontrados no planejamento de uma residência automatizada é o local no qual serão acomodados os equipamentos necessários para o controle da residência, a saber:

- O Centralizador: é um computador dedicado e com cabeamento suficiente, para monitorar e enviar comandos a todos os equipamentos da residência, como por exemplo, as luzes, climatizadores e telefone,
- A área suficiente para a acomodação desse equipamento seria em torno de um metro quadrado. Contudo, apesar de ser um pequeno espaço, nas condições atuais, várias residências não o possuem. E para este problema, uma solução é a utilização de *racks* para os equipamentos e de caixas embutidas para o cabeamento necessário,
- Outra dificuldade estrutural encontrada é o fato de que nem sempre o planejamento da automação residencial pode ser feito juntamente com a construção da residência, acarretando assim a necessidade de alteração na estrutura física do local, como, por exemplo, a modificação de quadros embutidos, alargamento de vias de cabeamento e inserção de novos equipamentos.

A necessidade de uma reforma estrutural numa residência, com certeza, gera gastos que, para uma maioria, serão considerados muito altos. Talvez, o alto custo seja considerado a maior dificuldade enfrentada pela domótica (BOLZANI, 2004).

Outro grande problema enfrentado pela domótica é a troca de informações entre equipamentos de diferentes marcas que pode ser resolvido utilizando apenas equipamentos de uma mesma marca ou que utilizem o mesmo padrão de comunicação (DAAMEN, 2005).

Em Miori *et al* (2006), apresenta-se uma proposta de um padrão aberto para a interoperabilidade de sistema de automação residencial baseado no paradigma computacional orientado a ser serviços. No entanto, devido à multiplicidade de soluções e seus provedores, acredita-se que a padronização ainda irá requerer alguns anos.

4.2 Plataforma .NET

A plataforma .NET é o principal componente da iniciativa Microsoft .NET, pois é a responsável pelo gerenciamento e pela execução dos aplicativos .NET, além de garantir outros recursos como segurança, serviços *Web* e o tratamento de exceções. A plataforma .NET é formada por vários componentes e características que serão descritos a seguir para uma melhor compreensão das vantagens e do funcionamento da plataforma (MSDN, 2008b).

Atualmente na versão 3.5, a plataforma .NET possui os seus detalhes descritos na CLS (*Common Language Specification*). Detalhes esses que especificam o esquema da estrutura da plataforma .NET, tornando, assim, possível a portabilidade da mesma, uma vez que, seguindo as especificações, é possível implementá-la para o sistema operacional alvo (MSDN, 2008a).

Outra parte importante da plataforma .NET é a CLR (*Common Language Runtime*), componente responsável pela execução dos programas a partir de um arquivo do tipo PE (*Portable Executable*) e por alguns serviços que serão mencionados mais adiante. Os arquivos do tipo PE mencionados são compostos pelo *Managed Code* que será analisado a seguir (MSDN, 2008b).

O *Managed Code* é um código gerenciado que é desenvolvido a partir de um compilador de linguagens, cujo objetivo é a execução do programa através da CLR. O *Managed Code* é composto por dois principais elementos: a MSIL (*Microsoft Intermediate Language*) e os Metadados, que, para fins de entendimento do processo de execução do aplicativo .NET, serão melhor definidos a seguir (MSDN, 2008b).

MSIL é o resultado da compilação do código fonte, escrito em qualquer linguagem suportada pela plataforma .Net, numa linguagem intermediária que independe do conjunto de instruções da arquitetura de computador subjacente que irá executar o programa. Em outras palavras, a MSIL define o conjunto de instruções de uma máquina virtual implementada em *software* e que traduz os programas da plataforma .NET em código nativo. Este componente do *Managed Code* torna possível a interoperabilidade de linguagens, uma das vantagens da plataforma .NET (MSDN, 2008c).

A interoperabilidade de linguagens é uma das grandes vantagens da plataforma .NET, pois possibilita a unificação de partes diferentes do código de um mesmo programa codificadas em linguagens diferentes. Isso se dá devido à existência do MSIL, que permite a desvinculação do programa .NET a uma linguagem de programação. Qualquer linguagem que possa ser compilada em MSIL é chamada de linguagem compatível com .NET (MSDN, 2008d).

O conjunto dos Metadados, segunda parte do *Managed Code*, é responsável por carregar as informações capazes de descrever os elementos definidos no código escrito, independente de qualquer linguagem suportada pela a plataforma .NET. Essas informações são as descrições dos tipos, dos membros de cada tipo, do código escrito, resoluções de chamadas de métodos e de versões diferentes de uma mesma aplicação (MSDN, 2008e).

Diante do que foi definido anteriormente, o processo de execução de um programa .NET pode ser descrito como segue. Um esquema de execução de um programa na plataforma .NET encontra-se ilustrado na Figura 4:

Inicialmente, o programa é compilado em *Managed Code*, que contém as instruções que serão utilizadas pela CLR. Os códigos em MSIL são reunidos pela CLR, e então um compilador da CLR compila os códigos MSIL em código da máquina nativa, criando assim um único aplicativo (MSDN, 2008d).

A execução do *Managed Code* pela CLR pode ser dividida em três principais etapas:

1. Carga de classes: Tarefa realizada pelo carregador de classes (*class loader*) – outro elemento constituinte da plataforma .NET. Seu propósito é carregar para a memória o código MSIL e os Metadados necessários para a execução do aplicativo,

2. **Compilação de Instruções:** Etapa em que os Metadados são traduzidos em código de máquina. Esta tarefa é realizada pelo Compilador JIT (*Just-in-Time*), outro componente da plataforma .NET. O resultado final dessa tarefa é suficiente para a máquina poder executar os comandos da MSIL (MSDN, 2008e),
3. **Execução:** Neste momento o código nativo de cada método que foi gerado é executado. É possível que instruções a serem executadas façam chamadas a outros métodos. Caso o sistema de execução detecte que um desses métodos está sendo chamado pela primeira vez, será chamado o compilador JIT que irá gerar o código nativo para o método, e, em seguida, ele será executado diretamente, desde que não haja modificações no código do programa.

Vale ressaltar que a compilação feita durante a segunda etapa do processo é chamada de *Just-in-Time* pelo motivo de que as classes vão sendo compiladas para instruções nativas à medida que seus métodos vão sendo chamados pela primeira vez. Uma vez que a classe já foi compilada para instruções nativas, não é mais necessário re-fazer a compilação (MSDN, 2008e).

Assim como a interoperabilidade de linguagens, outra vantagem que justifica a etapa de compilação da linguagem de programação em *Managed Code*, ao invés de ser compilada diretamente para a linguagem da máquina nativa, é o uso da CLR que libera o desenvolvedor para focar a análise dos aspectos fins de sua aplicação.

Essa vantagem se dá pelo fato de que a CLR é responsável pelos serviços de gerenciamento de memória, gerenciamento de ameaças, manipulação de exceções, coleta de lixo, gerenciamento de processo leves (*threads*) e segurança. Analogamente, tem-se a FCL (*Framework Class Library*), outro integrante da plataforma .NET, que possui muitas classes, interfaces e tipos de valores reutilizáveis (MSDN, 2008c).

No apêndice A, podem ser encontrados os conjuntos de classes e interfaces e os tipos de valores da FCL que foram utilizados no projeto, acompanhados por uma breve descrição.

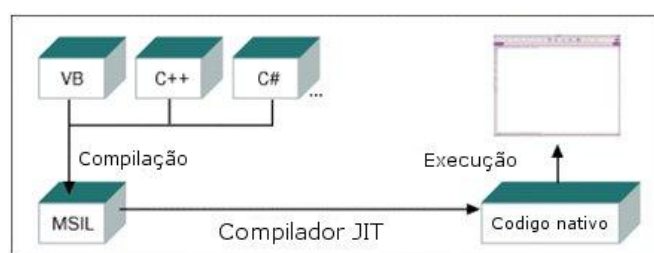


Figura 4: Esquema de execução na plataforma .NET (FINKELSTEIN, 2003)

4.3 Comunicação

Os periféricos utilizados atualmente como *mouses*, teclados e impressoras, necessitam de uma forma de comunicação e de uma sincronia de transmissão de dados entre eles e outro dispositivo, como um microcomputador, para poder funcionar e executar a sua tarefa corretamente.

A sincronia de transmissão de dados entre dois dispositivos é muito importante para que o sinal seja enviado, recebido e interpretado corretamente. Existem duas formas de sincronia de transmissão de dados, que serão analisadas a seguir, são elas: transmissão assíncrona e transmissão síncrona de dados.

Segundo Silveira (1991) e Monteiro (2001), a transmissão assíncrona é a forma mais simples de ser projetada e se caracteriza pela irregularidade entre o tempo de transmissão de sinais entre o emissor e o receptor que é estabelecido, através de um pulso inserido no início da informação, conhecido como *start*, e um pulso no final da informação, conhecido como *stop*, a cada informação enviada e recebida.

Ainda de acordo Silveira (1991) e Monteiro (2001), a transmissão síncrona tem como sua principal característica o envio de informações através de blocos de dados de forma sequencial respeitando uma frequência determinada por dispositivos de controle da comunicação, como exemplo podem ser citados os chips UART (*Universal Asynchronous Receiver Transmitter*) e USART (*Universal Synchronous Asynchronous Receiver Transmitter*).

Ao contrário da comunicação assíncrona, que utiliza pulsos de controle (*start* e *stop*), a transmissão síncrona usa grupos de *bits* no início e no final dos blocos de dados para identificar o começo e o fim de cada um.

Existem basicamente três interfaces de comunicação entre dois dispositivos, também chamadas de portas paralela, serial e USB.

A porta utilizada foi a paralela devido a sua simplicidade de operação que irá reduzir a complexidade do projeto e alta velocidade na transmissão de dados.

4.3.1 Porta Serial

A porta serial, uma das mais antigas e básicas interfaces de comunicação de computadores, integra a maioria dos computadores atuais e é muito utilizada para *mouses* e PDAs. Contudo, está perdendo o seu lugar para as portas USB (*Universal Serial Bus*) (TYSON, 2009a).

O nome serial é devido ao fato de que os dados são serializados, ou seja, os dados são enviados e recebidos a um *bit* por vez, fato que determina suas principais características que foram levadas em consideração no momento de escolha da interface de comunicação. São elas, a necessidade de apenas um fio para a transmissão de dados e uma velocidade menor do que a da interface paralela que pode enviar oito *bits* por vez (TYSON, 2009a).

4.3.2 Porta Paralela

Esta interface, que utiliza o modo de transmissão síncrona, tem como característica principal a alta velocidade de transmissão de dados, quando comparada à interface serial, devido ao fato de transmitir vários *bits* paralelamente. É utilizada por diversos tipos de equipamentos, entre os mais comumente utilizados citam-se impressoras, *scanners* e controladores de jogos (MESSIAS, 2006).

Os sistemas operacionais da família *Windows* reconhecem como *LPT*i** (*Line Printer Terminal*) a identificação padrão para portas paralelas, em que *i* representa um identificador numérico com valores entre **1** e *n*, inclusive, desde que existam *n* portas paralelas (*LPT1, LPT2, ..., LPTn*).

Segundo Messias (2006), as portas paralelas possuem três registradores: (*i*) *Data lines* ou linhas de dados (registrador utilizado para o envio de dados, tem por padrão o endereço 378h), (*ii*) *Control lines* ou linhas de controle (registrador utilizado para controlar os dispositivos conectados à porta paralela, tem por padrão o endereço 37Ah) e (*iii*) *Status lines* ou linhas de status (registrador utilizado para a recepção de dados, tem por padrão o endereço 379h).

Segundo Messias (2006), outra característica importante da interface paralela é que ela possui duas formas de operação. São elas:

- SPP (*Standard Parallel Port*): modo unidirecional com taxa de transferência de até 150 Kb/s (kilobits por segundo).
- EPP (*Enhanced Parallel Port*): modo bidirecional com taxa de transferência de até 2 Mb/s (megabits por segundo).

O padrão utilizado neste projeto foi o EPP, pois permite comunicação bidirecional, possibilitando a leitura e escrita simultânea dos registradores da porta paralela.

4.3.3 Barramento e Interfaces USB

Presente em praticamente todos os computadores atuais, a porta USB vem ganhando cada vez mais espaço no mercado e utilidades novas como conexões com impressoras, *scanners* e teclado (TYSON, 2009b).

Devido a sua simplicidade e capacidade de resolver problemas de comunicação, a interface USB está sendo adotada como padrão para todos os novos equipamentos desenvolvidos (TYSON, 2009b).

A inclusão desta seção dedicada ao padrão USB se deve à tendência da substituição das portas serial e paralela por interfaces USB. Em diversos computadores, notadamente nos *notebooks*, já não são encontradas as portas seriais. Acredita-se que o mesmo venha a ocorrer brevemente com a interface paralela. Para ambos os casos, existem adaptadores USB – Serial e USB – Paralelo, que permitem a comunicação com periféricos que disponham apenas destes tipos de interface. Portanto, uma das extensões deste trabalho seria a inclusão ao suporte USB.

4.4 Padrões de Projeto

Segundo Shalloway e Trott (2004), os padrões de projeto e de programação orientada a objetos sustentam a promessa de tornar mais fácil o trabalho do projetista e do desenvolvedor, pois os padrões de projeto propõem aos projetistas e aos desenvolvedores de *software* conjuntos de pares “problema-solução” que

comprovadamente solucionam problemas recorrentes de complexidade elevada em diferentes domínios da aplicação.

Ainda de acordo com Shalloway e Trott (2004), os padrões de projeto estão baseados no princípio da reutilização de idéias, não somente de código, proporcionando uma maior condição de gerar uma aplicação mais robusta e confiável, de maior simplicidade de manutenção, pois, ao padronizar soluções, reduz-se a complexidade do sistema e cria-se um vocabulário comum entre os projetistas e desenvolvedores, acarretando assim em uma maior integração entre a equipe, além de diminuir o tempo de aprendizado de uma plataforma de desenvolvimento.

Nas próximas seções são apresentados, em ordem alfabética, os padrões de projeto utilizados no protótipo de casa inteligente implementado neste projeto.

4.4.1 *Abstract Factory*

De acordo com Gamma *et al* (2005) e Buschmann *et al* (1996), a intenção do padrão *Abstract Factory* é “fornecer uma interface para criar famílias de objetos relacionados ou dependentes, sem especificar suas classes concretas”.

Ainda de acordo com Gamma *et al* (2005) e Buschmann *et al* (1996), algumas vezes, diversos objetos necessitam ser instanciados de uma forma coordenada. Por exemplo, ao lidar com interfaces de usuário, o sistema pode precisar utilizar um conjunto de objetos para trabalhar com um determinado sistema operacional e outro para trabalhar com um sistema operacional diferente. O padrão *Abstract Factory* garante que o sistema sempre obtenha os objetos corretos para cada situação.

O *Abstract Factory* é utilizado nas ocasiões em que se deve coordenar a criação de famílias de objetos. Isso abre espaço para remover as regras relacionadas ao modo como efetuar a instanciação para fora do objeto cliente que está usando esses objetos criados (GAMMA *et al*, 2005).

- Primeiro, identificar as regras para instanciação e definir uma classe abstrata com uma interface que tenha um método para cada objeto que necessita ser instanciado,
- Então, implementar classes concretas, a partir dessa classe, para cada família,

- O objeto cliente utiliza esse objeto fábrica para criar os objetos servidores de que ele necessita.

Neste projeto, o padrão *Abstract Factory* foi utilizado para que classes de uma camada possam instanciar objetos da camada diretamente abaixo, gerando assim total independência entre estas. Assim sendo, classes da camada de negócio utilizam o padrão *Abstract Factory* para criar objetos da camada de acesso a dados (DAL) que encapsulam os detalhes de como e onde é feito o acesso.

4.4.2 *Facade*

De acordo com Gamma *et al* (2005) e Buschmann *et al* (1996), a intenção do padrão *Facade* é a seguinte: “Fornecer uma interface unificada para um conjunto de interfaces de um subsistema. O *Facade* define uma interface de mais alto nível, tornando assim mais fácil utilizar o subsistema”.

Ainda de acordo com Gamma *et al* (2005) e Buschmann *et al* (1996), *Facades* podem ser utilizados não somente para criar uma interface mais simples em termos de chamadas a métodos, mas também para reduzir o número de objetos com os quais o objeto cliente deve lidar.

Por exemplo, supondo que um objeto Cliente deve lidar com uma base dados, modelos e elementos. Este objeto (cliente) deve primeiro abrir a base de dados e obter um modelo, que é então utilizado para obter um elemento. Finalmente, solicita informação ao elemento. Seria muito mais fácil utilizar uma *Facade*-Base-Dados que poderia ser consultado pelo Cliente (BUSCHMANN *et al*, 1996).

A Figura 5 apresenta um exemplo de como o padrão *Facade* pode ser utilizado.

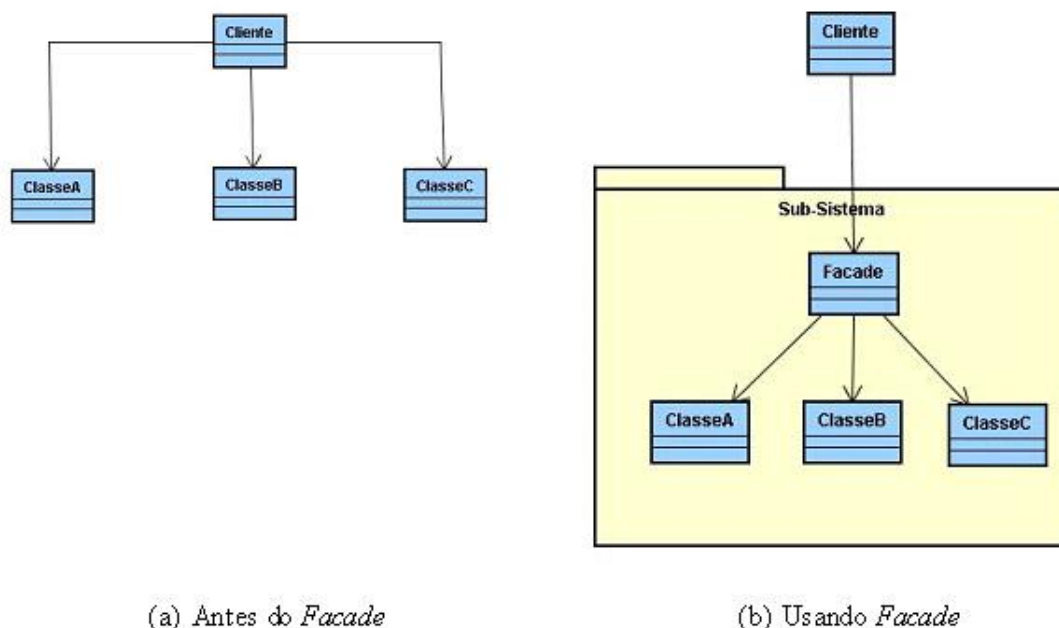


Figura 5: Resultado da aplicação do padrão *Facade*.

Dessa forma, a *Facade* tem a intenção de fornecer uma interface unificada para um conjunto de interfaces em um subsistema. *Facade* conceitua-se como uma interface de nível mais alto que torna o subsistema mais fácil de ser usado (BUSCHMANN *et al*, 1996).

Neste projeto, o padrão *Facade* foi utilizado para que na camada de apresentação, uma página possa acessar vários subsistemas das camadas inferiores sem a necessidade de instanciar vários objetos, gerando uma maior simplicidade para a camada de apresentação.

4.4.3 Padrão Camadas

De acordo com Deitel *et al* (2006), Horstmann (2007) e Gamma *et al* (2005), a proposta do padrão camadas é dividir a funcionalidade do sistema em camadas separadas.

O padrão arquitetural em camadas é comumente utilizado em sistemas computacionais e baseia-se nos princípios de dividir para conquistar e abstração, através do qual uma camada não conhece detalhes internos das camadas adjacentes, as quais são coesas, possuem grau de acoplamento fraco entre si e ocultam detalhes de suas implementações.

O padrão arquitetural prevê a organização de um sistema num número qualquer de camadas (padrão n-camadas), dependendo na natureza do problema e complexidade do sistema. No entanto, diversas aplicações encontram-se estruturadas em apenas três camadas – apresentação, negócios e acesso a dados, conforme ilustrado na Figura 6, que representa as disposições das camadas e suas interações.

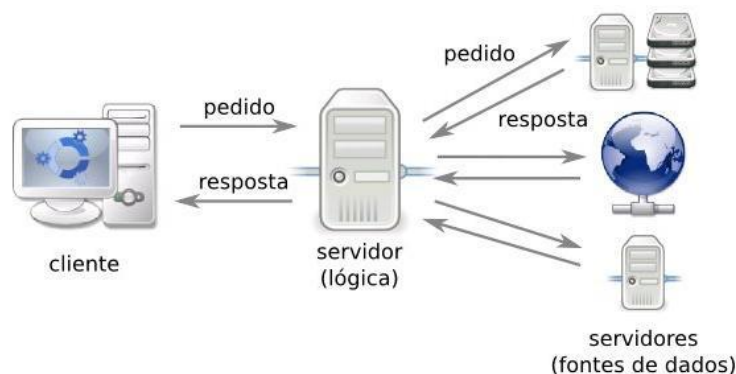


Figura 6: Arquitetura em três camadas. Fonte: Maziero (2008).

Nesta ilustração, a camada de negócio está representada pelo servidor que hospeda a aplicação (lógica), a camada de acesso a dados está representada pelo servidor que detém a fonte de dados e a camada de apresentação pelo cliente (navegador *WEB*).

Seguindo este modelo, o protótipo implementado utiliza este padrão arquitetural para organizar o sistema nas seguintes camadas:

- *DAL (Data Access Layer)*: responsável pela persistência dos dados da aplicação,
- *Business*: responsável pela regra de negócio da aplicação,
- *Apresentação*: responsável por fornecer uma interface para interação do usuário com o sistema.

5 Construção do Protótipo

Nas próximas seções são apresentados os aspectos referentes à concepção do protótipo organizado pelos tópicos do *software* de gerenciamento (seção 5.1), da

metodologia de testes utilizada (seção 5.2), do *hardware* de controle (seção 5.3) e da maquete construída para fins de demonstração (seção 5.4).

5.1 O *Software*

Siqueira Filho e Silva Filho (2006) definem o *software* como sendo toda a parte lógica do computador, são conjuntos de instruções relacionadas e não-ambíguas que determinam a realização de tarefas específicas.

5.1.1 Ferramentas utilizadas

A ferramenta escolhida para o desenvolvimento do software foi o ambiente de desenvolvimento integrado (IDE – *Integrated Development Environment*) *Visual Studio 2005*. As principais razões para a escolha deste ambiente são a produtividade e características como atualizações automáticas e compatibilidade com um conjunto amplo de linguagens, que tornam o ambiente uma das ferramentas de desenvolvimento de *software* mais modernas.

O *software* proposto no projeto foi implementado sobre plataforma *Microsoft .NET* versão 2.0, em linguagem C# versão 3.0 e utilizando a ferramenta *Microsoft SQL Server 2005 Express Edition* para gerenciar o banco de dados.

Conforme mencionado anteriormente, a plataforma .NET possui uma linguagem intermediária (MSIL – *Microsoft Intermediary Language*) que implementa os tipos e operações comuns em diversas linguagens (MICROSOFT, 2007).

Desta forma, qualquer compilador desenvolvido sobre esta linguagem intermediária torna seu código completamente portátil a qualquer outra linguagem compatível com a MSIL. Atualmente existem implementações de cerca 42 linguagens, incluindo C++, C#, VB.NET, J#, Delphi, COBOL, Python, Haskell, Perl, SmallTalk e Eiffel. Adicionalmente, a plataforma .NET têm por objetivos melhorar os sistemas operacionais, os modelos de componente COM+ do desenvolvedor e proporcionar a publicação e/ou acesso do sistema através da Internet de forma independente da linguagem de programação (PLATT, 2003).

O *Microsoft SQL Server 2005 Express Edition (SQL Server Express)* é uma ferramenta de gerenciamento de dados poderoso e confiável que fornece recursos robustos, proteção de dados e desempenho para clientes de aplicativos incorporados, aplicativos *WEB* simples e armazenamentos de dados locais. Criado para ser um protótipo rápido e de fácil implantação, o *SQL Server Express* está disponível gratuitamente e pode ser redistribuído com outros aplicativos. O *SQL Server Express* foi criado para se integrar perfeitamente com outros investimentos em infra-estrutura de servidor (MICROSOFT, 2008).

5.1.2 Requisitos

O *software* foi desenvolvido baseado em requisitos funcionais que estão descritos a seguir.

- Manter controladores: permite a inclusão, alteração, consulta e exclusão de controladores. Para cada controlador, devem ser cadastradas as seguintes informações: nome, endereço físico, número de portas e o tipo (placa de relés ou placa de sensores),
- Manter equipamentos: permite a inclusão, alteração, consulta e exclusão de equipamentos. Deve conter as seguintes informações: nome do equipamento, controlador a que está associado e a porta do na qual foi instalado,
- Manter comandos: permite a inclusão, alteração, consulta e exclusão de comandos dos equipamentos. Deve conter o nome do comando,
- Associar eventos e comandos: permite a inclusão, consulta e exclusão de associações entre eventos e comandos. Deve conter as seguintes informações: o sensor que disparará o evento e o comando do equipamento que será executado,
- Associar comandos e ações: permitir a inclusão, consulta e exclusão de associações entre comandos emitidos pelo *software* e ações do dispositivo. Deve conter as seguintes informações: o equipamento que receberá o comando, o comando que será executado e o valor lógico do comando – VERDADEIRO, que aciona uma carga elétrica, ligando um dispositivo, ou FALSO, que interrompe uma carga elétrica, desligando o dispositivo,

- Executar comandos,
- Monitorar os equipamentos associados às placas de sensores, e
- Capturar eventos dos sensores.

Os requisitos não funcionais abaixo também foram considerados para o desenvolvimento do *software*.

- Executar em plataforma *WEB*, e
- Ser acessado através de PDAs (*Personal Device Assistants*) e celulares com capacidade de carga de páginas *WEB* simples.

5.1.3 Modelagem

O *software* teve seus diagramas de caso de uso e de classe especificados através da linguagem UML (*Unified Modeling Language*).

5.1.3.1 Diagramas de casos de uso

O sistema pode ser dividido conceitualmente em dois aplicativos, o de configuração do sistema que é responsável pela interação do usuário com o sistema e o de monitoração que é responsável por monitorar os sensores e disparar eventos, contudo, os dois aplicativos compartilham das mesmas classes, métodos e pacotes.

O aplicativo de configuração tem cinco casos de uso.

- Manter controladores: responsável pela manutenção dos controladores, placas de relés e de sensores, que controlam os equipamentos,
- Manter equipamentos: responsável pela manutenção dos equipamentos da residência como luz e ventilador,
- Manter comandos: responsável pela manutenção dos comandos que serão associados aos equipamentos,
- Manter comandos do equipamento: responsável por associar comandos a equipamentos e executar comandos, e

- Manter eventos do equipamento: responsável por associar eventos do equipamento a comandos do equipamento.

O serviço de monitoração possui dois casos de uso.

- Monitorar equipamentos: responsável por monitorar os equipamentos do tipo sensor, e
- Disparar eventos do equipamento: responsável por disparar os eventos do equipamento.

A Figura 7 representa o diagrama de casos de uso do sistema citados anteriormente.

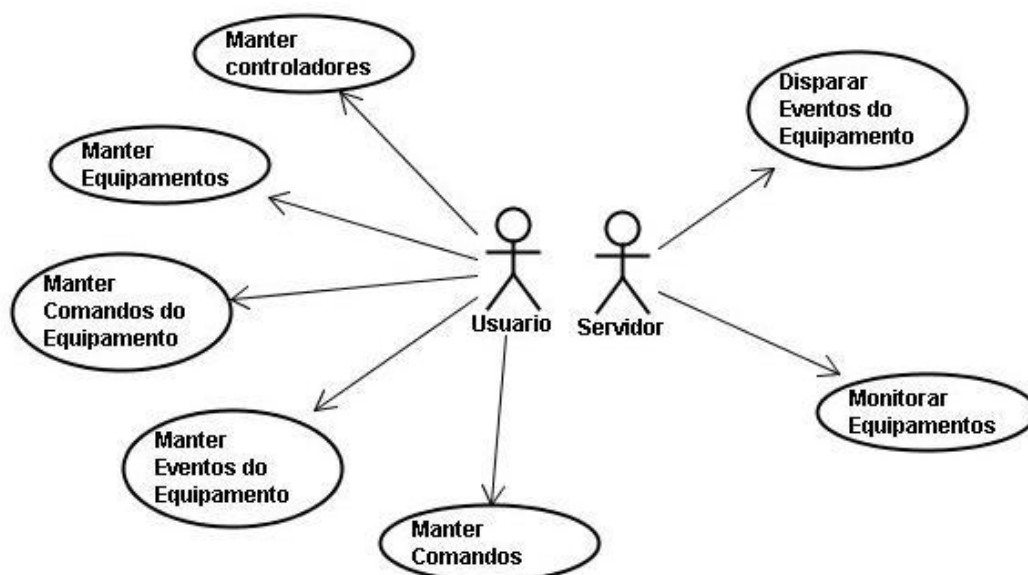


Figura 7: Diagrama de casos de uso do sistema.

5.1.3.2 Diagramas de classes

O software foi implementado utilizando uma arquitetura em três camadas que são analisadas a seguir:

- *DAL*: camada utilizada para agrupar classes e interfaces responsáveis pela persistência dos dados do sistema. Composta por oito classes concretas e seis interfaces,
- *Business*: camada utilizada para agrupar classes e interfaces responsáveis pela regra de negócio do sistema. Composta por oito classes concretas e oito interfaces,

- **Apresentação:** camada utilizada para permitir a interação do usuário com o sistema é composta pelas telas de configuração do sistema que podem ser acessadas pelo usuário.

Além das três camadas citadas, foi implementado o pacote *Common* que agrupa classes que definem as entidades utilizadas no protótipo e que podem ser instanciadas em qualquer camada do sistema. Composta por sete classes.

Encontra-se ilustrado na Figura 8 o relacionamento entre os pacotes da aplicação, onde as setas indicam de quais pacotes a poderá ser instanciado objetos.

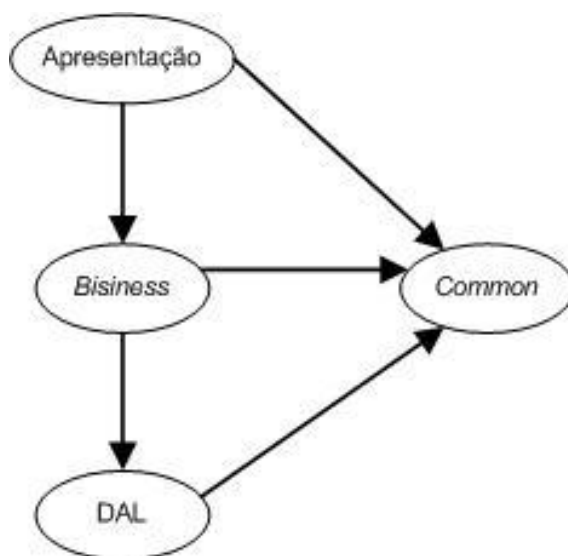


Figura 8: Relacionamentos entre as camadas do sistema.

A Figura 9 apresenta o diagrama de classes do pacote *Common*.

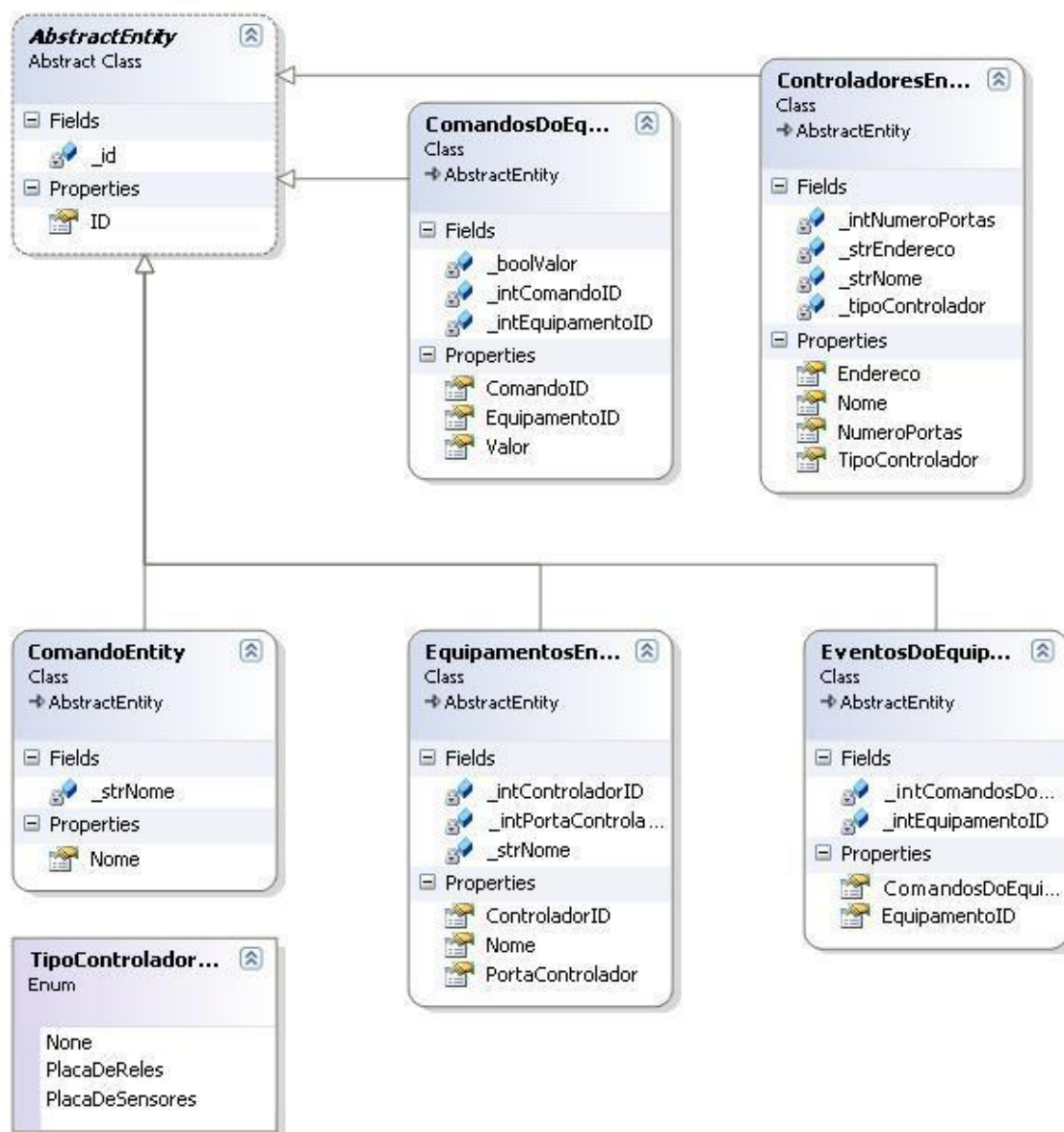


Figura 9: Diagrama de classes do pacote Common.

O Quadro 3 apresenta uma descrição das classes do pacote *Common*.

Quadro 3: Classes do pacote Common.

Interface	Propósito	Super Classe
AbstractEntity	A classe representa uma entidade abstrata, da qual as demais classes poderão herdar suas características que devem ser comuns a todas as entidades. No protótipo, a única característica identificada que deve ser comum em todas as entidades é a propriedade ID que caracteriza um identificador.	
ControladoresEntity	A classe representa um controlador que tem como propriedades um nome, um endereço físico, número de portas e um tipo.	AbstractEntity
ComandoEntity	A classe representa um comando que tem como propriedades apenas o nome.	AbstractEntity

EquipamentoEntity	A classe representa um equipamento que tem como propriedades um nome, uma associação com um controlador e a portado controlador onde está instalado.	AbstractEntity
ComandosDoEquipamentoEntity	A classe representa uma associação de um comando a um equipamento que tem como propriedades uma associação com um comando, uma associação com um equipamento e o valor lógico que o comando representa para o equipamento.	AbstractEntity
EventosDoEquipamentoEntity	A classe representa um evento associado a um comando do equipamento que tem como características um equipamento que deve ser do tipo sensor e o comando do equipamento.	AbstractEntity
TipoControladorEnum	A classe representa um enumerador com os tipos de controladores (placa de relés e de sensores).	

A Figura 10 apresenta o diagrama de interfaces da camada DAL. Uma melhor descrição dos métodos dessa camada poderá ser analisada no apêndice B.

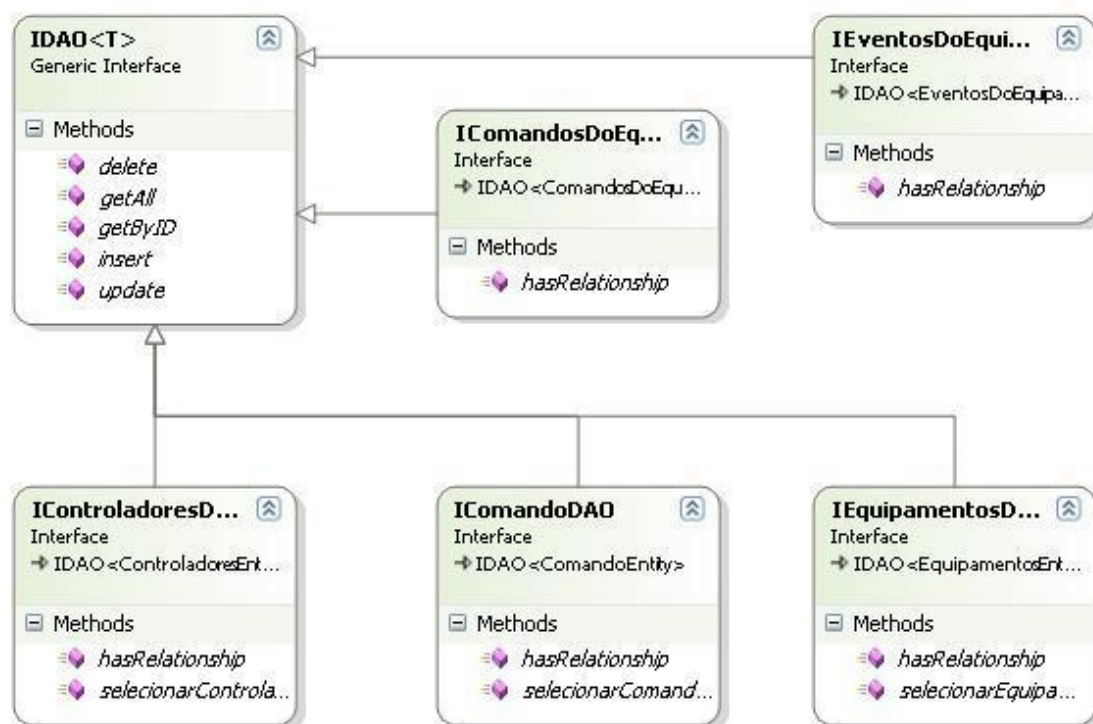


Figura 10: Diagrama de interfaces da camada DAL.

O Quadro 4 apresenta uma descrição das interfaces da camada DAL.

Quadro 4: Interfaces da camada DAL.

Interface	Propósito	Super Classe
IDAO	Interface que serve de super classe para as demais herdarem suas características que são assinaturas de métodos básicos em relação a comandos SQL.	
IControladoresDAO	Interface utilizada para exportar assinaturas dos métodos de acesso a base de dados da entidade Controladores.	IDAO
IComandoDAO	Interface utilizada para exportar assinaturas dos métodos de acesso a base de dados da entidade Comando.	IDAO
IEquipamentoDAO	Interface utilizada para exportar assinaturas de métodos de acesso a base de dados da entidade Equipamento.	IDAO
IComandosDo EquipamntoDAO	Interface utilizada para exportar assinatura de métodos para acesso a base de dados da entidade Comandos do Equipamento	IDAO
IEventosDo EquipamentoDAO	Interface utilizada para exportar assinatura dos métodos de acesso a base de dados da entidade Eventos do Equipamento.	IDAO

A Figura 11 apresenta o diagrama de classes concretas da camada DAL. Uma melhor descrição dos métodos dessa camada poderá ser analisada no apêndice B.

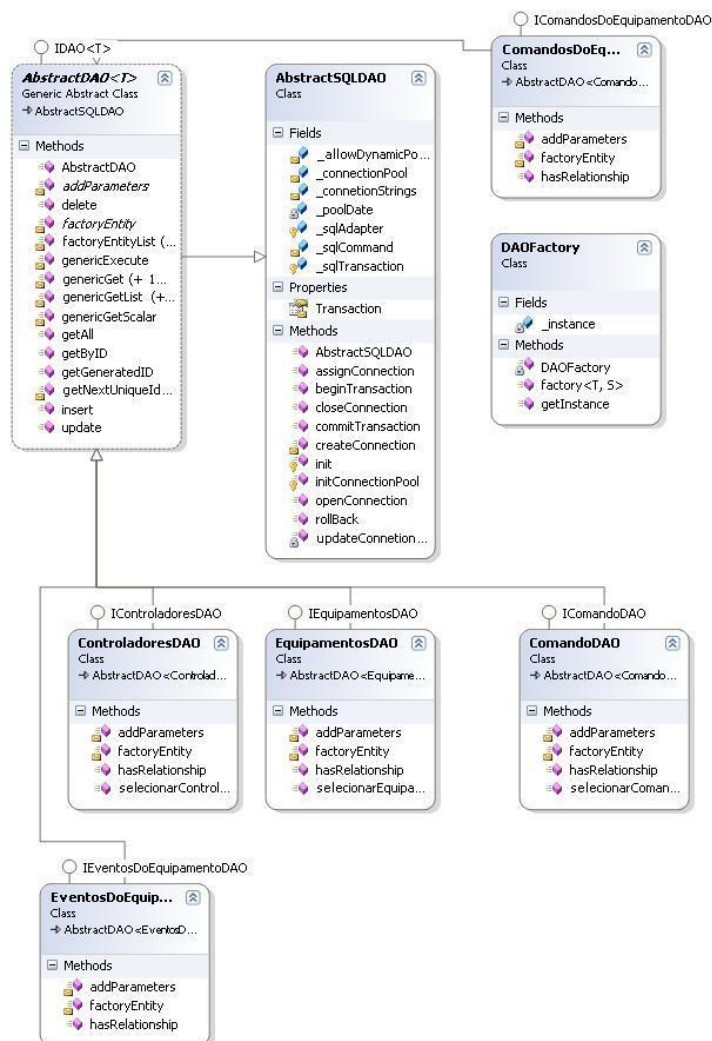


Figura 11: Classes concretas da camada DAL.

O Quadro 5: Classes concretas da camada DAL. apresenta uma descrição das classes concretas da camada DAL.

Quadro 5: Classes concretas da camada DAL.

Interface	Propósito	Super Classe
AbstractSQLDAO	Classe utilizada para encapsular a comunicação do sistema com o <i>Microsoft SQL Server</i> .	
AbstractDAO	Classe utilizada para encapsular a implementação dos métodos mais comuns de acesso a base de dados.	AbstractSQLDAO
ControladoresDAO	Classe responsável pela persistência dos dados da entidade Controlador.	AbstractDAO
ComandoDAO	Classe responsável pela persistência dos dados da entidade Comando.	AbstractDAO
EquipamentoDAO	Classe responsável pela persistência dos dados da entidade Equipamento.	AbstractDAO

ComandosDo EquipametntoDAO	Classe responsável pela persistência dos dados da entidade Comandos do Equipamento.	AbstractDAO
EventosDo EquipamentoDAO	Classe responsável pela persistência dos dados da entidade Eventos do Equipamento.	AbstractDAO
DAOFactory	Classe responsável pela fabricação de objetos para as camadas superiores.	

A Figura 12 apresenta o diagrama das interfaces da camada *Business*. Uma melhor descrição dos métodos dessa camada poderá ser analisada no apêndice C.

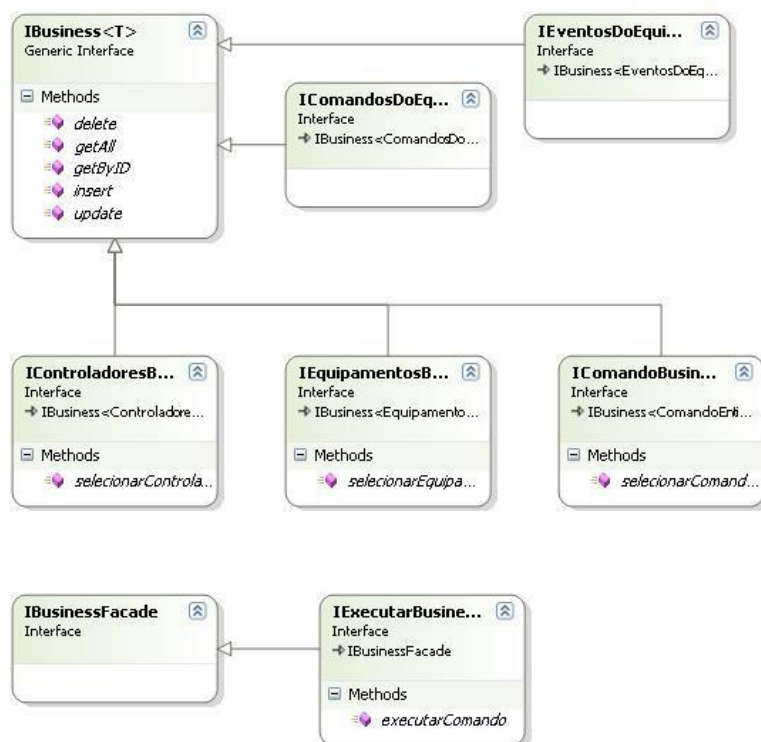


Figura 12: Diagrama de interfaces da camada Business

O Quadro 6: Interfaces da camada Business. apresenta uma descrição das interfaces da camada *Business*.

Quadro 6: Interfaces da camada Business.

Interface	Propósito	Super Classe
IBusiness	Interface que serve de super classe para as demais herdarem suas características que são os métodos básicos da camada de negócio.	
IControladores Business	Interface utilizada para exportar assinaturas dos métodos de regra de negócio da entidade Controlador.	IBusiness
IComando Business	Interface utilizada para exportar assinatura dos métodos de regra de negócio da entidade Comando.	IBusiness
IEquipamento Business	Interface utilizada para exportar assinatura dos métodos de regra de negócio da entidade Equipamento.	IBusiness
IComandosDo Equipametnto Business	Interface utilizada para exportar assinatura de métodos de regra de negócio da entidade Comando do Equipamento.	IBusiness
IEventosDo	Interface utilizada para exportar assinatura de métodos	IBusiness

EquipamentoBusiness	de regra de negócio da entidade Evento do Equipamento.	
IBusinesFacade	Interface que serve de super classe para as demais herdarem características que são os métodos básicos para uma classe que represente uma fachada.	
IExecutarBusinessFacade	Interface utilizada para exportar os métodos da fachada de execução de comando.	IBusinesFacade

A Figura 13 apresenta um diagrama de classes concretas da camada *Business*. Uma melhor descrição dos métodos dessa camada poderá ser analisada no apêndice C.

O Quadro 7: Classes concretas da camada *Business*. apresenta uma descrição das classes concretas da camada *Business*.

Quadro 7: Classes concretas da camada *Business*.

Interface	Propósito	Super Classe
AbstractBusiness	Classe abstrata que implementa os métodos básicos da camada de regra de negócio.	
ControladoresBusiness	Classe responsável pela regra de negócio da entidade Controlador.	AbstractBusiness
ComandoBusiness	Classe responsável pela regra de negócio da entidade Comando.	AbstractBusiness
EquipamentoBusiness	Classe responsável pela regra de negócio da entidade Equipamento.	AbstractBusiness
ComandosDoEquipamentoBusiness	Classe responsável pela regra de negócio da entidade Comandos do Equipamento.	AbstractBusiness
EventosDoEquipamentoBusiness	Classe responsável pela regra de negócio da entidade Eventos do Equipamento.	AbstractBusiness
AbstractBusinesFacade	Classe abstrata que implementa os métodos básicos de uma classe do que represente uma fachada.	
ExecutarBusinessFacade	Classe concreta, utilizada como uma fachada. É responsável pela execução dos comandos.	AbstractBusinesFacade

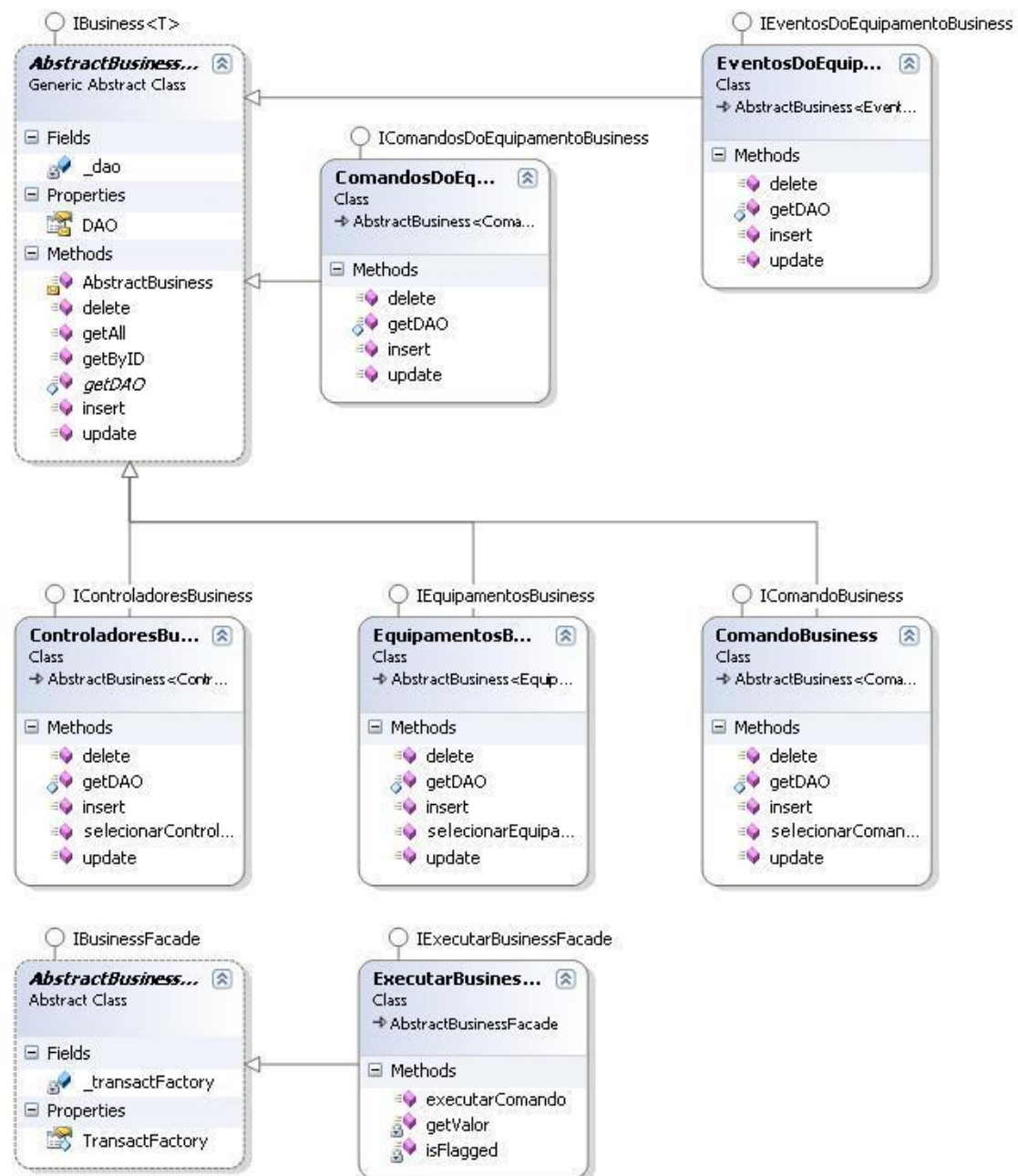


Figura 13: Classes Concretas da camada *Business*.

A camada de apresentação não possui diagrama de classes, pois a camada é composta de telas para a interação do usuário com o sistema. Essas telas serão analisadas a seguir.

5.1.3.3 As Telas da Aplicação

A Figura 14 apresenta a tela de cadastro de controladores que, de acordo com os requisitos, possibilita consultar, inserir, editar e excluir um controlador através de botões e *links* encontrados na tela.



Figura 14: Tela de consulta de controladores.

A Figura 15 apresenta a tela de cadastro de comandos que, de acordo com os requisitos, possibilita consultar, inserir, editar e excluir um comando através de botões e *links* encontrados na tela.

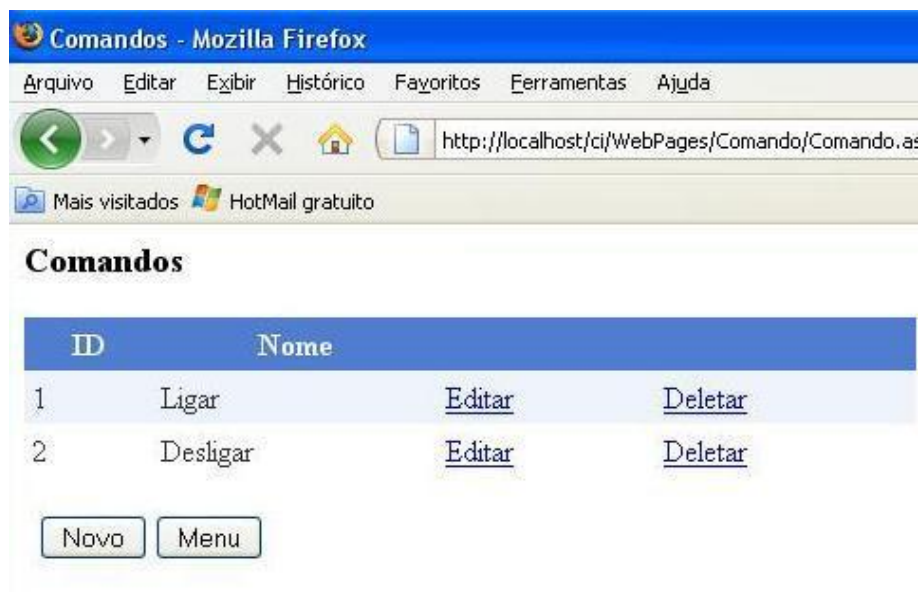
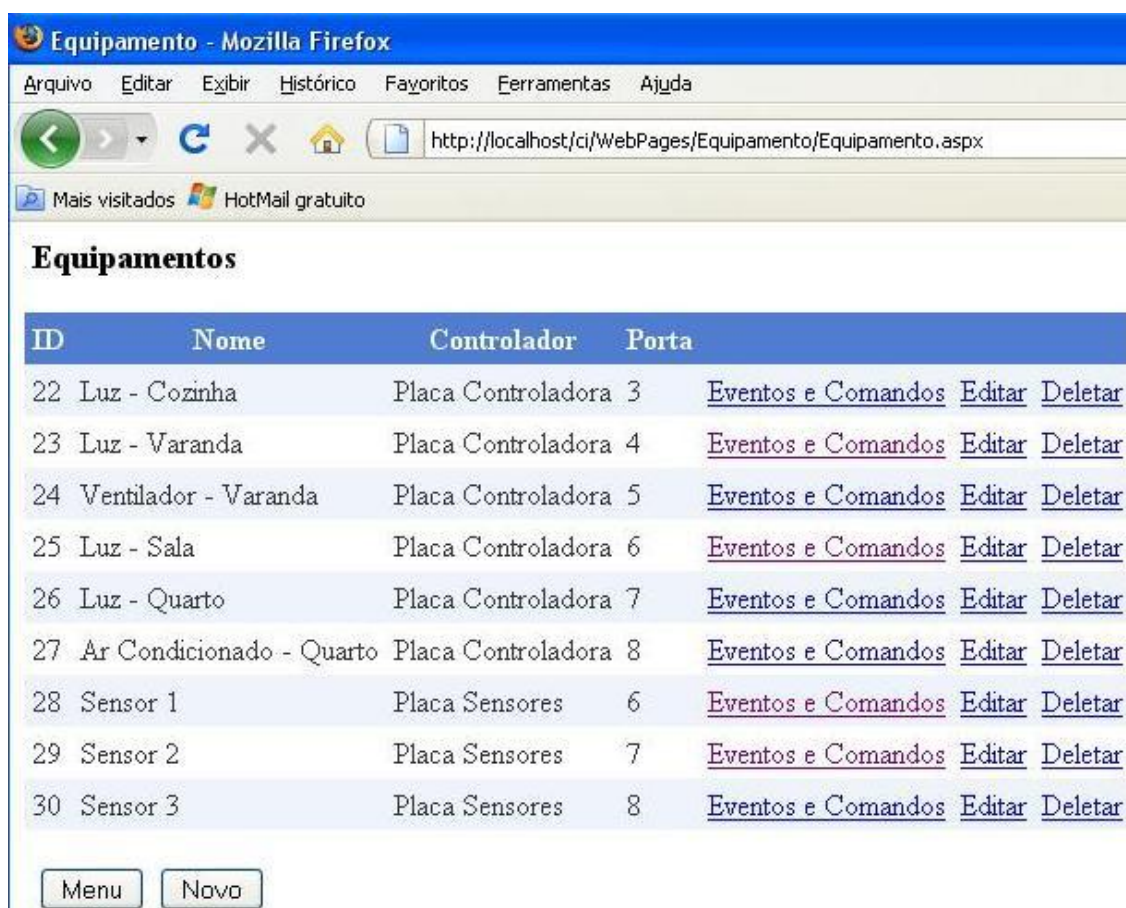


Figura 15: Tela de consulta de comandos.

A Figura 16 apresenta a tela de cadastro de equipamentos que, de acordo com os requisitos, possibilita, através de botões e *links* encontrados na tela, consultar, inserir, editar, excluir e direcionar a aplicação para as telas de associação de comandos e eventos, dependendo do tipo de equipamento. Caso o equipamento seja um sensor, a aplicação será direcionada para a tela de associação de eventos do equipamento, caso seja um atuador, a aplicação será direcionada para a tela de associação de comandos.



ID	Nome	Controlador	Porta			
22	Luz - Cozinha	Placa Controladora	3	Eventos e Comandos	Editar	Deletar
23	Luz - Varanda	Placa Controladora	4	Eventos e Comandos	Editar	Deletar
24	Ventilador - Varanda	Placa Controladora	5	Eventos e Comandos	Editar	Deletar
25	Luz - Sala	Placa Controladora	6	Eventos e Comandos	Editar	Deletar
26	Luz - Quarto	Placa Controladora	7	Eventos e Comandos	Editar	Deletar
27	Ar Condicionado - Quarto	Placa Controladora	8	Eventos e Comandos	Editar	Deletar
28	Sensor 1	Placa Sensores	6	Eventos e Comandos	Editar	Deletar
29	Sensor 2	Placa Sensores	7	Eventos e Comandos	Editar	Deletar
30	Sensor 3	Placa Sensores	8	Eventos e Comandos	Editar	Deletar

Menu Novo

Figura 16: tela de consulta de equipamentos.

A Figura 17 apresenta a tela de consulta dos comandos associados ao equipamento selecionado na tela de consulta de equipamentos que, de acordo com os requisitos, possibilitar associar, desassociar e executar comandos através de botões e *links* encontrados.



Figura 17: tela de consulta dos comandos associados ao equipamento.

A Figura 18 apresenta a tela de associação de eventos a comandos do equipamento que, de acordo com os requisitos, possibilita associar e desassociar comandos a equipamentos do tipo sensor.



Figura 18: tela de associação de eventos a comandos.

Para os requisitos referentes à monitoração dos sensores não existe tela, pois a monitoração é feita através de métodos que são executados de forma transparente para o usuário.

A Figura 19 apresenta a tela de consulta de controladores sendo acessada por um PDA (*iPhone*), satisfazendo assim o requisito de acessibilidade do sistema da através de dispositivos móveis. Outro equipamento utilizado nos testes foi o *Smartphone* N95 8G, da Nokia.

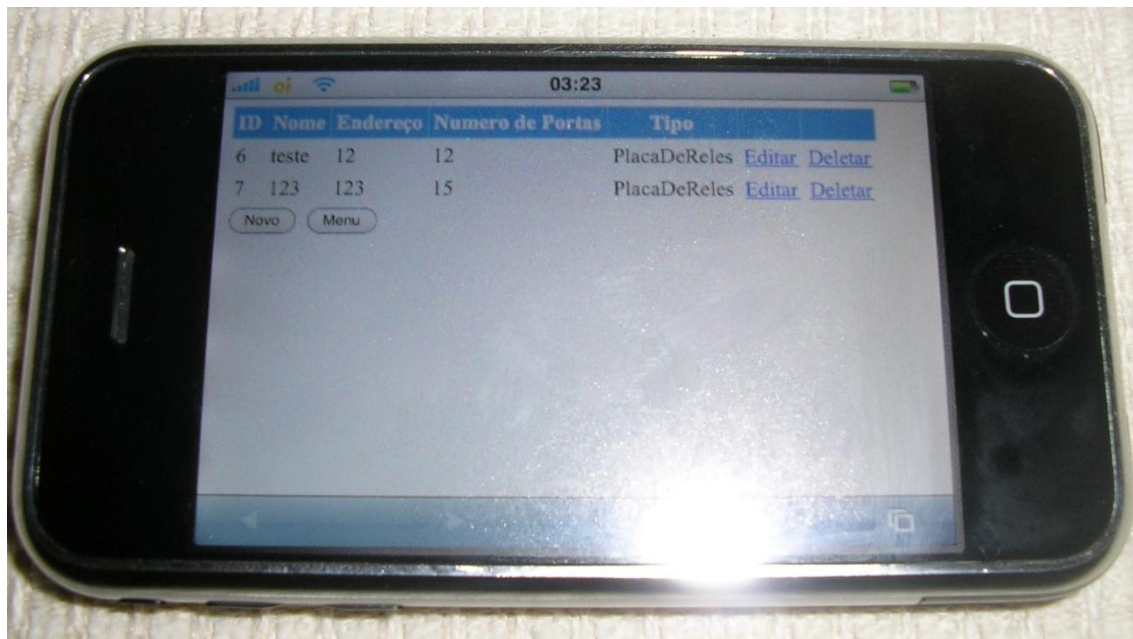


Figura 19: Aplicação sendo acessada através do *iPhone*.

5.2 Metodologia de Testes

Durante toda a fase de implementação, foram realizadas baterias de testes de integração à medida que novos pacotes eram construídos. Ao final, foram realizados testes sistêmicos, que visavam identificar inconformidades do protótipo, levando em consideração os requisitos.

À medida que eram detectados, os erros eram corrigidos até que entrassem em total conformidade com os requisitos do sistema.

5.3 O Hardware

De acordo com Siqueira Filho e Silva Filho (2006), o *hardware* consiste da parte mecânica e física do computador compreendendo seus componentes eletrônicos e peças.

O *hardware* que foi desenvolvido para fornecer informações de sensores ao microcomputador e ligar ou desligar aparelhos eletro-eletrônicos é composto de duas partes. São elas:

- Placa de relés: responsável por ligar e desligar equipamentos eletro-eletrônicos a partir de um comando do enviado pelo *software* através da porta paralela,
- Placa de sensores: responsável por enviar ao *software* informações sobre o estado dos sensores (acionados ou não) através da porta paralela.

A escolha da comunicação via porta paralela, em modo EPP, se dá pelas características da mesma: (i) taxa de transferência de 2 Megabits por segundo e (ii) transmissão bidirecional de informações, que permite tanto o recebimento como envio de dados simultaneamente. Este fato é muito importante, uma vez que a idéia de ter apenas um dispositivo para receber e enviar dados, a porta paralela, simplifica muito o projeto (MESSIAS, 2006).

5.3.1 Placa de Relés

Como mencionado anteriormente, a placa de relés é responsável por ligar e desligar os equipamentos eletro-eletrônicos associados à mesma. Para tal, a placa é composta dos seguintes componentes:

- *LEDs (Light Emitting Diode)*: para sinalizar quais portas estão acionadas,
- Conector de alimentação: responsável pela alimentação da placa.
- Terminais Conectores: para conectar a placa à porta paralela do microcomputador e aos equipamentos eletro-eletrônicos que serão controlados,
- Resistores: para controlar a passagem da corrente elétrica,
- Relés: componente eletromecânico do circuito que tem uma funcionalidade semelhante à de um interruptor comum, fechar e abrir circuitos, característica essa

que proporciona o controle de equipamentos de corrente alternada e de corrente contínua,

- Circuito impresso ULN2803A: responsável por ampliar o sinal recebido pela porta paralela,
- Diodo: para retificar o circuito.

A Figura 20 apresenta a imagem da placa de relés com os seus componentes destacados.

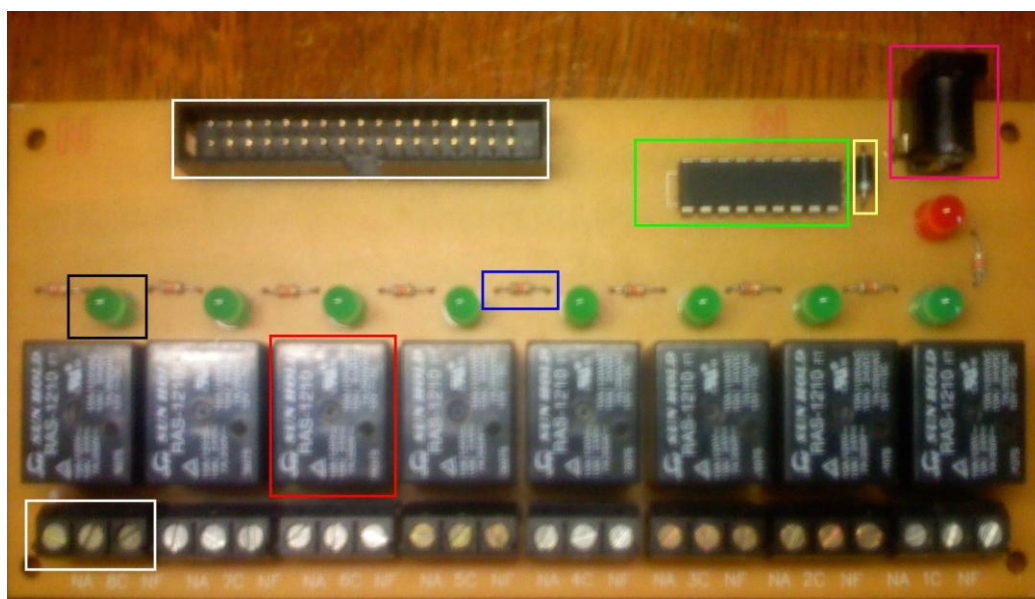


Figura 20: Placa de relés.

No Quadro 8 encontra-se a legenda das cores da Figura 20.

Quadro 8: Legenda de cores da placa de relés.

Cor	Componente
Preto	<i>LEDs</i>
Rosa	Conector de alimentação
Branco	Terminais conectores
Azul	Resistores
Vermelho	Relé
Verde	Circuito impresso ULN2803A
Amarelo	Diodo

A Figura 21 apresenta o circuito elétrico da placa de relés.

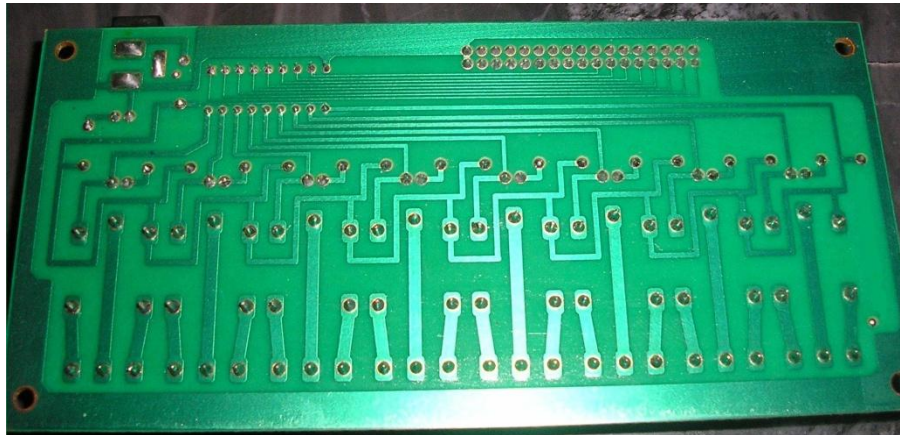


Figura 21: Circuito elétrico da placa de relés.

O esquema elétrico é ilustrado através da fotografia da placa por ser um circuito simples, cuja concepção foi manual, e pela indisponibilidade de *softwares* de CAD (*Computer Aided Design*), que normalmente são muito caros. Adicionalmente, por sua simplicidade, esta placa dispõe apenas de uma única camada.

5.3.2 Placa de Sensores

Como mencionado anteriormente, a placa de sensores é responsável por informar ao *software*, por meio de pulsos elétricos, o estado dos sensores associados à mesma. Para tal, a placa é composta dos seguintes componentes:

- *LEDs (Light Emitting Diode)*: para sinalizar o funcionamento da placa,
- Conector de alimentação: responsável pela alimentação da placa,
- Terminais Conectores: para conectar a placa à porta paralela do microcomputador e aos sensores que serão monitorados,
- *Chip LTV4N25*: responsável por ampliar o sinal recebido pelo sensor e enviá-lo para a porta paralela,
- Transistor UA7805C: responsável por regular a tensão da placa.

A Figura 22: Placa de sensores. apresenta a imagem da placa com os seus componentes destacados.

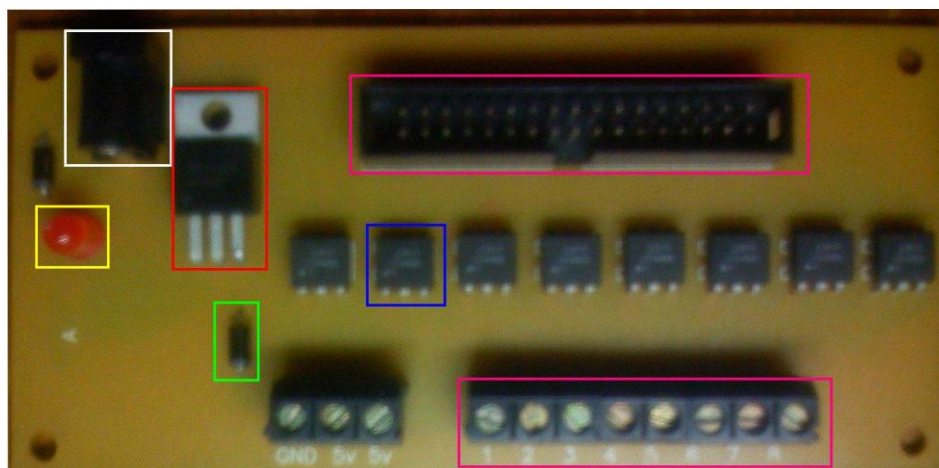


Figura 22: Placa de sensores.

No Quadro 9 encontra-se a legenda das cores da Figura 20.

Quadro 9: Legenda de cores da placa de sensores.

Cor	Componente
Amarelo	<i>LEDs</i>
Branco	Conector de alimentação
Rosa	Terminais conectores
Vermelho	Transistor UA7805C
Azul	<i>Chip</i> LTV4N25
Verde	Diodo

A Figura 23 apresenta o esquema elétrico da placa de sensores.

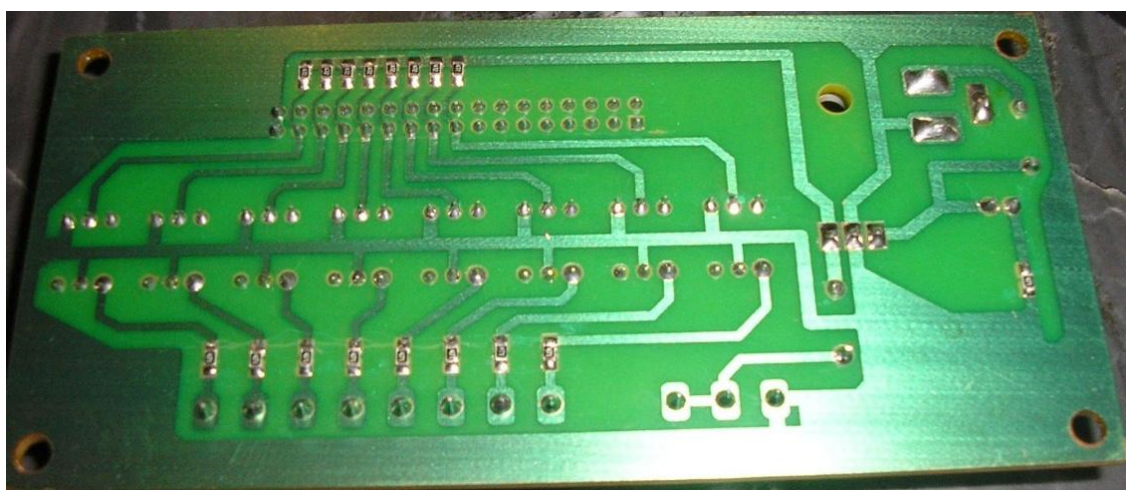


Figura 23: Circuito elétrico da placa de sensores.

O esquema elétrico é ilustrado através da fotografia da placa por ser um circuito simples, cuja concepção foi manual, e pela indisponibilidade de softwares de

CAD, que normalmente são muito caros. Adicionalmente, por sua simplicidade, esta placa dispõe apenas de uma única camada.

5.3.3 A Concepção do *Hardware*

O *hardware* foi produzido em duas etapas. Primeiramente, de forma artesanal, através do método subtrativo, que foi o mais rápido e simples encontrado. Posteriormente, o desenho do *hardware* foi enviado para uma empresa especializada na elaboração de placas de circuito impresso, no estado de São Paulo, onde foi confeccionada de forma industrial.

O método subtrativo é antigo, contudo muito utilizada para fabricação de placas de circuito impresso. Consiste em desenhar, sobre uma folha de fibra de vidro (material base) recoberta por uma fina camada de cobre, em uma ou em ambas as faces, que quando mergulhada em uma solução química corrosiva terá como produto final a folha de fibra de vidro com as trilhas, que foram desenhadas no início do processo (SOARES, 2006).

5.4 A Maquete da Casa Inteligente

Além do *software* e *hardware* envolvidos no protótipo, foi construída uma maquete de casa inteligente em madeira, utilizada para distribuir os sensores e dispositivos de controle (lâmpadas e ventoinhas (que representam os dispositivos de climatização)). A elaboração desta maquete tem o propósito de tornar mais realística a experiência da casa inteligente (Figura 24).

A maquete representa uma casa com quatro cômodos, sendo eles:

- Um dormitório: quadrante superior esquerdo,
- Uma varanda: quadrante superior direito,
- Uma sala de jantar: quadrante inferior esquerdo,
- Uma sala de televisão: quadrante inferior direito.



Figura 24: Maquete da Casa Inteligente.

Os cômodos da maquete tiveram os equipamentos eletro-eletrônicos distribuídos como representa o Quadro 10.

Quadro 10: Distribuição dos equipamentos por cômodo.

Cômodo	Equipamentos
Dormitório	Luz
	Uma representação de um condicionador de ar modelo <i>Split</i> .
Varanda	Luz
	Uma ventoinha representando um ventilador.
Sala de jantar	Luz
Sala de televisão	Luz

Para representar os sensores, foi utilizado um interruptor triplo como se pode observar na Figura 25 que, ao ser pressionado, simula o pulso elétrico enviado por sensores como os de presença e temperatura para a placa de sensores.



Figura 25: Interruptor.

Os sensores estão pré-configurados para efetuar os seguintes comandos:

- Sensor 1: ligar todos os equipamentos da maquete,
- Sensor 2: desligar todos os equipamentos da maquete,
- Sensor 3: livre para demonstração.

6 DIFICULDADES ENFRENTADAS

As principais dificuldades encontradas durante o desenvolvimento do projeto são:

- **Multidisciplinaridade:** A construção dos diversos elementos integrantes do protótipo envolveu o projeto de um *hardware*, de um *software* e uma maquete.
 - A concepção do *hardware* foi suportada pelos princípios estudados nas disciplinas de Sistemas Lógicos Digitais e Arquitetura de Computadores. Adicionalmente, para a construção física do equipamento, foi necessária a contratação de uma empresa especializada do estado de São Paulo,
 - A aplicação foi desenvolvida utilizando-se os princípios teóricos e práticos das áreas de programação e engenharia de *software*,
 - A maquete exigiu habilidades rudimentares de marcenaria.
- Do ponto de vista arquitetural, para prover acesso a dispositivos à uma aplicação que executa em um contêiner *WEB*, há necessidade de suplantando algumas barreiras

técnicas, dado que, por questões de segurança, o modelo não permite acesso direto aos componentes físicos do computador. Neste caso, foi preciso um estudo sobre configurações de segurança do *Firewall* do *Windows XP* e do *IIS 6.0*, servidor de aplicações da Microsoft.

7 TRABALHOS FUTUROS

Durante o desenvolvimento do projeto, foram identificadas as seguintes oportunidades de melhorias e / ou trabalhos futuros:

- Melhorias da interface de usuário, tornando a interface mais intuitiva e amigável,
- Melhorias arquiteturais da solução, tornando-a mais flexível ao suporte de novas facilidades como, por exemplo, o suporte a som ambiente. Tais modificações requerem alterações tanto no *hardware* (aumento no número de portas e suporte a outros tipos de dispositivos) como no *software* (parametrização, facilidades de configuração através de arquivos, aumento da facilidade de manutenção e melhorias de processos como, a monitoração dos ambientes e controle de acesso),
- Adição de novas funcionalidades, como, por exemplo, a disponibilização de imagens de câmeras na Internet e serviços de SMS (*Short Message Service*),
- Adição de suporte à comunicação sem fio através de protocolos como *Bluetooth*, Wi-Fi (padrão IEEE 802.11 (IEEE, 2009)) ou outros,
- Adição de suporte ao padrão USB para evitar a obsolescência do equipamento, em caso de descontinuidade das interfaces paralelas, previsíveis num futuro próximo, e
- Formatação desta prova de conceito como um produto comercial.

8 CONCLUSÕES

A finalidade deste projeto foi construir um protótipo de sistema de automação residencial de baixo custo – denominado “Casa Inteligente”, com a capacidade de tomar decisões autônomas através da leitura de sensores e do acionamento de cargas elétricas. Deste modo, é possível aumentar a segurança,

comodidade, o conforto e o grau de acessibilidade a portadores de necessidades especiais.

As soluções comerciais de automação residencial são tipicamente proprietárias, baseadas em um dispositivo de controle específico com um *software* embarcado e que possuem custo bastante elevado. Em algumas destas soluções, somente o *hardware* de controle tem custo superior a R\$ 10.000,00 (dez mil reais).

O protótipo apresentado neste projeto baseia-se em um *hardware* bastante simples, com custo estimado de, aproximadamente, R\$ 600,00 (seiscentos reais) e um *software*, que pode ser executado em microcomputadores com sistema operacional Windows 2003, XP ou Vista. Considerando-se o fato de que um número cada vez maior de residências dispõe de microcomputadores, este componente da solução não representaria um custo adicional.

Portanto, a primeira, e também a maior contribuição deste trabalho, consiste exatamente em demonstrar a viabilidade de soluções de domótica de baixo custo.

Outra contribuição direta deste projeto consiste em despertar o interesse acadêmico, notadamente para alunos de graduação, para esta área do conhecimento que possui um conjunto bastante amplo de aplicações, além de um forte apelo social.

Outra contribuição importante consiste em desmistificar a complexidade de concepção do *hardware* de controle. Através de um projeto simples, que utiliza conceitos básicos de sistemas e eletrônica digitais, é possível construir um equipamento suficientemente genérico para controlar diversos tipos de dispositivos, simplesmente através do acionamento de cargas elétricas.

Destaca-se a simplicidade e a versatilidade do *software* de gerenciamento, que possui acesso através da Internet e suporte a dispositivos móveis sem fio, como celulares e PDAs (*Personal Device Assistants*). Esta característica possibilita o monitoramento e controle remoto de todo o sistema de automação residencial.

Apesar de este projeto ter optado pela plataforma .NET, por questões de facilidade e produtividade para o desenvolvimento do *software*, tecnicamente é possível sua implementação utilizando-se outras tecnologias, como por exemplo, o JavaTM, da *Sun Microsystems*.

Esta mudança representa a independência da plataforma de execução do *software* de gerenciamento em relação ao *hardware* e sistema operacional hospedeiros, possibilitando, inclusive, a utilização de sistemas operacionais baseados *software* livre. No entanto, destaca-se a necessidade de adaptações, as quais dependem do conjunto de *hardware* e *software* escolhido. Estas adaptações referem a detalhes de implementação do modelo de comunicação paralela, ao contêiner a ser utilizado (Apache Tomcat, JBoss ou outros) e também à própria compatibilidade com navegadores utilizados para acesso à aplicação.

Como consequência direta da adoção de *software* livre, pode-se alcançar uma redução ainda mais significativa no custo final da solução.

REFERÊNCIAS BIBLIOGRÁFICAS

[ANGEL, 1993] Angel, P. M. Introducción a la domótica; Domótica: controle e automação. Escuela Brasileño-Argentina de Informática. EBAI. 1993

[BOLZANI, 2004] BOLZANI, C.A.M. **Residências inteligentes** - domótica, redes domésticas, automação residencial. São Paulo: Livraria da Física, 2004.

[BRUGNERA, 2008] BRUGNERA, Mauro Ricardo. **Domótica**. Disponível em: <<http://www.unibratec.com.br/jornadacientifica/diretorio/FEEVALE+MRB.pdf>>. Acesso em: 02 dez. 2008.

[BUSCHMANN *et al* 1996] BUSCHMANN, F., Meunier, R., Rohnert, H., Sommerland, P., and Stal, M. Pattern- oriented software architecture: a system of patterns. John Wiley & Sons Ltd, New York, 1996.

[DAAMEN, 2005] DAAMEN, David. Sistemas de barramentos domésticos. Elektor: eletrônica e microinformática, Barueri-SP, n. 41, 2005.

[DECOESFERA, 2009] DECOESFERA. *Iluminación y domótica*. Disponível em: <<http://www.decoesfera.com/2007/04/21-iluminacion-y-domotica>> acesso em: 18 jan 2009.

[DEITEL *et al*, 2006] DEITAL, H. M., DEITEL, P. J., LISTFIELD, J., NIETO, T. R., YAEGER, C., ZLATKINA, M. **C# Como Programar**. São Paulo: Makron Books, 2006.

[FERREIRA, 2008] FERREIRA, João Alexandre Oliveira. **Interface homem-máquina para domótica baseado em tecnologias WEB**, Faculdade de Engenharia da Universidade do Porto, Porto – Portugal, 2008.

[FINKELSTEIN, 2003] FINKELSTEIN, Clive. *Microsoft WEB services support*. Austrália, fev. 2003. Disponível em: <<http://www.dmreview.com/issues/20030201/6304-1.html>>. Acesso em: 17 jan. 2009.

[GAMMA *et al*, 2005] Gamma, E., Johnson, R., Helm, R., e Vlissides, J. **Padrões de projeto**: soluções reutilizáveis de *software* orientado a objetos. Porto Alegre: Bookman, Editora. 2005.

[HORSTMANN, 2007] HORSTMANN, C. **Padrões e Projetos Orientados a Objetos**. São Paulo: BOOKMAN COMPANHIA ED, 2007.

[IEEE, 2009] IEEE. **IEEE 802.11 Architecture**. Disponível em: < http://www.tutorial-reports.com/wireless/wlanwifi/wifi_architecture.php > . Acessado em: 20 jan. 2009.

[MAZIERO, 2008] MAZIERO, Carlos. **Introdução aos Serviços de Rede**. Disponível em: < <http://www.ppgia.pucpr.br/~maziero/doku.php/espec:introducao> >. Acesso em: 19 jan. 2009.

[MESSIAS, 2006] MESSIAS, R. **Curso C/C++ Porta Paralela**: controle de dispositivos. São Paulo: Rogercom, 2006.

[MICROSOFT, 2007] MICROSOFT. **Microsoft Developer Network**. Disponível em: <<http://www.microsoft.com/brasil/msdn>>. Acesso em: 6 abr. 2007.

[MICROSOFT, 2008] MICROSOFT. **Microsoft SQL Server 2005 Express Edition**. Disponível em: <<http://www.microsoft.com/downloads/details.aspx?familyid=220549b5-0b07-4448-8848-dcc397514b41&displaylang=pt-br>>. Acesso em: 19 dez. 2008.

[MIORI *et al*, 2006] MIORI, V., TARRINI, L., MANCA, M. TOLOMEI, G. **An open standard solution for domotic interoperability**. In: *Consumer Electronics*, IEEE Transactions on. 2006. Disponível em: < http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1605032 >. Acesso em: 20 jan. 2009.

[MONTEIRO, 2001] MONTEIRO, L. A. Internet como meio de comunicação possibilidades e limitações. In: CONGRESSO BRASILEIRO DE CIÊNCIAS DA COMUNICAÇÃO, 24., 2001. Campo Grande. Anais... São Paulo: Intercom, 2001. CD-ROM. Disponível na Internet. URL: <<http://hdl.handle.net/1904/4714>>. Acesso em: 20 out. 2008.

[MSDN, 2008a] MSDN. *Common Language Runtime Overview*. Disponível em: <<http://msdn.microsoft.com/pt-br/library/ddk909ch.aspx>>. Acesso em: 20 out. 2008a.

[MSDN, 2008b] MSDN. *Common Language Specification*. Disponível em: <<http://msdn.microsoft.com/pt-br/library/12a7a7h3.aspx>>. Acesso em: 20 out. 2008b.

[MSDN, 2008c] MSDN. *Compiling to MSIL*. Disponível em: <<http://msdn.microsoft.com/pt-br/library/c5tkafs1.aspx>>. Acesso em: 20 out. 2008c.

[MSDN, 2008d] MSDN. *Language Interoperability Overview*. Disponível em: <<http://msdn.microsoft.com/pt-br/library/a2c7tshk.aspx>>. Acesso em: 20 out. 2008d.

[MSDN, 2008e] MSDN. *Metadata Overview*. Disponível em: <<http://msdn.microsoft.com/pt-br/library/xcd8txaw.aspx>>. Acesso em: 20 out. 2008e.

[MSDN, 2009a] MSDN. *Class Library*. Disponível em: <[http://msdn.microsoft.com/en-us/library/d11h6832\(VS.71\).aspx](http://msdn.microsoft.com/en-us/library/d11h6832(VS.71).aspx)>. Acesso em: 10 jan. 2009a.

[MSDN, 2009b] MSDN. *Introduction to the .NET Framework Class Library*. Disponível em: <[http://msdn.microsoft.com/en-us/library/hfa3fa08\(VS.71\).aspx](http://msdn.microsoft.com/en-us/library/hfa3fa08(VS.71).aspx)>. Acesso em: 10 jan. 2009b.

[NICOSHOP, 2009] NICOSHOP. Disponível em: <http://www.nicoshop.com/cat039_l2.html>. Acesso em: 18 jan 2009.

[NUNES, 2002] NUNES, Renato Jorge Caleira. **Arquitetura de um Controlador Domótico Flexível e de Baixo Custo**. INESC-ID. Lisboa – Portugal, 2002.

[PLATT, 2003] PLATT, D.S. **Introducing Microsoft .Net**. Redmond. Microsoft Press, 2003.

[SGARBI, 2007] SGARBI, Julio André; Tonidandel, Flavio. **Domótica Inteligente: Automação Residencial baseada em Comportamento**. Centro Universitário da FEI - São Bernardo do Campo – SP, 2007.

[SHALLOWAY e TROTT, 2004] SHALLOWAY, A.; TROTT, J.R. **Explicando padrões de projeto: uma nova perspectiva em projeto orientado a objeto**. Porto Alegre: Bookman, 2004

[SILVEIRA, 1991] SILVEIRA, JORGE LUIS DA. **Comunicação de dados e sistemas de teleprocessamento**. São Paulo: Makron, McGraw-Hill, 1991.

[SIQUEIRA FILHO e SILVA FILHO, 2006] SIQUEIRA FILHO, J.B.; SILVA FILHO, J.B. **Tecnologia da informação para administradores**. 2.ed. Fortaleza: Universidade de Fortaleza, 2006.

[SOARES, 2006] SOARES, Marcio J., Programação Placa Logo Control. **Revista Mecatrônica Fácil**. São Paulo, v. 05, n. 27, Mar.-Abr. 2006.

[TAKIUCHI *et al*, 2004] TAKIUCHI, Marcelo; MELO, Érica; TONIDANDEL, Flávio. **Domótica Inteligente: Automação Baseada Em Comportamento**. Centro Universitário da FEI - São Bernardo do Campo – SP, 2004.

[TYSON, 2009a] TYSON, J. **How Serial Ports Work**. Disponível em:<<http://computer.howstuffworks.com/serial-port3.htm>> Acessado em: 19 jan. 2009a.

[TYSON, 2009b] TYSON, J. **How USB Ports Work**. Disponível em:<<http://computer.howstuffworks.com/usb.htm>> Acessado em: 19 jan. 2009b.

APÊNDICES

APÊNDICE A – Namespaces, Classes, Interface e Tipos de dados utilizados

No Quadro 11 é possível encontrar os *Namespaces*, conjuntos de classes e interfaces, da FCL que foram utilizados na implementação do projeto.

Quadro 11: Namespaces utilizados na implementação do projeto (MSDN, 2009a).

Namespaces	Descrição	Justificativa de uso
System	Principal Namespace, pois contém as classes fundamentais. Dentre essas classes encontra-se as responsáveis por definirem os tipos de dados, eventos, manipuladores, interfaces, exceções, entre outras.	Utilização indispensável, pois fornece diversos tipos de classes de grande importância para a implementação de qualquer aplicação.
System.Collections	Contém um conjunto de interfaces e classes que definem coleções de objetos, como listas, filas, <i>arrays bit</i> , <i>hashtables</i> e dicionários, por exemplo.	Utilizado principalmente para o manuseio e transferência, entre camadas, de registros recuperados do banco de dados.
System.ComponentModel	Fornecer classes que são usadas para implementar o tempo de execução e para controlar o comportamento de componentes e controles.	Extremamente útil para o controle de componentes e controles, utilizado para controlar componentes como os de acesso a fontes de dados.
System.Configuration	Fornecer classes e interfaces que permitem o acesso e manutenção dos arquivos de configurações do .NET Framework.	Utilizado para acessar e manter o arquivo de configuração da aplicação.
System.Data	Contém um conjunto de interfaces e classes que permitem a construção de componentes capazes de gerenciar dados de fontes de dados.	Utilizado para criação de componentes para manipular dados de fontes de dados.
System.Data.SqlClient	Encapsula uma coleção de classes para acesso a bancos de dados da família SQL Server.	Utilizado para armazenar no / recuperar informações do banco de dados SQL Server 2005.
System.IO	Fornecer interfaces e classes para leitura e escrita de arquivos.	Utilizado para ler e escrever arquivos no sistema de arquivos.
System.Reflection	Contém interfaces e classes para a criação e invocação dinâmica de tipos de valores.	Utilizado para criar e invocar tipos de valores.
System.Text	Fornecer interfaces e classes para representar ASCII (<i>American Standard Code for Information Interchange</i>), Unicode, UTF-7 (<i>Unicode Transformation Format de 7 bits</i>) e codificações de caracteres UTF-8 (<i>Unicode Transformation Format de 8 bits</i>), além de conter classes para manipulação de objetos do tipo <i>string</i> .	Utilizado para manipular objetos do tipo <i>string</i> .
System.Web	Fornecer interfaces e classes para permitir a comunicação entre o navegador e o servidor, bem como o gerenciamento de	Utilizado para tornar possível a troca de informações entre o navegador do cliente e o

	<i>cookies</i> e transferência de arquivos.	servidor.
System.Web.Security	Contém classes que são usadas para implementar a segurança no servidor de aplicação.	Utilizado para tornar a aplicação segura, através da autenticação de usuários e encriptação de dados.
System.Web.UI	Fornecer interfaces e classes que permitem a criação de controles e páginas para aplicações Web.	Utilizado para criar controles visuais WEB, como <i>labels</i> e <i>textboxes</i> .
System.Xml	Oferece interfaces e classes para suporte a arquivos com base no padrão XML.	Utilizado para trabalhar com arquivos XML.

No Quadro 12 é possível encontrar os tipos de valores disponíveis na FCL que foram utilizados na implementação do projeto.

Quadro 12: Tipos de valores encontrados na FCL utilizados na implementação do projeto (MSDN, 2009b).

Categoria	Nome da Classe	Tipo de Valor do C#
Inteiro	Int32	Int
Lógico	Boolean	Bool
Class Object	Object	Object
	String	string

Documentar exhaustivamente todos os *Namespaces*, classes, interfaces e tipos de dados da plataforma .NET está fora do escopo deste trabalho. Recomenda-se a leitura das referências MSDN, 2009a e MSDN, 2009b ao leitor interessado em obter detalhes adicionais.

APÊNDICE B – Descrição dos métodos da camada DAL

Este apêndice tem a função de detalhar os métodos apresentados no diagrama de interfaces, Figura 10, e diagrama de classes, Figura 11, da camada DAL.

O Quadro 13 detalha as assinaturas dos métodos exportadas pela interface IDAO.

Quadro 13: Descrição das assinaturas dos métodos da interface IDAL.

Assinaturas	Descrição
<i>Insert</i>	Insere na base de dados um novo objeto da classe concreta em questão.
<i>Update</i>	Altera objetos já inseridos na base de dados da classe concreta em questão.
<i>Delete</i>	Exclui um objeto da base de dados da classe concreta em questão.
<i>GetAll</i>	Retorna da base de dados todos os objetos da classe concreta em questão.
<i>GetByID</i>	Retorna um objeto da base de dados filtrado pelo campo ID.

O Quadro 14 detalha as assinaturas dos métodos exportadas pela interface IControladorDAO.

Quadro 14: Descrição das assinaturas dos métodos da interface IControladorDAO.

Assinaturas	Descrição
<i>hasRelationship</i>	Retorna um valor booleano que indica se o objeto em questão possui algum relacionamento com outros objetos.
<i>selecionarControladoresPorNome</i>	Retorna todos os objetos persistidos no banco que possuem o nome passado por parâmetro.

O Quadro 15 detalha as assinaturas dos métodos exportadas pela interface IComandoDAO.

Quadro 15: Descrição das assinaturas dos métodos da interface IComandoDAO.

Assinaturas	Descrição
<i>hasRelationship</i>	Retorna um valor booleano que indica se o objeto em questão possui algum relacionamento com outros objetos.
<i>selecionarComandosPorNome</i>	Retorna todos os objetos persistidos no banco que possuem o nome passado por parâmetro.

O Quadro 16 detalha as assinaturas dos métodos exportadas pela interface IEquipamentosDAO.

Quadro 16: Descrição das assinaturas dos métodos da interface IEquipamentosDAO.

Assinaturas	Descrição
<i>hasRelationship</i>	Retorna um valor booleano que indica se o objeto em questão possui algum relacionamento com outros objetos.
<i>selecionarEquipamentosPorNome</i>	Retorna todos os objetos persistidos no banco que possuem o nome passado por parâmetro.

O Quadro 17 detalha as assinaturas dos métodos exportadas pela interface IControladorDAO.

Quadro 17: Descrição das assinaturas dos métodos da interface IControladorDAO.

Assinaturas	Descrição
<i>hasRelationship</i>	Retorna um valor booleano que indica se o objeto em questão possui algum relacionamento com outros objetos.
<i>selecionarControladoresPorNome</i>	Retorna todos os objetos persistidos no banco que possuem o nome passado por parâmetro.

O Quadro 18 detalha a assinatura do método exportada pelas interfaces IComandoDoEquipamentoDAO e IEventoDoEquipamentoDAO.

Quadro 18: Descrição da assinatura do método das interfaces IComandoDoEquipamentoDAO e IEventoDoEquipamentoDAO.

Assinaturas	Descrição
<i>hasRelationship</i>	Retorna um valor booleano que indica se o objeto em questão possui algum relacionamento com outros objetos.

O Quadro 19 detalha os métodos da classe AbstractDAO.

Quadro 19: Descrição dos métodos da classe AbstractDAO

Métodos	Descrição
<i>AddParameters</i>	É responsável por informar no comando SQL os valores dos parâmetros do objeto entidade.
<i>Delete</i>	Responsável por excluir um objeto da base de dados.
<i>FactoryEntity</i>	É responsável por criar um objeto a partir de um registro da base de dados.
<i>FactoryEntityList</i>	Responsável por fabricar uma lista de objetos provenientes de um <i>DataSet</i> .
<i>GenericExecute</i>	Executa um comando SQL que não retorna resultados da base de dados.
<i>GenericGet</i>	Executa um comando SQL que retorne um objeto da base de dados.

<i>GenericGetList</i>	Executa um comando SQL que retorne uma lista de objetos da base de dados.
<i>GetAll</i>	Recupera da base de dados todos os registros do objeto do tipo da classe concreta.
<i>GetByID</i>	Recupera um objeto da base de dados filtrado a partir do campo ID.
<i>GetGeneratedID</i>	Recupera da base de dados o ultimo identificador gerado.
<i>GetNextUniqueIdentifier</i>	Retorna o próximo identificador do objeto.
<i>Insert</i>	Responsável por incluir um objeto do tipo da classe concreta na base de dados.
<i>Update</i>	Responsável por alterar um registro na base de dados.

O Quadro 20 detalha os métodos da classe DAOFactory.

Quadro 20: Descrição dos métodos da classe DAOFactory

Métodos	Descrição
<i>Factory</i>	Fabrica as classes concretas a partir de uma interface.
<i>GetInstance</i>	Fornece a instancia da classe passada por parâmetro.

Quadro 21 detalha os métodos da classe AbstractSQLDAO.

Quadro 21: Descrição dos métodos da classe AbstractSQLDAO.

Métodos	Descrição
<i>AssignConnection</i>	Responsável por associar uma conexão do banco de dados com um comando SQL.
<i>BeginTransaction</i>	Responsável por iniciar uma transação de banco de dados para uma conexão.
<i>CloseConnection</i>	Responsável por encerrar uma conexão com o banco de dados.
<i>CommitTransaction</i>	Responsável pelo comando “commit” no banco de dados.
<i>CreateConnection</i>	Responsável por criar uma nova conexão com o banco de dados.
<i>Init</i>	Responsável por criar novas instâncias de membros provados.
<i>InitConnectionPool</i>	Responsável por iniciar um <i>pool</i> de conexões com o banco de dados.
<i>OpenConection</i>	Responsável por abrir uma conexão.
<i>RollBack</i>	Responsável por desfazer uma transação com o banco de dados.
<i>UpdateConnectionPool</i>	Responsável por atualizar o <i>pool</i> de conexões com o banco de dados.

Todos os métodos das classes concretas ComandosDoEquipamentoDAO, ControladoresDAO, EquipamentosDAO, EventosDoEquipamentoDAO e ComandosDAO já foram mencionados anteriormente quando foi falado sobre a assinatura dos mesmos nas interfaces da camada DAL.

APÊNDICE C – Descrição dos métodos da camada *Business*

Este apêndice tem a função de detalhar os métodos apresentados no diagrama de interfaces, Figura 12, e diagrama de classes, Figura 13, da camada *Business*.

O Quadro 22 detalha as assinaturas dos métodos exportadas pela interface *IBusiness*.

Quadro 22: Descrição das assinaturas dos métodos da interface *IBusiness*.

Assinaturas	Descrição
<i>Insert</i>	Responsável pela regra de negócio de inserção de um objeto novo na base de dados.
<i>Update</i>	Responsável pela regra de negócio de alteração de objetos já inseridos na base de dados.
<i>Delete</i>	Responsável pela regra de negócio de exclusão de um objeto da base de dados.
<i>GetAll</i>	Responsável pela regra de negócio para retornar da base de dados todos os objetos da classe concreta em questão.
<i>GetByID</i>	Retorna um objeto da base de dados filtrado pelo campo ID.

O Quadro 23 detalha a assinatura do método exportada pela interface *IControladoresBusiness*.

Quadro 23: Descrição das assinaturas dos métodos da interface *IControladoresBusiness*.

Assinaturas	Descrição
<i>selecionarControladoresPorNome</i>	Responsável pela regra de negócio para retornar da base de dados todos os objetos com o campo nome igual ao campo nome passado por parâmetro e chamada do método correspondente na camada de acesso a base de dados.

O Quadro 24 detalha a assinatura do método exportada pela interface *IComandoBusiness*.

Quadro 24: Descrição das assinaturas dos métodos da interface *IComandoBusiness*.

Assinaturas	Descrição
<i>selecionarComandoPorNome</i>	Responsável pela regra de negócio para retornar da base de dados todos os objetos com o campo nome igual ao campo nome passado por parâmetro e chamada do método correspondente na camada de acesso a base de dados.

O Quadro 25 detalha a assinatura do método exportada pela interface `IEquipamentosBusiness`.

Quadro 25: Descrição das assinaturas dos métodos da interface `IEquipamentosBusiness`.

Assinaturas	Descrição
<code>selecionarEquipamentosPorNome</code>	Responsável pela regra de negócio para retornar da base de dados todos os objetos com o campo nome igual ao campo nome passado por parâmetro e chamada do método correspondente na camada de acesso a base de dados.

O Quadro 26 detalha a assinatura do método exportada pela interface `IExecutarBusinessFacade`.

Quadro 26: Descrição das assinaturas dos métodos da interface `IExecutarBusinessFacade`.

Assinaturas	Descrição
<code>executarComando</code>	Responsável por executar comandos dos equipamentos.

O Quadro 27 detalha a assinatura do método exportada pela interface `AbstractBusiness`.

Quadro 27: Descrição dos métodos da classe `AbstractBusiness`.

Assinaturas	Descrição
<code>Delete</code>	Método genérico para exclusão de um objeto.
<code>GetAll</code>	Método genérico para recuperar todos os objetos de uma base de dados.
<code>GetByID</code>	Método genérico para recuperação de um objeto filtrado pelo campo ID.
<code>GetDAO</code>	Responsável pela construção da classe de acesso a base de dados.
<code>Insert</code>	Método genérico para inclusão de um objeto na base de dados.
<code>Update</code>	Método genérico para a alteração de um objeto na base de dados.

O Quadro 28 detalha a assinatura do método exportada pela interface `IExecutarBusinessFacade`.

Quadro 28: Descrição dos métodos da classe `ExecutarBusinessFacade`.

Assinaturas	Descrição
<code>executarComando</code>	Responsável por executar comandos dos equipamentos.
<code>getValor</code>	responsável por ler os valores dos sensores.
<code>isFlagged</code>	responsável por verificar se uma determinada porta da paralela está acionada.

Todos os métodos das classes concretas `ControladoresBusiness`, `EventosDoEquipamentoBusinessm`, `EquipamentosBusiness`, `ComandosDoEquipamentoBusiness` e `ComandoBusiness` já foram mencionados

anteriormente quando foi falado sobre a assinatura dos mesmos nas interface da camada *Business*.