CENTRO UNIVERSITÁRIO DE BRASÍLIA – UniCEUB
CURSO DE ENGENHARIA DE COMPUTAÇÃO

WILSON GOTTI NETO

AUTOMATIZAÇÃO PARA PAINÉIS ELETRÔNICOS PUBLICITÁRIOS DE ALTA DEFINIÇÃO

Orientadora: Professora Msc Maria Marony Sousa Farias

Brasília
Junho, 2012

WILSON GOTTI NETO

AUTOMATIZAÇÃO PARA PAINÉIS ELETRÔNICOS PUBLICITÁRIOS DE ALTA DEFINIÇÃO

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.
Orientadora: Prof. Maria Marony Souza Farias Nascimento

Brasília

Junho, 2012

WILSON GOTTI NETO

AUTOMATIZAÇÃO PARA PAINÉIS ELETRÔNICOS PUBLICITÁRIOS DE ALTA DEFINIÇÃO

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.
Orientador: Prof. Maria Marony Souza Farias Nascimento

Este Trabalho foi julgado adequado para a obtenção do Título de Engenheiro de Computação, e aprovado em sua forma final pela Faculdade de Tecnologia e Ciências Sociais Aplicadas -FATECS.

_____

Prof. Abiezer Amarilia Fernandes
Coordenador do Curso

Banca Examinadora:

_____

Professora Msc Maria Marony Sousa Farias
Orientador

_____

Professor Msc Marco Antônio Araújo
UniCEUB

_____

Professor Doutor Sidney Cerqueira Bispo dos Santos
UniCEUB

Dedico esse trabalho e a minha vida aqueles com que sempre pude contar: minha mãe, meus avós e minha querida prima Munik, que com seu jeito menina sempre consegue alegrar meus dias, por mais sombrios que eles tenham sido!

AGRADECIMENTOS

Nesses anos de faculdade, muitas pessoas fizeram parte da minha vida, umas mais presentes e outras mais distantes. Agradeço a todas, pois com certeza essas pessoas são direta, ou indiretamente responsáveis por mais essa vitória em minha vida.

O amigo Thiago Rider foi uma das pessoas mais presentes nesse período acadêmico e me lembro de boas madrugadas regadas a cálculos matemáticos, coca-cola, pizza, muito jazz e MPB. Deixo aqui meus agradecimentos especiais pela sua eterna paciência em tolerar minhas intempestividades nesses anos de estudo e até mesmo nos dias de hoje.

Ao meu pai, que me olha lá do céu, só posso agradecer em orações, mas mesmo assim deixo aqui registrado um “muito obrigado”, tendo a certeza que ele está vibrando tanto quanto eu com essa graduação.

Palavras não suficientes para agradecer a minha mãe, guerreira, que me tolera há 31 anos. Mesmo assim mãezinha, obrigado por tudo que fez e ainda faz por esse seu filho impaciente, que tanto lhe ama.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Visando principalmente melhorar a automatização de painéis eletrônicos de alta definição, dada à expansão do segmento mídia *out of home* no Brasil e no mundo, é proposta uma solução gerenciadora que permite a publicação segmentada de conteúdos em uma rede de painéis eletrônicos. Neste projeto, foi desenvolvida uma solução gerenciadora, composta de dois softwares, um software cliente que deve ser instalado em cada um dos painéis digitais, responsável por ligar e desligar o equipamento em horários definidos e exibir o conteúdo publicitário, e um software servidor, que se comunica com o software cliente informando e disponibilizando o conteúdo a ser exibido, de acordo com a programação definida pelo administrador. Além da função intrínseca do sistema, que é disponibilizar e veicular conteúdo, o mesmo é capaz de gerar relatórios de veiculação, de forma a auxiliar a medição e análise dos resultados obtidos nas campanhas de comunicação e marketing.

Palavras Chave: painel eletrônico, digital, alta definição, mídia, out of home, solução gerenciadora, segmentada, software, comunicação, marketing

ABSTRACT

Aimed mainly to improve the automation of high-definition electronic billboards, given the expansion of out of home media segment in Brazil and worldwide, a managing solution is proposed which enables the targeted content in a network of electronic billboards. In this project, we developed a managing solution that is composed of two softwares, a client software that must be installed on each of the digital billboards, responsible for turning on and off the equipment at set times and display advertising content, and a server software, that communicates with the client software informing and providing content to be displayed, according to a schedule set by the administrator. In addition to the intrinsic function of the system that is provide and serve content, it is able to generate exhibition reports, in order to assist the measurement and analysis of results obtained in communication and marketing campaigns.

Keywords: electronic billboard, digital, high definition media, out of home, managing solution, segmented, software, communication, marketing

CAPÍTULO 1 - INTRODUÇÃO

1.1 - Apresentação do Problema

O forte impacto dos avanços tecnológicos no comportamento dos consumidores tem levado anunciantes e profissionais de marketing e comunicação à busca de soluções de mídia e comunicação mais atuais e adequadas. A dispersão da audiência e a fragmentação das mensagens recebidas pelos consumidores delineiam um perfil de espectador que não mais é atraído pela tradicional "disputa pela audiência". Sua atenção precisa ser conquistada em ambientes cheios de estímulos e informações, e uma das formas mais eficazes de se conseguir é quando as informações são transmitidas em sincronia com o momento vivenciado pelo público.

Nesse contexto, a Mídia Digital *Out of Home* (OOH), uma nova tecnologia em mídia de comunicação, surgiu nesse milênio nos países do primeiro mundo como solução para alcançar os consumidores no momento certo. Busca-se, com esse tipo de mídia, "reconquistar a atenção do consumidor, através da relevância que uma mensagem adquire ao ser transmitida em sincronia com o ambiente em que ela se realiza" (ARAÚJO, 2010, p. 22). Utilizando telas de LCD, Plasma ou LED que funcionam como painéis eletrônicos publicitários de alta definição, procura-se aproveitar todo tipo de contato com o cliente, seja no ambiente interno ou externo do estabelecimento.

A OOH é uma das mídias que oferece uma das melhores relações custo-benefício do mercado. Permite que o anunciante passe sua mensagem a um segmento específico, com custo relativamente baixo, e mensure o retorno de sua ação e curto e médio prazo. Uma pequena padaria, por exemplo, que provavelmente não teria recursos para divulgação de sua empresa e seus produtos por outros meios de comunicação como TV ou rádio, pode encontrar nas soluções de mídia digital *Out of Home* uma importante alternativa.

1.2 - Topologia da Solução

Dentre as possibilidades de configuração de um sistema OOH, pode-se trabalhar com painéis eletrônicos conectados a uma solução gerenciadora, onde serão arquivadas e disponibilizadas as mídias a serem veiculadas.

Conforme ilustra a Figura 1.1 - Topologia geral do trabalho, os painéis são configurados e controlados por um administrador, que pode encontrar-se fisicamente próximo ou distante dos painéis.

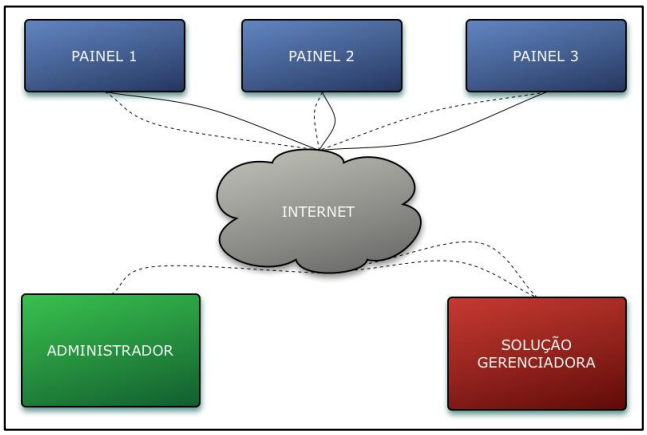


Figura 1.1 - Topologia geral do trabalho

Cada painel é composto por um uma TV LCD, que funciona como monitor, e um computador conectado à internet, que transmite os dados recebidos, conforme demonstrado na Figura 1.2 - Painel de mídia digital OOH.

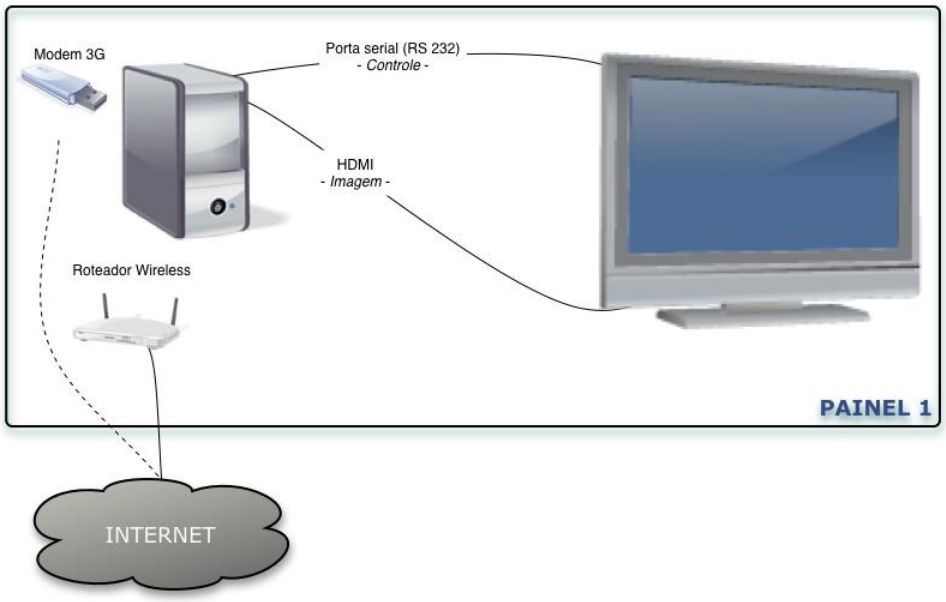


Figura 1.2 - Painel de mídia digital OOH

A Solução gerenciadora, ilustrada na Figura 1.3 - Solução Gerenciadora, é composta por um servidor de banco de dados (SQL Server 2008) e um servidor web (Internet Information Services).

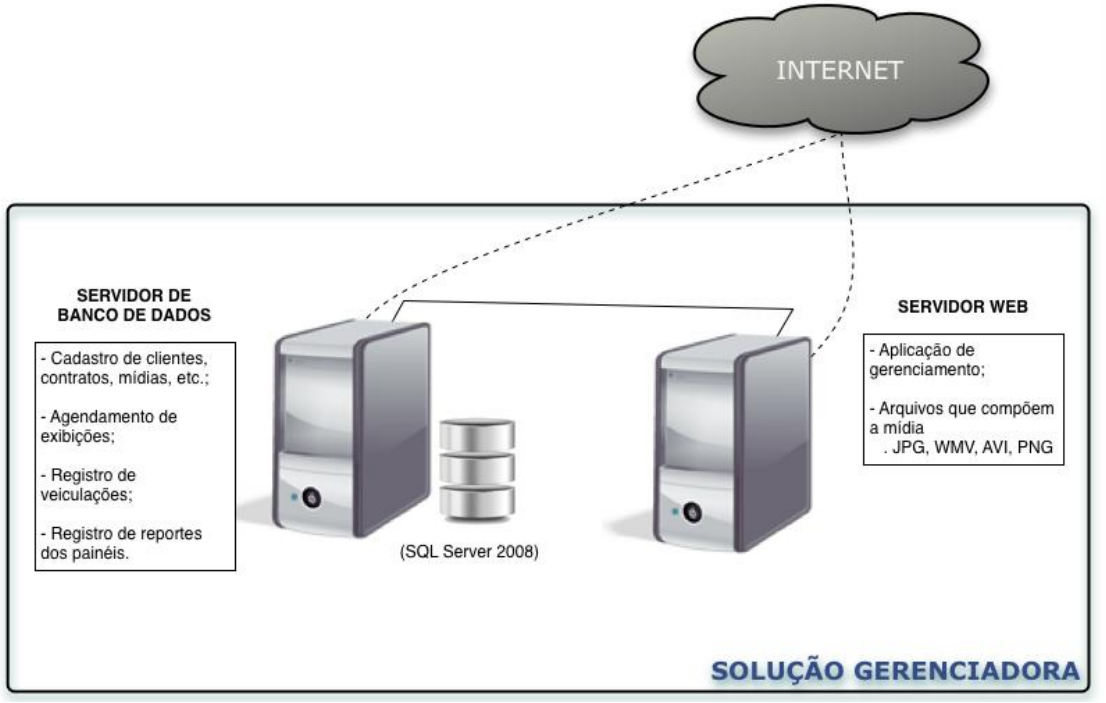


Figura 1.3 - Solução Gerenciadora

O servidor de banco de dados é responsável por: a- cadastro de clientes, contratos, mídias, entre outros; b- agendamento de exibições; registro de veiculações; e c- registro de reportes dos painéis. Já o servidor web possui como responsabilidades: a- aplicação de gerenciamento; b- arquivos que compõem a mídia (JPG, WMV, AVI, PNG, etc.).

Tudo isso é gerenciado por um administrador que pode encontrar-se distanciado do sistema, conectado via internet, conforme ilustra a Figura 1.4- Administrador.

Figura 1.4- Administrador

Diante de tal contexto, interessa-nos verificar: De que forma a mídia digital *out of home* pode proporcionar publicidade em alta resolução e automatizada, de forma a influenciar ainda mais a decisão de compra dos consumidores?

1.3 - Objetivos do Trabalho

O objetivo geral deste trabalho é contribuir com melhorias na logística de publicação de mídias digitais *out of home* (OOH), por meio da criação de um sistema de automatização para painéis eletrônicos publicitários de alta definição, que permitirá o gerenciamento dos dados publicados de forma automática e personalizada.

A solução aqui proposta, tem como objetivos específicos:

- Modelar e alimentar o banco de dados, de forma a fornecer informações de controle sobre os painéis e gerir de maneira eficaz a publicação de conteúdo nos mesmos;
- Analisar e testar os comandos seriais disponíveis em uma TV de LED da marca LG (modelo 42LE5300), incluindo criação de cabo serial específico para este fim;
- Desenvolver o software cliente que é instalado em cada um dos painéis e o software de gerenciamento, que é hospedado na plataforma web, em um servidor na internet;
- Testar a solução como um todo, incluindo a exibição de conteúdo e geração de registros de veiculações.

1.4 - Justificativa e Importância do Trabalho

O uso de Mídia Digital *Out Of Home* é uma realidade crescente enquanto estratégia de marketing. Vem se destacando por sua excelente relação custo-benefício, tornando-se uma alternativa viável de publicidade para as pequenas e médias empresas.

Esse trabalho contribuirá com melhorias no que diz respeito à qualidade das imagens transmitidas e automatização do processo, trazendo ganhos significativos para a comunicação, a publicidade e a logística. Além disso, dada a preocupação com a poluição visual provocada pelos *outdoors*, que já se transformou em proibição legal em algumas grandes cidades, a mídia digital *out of home* se apresenta como uma solução mais sustentável e adequada às necessidades dos empresários.

Torna-se necessário, portanto, buscar alternativas que proporcionem aumento na qualidade das mídias veiculadas, e que facilitem o processo de gerenciamento dos conteúdos publicados, de forma automática, personalizada e rápida.

1.5 - Escopo do Trabalho

Este projeto limitou-se à criação de uma solução de automatização para painéis eletrônicos publicitários de alta definição, incluindo um software cliente instalado em cada um dos painéis e o software de gerenciamento, que é hospedado na plataforma web, em um servidor na internet.

1.6 - Resultados esperados

Com este trabalho pretende-se contribuir com a automatização dos painéis eletrônicos de mídia digital *out of home*, com um software que além de publicar conteúdo em alta definição de forma segmentada para cada um dos painéis, também permitirá ligar os aparelhos automaticamente e elaborar relatórios de veiculação das mídias, tornando a relação entre a empresa e o cliente ainda mais transparente.

1.7 - Estrutura do Trabalho

O trabalho está organizado em seis capítulos. Os primeiros quatros capítulos fazem a apresentação do tema do projeto, fornecem o embasamento teórico do trabalho e as tecnologias utilizadas. No capítulo 5 é apresentado o protótipo do sistema e sua aplicação em

um estudo de caso em Uberaba-MG, restando para o último capítulo as considerações finais e propostas futuras. A organização detalhada é descrita a seguir:

- Capitulo 1: Capítulo introdutório, contendo breve apresentação do problema, a topologia da solução, os objetivos, justificativa e relevância do trabalho, seu escopo, resultados esperados e estrutura.
- Capitulo 2: Detalhamento do problema, apresentando a relação entre publicidade e mídia digital, conceitos de Mídia Digital *Out Of Home* (OOH) e Mídia Indoor, os avanços observados na OOH no mundo e no contexto brasileiro, a caracterização dos painéis eletrônicos publicitários, considerando-se em especial a qualidade das imagens e a automatização dos conteúdos a serem neles disponibilizados.
- Capitulo 3: Bases Metodológicas para Resolução do Problema, contendo explicações detalhadas sobre os equipamentos e hardware utilizado na solução.
- Capitulo 4: Modelo Proposto, com descrição detalhada da solução, tanto em nível de hardware quanto de software.
- Capitulo 5: Aplicação do Modelo Proposto, com estudo de caso realizado na cidade de Uberaba-MG.
- Capitulo 6: Conclusão, finalizando o projeto com as observações finais e sugestões para projetos futuros.

CAPÍTULO 2 - APRESENTAÇÃO DO PROBLEMA

2.1 - Publicidade e a mídia digital OOH

Kelsen (2010)[1], um dos maiores especialistas do setor, afirma que a mídia digital *out of home* (OOH) é a "quinta tela", conforme ilustra a Figura 2.1 - As quatro telas que, segundo Kelsen, antecederam a Mídia OOH, surgida após o cinema, a televisão, os computadores e celulares, a qual cria uma nova conexão visual e pode influenciar diretamente a decisão de compra dos consumidores.

Figura 2.1 - As quatro telas que, segundo Kelsen, antecederam a Mídia OOH
Fonte: Kelsen, 2010.

Utilizando painéis digitais de alta tecnologia estrategicamente localizados, a OOH destaca-se por atingir os consumidores em seu cotidiano, quando estão em plena atividade.

Um dos fatores que diferencia a mídia OOH de outras mídias é o fato de que o consumidor é alcançado pela propaganda de forma compulsória, passiva, enquanto realiza

[1] Kelsen é mundialmente reconhecido por sua atuação na indústria de Mídia Digital Out of Home, tendo publicado livro tratando a respeito (Unleashing the Power of Digital Signage – Content Strategies For The 5th Screen). É Chair of the Content Best Practices Committee for the International Digital Screen Media (Trade) Association e atua como representante das mais importantes companhias do setor, buscando a difusão do conhecimento e a utilização correta desse poderoso meio de comunicação. Recebeu em 2009 o título de “Digital Signage Man of the Year” e recentemente foi nomeado um dos “Top 5 Executives to Watch in 2010”, pela Digital Signage Today. Também é membro do Conselho da Digital Screen Association e é autor do prestigiado “Top 10 Annual Trends” para a indústria de Digital Signage/DOOH.

suas atividades cotidianas, sem que precise realizar alguma ação como ligar a TV, comprar um jornal, ou entrar na internet. Além disso, pode ser considerada uma mídia “pura”, pois diferente de outros veículos de comunicação não possui editorial, o conteúdo é disponibilizado diretamente (CURY, 2004).

2.1.1 - Processo de comunicação

Para Cury (2004), a comunicação proporcionada pelas mídias digitais *out of home* permite modelos de publicidade que compreendam todo o processo comunicacional, desde sua origem até o momento de recepção da mensagem. O processo de comunicação é iniciado por um *anunciante* (remetente) que tem um *problema* ou *oportunidade*, um *foco*, um *objetivo* e uma *verba* para elaborar e enviar uma mensagem a um *destinatário* (consumidor).

Para que atenda ao foco e aos objetivos da comunicação, a mídia digital deverá ter seus conteúdos e sua forma de apresentação coerentes com a estratégia publicitária adotada.

De acordo com Kelsen (2010), quando uma mídia de qualidade é incorporada adequadamente à ambientes de venda, estas são elevadas. Kelsen (2010) afirma ainda que trabalhando o conteúdo de forma segmentada, e instalando os painéis levando em consideração o local onde ocorre a decisão de compra, como a estante onde está o produto, a mídia OOH desempenha seu papel de forma mais eficaz.

É importante também medir o retorno em relação à eficácia da transmissão da mensagem. Isso pode ser viabilizado por meio de ferramentas de acompanhamento de vendas, vinculadas à rede de publicidade. Esta integração pode inclusive estar conectada ao estoque da empresa, trocando o anúncio quando um produto é esgotado, por exemplo.

Assim, o marketing digital, utilizando adequadamente os conceitos relacionados à comunicação publicitária e usufruindo dos benefícios da OOH, atinge aos consumidores com mais eficiência, podendo até mesmo interagir com os mesmos, quando utilizados recursos tecnológicos como *touchscreen*, câmeras, microfones, sensor de movimento, entre outros.

2.1.2 - Programação segmentada

Uma das maiores vantagens da mídia digital OOH é que, pela sua característica tecnológica, ela permite trabalhar com programação segmentada, definindo-se diferentes conteúdos a serem disponibilizados em locais e horários distintos.

Assim, ao analisar o perfil dos consumidores, conhecendo seu cotidiano, os locais que frequenta em cada hora do dia, pode-se programar conteúdos específicos em cada painel, cada qual adequado ao contexto e ao perfil de consumidor que será atingido naquele momento do dia. A Figura 2.2 - Ubiquidade por perfil, ilustra como esta analise do perfil dos consumidores pode levar a escolhas de mídias diferenciadas para horários distintos, em um mesmo local, como uma loja de conveniência, por exemplo.

Figura 2.2 - Ubiquidade por perfil
Fonte: AABOH, 2008

Como a mídia OOH pode ser inserida em diferentes contextos, a escolha do momento e do local de disponibilização da mensagem pode interferir diretamente na decisão de compra do consumidor. Por exemplo, a propaganda de uma marca de pão de queijo em uma padaria será mais eficiente em alguns horários do que em outros, ao longo do dia. Da mesma forma, o anúncio de uma churrasqueira elétrica provavelmente surtirá mais efeito se veiculado no açougue de um supermercado do que na seção de frutas.

2.2 - Mídia Out Of Home x Mídia Indoor

A Mídia Digital *Out of Home*, conforme já apresentado, caracteriza-se pela veiculação de informações em telas em diferentes locais, normalmente pontos de venda e lugares com grande fluxo de pessoas. Entretanto, por tratar-se de uma mídia nova e sua definição ainda não estar fundamentada, encontram-se variações em seu emprego em diferentes países.

O termo "out of home" esteve originalmente ligado à noção geral de algo externo, aquilo que está fora de casa, em alguns lugares adotado como sinônimo de mídia ou comunicação exterior[2].

Por outro lado, conforme afirma Santana (2009), o termo também é adotado como sinônimo de mídia *indoor*, ou seja, aquela que é realizada dentro de um estabelecimento ou local de espera forçada. Ao agregar-se o suporte digital à mídia, a utilização de Mídia Digital OOH como sinônimo de indoor reforçou-se.

De acordo com a Associação Brasileira de Mídia Indoor (2008), o termo Mídia Indoor conceitua-se por

> [...] toda e qualquer forma de atividade correspondente à produção, montagem e veiculação de qualquer manifestação publicitária, que possua ou não movimento ou iluminação, exibida de forma mecânica ou digital, ou por outro meio que venha a surgir, instalada em ambientes fechados, com grande fluxo de trânsito de pedestres ou veículos.

Entretanto, por considerar inclusas entre as mídias digitais OOH também as telas que se encontram a céu aberto, considera-se mais adequada a definição de "out of home" para toda mídia veiculada fora da casa do consumidor, seja interna (mídia indoor) ou externamente (mídia exterior) aos estabelecimentos.

2.3 - Classificação das mídias digitais OOH

A OOH atinge espontaneamente o consumidor fora de sua casa, seja em um corredor de um shopping, em bares, restaurantes, padarias, academias, etc. A comunicação pode ocorrer em momentos de espera forçada, quando o consumidor aguarda atendimento e está

[2] Inclui, além de painéis digitais, totens, letreiros luminosos, outdoors, entre outros.

carente de algum atrativo para amenizar sua espera – como em lotéricas, elevadores, ônibus, metrô, trens, consultórios, postos de gasolina, etc. – ou no momento de sua decisão de compra, podendo influenciá-la de forma significativa – como em redes de supermercados, lojas de departamento, vitrines, etc.

Diversas podem ser as classificações em relação ao tipo de mídia OOH, em geral referindo-se ao momento ou o local em que a mídia atinge o consumidor, refletindo diferentes níveis de influência em sua decisão de compra.

De acordo com definições internacionais, a Mídia Digital *Out of Home* pode ser classificada em três setores (ABDOH, 2008):

- Alto Impacto: Enormes monitores de LCD, e até mesmo conjuntos de monitores, disponíveis em diferentes locais ao ar livre e atingindo motoristas, pedestres e pessoas em trânsito;
- Ponto de venda: Monitores instalados em pontos de venda como no interior ou exterior de supermercados, lojas, restaurantes e Shopping Centers;
- Audiência Cativa: Comunicação exibida em um local específico, com público definidos, onde consumidor está disponível e inclui ônibus, metro, trem, elevador, aeroporto, maternidade, refeitórios, áreas de convivência de funcionários, etc.

2.4 - Avanço da Mídia *Out of Home* (OOH)

A mídia *Out of Home* surgiu há cerca de 15 anos nos Estados Unidos, com faturamento crescente especialmente nos últimos anos. De acordo com a *Digital Place-based Association* (DPPA), o setor digital tem crescido significativamente na área de publicidade, com aumento de 14,2% nos Estados Unidos de 2010 para 2011.

Esse crescimento é 17 vezes maior do que o observado no setor de publicidade em geral, ficando atrás apenas da TV *Syndication* (15,4%)[3], conforme apresentado na

[3] TV "Syndication" é a venda de programas de TV para diversos canais, para transmissão ao vivo ou gravada, no rádio ou na televisão. É comum em países onde a televisão é composta por cadeias de emissoras locais afiliadas, como os Estados Unidos.

Tabela 1 - Crescimento da indústria publicitária entre 2010 e 2011.

Tabela 1 - Crescimento da indústria publicitária entre 2010 e 2011.

Mídia	**2011 x 2010 Crescimento (%)**
TV *Syndication*	15.4
Digital Place-based (midia digital)	14.2
TV a cabo	7.7
Outdoor	6.5
Network Radio	2.7
Internet	0.4
Revistas Nacionais	.0
Rede TV	2.0
Jornais nacionais	3.6
Jornais locais	3.8
TV local	4.5
Radio local	5.4
Publicidade nos EUA total	0.8

Fonte: http://www.prnewswire.com/-releases/digital-place-based-media-revenue-growth-rate-for-2011-exceeds-that-of-overall-us-ad-industry-by-more-than-171-margin-148668545.html.

A The Michael J. Fox Foundation[4], ONG que vem contribuindo com pesquisas sobre a cura do câncer, divulgou sua causa durante dois meses apenas utilizando mídia digital, para testar seu alcance. Com 137 mil telas em 230 mercados e atingindo 1,3 bilhões de pessoas, a empresa constatou aumento de 27% no número de membros da entidade, atribuindo grande eficácia ao meio de comunicação. Para essa campanha, foi demandado esforço conjunto 22 empresas, 2 agências e grande quantidade de peças criativas (ARAÚJO, 2010).

Sem um sistema para gerenciar essa quantidade de peças criativas, toda a logística de disponibilização das mídias e geração de relatórios de veiculação era manual, e quase inviabilizou ou projeto, evidenciando a necessidade de uma automatização do processo, proposta neste trabalho.

Portanto, apesar do grande potencial publicitário da mídia digital OOH, alguns entraves tecnológicos são os que mais impedem essa indústria de assumir papel importante no cenário da mídia. Como exemplo, Brian Dusho, membro da DPAA, cita em 2010 “a falta de sistemas automatizados e padrões abertos que permitem ao comprador de mídia fazer milhares de operações, sejam elas locais, regionais ou nacionais” (ARAÚJO, 2008). Para

[4] Michael Fox, ator conhecido desde a década de 1980 por sua atuação em "De volta para o futuro", foi diagnosticado em 1991 com mal de Parkinson. Dada o avanço de sua doença e a impossibilidade de continuar atuando, Fox criou a The Michael J. Fox Foundation, uma ONG que angaria recursos para financiar grupos que pesquisam cura para o Parkinson.

Dusho, o ideal seria a criação de redes automatizadas, com soluções cruzadas de execução e medição das campanhas, para avaliação de sua eficácia.

2.5 - Mídia *Out of Home* no Brasil

A OOH surgiu no Brasil por volta de 2003 (ABDOH, 2008), com a instalação de monitores de LCD em supermercados, shopping centers, restaurantes e maternidades. À medida em que seu potencial foi sendo reconhecido, as telas também passaram a ser instaladas em outros locais, inicialmente em elevadores de edifícios comerciais e em seguida em ônibus, metrô, trens e aeroportos.

Com o aumento nos investimentos publicitários, a OOH, oferecendo programação segmentada de acordo com o perfil de público e seu contexto, além do seu custo reduzido, tornou-se atrativa a academias, panificadoras, farmácias e outros estabelecimentos do pequeno varejo, para divulgação de lançamentos, serviços e promoções.

O segmento tem tido significativo crescimento no contexto brasileiro, especialmente nos últimos anos. Comparando-se os quatro primeiros meses de 2009 e de 2010, por exemplo, observou-se crescimento de 80% no mercado de mídia OOH. Em 2009 eram 50 mil telas instaladas, com faturamento de R$ 94 mil (ABDOH, 2008).

A empresa Mídia Bay, por exemplo, ao anunciar uma promoção para venda de frios da perdigão com sorteio de porta-frios, conforme ilustra a Figura 2.3 - Mídia OOH nas panificadoras gerou aumento de 20% nas vendas de produtos Perdigão, utilizando OOH conseguiu superar a meta de vendas. Foram instalados 2,2 mil aparelhos de LCD em 680 locais em São Paulo, Curitiba, Florianópolis, Porto Alegre e Juiz de Fora.

Figura 2.3 - Mídia OOH nas panificadoras gerou aumento de 20% nas vendas de produtos Perdigão
Fonte: ARAÚJO, 2010.

Estima-se que a curva de crescimento da mídia OOH chegará a equivaler-se à da internet, atingindo 5% do bolo publicitário brasileiro até 2020 (PR Newswire Association, 2012).

Em pesquisa realizada pela Ipsos Marplan[5], foi constatado que 67% da população de São Paulo já teria sido impactada por mídia OOH nos últimos 30 dias, principalmente em ônibus, supermercados e trens de metrô. Observa-se também significativo crescimento no uso desse tipo de mídia no contexto do pequeno varejo. Esse é um segmento que, conforme Raphael do Amaral Cumplido, coordenador de trade marketing da Brasil Foods, "compra bem, tem boas margens de lucro, mas ainda é carente de promoções" (ARAÚJO, 2010).

Conforme evidenciado no Capítulo 5 deste trabalho, a mídia OOH pode ser uma solução acessível aos pequenos empresários, dada sua boa relação custo-benefício.

[5] Considerada referência em pesquisa de mídia com os 'Estudos Marplan', que analisam os hábitos de mídia e consumo.

CAPÍTULO 3 - BASES METODOLÓGICAS PARA RESOLUÇÃO DO PROBLEMA

3.1 - Internet

Qualquer equipamento de mídia digital OHH automatizado tem como premissa básica o uso da internet. A logística de atualização de conteúdo para uma rede de painéis sem o uso da internet praticamente inviabiliza todo o negócio.

Além disso, a internet é o recurso que possibilita que painéis localizados em cidades, estados e países diferentes possam ser acessados para manutenção de forma remota e segura, com autenticação e até mesmo criptografia dos dados, caso necessário.

Dada essa necessidade de conexão e a dificuldade enfrentada em alguns casos para conectar os painéis a internet por cabo, especialmente devido a posição física dos mesmos, frequentemente localizados próximos ao teto, em pilares, e no centro de grandes pátios, fica evidente que esse acesso a internet deve ser realizado sem o uso de cabos, ou seja, *wireless*. Dentre os métodos para acesso *wireless* a internet, os mais usados são:

3.1.1 - Wifi

A internet, seja por ADSL, cabo coaxial ou óptico é instalada em um ponto relativamente distante do painel e através de um roteador *wireless* é transmitida até o equipamento. A mesma solução é utilizada para acessar a internet na maioria dos notebooks, seja em casa, faculdade, escritórios ou até mesmo em restaurantes.

Esse tipo de acesso a internet é muito utilizado para redes privadas de mídia digital OOH, onde existe mais de um painel instalado. Nesse caso, usando a mesma conexão física (adsl, cabo coaxial ou óptico) e vários *access points* distribuindo a conexão entre os painéis é possível reduzir drasticamente o custo de acesso a internet para os painéis da rede.

3.1.2 - Tecnologia 3G

Desde a tecnologia móvel 1G, que ainda era analógica, utiliza-se um processo denominado modulação para transmissão de voz e/ou dados na telefonia móvel.

Modulação é a técnica pela qual "as características da portadora (sinal que é modulado) são modificados com a finalidade de transmitir informações, utilizada em transmissões e em *modems*[6]. É o processo pelo qual se modificam as características de uma onda de rádio ou elétrica, de forma que as alterações representem informações significativas para o ser humano ou para uma máquina" (PIZZOTTI, s/d).

A modulação modifica o formato da informação elétrica com o objetivo de transmiti-la com a menor potência possível, com a menor distorção possível, com facilidade de recuperação da informação original e ao menor custo possível.

Transforma-se uma das características da onda: amplitude, fase ou frequência. Dessa maneira, qualquer tipo de informação, até a voz humana ou transação de dados numa aplicação interativa é transmitida numa onda eletromagnética. O transmissor adiciona a informação numa onda básica (denominada onda portadora) de tal forma que poderá ser recuperada na outra parte através de um processo reverso chamado demodulação.

A tecnologia 3G é um padrão de tecnologia móvel que permite às operadoras oferecerem aos usuários serviços com velocidade de transferência maior do que na rede tradicional 2G, mas ainda inferior as praticadas pelos acessos convencionais, como adsl e cabo coaxial.

O 3G, possui melhor *eficiência espectral*, uma medida da taxa de informação que pode ser transmitida em uma determinada largura de banda, em função da melhoria dos equipamentos disponíveis (inclusive dos aparelhos celulares) e das técnicas de modulação aplicadas.

No âmbito de mídia digital OOH, esse tipo de conexão tem alguns pontos positivos interessantes:

---

[6] Dispositivo utilizado para alterar um sinal, tendo seu nome baseado nos procedimentos que o mesmo utiliza para atingir esse objetivo: **mod**ulação e **dem**odulação. Assim, a função principal de um modem é alterar o sinal a ser transmitido através de um meio físico, e no destino realizar o procedimento inverso, reconstruindo o sinal a seu formato original.

- Independência de cada um dos painéis, ou seja, se houver algum problema em uma conexão, os outros painéis continuam funcionando perfeitamente;
- Ausência de qualquer vínculo com a internet usada pelo estabelecimento onde o painel está instalado, evitando transtornos de redução de largura de banda e de reestabelecimento da conexão, em caso de problemas técnicos e/ou financeiros junto a fornecedora de serviços de internet.
- Maior mobilidade, requisito indispensável em painéis móveis de mídia digital OOH;

3.2 - Conexões da televisão LCD

Os aparelhos televisores modernos, sejam eles de LCD, LED ou Plasma têm oferecido uma vasta gama de portas para conexão com outros dispositivos.

Dentre as portas existentes, existem as conexões de entrada e saída de vídeo e áudio, a conexão de rede (geralmente uma porta ethernet comum), conexões USB para pen-drives e uma porta de serviço, que também pode ser USB ou uma conexão serial específica para este fim.

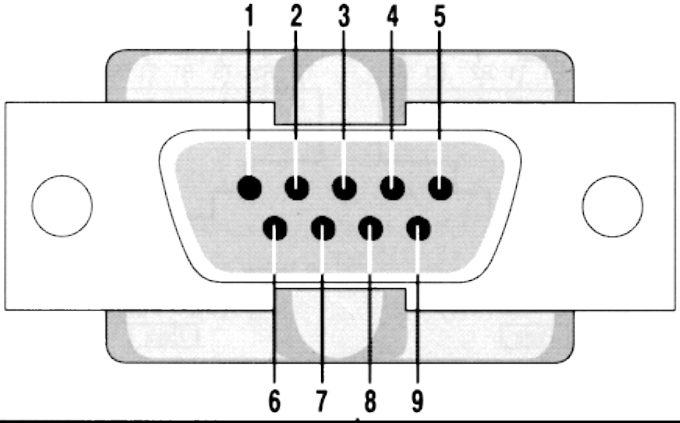
Evidentemente, as conexões de áudio e vídeo são as mais utilizadas, mas quando há a necessidade de manutenção no equipamento ou qualquer interação não humana, a porta de serviço é utilizada, seja para uma simples atualização de *firmware* ou para controlar o equipamento através de um computador.

Essa última utilidade para a porta de serviço é de extrema importância para este projeto, pois através dela será possível automatizar comandos como e desligar ligar a TV do painel de mídia digital OOH.

3.2.1 - Porta Serial (RS-232)

Considerada uma das conexões externas mais básicas para um computador, a porta serial tem sido uma parte integral da maioria dos computadores há mais de 20 anos. A interface 20serial ou porta serial, também conhecida como RS-232, é uma porta de

comunicação utilizada para conectar modems, *mouses*, algumas impressoras, *scanners* e outros equipamentos. Na interface serial, diferentemente da porta paralela, os bits são transferidos em fila, ou seja, um bit de cada vez. O padrão RS-232 foi originalmente definido para uma comunicação por meio de 25 fios diferentes, porém conforme ilustra a Figura 3.1 - Pinagem da porta serial, a IBM definiu que apenas 9 pinos seriam necessários, o que foi largamente adotado.



Pin	Signal	Pin	Signal
1	Data Carrier Detect	6	Data Set Ready
2	Received Data	7	Request to Send
3	Transmitted Data	8	Clear to Send
4	Data Terminal Ready	9	Ring Indicator
5	Signal Ground		

Figura 3.1 - Pinagem da porta serial
Fonte: http://www.aggsoft.com/rs232-pinout-cable/images/9-pinout.gif

3.2.1.1 - Configurações

Muitos ajustes são necessários para as conexões seriais utilizada para comunicação assíncrona, é necessário selecionar a velocidade, o número de bits de dados por caractere, paridade e número de bits de parada por caractere. Nas portas seriais construídas através de um circuito integrado UART[7], todas as configurações geralmente são controladas por software, já em portas seriais da década de 80 e anteriores o estabelecimento dessas configurações pode exigir alterações em *switches* ou *jumpers* em uma placa de circuito. Muitas vezes, se as configurações

[7] **U**niversal **A**synchronous **R**eceiver **T**ransmitter, circuito responsável por converter um sinal paralelo, proveniente de um computador, por exemplo, em um sinal serial, e vice e versa. Para atingir sua função principal, estabelecer comunicação serial assíncrona entre dois dispositivos, o sinal a ser transmitido sofre algumas alterações como a adição de bits de parada, e de paridade, por exemplo.

foram informadas incorretamente a conexão não será descartada, no entanto os dados enviados serão recebidos na outra extremidade incorretamente.

A **velocidade** é a quantidade de bits por segundo transmitida de um dispositivo para outro. Taxas comuns de transmissão são 300, 1200, 2400, 9600, 19200, etc. Tipicamente ambos os dispositivos devem estar configurados com a mesma velocidade, alguns dispositivos, porém, podem ser configurados para auto-detectar a velocidade.

O número de **bits de dados** em cada caractere da mensagem pode ser de 5 (para código Baudot), 6, 7 (para ASCII true), 8 (para qualquer tipo de dados, este corresponde ao tamanho de um byte), ou 9. A utilização do byte como caractere da mensagem é uma prática quase universalmente utilizada, especialmente em aplicações mais recentes. Geralmente 5 ou 7 bits só fazem sentido com equipamentos mais antigos, como telégrafos.

Paridade é um método de verificar a precisão dos dados, detectando possíveis erros de transmissão. Podendo ser par, ímpar ou nula (não usada), ela funciona modificando os dados, em cada pacote enviado (geralmente um byte). Na paridade par, os dados são acomodados de modo que o quantidade de bits 1 (isto é, sua contagem no pacote a ser trafegado) seja um número par; isto é feito definindo o bit de paridade (geralmente o bit mais ou menos significativo) como 0 ou 1. Na paridade impar, a quantidade de bits 1 deverá ser um número impar. Evidentemente, na paridade nula os dados não são modificados.

A **paridade** pode ser usada pelo receptor para detectar a transmissão de erros: se um byte foi recebido com o número errado de bits 1, então ele deve estar corrompido. Se a paridade estiver correta então não devem haver erros, ou então há um número par de erros.

Bits de parada (stop bits) são enviados no fim de cada byte transmitido em um conexões assíncronas, com o intuito de permitir que o receptor do sinal se sincronize. Podem haver 1, 1,5 ou 2 bits de parada. Dispositivos eletrônicos geralmente usam um bit de parada. Os telégrafos eletromecânicos utilizavam, um 1,5 ou 2 bits de parada.

Existe uma convenção para a notação se uma configuração de software de uma conexão serial, esta notação é da forma D/P/S. Sendo que a configuração mais comum é a 8/N/1 que especifica que são transmitidos 8 bits de dados, paridade nula e um bit de parada. O número de bits de dados pode ser 7, 8 ou 9. Paridade pode

ser nula (N), impar (O) ou par (E); o bit de paridade é emprestado dos bits de dados, então 8/E/1 significa que um dos oito bits de dados é utilizado como bit de paridade.

3.2.1.2 - Controle de fluxo

Uma porta serial pode usar sinais do dispositivo para fazer uma pausa e posteriormente retomar a transmissão de dados. Esse procedimento é denominado controle de fluxo. Por exemplo, uma impressora lenta pode precisar de um *handshake*[8] com a porta serial para indicar que o fluxo de dados deve ser interrompido enquanto ela executa procedimentos mecânicos, como puxar um novo papel.

O controle de fluxo pode acontecer por *hardware,* sendo RTS/CTS ou DTR/DSR e utilizando pinos do conector, ou por *software*, sendo XON/XOFF e utilizando caracteres especiais no fluxo dos dados.

No controle de fluxo por *software*, o caractere XON diz ao receptor que o remetente do caractere está pronto para receber mais dados. O caractere XOFF diz ao receptor para parar de enviar caracteres. XON/XOFF é um método "em banda" que funciona entre dois pontos, mas ambos devem suportar o protocolo, e há uma confusão em potencial no início. O XON/XOFF está em desuso, e em linhas gerais, é preferível que se utilize o controle de fluxo RTS/CTS.

O controle de fluxo RTS/CTS foi desenvolvido com o intuito de permitir ligações *half-duplex*[9]*,* onde apenas um equipamento pode transmitir por vez. O terminal deve sinalizar “Pronto Para Enviar” e esperar que o destinatário responda com o sinal “Envie os Dados”. RTS/CTS é um *handshake* no nível do *hardware* e tem suas vantagens.

Uma das simplificações introduzidas em padrões mais modernos de barramento serial como Ethernet, FireWire e USB é que muitos desses parâmetros têm valores fixos para que os usuários não possam e não precisem alterar a configuração.

---

[8] É o processo onde dois dispositivos “negociam” e definem parâmetros para estabelecer a conexão entre si, como taxa de transmissão, paridade, dentre outros.

[9] Comunicação onde ambos dispositivos podem transmitir e receber, porem não simultaneamente. Assim, em um determinado momento um dispositivo é transmissor e o outro é receptor, e em outro momento há uma inversão destes papéis.

3.2.2 - High-Definition Multimedia Interface (HDMI)

O HDMI é uma tecnologia de conexão de dispositivos de áudio e vídeo que tem tudo para substituir os padrões existentes até então. Por trás de seu desenvolvimento está um time de gigantes da indústria eletrônica, tais como Sony, Philips, Toshiba, Silicon Image, entre outras.

Com essa tecnologia, é possível, por exemplo, conectar um reprodutor de *Blue-ray* a uma TV de alta definição e ter como resultado imagens de excelente qualidade. Por meio de um cabo HDMI pode-se transmitir sinais de áudio e vídeo, sendo que em outros padrões é necessário ter, pelo menos, um cabo para cada coisa.

Mas, as vantagens do HDMI não se limitam a isso. Essa é uma tecnologia que transmite sinais de forma totalmente digital. Graças a isso, é possível ter imagens de excelente qualidade e resoluções altas, inclusive maiores que as suportadas pela tecnologia DVI (Digital Visual Interface), que substituiu o padrão VGA para as conexões de monitores em computadores.

O conector do cabo HDMI também leva vantagem em relação aos demais padrões, já que possui tamanho reduzido e encaixe fácil, semelhante aos conectores USB. Sua pinagem pode ser conferida através da Figura 3.2 - Pinagem HDMI, a seguir.

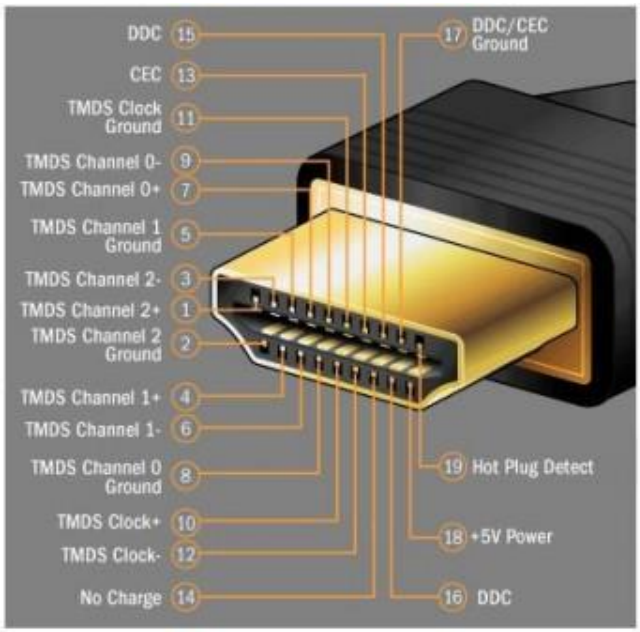


Figura 3.2 - Pinagem HDMI
Fonte: http://www.eletronica.com/hdmi-pinagem-dos-conectores

A indústria definiu dois tipos de conectores inicialmente: o HDMI tipo A e HDMI tipo B, com 19 e 29 pinos, respectivamente. O conector tipo A é o mais comum do mercado, já que consegue atender a toda a demanda existente, sendo inclusive compatível com a tecnologia DVI-D. Neste caso, basta que uma ponta do cabo seja DVI-D e, a outra, HDMI. O conector HDMI tipo B é destinado a resoluções mais altas e pode trabalhar com o esquema dual link, fazendo com que a transmissão dobre a sua capacidade.

Além dessas vantagens, utilizando-se HDMI é possível criar sistemas de *home theater* que se configuram automaticamente, e comandados por um único controle remoto, integrando aparelhos como DVD, CD, TV a cabo ou por satélite (HDMI, 2012).

3.2.2.1 - Versões

A tecnologia HDMI passou por várias revisões em suas especificações desde a disponibilização da primeira versão. A vantagem disso é que cada versão adiciona melhorias à tecnologia, sendo que atualmente já contamos com 4 versões:

- HDMI 1.0: lançado oficialmente no final de 2002, a primeira versão do HDMI é caracterizada por utilizar cabo único para transmissão de vídeo e áudio com um taxa de transmissão de dados de 4,95 Gb/s à uma frequência de 165 MHz. É possível ter até 8 canais de áudio;

- HDMI 1.1: semelhante à versão 1.0, porém com a adição de compatibilidade ao padrão DVD-Audio. Lançado em maio de 2004;

- HDMI 1.2: adicionado suporte a formatos de áudio do tipo One Bit Audio, usados, por exemplo, em SACD (Super Audio CD). Incluído suporte à utilização do HDMI em PCs e a novos esquemas de cores. Lançado em agosto de 2005;

- HDMI 1.2a: lançado em dezembro de 2005, esta revisão adotou as especificações Consumer Electronic Control (CEC) e recursos específicos para controle remoto;

- HDMI 1.3: nesta versão, o HDMI passou a suportar frequências de até 340 MHz, permitindo transmissões de até 10,2 Gb/s. Além disso, a versão 1.3 permite a

utilização de uma gama maior de cores e suporte às tecnologias Dolby TrueHD e DTS-HD Master Audio. Essa versão também possibilitou o uso de um novo miniconector (HDMI tipo C - mini), apropriado a câmeras de vídeo portáteis, e elimina um problema de sincronismo entre o áudio e o vídeo. O lançamento do HDMI 1.3 se deu em junho de 2006;

- HDMI 1.3a e 1.3b: lançado em novembro de 2006 e outubro de 2007, respectivamente, essas revisões contam com leves alterações nas especificações da versão 1.3 e com a adição de alguns testes, inclusive em relação ao HDCP[10].

- HDMI 1.4: esta versão foi anunciada em maio de 2009 e oferece tantas novidades que poderia até ser chamada de 2.0. Suas principais características são:

    - Capacidade de trabalhar com resoluções de até 4096x2160 pixels;

    - Compatibilidade com um número maior de cores;

    - Suporte a um canal de retorno de áudio (Audio Return Channel - ARC);

    - Possibilidade de transmissão por meio de conexões Ethernet de até 100 Mb/s (HDMI Ethernet Channel - HEC), permitindo que dispositivos interconectados compartilhem acesso à internet;

    - Melhor suporte para tecnologias de imagens em 3D;

    - Padronização para transmissão em veículos (aparelhos de DVD de ônibus, por exemplo).

E nesta última versão (1.4), ainda há novidades, pois o padrão traz consigo novos tipos de cabo:

- Standard HDMI Cable: cabo padrão que suporta transmissões de 1080i;

- High Speed HDMI Cable: cabo para transmissões de 1080p, incluindo suporte a um número maior de cores e tecnologias 3D;

- Standard HDMI Cable with Ethernet: cabo padrão com suporte à tecnologia Ethernet;

[10] High-bandwidth Digital Content Protection é uma tecnologia desenvolvida pela Intel que previne que conteúdo digital, áudio ou vídeo, seja copiada através das conexões.

- High Speed HDMI Cable with Ethernet: cabo para transmissões de alta velocidade com suporte à tecnologia Ethernet;
- Automotive HDMI Cable: cabo apropriado para transmissões em veículos.

Conforme mostra a Figura 3.3 - Conectores HDMI tipos A, C e D, o HDMI 1.4 também introduz um novo tipo de conector (HDMI tipo D - micro) de 19 pinos, que de tão pequeno pode ser facilmente utilizado em dispositivos portáteis, como câmeras digitais e smartphones.

Figura 3.3 - Conectores HDMI tipos A, C e D
Fonte: http://www.infowester.com/hdmi.php

3.3 - Qualidade de imagem e vídeo

Toda e qualquer mídia que deseje provocar impacto visual precisa se preocupar com qualidade de imagem. Este projeto visa criar as condições necessárias, tanto em *hardware* quanto em *software*, para que o anunciante possa veicular sua mídia em altíssima qualidade de imagem. Em suma, pretende-se permitir que o cliente tenha sua propaganda exibida em qualidade de *Blue-ray*, de forma a chocar o público com a nitidez e vividez das imagens, de forma a maximizar os resultados da campanha.

3.3.1 - Resolução

Quando o assunto é HDMI ou outras tecnologias relacionadas, como o HDTV - High-Definition Television, é comum a menção de resoluções como 720p e 1080p. Essas nomenclaturas simplesmente facilitam a identificação da quantidade de *pixels*[11] suportava pelo dispositivo, além do uso de *progressive scan* ou *interlaced scan*.

No *progressive scan*, todas as linhas de pixels da tela são exibidas simultaneamente, a cada quadro de imagem. Por sua vez, no modo *interlaced scan*, primeiro as linhas pares são exibidas, e em seguida, já no próximo quadro, as linhas ímpares são exibidas. De acordo com a Philips, o modo *progressive scan* oferece maior nitidez e definição de imagem.

Assim sendo, a letra 'p' existente em 720p, 1080p e outras resoluções indica que o modo usado é *progressive scan*. Se for utilizado *interlaced scan*, a letra aplicada é 'i' (por exemplo, 1080i). O número, por sua vez, indica a quantidade de linhas de pixels na vertical. Isso significa que a resolução 1080p, por exemplo, conta com 1080 linhas verticais e funciona com *progressive scan*. Algumas das resoluções mais comuns são:

- 480i = 640x480 pixels com *interlaced scan;*
- 480p = 640x480 pixels com *progressive scan;*
- 720i = 1280x720 pixels com *interlaced scan;*
- 720p = 1280x720 pixels com *progressive scan;*
- 1080i = 1920x1080 pixels com *interlaced scan;*
- 1080p = 1920x1080 pixels com *progressive scan.*

O termo *Full HD (High Definition)*, cuja tradução fidedigna seria "Alta Definição Máxima", indica que o aparelho trabalha na resolução máxima, que é de 1080p. Isso significa que o equipamento em questão será capaz de executar em qualidade máxima vídeos provenientes de um disco *Blu-ray*, por exemplo, que são feitos para este nível de resolução.

[11] Um ponto que representa a menor parte da imagem em uma tela

3.3.2 - Bit rate ou taxa de bits

Uma *bit rate* ou taxa de bits refere-se à quantidade de bits convertidos ou processados por unidade de tempo, medido em segundos (bps ou b/s). Representa a quantidade de informação ou detalhe guardada por unidade de tempo em uma gravação digital de áudio ou vídeo, e em linhas gerais, quanto maior a taxa de bits, melhor será a qualidade, pois mais bits estão sendo utilizados para representar a mesma imagem/áudio. Pode variar devido a fatores que dependem da finalidade da mídia:

- diferença de frequência de amostragem na digitalização do original;
- número de bits utilizados pelas amostragens;
- técnica utilizada para codificar os dados;
- técnica de ou grau utilizado na compressão da informação.

Quanto mais alta a taxa de bits, maior será a qualidade final e consequentemente o tamanho do arquivo. Porém, especialmente quando o aspecto é transmissão via internet, deve existir um balanceamento entre a qualidade e o tamanho da mídia, visto que o dado precisará trafegar pela rede.

Existem duas classificações para a de taxa de bits, a *Constant bitrate* (CBR) ou “taxa de bits constante”, onde um mesmo valor de taxa de bits é utilizado para todo o arquivo, e o *Variable bit rate* (VBR) ou "taxa de bits variável" onde há uma variação desse valor da taxa de bits ao longo do arquivo.

Quando um arquivo é codificado com VBR, a quantidade de informação guardada por segmento de tempo varia, economizando-se espaço em disco de forma mais eficiente. Uma taxa de bits maior é utilizada para segmentos mais complexos, com maior alteração em relação ao quadro anterior e que ocupam mais espaço em disco, como explosões e cenas rápidas, e uma taxa de bits reduzida para os segmentos de menor complexidade, ocupando menos espaço em disco.

Em linhas gerais, para um mesmo valor de taxa de bits, a VBR proporciona maior eficiência nas gravações digitais (áudio e vídeo), pois varia essa taxa em função da necessidade, gerando arquivos menores se comparada a VBR.

Dessa forma, consegue-se obter uma qualidade final superior, para um mesmo tamanho de arquivo, que será transmitido pela internet e necessita ser o menor possível (em termos de tamanho em bytes).

3.4 - Formatos de arquivo

Como se trata de uma mídia digital, é necessário manter compatibilidade com os principais formatos de arquivos disponíveis no mercado, sempre procurando atingir a melhor qualidade possível, sem consumo excessivo de banda. Ou seja, é preciso balancear a qualidade de imagem e/ou som em contrapartida ao tamanho do arquivo (em kbytes), pois é preciso lembrar que o mesmo será transmitido via internet.

Pretende-se que a solução a ser desenvolvida seja compatível com os seguintes tipos de arquivo de áudio e vídeo:

3.4.1 - Áudio Digital

Tabela 2 - Alguns formatos de áudio

EXTENSÃO	MAIS INFORMAÇÕES
.mp3	**MPEG Audio Layer 3** Esse é um arquivo de som compactado pela utilização do codec MPEG Audio Layer 3, desenvolvido pelo Instituto Fraunhofer.
.wma	**Windows Media Audio** Esse é um arquivo de som compactado pela utilização do codec Microsoft Windows Media Audio, um esquema de codificação de áudio digital desenvolvido pela Microsoft que é utilizado para se distribuir música gravada, geralmente pela Internet.

Fonte: http://office.microsoft.com/pt-br/powerpoint-help/formatos-de-arquivos-multimidia-compativeis-HA001230325.aspx.

3.4.2 - Vídeo digital

Tabela 3 - Alguns formatos de áudio

EXTENSÃO	MAIS INFORMAÇÕES
.avi	**Audio Video Interleave** Formato de arquivo multimídia para armazenamento de som e imagens em movimento no formato RIFF (Microsoft Resource Interchange File Format). É um dos formatos mais populares em função do conteúdo de áudio ou vídeo ser compactado por uma ampla variedade de *codecs*[12].
.mpg ou .mpeg	**Moving Picture Experts Group** Esse é um conjunto de padrões para compressão de áudio e vídeo desenvolvido por Moving Picture Experts Group. Esse formato de arquivo foi projetado especificamente para ser usado com mídias Video-CD e CD-i.
.wmv	**Windows Media Video** Esse formato de arquivo compacta áudio e vídeo por meio do *codec* Windows Media Video, um formato de compressão proprietário da Microsoft e não compatível a alguns aparelhos reprodutores de mesa.

Fonte: adaptada de http://office.microsoft.com/pt-br/powerpoint-help/formatos-de-arquivos-multimidia-compativeis-HA001230325.aspx.

3.5 - Framework .NET

Framework de software nada mais é do que um conjunto de classes criadas em uma linguagem específica, usadas para auxiliar o desenvolvimento de software. Obviamente, um framework atua onde existem funcionalidades em comum a várias aplicações, aumentando a produtividade da equipe como um todo, que irá reutilizar estas bibliotecas, sem a necessidade de recodificá-las.

Analogamente, o Framework .NET é um conjunto de componentes para Windows que suporta construção e execução de aplicações desktop e web. Ele fornece um ambiente de execução gerenciado, desenvolvimento e implantação simplificados e suporte para uma ampla variedade de linguagens de programação.

[12] Software que permite compressão de arquivos de áudio e/ou vídeo, com o intuito de reduzir o espaço em disco utilizado pelos mesmos, geralmente com o intuito de realizar uma transmissão dos mesmos pela internet. Existem vários codecs disponíveis no mercado, alguns proprietários e outros livres,

3.5.1 - Linguagens de Programação

A plataforma .NET baseia-se em um dos princípios utilizados na tecnologia Java: os programas desenvolvidos para ela são duplo-compilados (compilados duas vezes), uma na distribuição, gerando um código que é conhecido como bytecodes, e outra na execução.

Um programa é escrito em qualquer das mais de vinte linguagens de programação disponíveis para a plataforma, o código fonte gerado pelo programador é então compilado pela linguagem escolhida gerando um código intermediário em uma linguagem chamada Microsoft Intermediate Language (MSIL).

Este novo código fonte gera um arquivo na linguagem de baixo nível Assembly, de acordo com o tipo de projeto:

- EXE - Arquivos Executáveis, Programas
- DLL - Biblioteca de Funções
- ASPX - Página Web
- ASMX - Web Service

3.5.2 - Common Language Runtime (CLR)

Base comum a todas as linguagens escritas para a plataforma. O CLR é o ambiente que gerencia a execução de código escrito em qualquer linguagem e é responsável pelo gerenciamento da memória, da execução de código, e outros serviços do sistema.

Conforme ilustra a Figura 3.4 - Estrutura de linguagens do Framework .NET, toda aplicação .NET compilada é convertida para a Microsoft Intermediate Language (MSIL), também conhecida como Common Intermediate Language (CIL), que é uma linguagem intermediária composta por um conjunto de instruções independentes de CPU. Esta é a última linguagem de baixo nível legível para humanos do framework .NET.

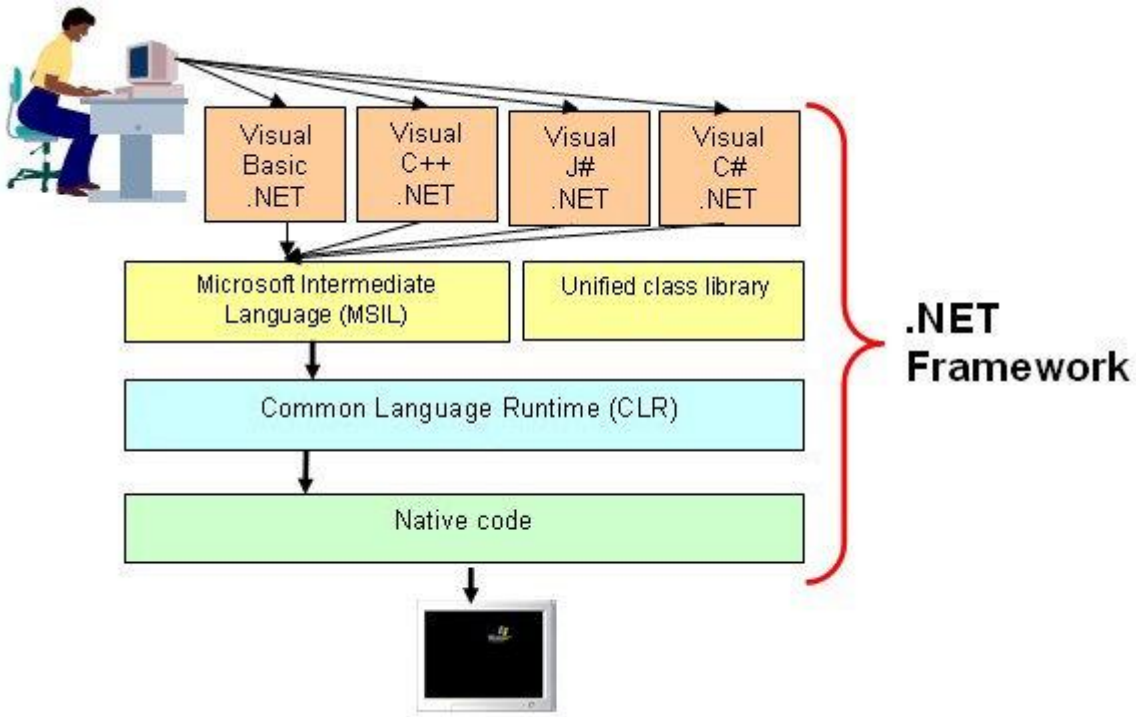


Figura 3.4 - Estrutura de linguagens do Framework .NET
Fonte: http://improve.dk/articles/dotnet/securing-dotnet-code/images/framework.jpg

Na hora da execução do programa, um novo compilador chamado Just-in-time Compiler (JIT), converte o MSIL para código nativo, específico para o processador da máquina, conforme ilustrado na Figura 3.5 - Compilação e execução no Framework .NET.

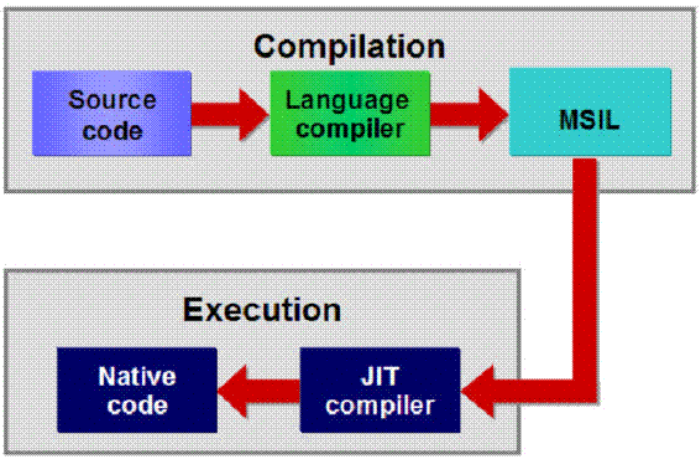


Figura 3.5 - Compilação e execução no Framework .NET
Fonte: http://www.c-sharpcorner.com/UploadFile/Dada%20Kalander/MigratingASPtoASP.NET11262005035157AM/Images/01.GIF

3.5.3 - Biblioteca de Classes

A biblioteca de classes .NET fornece acesso a todas as funcionalidades da CLR e é organizada em *namespaces*. Cada *namespace* contem um grupo de classes relacionadas, buscando facilidade de acesso as mesmas, como por exemplo o *namespace* System.IO, que aglomera todas as classes para realizar entrada e saída de dados, dentre elas, as responsáveis por escrita e leitura de arquivos.

Resumidamente, esta biblioteca é formada por classes reutilizáveis para desenvolver seus aplicativos de forma mais rápida e segura.

3.6 - WPF (Windows Presentation Foundation)

O WPF apareceu em 2001 com o codinome “Avalon”, como a nova tecnologia de apresentação para criação de aplicações Windows, com a prerrogativa de substituição do amplamente utilizado Windows Forms.

Sua capacidade de criar aplicativos visualmente inovadores, com janelas transparentes, vídeos e outros recursos diferenciados a tecnologia Windows Forms, e com excelente experiência de usuário, o elevaram a melhor opção para desenvolvimento de aplicativos "ricos" no ambiente .Net. Suas principais características são:

- Flexibilidade da interface, que pode ser independente do código: é possível ter duas apresentações completamente diferentes compartilhando o mesmo comportamento;
- Incorpora todas as funções já presentes no framework .Net, acrescentando novos recursos como 3D, animações, gráficos vetoriais, reconhecimento de voz, *layouts* avançados, entre outros;
- Traz para as aplicações desktop o conceito já existente na Web de separação entre o design e o código, permitindo que a interface seja criada por um *designer* e o código por um programador especializado, de maneira independente e eficiente;
- Usa diretamente recursos do sistema operacional, de maneira a otimizar a performance da interface para o *hardware* do usuário;

- Os controles podem ser personalizados, é possível, por exemplo, criar um botão não retangular que contém uma animação 3D, sem a necessidade de escrever um código específico para isso, e com um esforço mínimo.

- É independente de plataforma: o mesmo código-fonte funciona tanto na web (com o uso de Silverlight) quanto para uma aplicação *desktop*.

Um software desenvolvido em WPF é normalmente composto por duas partes: um arquivo XML com características especiais chamado XAML (eXtended Aplication Markup Language), e um código .Net (em qualquer uma das linguagens que o compilador permite: C#, Vb.Net, etc).

O arquivo XAML contém as diretrizes de interface, podendo ser comparado ao XHTML em relação a uma aplicação web. Por outro lado, o código .Net é responsável pelos processamento das requisições, seja ela um clique de botão ou um evento qualquer disparado pelo usuário.

O modelo de programação principal WPF é exposto através de código gerenciado[13]. De acordo com a Microsoft, no início do desenvolvimento do WPF houve uma série de debates sobre onde deveria ser traçada a linha divisória entre os componentes gerenciados do sistema e os não gerenciados. O CLR fornece uma série de características que tornam o desenvolvimento mais produtivo e robusto (incluindo gerenciamento de memória, tratamento de erros, o sistema de tipo de dados comuns, etc), contudo há um custo. Os principais componentes do WPF estão ilustrados na Figura 3.6 - Componentes do WPF.

[13] Código gerenciado é um termo criado pela Microsoft para se referir a qualquer código que precisa do gerenciamento do CLR para executar. Ou seja, que execute dentro dos limites do CLR, sujeito à suas regras de segurança, gerenciamento de memória e recursos, etc. Define-se por código não gerenciado aqueles softwares que acessam diretamente os recursos do sistema operacional, sem a intervenção da CLR, sendo por sua vez extremamente mais complexos e de execução mais rápida.

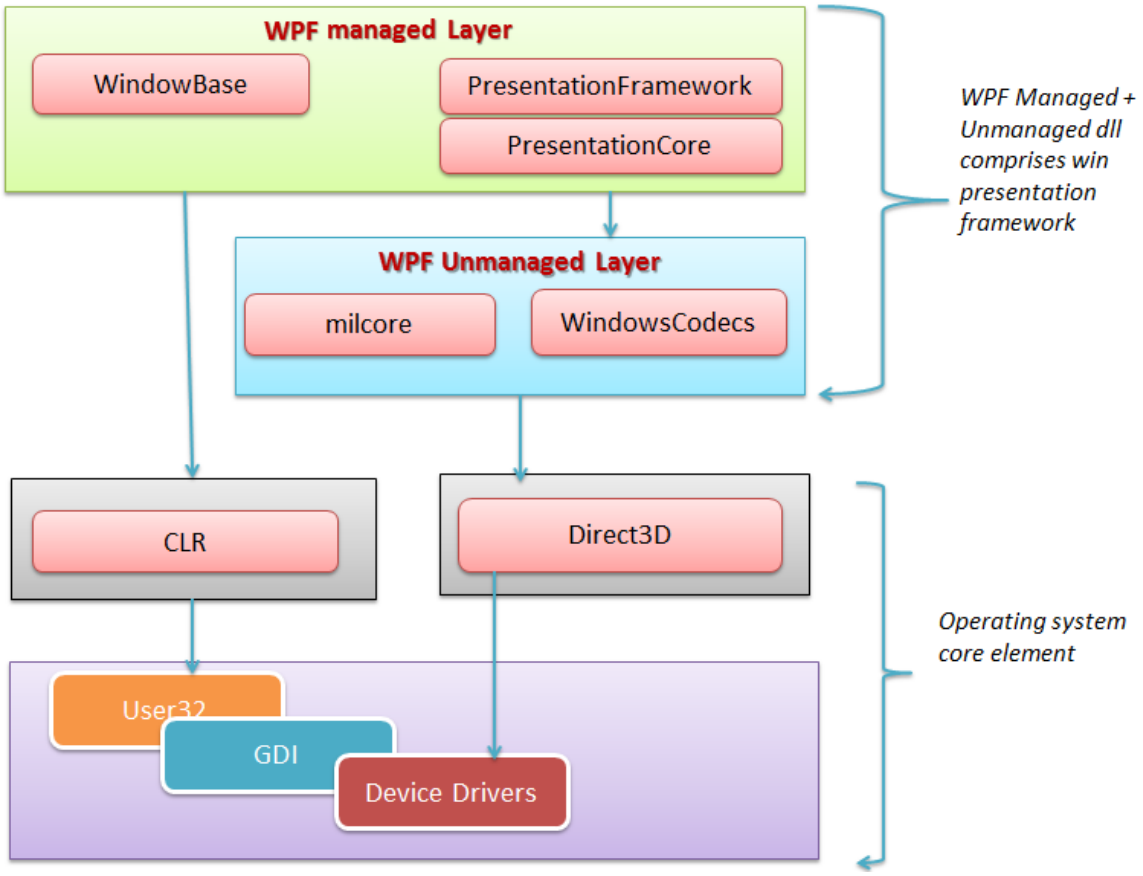


Figura 3.6 - Componentes do WPF
Fonte: http://www.codeproject.com/Articles/140611/WPF-Tutorial-Beginning

As bibliotecas PresentationFramework, PresentationCore e milcore representam as principais partes do código do WPF. Destas, apenas uma é um componente não gerenciado: milcore. Abreviação de Media Integration Layer, que significa Camada de Integração com Mídias, o milcore é escrito em código não gerenciado, a fim de permitir uma integração forte com o DirectX.

Todas as renderizações[14] em WPF são feitas através do milcore e do DirectX, permitindo uma utilização eficiente do hardware e renderização por software. Foi necessário também um rigoroso controle sobre a memória e a execução em milcore, logo esse mecanismo de renderização é extremamente voltado para a otimização do desempenho, e foi necessário abrir mão de muitas vantagens do CLR para obter esse ganho.

[14] Renderização em desenvolvimento de software se refere ao momento em que a aplicação é "desenhada", ou enviada, para a tela. Pode-se dizer que a renderização é a conversão do código em exibição em tela.

3.7 - ASP Net Membership

Da mesma forma que existe a necessidade de controlar o acesso a áreas restritas em empresas, ao desenvolver aplicações comerciais, uma das principais preocupações é o controle de acesso a informação, ou seja, a segurança dos dados.

Seja para um simples aplicativo de correio eletrônico, um site de banco, ou um complexo sistema corporativo, é importante que a informação não esteja acessível a todos os usuários, e sim apenas aqueles que realmente tem direito.

Na maioria dos casos, são criados perfis diferenciados para conceder o nível de segurança desejado a cada um dos usuários do sistema. Portanto, cada um destes perfis é um conjunto de permissões de acesso, seja a uma determinada funcionalidade, ou manutenção de dados.

Esse controle de segurança, contendo funcionalidades, perfis, usuários e suas permissões exige um esforço de codificação e modelagem de dados. A Microsoft, visando atender necessidades semelhantes de vários clientes criou uma equipe com o objetivo primário de construir uma biblioteca de componentes reutilizáveis, reduzindo de maneira significativa o tempo de codificação com a garantia de revisão e aprovação de equipes qualificadas da Microsoft e de inúmeros clientes ao redor do mundo.

Um dos componentes dessa biblioteca é o ASP Net Membership, cujo objetivo primário é assegurar a segurança no controle de acesso, permitindo cadastro de usuários, perfis e suas associações aos usuários.

Dotado de um modelo de dados específico, conforme ilustra a Figura 3.7 - Tabelas do Asp Net Membership, o ASP Net Membership armazena informações de segurança em tabelas próprias, que podem ser incorporadas ao modelo de dados do sistema que irá fazer uso do componente, ou até mesmo em um banco de dados específico para este fim, de forma mais corporativa.

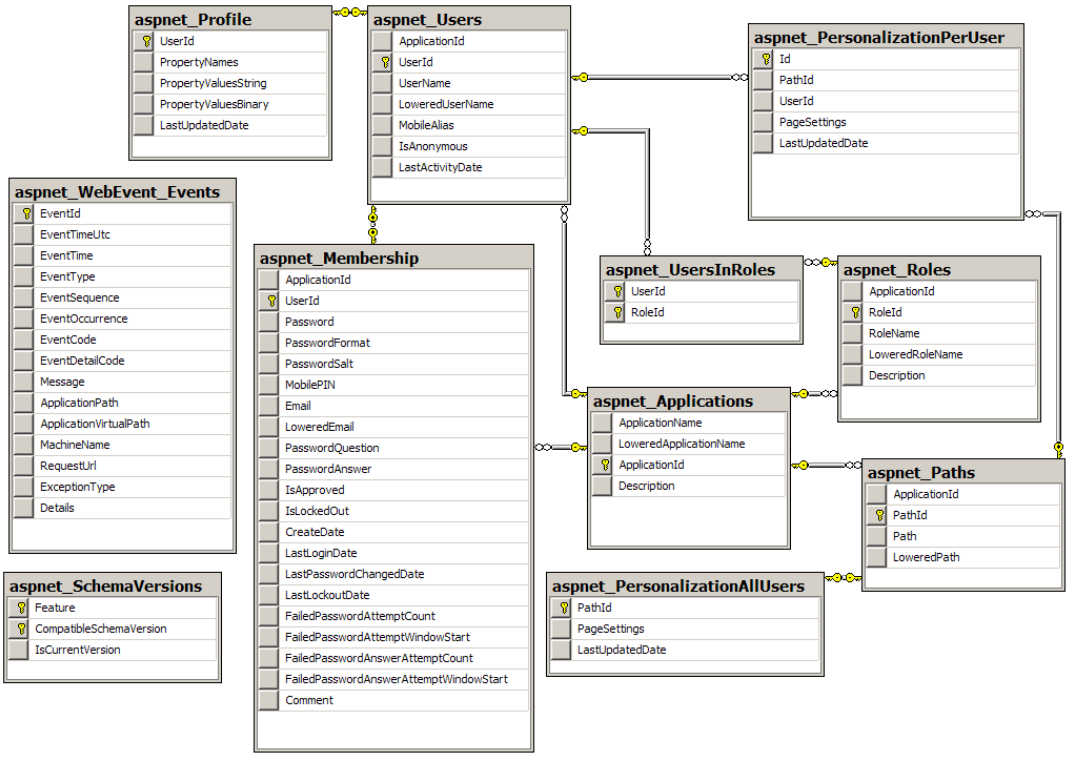


Figura 3.7 - Tabelas do Asp Net Membership

Há ainda, um website desenvolvido pela mesma equipe da Microsoft, que fornece uma interface intuitiva e de fácil uso para a gerência e configuração desses dados. Neste website, ilustrado na Figura 3.8 - Website do Asp Net Membership, é possível gerenciar usuários, perfis e associações entre os mesmos.

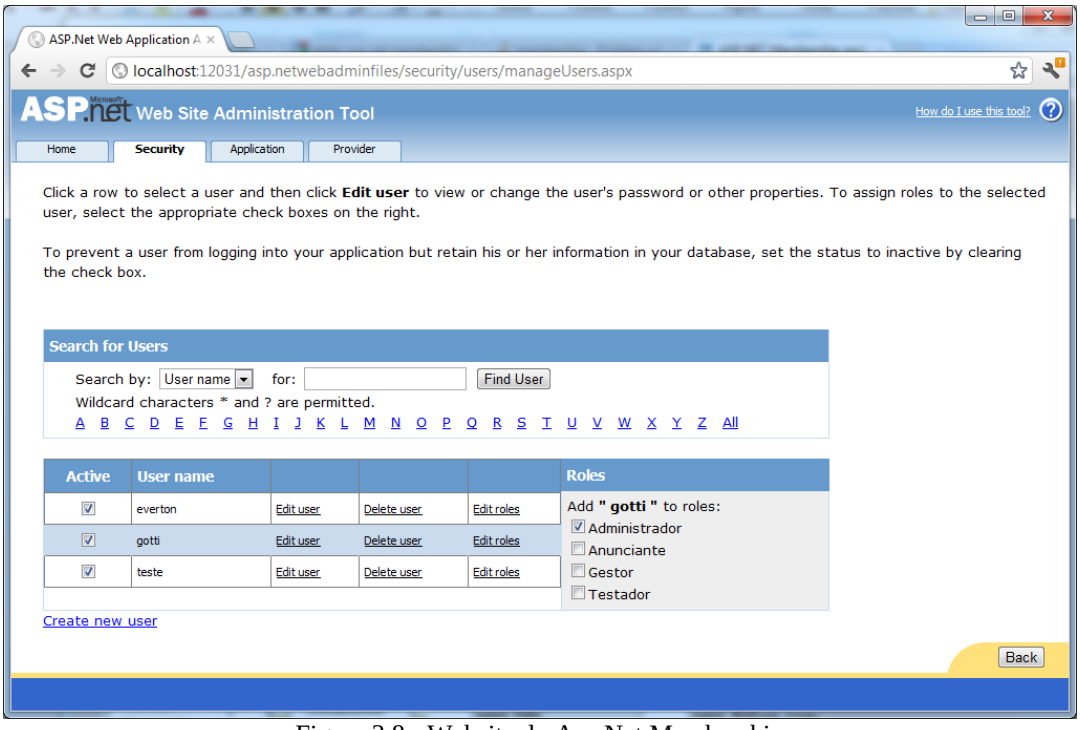


Figura 3.8 - Website do Asp Net Membership

3.8 - Entity Framework 4.0 (EF4)

O Entity Framework 4.0 é uma biblioteca de ORM produzida pela equipe da Microsoft para o programador não precisar se preocupar com a linguagem SQL.

ORM (Object-Relational Mapping ou Mapeamento Objeto Relacional) é uma técnica de desenvolvimento utilizada no mercado para realizar um mapeamento entre o banco de dados relacional e a programação orientada a objetos.

Comparada a outras técnicas de transferência de dados entre um banco de dados relacional e uma linguagem orientada a objetos, a abordagem ORM reduz significativamente a quantidade de código a ser produzida, elevando a produtividade da equipe como um todo.

Por ter sido criada logo acima do conhecido provedor ADO.NET, como mostra a Figura 3.9 - Arquitetura Entity Framework, o EF4 proporciona uma migração descomplicada, para aplicações legadas que ainda utilizem esse antigo provedor no acesso a dados e desejem migrar para Entity Framework 4.0.

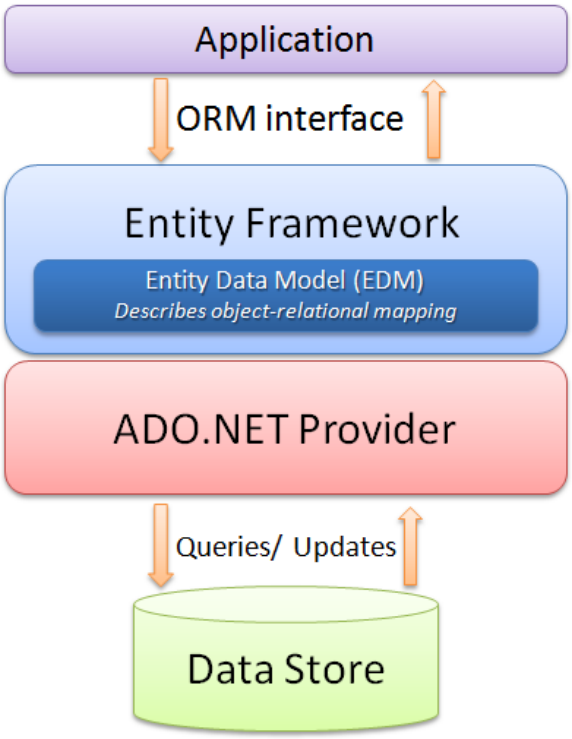


Figura 3.9 - Arquitetura Entity Framework
Fonte: http://msdn.microsoft.com/en-us/data/aa937709

O Entity Framework representa as tabelas do banco de dados como classes e os registros das tabelas como instâncias destas classes. As procedures e functions do banco de

dados são mapeadas como métodos. Assim, o EF4 cria efetivamente um “banco de dados virtual de objetos”.

3.9 - Generics

O *Generics* é um recurso de programação que existe desde a versão 2.0 do .Net Framework e introduziu o conceito de parâmetro de tipos, tornando possível a estruturação de classes e métodos que adiam a especificação de um ou mais tipos até que a classe ou método seja declarada e instanciada pelo código que irá usá-la.

A criação de uma classe genérica é realizada utilizando-se os símbolos < e >, ao redor de um tipo genérico, conforme ilustrado na Figura 3.10 - Generics .

C#

```
// Declare the generic class
public class GenericList<T>
{
    void Add(T input) { }
}

class TestGenericList
{
    private class ExampleClass { }
    static void Main()
    {
        // Declare a list of type int
        GenericList<int> list1 = new GenericList<int>();

        // Declare a list of type string
        GenericList<string> list2 = new GenericList<string>();

        // Declare a list of type ExampleClass
        GenericList<ExampleClass> list3 = new GenericList<ExampleClass>();
    }
}
```

Figura 3.10 - Generics
Fonte: http://msdn.microsoft.com/en-us/library/512aeb7t(v=vs.80).aspx

O framework .NET possui várias classes e métodos utilizando Generics, especialmente as que se encontram no *namespace* System.Collections.Generic. Porém é evidente que é permitido ao programador criar classes e métodos fazendo uso deste recurso, sendo possível ainda, criar padrões de desenvolvimento que sejam *type-safe*[15], ou seja, com tipagem segura.

[15] Quando uma linguagem possui tipagem segura, significa dizer que ela não permite que objetos executem ações que não são permitidas para aquele tipo de objeto. Em suma, a linguagem previne que erros de tipo aconteçam, como tratar um inteiro como número real e outros.

Além do objetivo primodial de adiar a especificação de um ou mais tipos, o Generics propicia o reuso[16] de código, sendo um excelente aliado na programação orientada a objetos.

3.10 - Visual Studio 2010

O Visual Studio 2010 é uma IDE (Integrated Development Environment ou Ambiente Integrado de Desenvolvimento), ou seja, é um programa que reúne características e ferramentas de apoio ao desenvolvimento de software com o objetivo de agilizar este processo.

As características e ferramentas mais comuns encontradas nas IDEs são:

- Editor - edita o código-fonte do programa escrito na(s) linguagem(ns) suportada(s) pela IDE;
- Compilador (*compiler*) - compila o código-fonte do programa, editado em uma linguagem específica e a transforma em linguagem de máquina;
- Linker - liga (linka) os vários "pedaços" de código-fonte, compilados em linguagem de máquina, em um programa executável que pode ser executado em um computador ou outro dispositivo computacional.
- Depurador (*debugger*) - auxilia no processo de encontrar e corrigir defeitos no código-fonte do programa, na tentativa de aprimorar a qualidade de software;
- Modelagem (*modeling*) - criação do modelo de classes, objetos, interfaces, associações e interações dos artefatos envolvidos no software com o objetivo de solucionar as necessidades-alvo do software final.
- Geração de código - característica mais explorada em Ferramentas CASE, a geração de código também é encontrada em IDEs, contudo com um escopo mais direcionado a *templates* de código comumente utilizados para solucionar problemas rotineiros. Todavia, em conjunto com ferramentas de modelagem, a geração pode gerar todo ou

[16] O reúso de software oferece vários benefícios para a empresa: aumento da produtividade, economia, agilidade, qualidade e eficiência, mas depende de uma arquitetura elaborada de maneira que propicie essa reutilização do código.

grande parte do código-fonte do programa com base no modelo proposto, tornando muito mais rápido o processo de desenvolvimento e distribuição do software;

- Distribuição (*deploy*) - auxilia no processo de criação do instalador do software, ou outra forma de distribuição, seja discos ou via internet.

- Testes Automatizados (*automated tests*) - realiza testes no software de forma automatizada, com base em scripts ou programas de testes previamente especificados, gerando um relatório, assim auxiliando na análise do impacto das alterações no código-fonte. Ferramentas deste tipo mais comuns no mercado são chamadas robôs de testes.

- Refatoração (*refactoring*) - consiste na melhoria constante do código-fonte do software, seja na construção de código mais otimizado, mais limpo e/ou com melhor entendimento pelos envolvidos no desenvolvimento do software. A refatoração, em conjunto com os testes automatizados, é uma poderosa ferramenta no processo de erradicação de "bugs".

O Visual Studio possui uma interface semelhante a de um editor de textos comum, conforme ilustra a Figura 3.11 - Microsoft Visual Studio 2010, e é a ferramenta padrão para desenvolvimento de software nas tecnologias e linguagens .NET. É um produto Microsoft e juntamente com o Team Foundation Server, permite todo o gerenciamento do ciclo de vida do software, incluindo o levantamento de requisitos, planejamento, codificação, testes, etc.

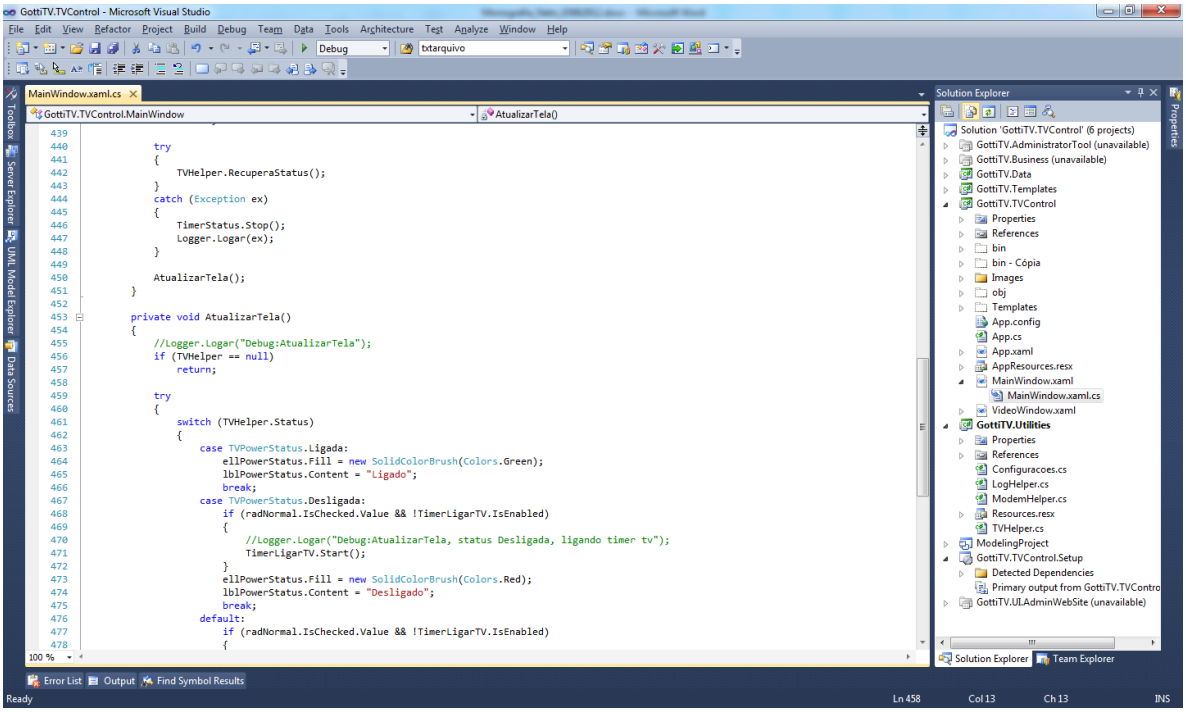


Figura 3.11 - Microsoft Visual Studio 2010

3.11 - Administração remota

Dentre as possibilidades de acesso remoto a computadores, encontra-se o Virtual Network Computing (VNC). Utilizando uma área de trabalho virtual e o protocolo RFB, softwares desse tipo permitem o uso de todas as funcionalidades de um computador a partir de outro, e toda a interface utilizada em um computador reflete-se no outro, incluindo o uso de mouse e teclado.

Todos os painéis eventualmente necessitam de manutenção e se esta puder ser efetuada remotamente, o tempo de interrupção dos serviços com certeza será reduzido sensivelmente.

No mercado existem vários softwares para esse tipo de administração remota, incluindo softwares livres. Portanto, esta solução não pretende desenvolver um software para administração remota, e sim fazer uso de algum aplicativo que seja disponível no mercado e não gere custos de licença.

CAPÍTULO 4 - MODELO PROPOSTO

A solução proposta, detalhada neste capítulo, é composta de *hardware* (uma TV e um computador) e dois *softwares* desenvolvidos na plataforma .NET.

4.1 - Hardware

4.1.1 - TV

O aparelho televisor utilizado na solução foi o modelo 42LE5300 da marca LG, de excelente qualidade de imagem devido a sua alta taxa de contraste, 3.000.000 por 1, taxa atingida atualmente apenas por modelos que utilizam tecnologia LED, e resolução *full-hd*, 1920x1080 pixels.

Porém, desde que seja um televisor do mesmo fabricante (LG), seja de plasma, LCD ou LED, e que possua as conexões necessárias (RS-232 e HDMI), é provável que não será necessária nenhuma modificação no sistema, pois foi possível verificar que a conexão serial dessa marca dispõe dos mesmos comandos básicos em vários modelos.

Não é escopo dessa solução a compatibilidade com todas as marcas e modelos de aparelhos televisores disponíveis no mercado, especialmente porque nem todos os equipamentos dispõe de porta serial para este tipo de controle. Todavia, alguns parâmetros ficarão disponíveis para configuração, já levando em consideração essa necessidade imposta pelo mercado.

4.1.1.1 - Automação via porta serial (RS-232)

A automação do aparelho televisor consiste basicamente em ligar e desligar o equipamento em horários pré-definidos, de acordo com o horário de funcionamento do local de instalação do painel, geralmente um estabelecimento parceiro com grande fluxo de pessoas.

Inicialmente, procurou-se atingir esse objetivo através do método mais simples: a configuração dos horários para ligar e desligar a TV no menu Hora, da próprio aparelho televisor, como ilustra a Figura 4.1 - Menu Hora da TV.

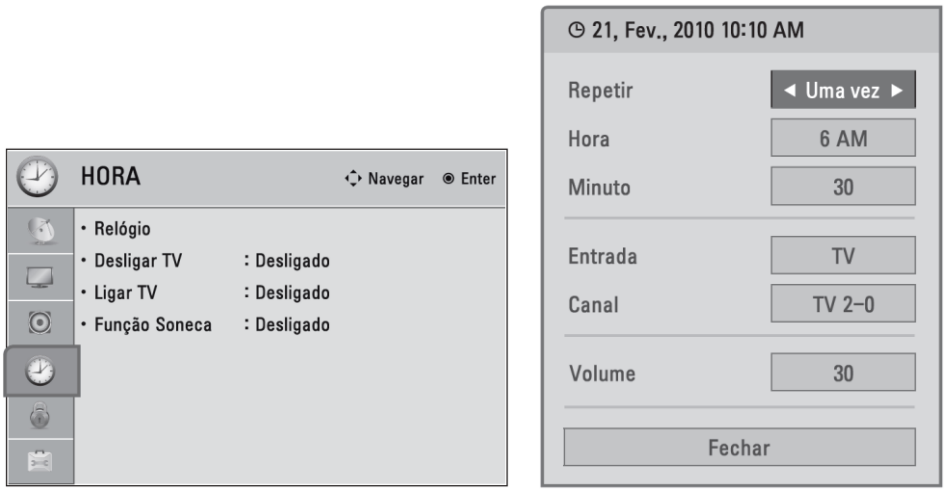
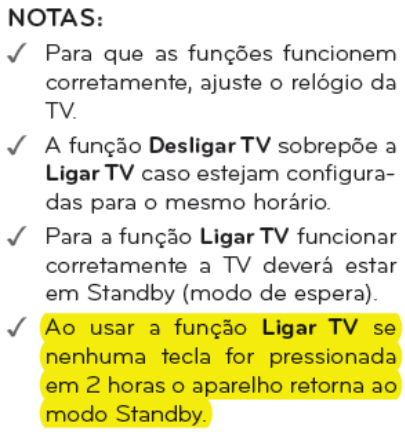


Figura 4.1 - Menu Hora da TV
Fonte: manual da TV, páginas 36 e 37

Porém, observou-se que após duas horas que o aparelho televisor se encontrava ligado, ele desligava automaticamente, sem uma explicação lógica.

Depois de várias pesquisas, uma leitura detalhada do manual da TV (veja notas do manual na Figura 4.2 - Notas da função Desligar/Ligar TV) e vários contatos com o fabricante, evidenciou-se que como medida preventiva, o *firmware* do aparelho é programado para desligar a TV, caso haja um período de inatividade de duas horas, sem pressionar nenhuma tecla do controle ou do painel da TV, e apenas quando a TV é programada para ligar automaticamente através de opção definida em seu menu.


NOTAS:

- ✓ Para que as funções funcionem corretamente, ajuste o relógio da TV.
- ✓ A função **Desligar TV** sobrepõe a **Ligar TV** caso estejam configuradas para o mesmo horário.
- ✓ Para a função **Ligar TV** funcionar corretamente a TV deverá estar em Standby (modo de espera).
- ✓ Ao usar a função **Ligar TV** se nenhuma tecla for pressionada em 2 horas o aparelho retorna ao modo Standby.

Figura 4.2 - Notas da função Desligar/Ligar TV
Fonte: manual da TV, página 37

Devido a este problema, decidiu-se controlar a TV através da porta serial, como ilustra a Figura 4.3 - Conexão do PC com a TV, via porta serial utilizando o software cliente que foi desenvolvido, instalado em cada um dos painéis, e comandos específicos para ligar e desligar o equipamento em horários predefinidos.

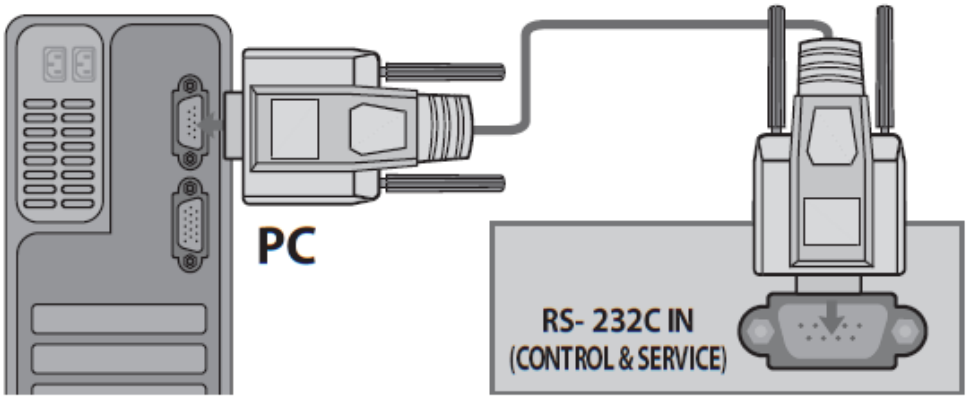


Figura 4.3 - Conexão do PC com a TV, via porta serial
Fonte: manual da TV, página 56

A conexão serial com o aparelho televisor só é possível através da confecção de um cabo específico para este fim, que utiliza sete fios com o esquemático definido na Figura 4.4 - Esquemático do cabo para conexão via porta serial .

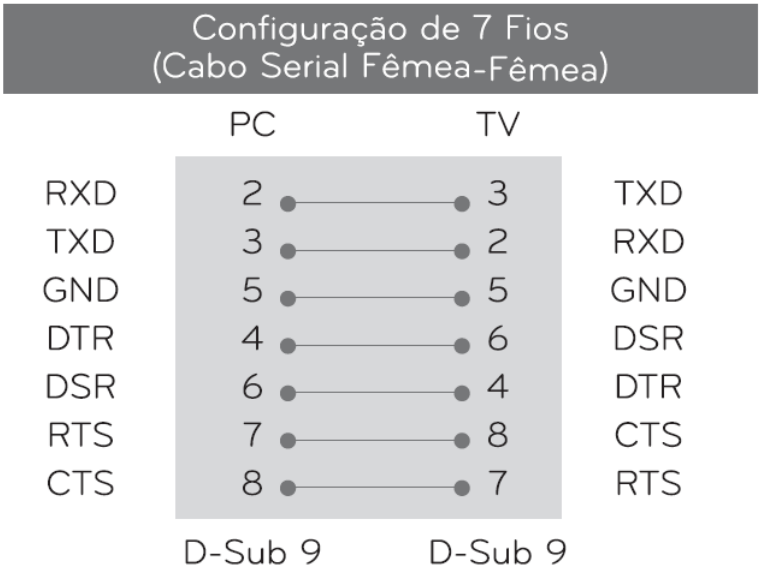


Figura 4.4 - Esquemático do cabo para conexão via porta serial
Fonte: manual da TV, página 56

Com o esquemático em mãos, escolheu-se um cabo com aproximadamente dois metros de comprimento e oito fios internos flexíveis, conforme ilustra a Figura 4.5 - Cabo serial confeccionado para conexão com a TV, com o intuito de evitar rupturas na solda com o conector, que ocorreu ao utilizar cabos de rede convencionais, em decorrência da rigidez dos fios internos do mesmo.

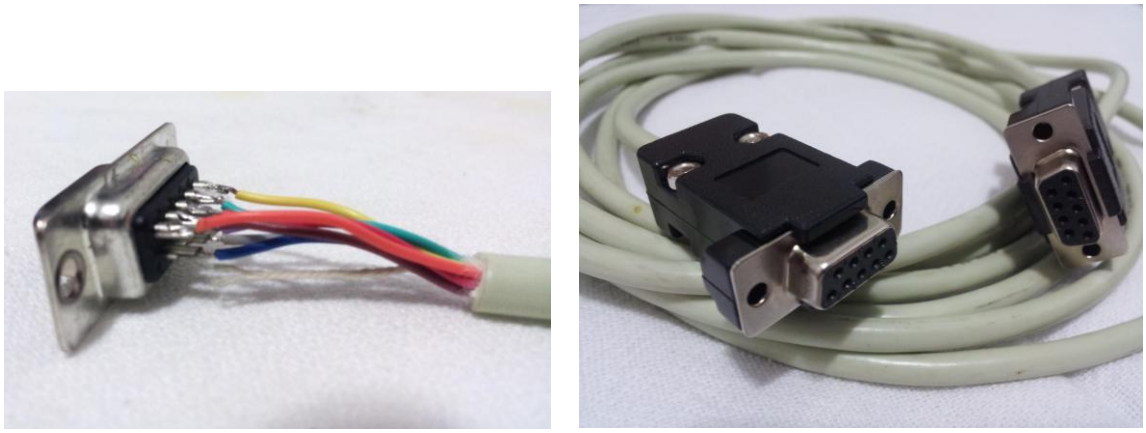


Figura 4.5 - Cabo serial confeccionado para conexão com a TV

Os comandos seriais para automação da TV são compostos de duas letras e em alguns casos uma informação adicional (em formato hexadecimal), conforme a Tabela 4 - Comandos seriais para controle da TV.

Tabela 4 - Comandos seriais para controle da TV

FUNÇÃO	COMANDO 1	COMANDO 2	DADOS (Hexa)
01. Power	k	a	00 - 01
02. Input Select	x	b	-
03. Aspect Ratio	k	c	-
04. Screen Mute	k	d	00 - 01
05. Volume Mute	k	e	00 - 01
06. Volume Control	k	f	00 - 64
07. Contrast	k	g	00 - 64
08. Brightness	k	h	00 - 64
09. Color	k	i	00 - 64
10. Tint	k	j	00 - 64
11. Sharpness	k	k	00 - 64
12. OSD Select	k	l	00 - 01
13. Remote Control Lock Mode	k	m	00 - 01
14. Treble	k	r	00 - 64
15. Bass	k	s	00 - 64
16. Balance	k	t	00 - 64
17. Color Temperature	x	u	00 - 64
18. ISM Method	j	p	-
19. Energy Saving	j	q	-
20. Auto Configuration	j	u	-

Fonte: manual da TV, página 57

4.2 - Banco de dados

Para a solução proposta, utilizou-se o SQL Server 2008 devido a sua integração nativa com a plataforma .NET. Porém, esta não é uma escolha obrigatória e é perfeitamente possível utilizar outro aplicativo de banco de dados, como o Oracle, MySQL, Postgree, etc.

O modelo de dados do sistema, ilustrado na Figura 4.6 - Modelo de dados , não inclui as tabelas do ASP Net Membership, pois o detalhamento deste não é escopo desta solução, que apenas faz uso do mesmo para controle de acesso ao software web.

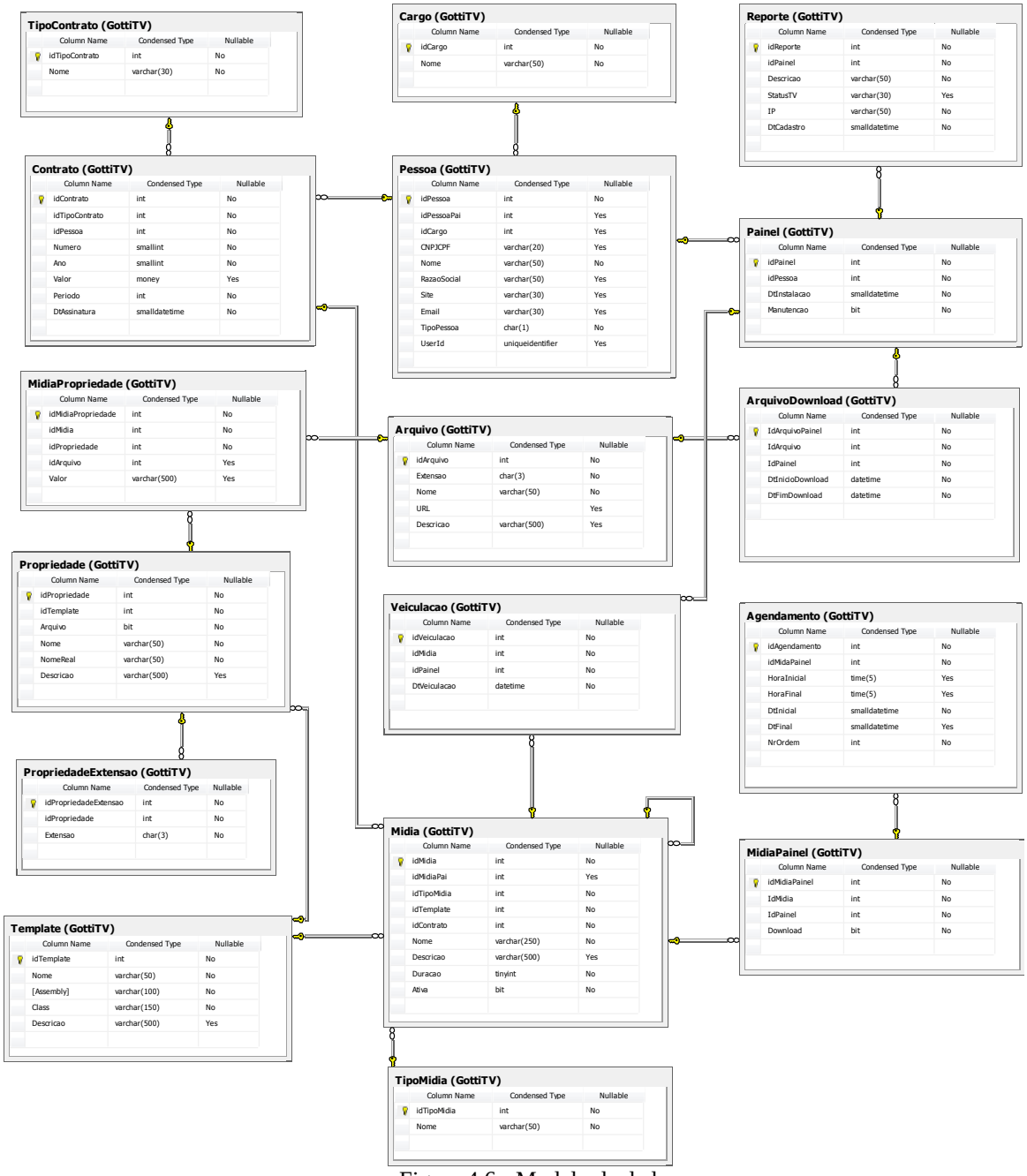


Figura 4.6 - Modelo de dados

4.2.1 - Dicionário de dados

A Tabela 5 – Dicionário de dados, assim como no modelo de dados, não inclui detalhamento sobre as tabelas do ASP Net Membership, pois conforme mencionado, o mesmo é utilizado apenas como um recurso para controle de acesso no software web.

Tabela 5 – Dicionário de dados

TipoContrato			
Propriedades	Tipo	Tamanho	Descrição
IdTipoContrato	Int		Código do tipo de contrato
Nome	Varchar	30	Nome do tipo de contrato

ArquivoDownload			
Propriedades	Tipo	Tamanho	Descrição
IdArquivoPainel	Int		Código do arquivopainel
IdArquivo	int		Código do arquivo
IdPainel	int		Código do painel
DtInicioDownload	smalldatetime		Data do início do download
DtFimDownload	smalldatetime		Data do fim do download

Arquivo			
Propriedades	Tipo	Tamanho	Descrição
IdArquivo	int		Código do arquivo
Extensão	char	3	Extensão do arquivo
Nome	Varchar	50	Nome do arquivo
URL	Varchar	250	Endereço de localização do arquivo
Descrição	Varchar	500	Descrição do arquivo

Reporte			
Propriedades	Tipo	Tamanho	Descrição
IdReporte	Int		Código do reporte
IdPainel	Int		Código do painel
Descrição	Varchar	50	Descrição do reporte
StatusTV	Varchar	30	Status do painel eletrônico
IP	Varchar	50	Endereço IP do painel
DtCadastro	smalldatetime		Data do cadastro do reporte

Agendamento			
Propriedades	Tipo	Tamanho	Descrição
IdAgendamento	Int		Código do agendamento
IdMidiaPainel	Int		Código da midiapainel
HoraInicial	Time	5	Horário inicial do agendamento

HoraFinal	Time	5	Horário final do agendamento
DtInicial	smalldatetime		Data inicial do agendamento
DtFinal	smalldatetime		Data final do agendamento
NrOrdem	Int		Número de ordem do agendamento

Contrato			
Propriedades	Tipo	Tamanho	Descrição
IdContrato	Int		Código do contrato
IdTipoContrato	Int		Código do tipo de contrato
IdPessoa	Int		Código da pessoa
Numero	Smallint		Numero do contrato
Ano	Smallint		Ano de contrato
Valor	Money		Valor do contrato
Período	Int		Período de vigência do contrato
DtAssinatura	smalldatetime		Data de assinatura do contrato

Painel			
Propriedades	Tipo	Tamanho	Descrição
IdPainel	Int		Código do painél
IdPessoa	Int		Código da pessoa
DtInstalação	smalldatetime		Data da instalação do painel
Manutenção	Bit		Chave para manutenção remota do painel

MidiaPropriedade			
Propriedades	Tipo	Tamanho	Descrição
idMidiaPropriedade	Int		Código da midiapropriedade
IdMidia	Int		Código da mídia
IdPropriedade	Int		Código da propriedade
IdArquivo	Int		Código do arquivo
Valor	Varchar	500	Valor da mídia

MidiaPainel			
Propriedades	Tipo	Tamanho	Descrição
IdMidiaPainel	Int		Código da midiapainel
IdMidia	Int		Código da mídia
IdPainel	Int		Código do painel
Download	Bit		Download da mídia para o painel

Template			
Propriedades	Tipo	Tamanho	Descrição
IdTemplate	Int		Código do template
Nome	Varchar	50	Nome do template
Assembly	Varchar	100	Nome da dll do template

Class	Varchar	150	Nome da classe do template
Descricao	Varchar	500	Descrição do template

Midia			
Propriedades	Tipo	Tamanho	Descrição
IdMidia	Int		Código da mídia
IdMidiaPai	Int		Código da mídia principal
IdTipoMidia	Int		Código de tipo de mídia
IdTemplate	Int		Código do template
IdContrato	Int		Código do contrato
Nome	Varchar	250	Nome da mídia
Descricao	Varchar	500	Descrição da mídia
Duracao	Tinyint		Duração da mídia
Ativa	Bit		Ativação da mídia

Veiculacao			
Propriedades	Tipo	Tamanho	Descrição
IdVeiculacao	Int		Código da veiculação
IdMidia	Int		Código da mídia
IdPainel	Int		Código do painel
DtVeiculacao	Datetime		Data da veiculação

TipoMidia			
Propriedades	Tipo	Tamanho	Descrição
IdTipoMidia	Int		Código do tipo de mídia
Nome	Varchar	50	Nome do tipo de mídia

Propriedade			
Propriedades	Tipo	Tamanho	Descrição
IdPropriedade	Int		Código da propriedade
IdTemplate	Int		Código do template do arquivo
Nome	Varchar	50	Nome do arquivo
NomeReal	Varchar	50	Nome real do arquivo
Descricao	Varchar	500	Descrição do arquivo
Arquivo	Varchar	50	Arquivo da propriedade

PropriedadeExtensao			
Propriedades	Tipo	Tamanho	Descrição
idPropriedadeExtensao	Int		Código da extensão da propriedade
IdPropriedade	Int		Código da propriedade
Extensao	Char	3	Extensão do arquivo

Pessoa			
Propriedades	Tipo	Tamanho	Descrição
IdPessoa	Int		Código da pessoa
IdPessoaPai	Int		Código da pessoa principal
CNPJCPF	Varchar	20	Cpf ou cnpj da pessoa
Nome	Varchar	50	Nome da pessoa/Nome fantasia
RazaoSocial	Varchar	50	Razão social (pessoa jurídica)
Site	Varchar	20	Site da pessoa (física ou jurídica)
Email	Varchar	30	Email da pessoa
Tipo Pessoa	Char	1	Tipo de pessoa
IdCargo	Int		Código do cargo da pessoa
UserId	uniqueidentifier		Código do usuário (proveniente da tabela do Asp Net Membership)

Cargo			
Propriedades	Tipo	Tamanho	Descrição
IdCargo	Int		Código do cargo
Nome	Varchar	50	Nome do cargo

4.3 - Arquitetura de camadas

De acordo com Fowler (2003), a comunidade de programação orientada a objetos forneceu a solução para o problema das regras de negócio: utilizar um modelo de três camadas. Nessa abordagem, há uma camada de apresentação para a UI[17] (*User Interface*, ou interface com o usuário), uma camada de domínio, para as regras de negócio, e uma camada de acesso a dados. Dessa maneira, todas as regras de negócio intrínsecas são removidas da UI e colocadas em uma camada onde é possível estruturá-las corretamente.

Esse modelo de arquitetura foi utilizado para a solução proposta, seguindo os princípios da programação orientada a objetos e as boas práticas de mercado, ou seja, o desenvolvimento foi estruturado em camadas de código propiciando o reuso e reduzindo o acoplamento[18].

Assim, como ilustrado pela Figura 4.7 - Arquitetura da solução, a arquitetura definida para a solução proposta permite o reuso das camadas de negócio e de acesso a dados em ambos os softwares desenvolvidos.

[17] Camada de interação com o usuário, como um aplicativo web (website), ou a porção de código referente a tela de um software cliente.
[18] O acoplamento significa o quanto uma classe, ou uma porção de código, depende da outra para funcionar. O forte acoplamento torna muito oneroso a sua manutenção e o seu gerenciamento, pois qualquer mudança vai afetar toda a cadeia de classes.

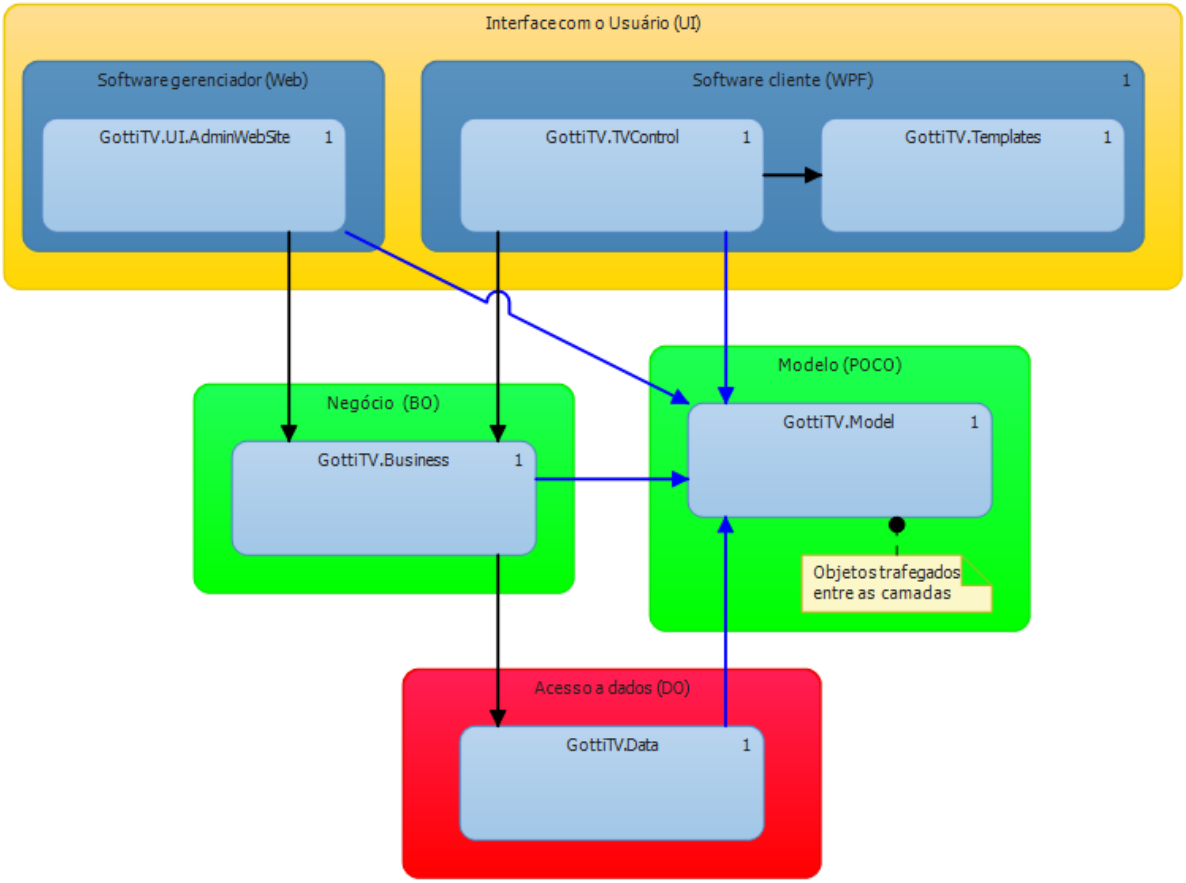


Figura 4.7 - Arquitetura da solução

4.3.1 - Camada de modelo (POCO[19])

A camada de modelo contém todas as classes que representam entidades de banco e são trafegadas entre as outras camadas. Estas classes não contém métodos ou códigos complexos, pois são apenas *containers* de dados, com o propósito único de representar informações que ainda serão ou já foram persistidas no banco de dados.

Nesta camada ocorre uma interação entre o Entity Framework 4.0 e o SQL Server 2008 para definir o diagrama de classes ilustrado na Figura 4.8 - Diagrama de classes, que será utilizado por toda a solução proposta.

[19] Plain Old CLR Object é uma analogia ao termo POJO (Plain Old Java Object) e se refere a uma classe simples, com pouca ou nenhuma codificação adicional, com o propósito de representar uma entidade do banco de dados, para utilização em frameworks de ORM.

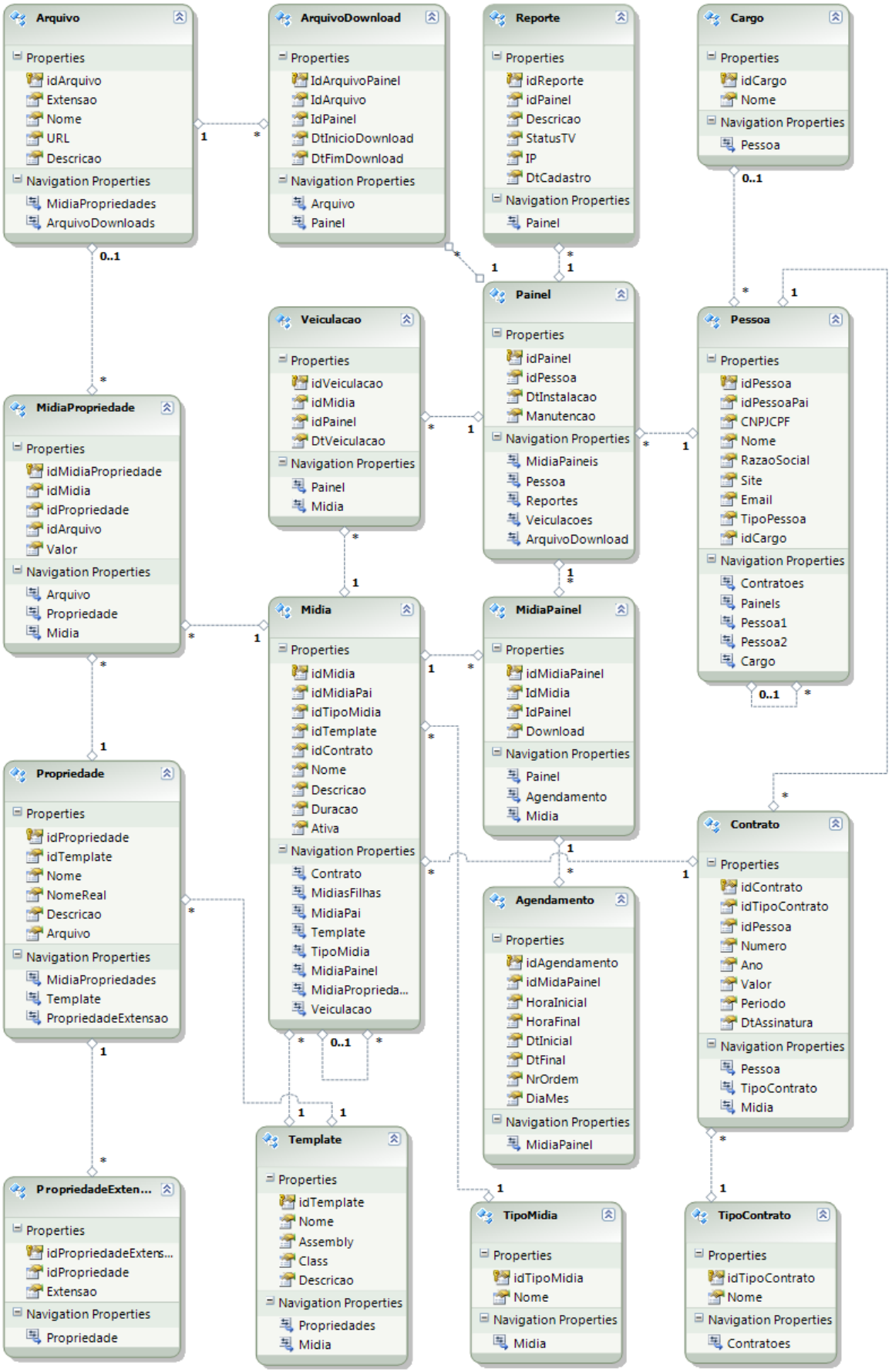


Figura 4.8 - Diagrama de classes

4.3.2 - Camada de acesso à dados (DO)

A camada de acesso a dados é única para os softwares desenvolvidos, unificando os procedimentos de persistência em um único código. O Entity Framework 4.0 foi utilizado como ferramenta de ORM pela facilidade de comunicação com o SQL Server 2008 e procedimento de sincronização simplificado, em caso de mudanças no banco de dados.

Para cada entidade do banco de dados, que fica mapeada para uma classe do código, uma outra classe com sufixo DO (Data Object ou Objeto de dados) é criada. Nesta classe fica a implementação, usando o Entity Framework, que é responsável pela persistência dos dados.

De maneira a prover a reutilização de código e aumento na produtividade, utilizou-se Generics para criar uma classe base, da qual todas as classes desta camada herdarão. Esta classe, conforme mostra a Figura 4.9 - Classe BaseDO, já contem toda a implementação para os procedimentos mais comuns de uma dada entidade: inserção, alteração, consulta e recuperação.

C#

```
public abstract class BaseDO<T> : IDisposable, IOperations<T> where T : class
{
    public String NomeEntidade { get; set; }

    private GottiTVContext contexto;
    protected GottiTVContext Contexto
    {
        get
        {
            if (contexto == null)
                contexto = new GottiTVContext();

            return contexto;
        }
        set
        {
            contexto = value;
        }
    }

    public T Carregar(T pEntidadeFiltro)
    {
        return Carregar(pEntidadeFiltro, false);
    }

    public T Carregar(T pEntidadeFiltro, bool useOnlyKey)
    {
        object entRetorno = null;
        try
        {
            string entitySetName = Contexto.CreateObjectSet<T>().EntitySet.Name;
```

```
            EntityKey entKey = Contexto.CreateEntityKey(entitySetName,
pEntidadeFiltro);

            foreach (EntityKeyMember item in entKey.EntityKeyValues)
            {
                if (!item.Value.Equals(0) || useOnlyKey)
                {
                    Contexto.TryGetObjectByKey(entKey, out entRetorno);
                    break;
                }
            }

            if (entRetorno == null && !useOnlyKey)
            {
                ObjectSet<T> objSet = Contexto.CreateObjectSet<T>();
                entRetorno = objSet.FirstOrDefault(Filtrar(pEntidadeFiltro));
            }
        }
        catch (Exception ex)
        {
            throw new Exception(string.Format(Resources.ERRO_012, NomeEntidade), ex);
        }
        return (T)entRetorno;
    }


    public List<T> Buscar()
    {
        List<T> lstRetorno = null;

        try
        {
            ObjectSet<T> objSet = Contexto.CreateObjectSet<T>();
            lstRetorno = objSet.ToList();
        }
        catch (Exception ex)
        {
            throw new Exception(string.Format(Resources.ERRO_015, NomeEntidade), ex);
        }

        return lstRetorno;
    }

    public List<T> Buscar(T pEntidadeFiltro)
    {
        List<T> lstRetorno = null;

        try
        {
            ObjectSet<T> objSet = Contexto.CreateObjectSet<T>();
            lstRetorno = objSet.Where(Filtrar(pEntidadeFiltro)).ToList();
        }
        catch (Exception ex)
        {
            throw new Exception(string.Format(Resources.ERRO_015, NomeEntidade), ex);
        }
        return lstRetorno;
    }

    public abstract Expression<Func<T, bool>> Filtrar(T pEntidadeFiltro);

    public void Salvar(T pEntidadeSalvar)
```

```
    {
        try
        {
            if (Carregar(pEntidadeSalvar, true) == null)
                Contexto.CreateObjectSet<T>().AddObject(pEntidadeSalvar);
            else
                Contexto.CreateObjectSet<T>().ApplyCurrentValues(pEntidadeSalvar);

            Contexto.SaveChanges();
        }
        catch (Exception ex)
        {
            throw new Exception(string.Format(Resources.ERRO_013, NomeEntidade), ex);
        }
    }

    public void Salvar(List<T> pEntidadesSalvar)
    {
        try
        {
            foreach (T item in pEntidadesSalvar)
            {
                if (Carregar(item) == null)
                    Contexto.CreateObjectSet<T>().AddObject(item);
                else
                    Contexto.CreateObjectSet<T>().ApplyCurrentValues(item);
            }

            Contexto.SaveChanges();
        }
        catch (Exception ex)
        {
            throw new Exception(string.Format(Resources.ERRO_013, NomeEntidade), ex);
        }
    }

    public void Excluir(T pEntidadeExcluir)
    {
        try
        {
            pEntidadeExcluir = Carregar(pEntidadeExcluir, true);

            if (pEntidadeExcluir != null)
                Contexto.CreateObjectSet<T>().DeleteObject(pEntidadeExcluir);

            Contexto.SaveChanges();
        }
        catch (Exception ex)
        {
            if (ex.GetType().BaseType.Name.Equals("DataException"))
                throw new Exception(Resources.ERRO_017);
            else
                throw new Exception(string.Format(Resources.ERRO_016, NomeEntidade),
ex);
        }
    }
}
```

Figura 4.9 - Classe BaseDO

Dessa forma, ao criar as classes DO para cada entidade do sistema, e em face destas herdarem da classe BaseDO, não é necessário realizar toda a implementação de persistência de dados, ficando apenas a responsabilidade de codificar a parte inerente aquela entidade, como demonstrado na Figura 4.10 - Classe PessoaDO.

C#

```
public class PessoaDO : BaseDO<Pessoa>
{
    public PessoaDO()
    {
        NomeEntidade = "Pessoa";
    }

    public override Expression<Func<Pessoa, bool>> Filtrar(Pessoa pEntidadeFiltro)
    {
        Expression<Func<Pessoa, bool>> funcao = p => true;

        if (pEntidadeFiltro.idPessoa > 0)
            funcao = funcao.AndAlso(p => p.idPessoa == pEntidadeFiltro.idPessoa);

        if (!string.IsNullOrEmpty(pEntidadeFiltro.Nome))
            funcao = funcao.AndAlso(p => p.Nome.Contains(pEntidadeFiltro.Nome));

        if (!string.IsNullOrEmpty(pEntidadeFiltro.TipoPessoa))
            funcao = funcao.AndAlso(p =>
p.TipoPessoa.Equals(pEntidadeFiltro.TipoPessoa));

        if (!string.IsNullOrEmpty(pEntidadeFiltro.CNPJCPF))
            funcao = funcao.AndAlso(p => p.CNPJCPF.Equals(pEntidadeFiltro.CNPJCPF));

        if (!string.IsNullOrEmpty(pEntidadeFiltro.Email))
            funcao = funcao.AndAlso(p => p.Email.Contains(pEntidadeFiltro.Email));

        if (pEntidadeFiltro.idPessoaPai > 0)
            funcao = funcao.AndAlso(p => p.idPessoaPai ==
pEntidadeFiltro.idPessoaPai);

        if (pEntidadeFiltro.idCargo > 0)
            funcao = funcao.AndAlso(p => p.idCargo == pEntidadeFiltro.idCargo);

        return funcao;
    }
}
```

Figura 4.10 - Classe PessoaDO

Conforme a Tabela 6 – Resources da camada de acesso a dados, um arquivo de *Resources*, ou recursos, no formato XML, foi adicionado com as mensagens de erro nesta camada, a fim de promover o reuso e centralizar as mensagens em um único local, facilitando modificação e evitando textos dentro de métodos e classes.

Tabela 6 – Resources da camada de acesso a dados

Chave	Descrição
ERRO_001	Erro ao realizar o download dos arquivos!
ERRO_002	Erro ao serializar as listas!
ERRO_003	Erro ao criar diretório {0}!
ERRO_004	Erro ao realizar download do arquivo {0}!
ERRO_005	Erro ao salvar registro da entidade {0}!
ERRO_006	Erro ao copiar arquivo do diretório temporário!
ERRO_007	Erro ao listar arquivos do diretório {0}!
ERRO_008	Erro ao excluir diretório de veiculações!
ERRO_009	Erro ao serializar arquivo {0}!
ERRO_010	Erro ao desserializar arquivo {0}!
ERRO_011	Erro ao mover arquivos para o diretório {0}!
ERRO_012	Erro ao carregar entidade {0}!
ERRO_013	Erro ao salvar entidade {0}!
ERRO_014	Erro ao salvar lista de entidades {0}!
ERRO_015	Erro ao buscar entidade {0}!
ERRO_016	Erro ao excluir entidade {0}!
ERRO_017	Não foi possível excluir o registro devido a um conflito na base de dados.

4.3.3 - Camada de negócio (BO[20])

A camada de negócio é a ponte entre a camada de dados e a camada de interface de usuário. As interfaces com o usuário devem conhecer apenas o negócio, permitindo ao arquiteto de software mudar toda a persistência de dados, sem que seja necessária qualquer alteração na interface ou na camada de negócio.

Seguindo o mesmo padrão da camada de acesso a dados, foi utilizado o Generics, para evitar replicação de dados e aumentar a produtividade. Também foi criada uma classe base, conforme a Figura 4.11 - Classe BaseBO, da qual todas as classes de BO devem herdar.

[20] Business Object em arquitetura de software se refere geralmente a classes criadas para conter as regras de negócio do sistema. O intuito dessa arquitetura é desacoplar a interface do negócio, permitindo que um sistema tenha várias interfaces e continue a obedecer regras específicas.

C#

```
public abstract class BaseBO<T> : IOperations<T> where T : class
{
    public BaseDO<T> DataObject { get; set; }

    public BaseBO(BaseDO<T> dataObject)
    {
        DataObject = dataObject;
    }

    public List<T> Buscar()
    {
        return DataObject.Buscar();
    }

    public List<T> Buscar(T pEntidadeFiltro)
    {
        return DataObject.Buscar(pEntidadeFiltro);
    }

    public T Carregar(T pEntidadeFiltro)
    {
        return DataObject.Carregar(pEntidadeFiltro);
    }

    public void Dispose()
    {
        DataObject.Dispose();
    }

    public void Excluir(T pEntidadeExcluir)
    {
        DataObject.Excluir(pEntidadeExcluir);
    }

    public Expression<Func<T, bool>> Filtrar(T pEntidadeFiltro)
    {
        return DataObject.Filtrar(pEntidadeFiltro);
    }

    public void Salvar(T pEntidadeSalvar)
    {
        if (ValidarSalvar(pEntidadeSalvar))
            DataObject.Salvar(pEntidadeSalvar);
    }

    public abstract bool ValidarSalvar(T pEntidade);
}
```

Figura 4.11 - Classe BaseBO

Assim, ao criar as classes BO para cada entidade do sistema, e tendo em vista a herança da classe BaseBO, só é necessário realizar modificações quando realmente houver uma regra de negócio a ser especificada, o que não ocorre para todos os casos, conforme ilustra a Figura 4.12 - Classe ContratoBO. Ou seja, as classes filhas poderão tanto modificar os comportamentos herdados, quanto adicionar novos métodos, caso necessário.

C#

```
public class ContratoBO : BaseBO<Contrato>
{
    public ContratoBO(): base(new ContratoDO())
    {
    }

    public override bool ValidarSalvar(Contrato pEntidade)
    {
        if (pEntidade.Ano == 0)
            throw new Exception(string.Format(Resources.ERRO_003,
DataObject.NomeEntidade, "Ano"));

        if (pEntidade.DtAssinatura == DateTime.MinValue)
            throw new Exception(string.Format(Resources.ERRO_003,
DataObject.NomeEntidade, "Data de assinatura"));

        if (pEntidade.idPessoa == 0)
            throw new Exception(string.Format(Resources.ERRO_003,
DataObject.NomeEntidade, "Código da pessoa"));

        if (pEntidade.idTipoContrato == 0)
            throw new Exception(string.Format(Resources.ERRO_003,
DataObject.NomeEntidade, "Tipo"));

        if (pEntidade.Numero == 0)
            throw new Exception(string.Format(Resources.ERRO_003,
DataObject.NomeEntidade, "Número"));

        if (pEntidade.Periodo == 0)
            throw new Exception(string.Format(Resources.ERRO_003,
DataObject.NomeEntidade, "Período"));

        if (Carregar(new Contrato() { Numero = pEntidade.Numero, Ano = pEntidade.Ano,
idTipoContrato = pEntidade.idTipoContrato }) != null)
            throw new Exception(string.Format(Resources.ERRO_004,
DataObject.NomeEntidade));

        if (Carregar(pEntidade) != null)
            throw new Exception(string.Format(Resources.ERRO_001,
DataObject.NomeEntidade));

        return true;
    }
}
```

Figura 4.12 - Classe ContratoBO

Em coerência com o padrão de desenvolvimento adotado, e conforme ilustra a Tabela 7 – Resources da camada de negócio, um arquivo de *Resources*, ou recursos, no formato XML, foi adicionado com as mensagens de erro nesta camada, a fim de promover o reuso e centralizar as mensagens em um único local, facilitando modificação e evitando textos dentro de métodos e classes.

Tabela 7 – Resources da camada de negócio

Chave	Descrição
ERRO_001	A entidade {0} já existe no banco de dados!
ERRO_002	Já existe um contrato ativo com esta vigência para o cliente em questão.
ERRO_003	Não é permitido inserir/alterar a entidade {0}, o campo obrigatório {0} não foi informado!
ERRO_004	Já existe um contrato deste tipo, com este número para este ano!
ERRO_005	A entidade {0} NÃO existe no banco de dados!

4.3.4 - Camada de interface com o usuário (UI)

Na camada de interface com o usuário existem dois softwares, um instalado em cada um dos painéis e outro web, que gerencia todo o conteúdo a ser exibido.

Esses softwares serão descritos neste capítulo, nos itens 4.4 e 4.5, a seguir.

4.4 - Software cliente (WPF)

O software cliente é instalado no computador de cada um dos painéis e tem como responsabilidades:

- Ligar e desligar a TV em horários definidos;
- Desligar o computador no horário definido;
- Exibir o conteúdo publicitário;
- Alimentar o banco de dados com informações sobre veiculações das mídias;
- Alimentar o banco de dados com informações sobre o equipamento (status da TV e IP);
- Conectar-se ao banco de dados, verificar se há atualizações à grade de exibição e realizar o download dos arquivos;

4.4.1 - Tecnologia

O software cliente foi desenvolvido utilizando WPF (Windows Presentation Foundation) em decorrência principalmente do seu acesso direto ao DirectX, importante para a exibição de vídeos em alta resolução com fidelidade, devido ao seu acesso direto ao *hardware* da placa de vídeo.

4.4.2 - Banco de dados off-line

O software cliente precisa operar 100% do tempo, ou seja, em alta disponibilidade, por isso apesar de depender de uma conexão com a internet para atualizar o conteúdo publicitário e enviar dados estatísticos e de reporte, ele precisa continuar operando caso haja indisponibilidade de uma conexão com a internet.

Com essa premissa em mente, foi desenvolvida uma solução que salva os dados de exibição localmente, sempre que uma conexão com a internet é estabelecida. Evidentemente, esta base local é apenas dos registros do painel em questão, evitando arquivos grandes e *downloads* longos, conforme ilustra a Figura 4.13 - Trecho de código que gera a base local, arquivo BaseLocal.cs.

C#

```
private List<MidiaPainel> SerializarListas()
{

    List<MidiaPainel> lstMidiaPainel = null;
    try
    {
        using (var ctx = new GottiTVContext())
        {
            List<Agendamento> results =
ctx.AgendamentosPorPainel(Convert.ToInt32(IdPainel)).ToList();

            Serializar<Agendamento>(results);

            lstMidiaPainel = new List<MidiaPainel>();
            results.ForEach(n => lstMidiaPainel.Add(n.MidiaPainel));
                                  lstMidiaPainel = lstMidiaPainel.Distinct().ToList();
            Serializar<MidiaPainel>(lstMidiaPainel);

            List<Midia> lstMidias = new List<Midia>();
            foreach (MidiaPainel item in lstMidiaPainel)
            {
```

```
                lstMidias.Add(item.Midia);
                if (item.Midia.MidiasFilhas != null && item.Midia.MidiasFilhas.Count >
0)
                    lstMidias.Add(item.Midia.MidiasFilhas.LastOrDefault(f =>
f.Ativa));
            }
            lstMidias = lstMidias.Distinct().ToList();
            Serializar<Midia>(lstMidias);

            List<Template> lstTemplates = new List<Template>();
            lstMidias.ForEach(n => lstTemplates.Add(n.Template));
            lstTemplates = lstTemplates.Distinct().ToList();
            Serializar<Template>(lstTemplates);

            List<MidiaPropriedade> lstMidiasPropriedades = new
List<MidiaPropriedade>();
            lstMidias.ForEach(n =>
lstMidiasPropriedades.AddRange(n.MidiaPropriedades));
            lstMidiasPropriedades = lstMidiasPropriedades.Distinct().ToList();
            Serializar<MidiaPropriedade>(lstMidiasPropriedades);

            List<Propriedade> lstPropriedades = new List<Propriedade>();
            lstMidiasPropriedades.ForEach(n => lstPropriedades.Add(n.Propriedade));
            lstPropriedades = lstPropriedades.Distinct().ToList();
            Serializar<Propriedade>(lstPropriedades);

            List<Arquivo> lstArquivos = new List<Arquivo>();
            lstMidiasPropriedades.ForEach(n => lstArquivos.Add(n.Arquivo));
            lstArquivos = lstArquivos.Where(n => n != null).Distinct().ToList();
            Serializar<Arquivo>(lstArquivos);

            List<ArquivoDownload> lstArquivoDownloads = new List<ArquivoDownload>();
            lstArquivos.ForEach(n =>
lstArquivoDownloads.AddRange(n.ArquivoDownloads));
            lstArquivoDownloads = lstArquivoDownloads.Distinct().ToList();
            Serializar<ArquivoDownload>(lstArquivoDownloads);
        }
    }
    catch (Exception ex)
    {
        throw new Exception(Resources.ERRO_002, ex);
    }

    return lstMidiaPainel;
}


private void Serializar<T>(IEnumerable<T> lista, bool temp)
{
    if (lista != null && lista.Count() > 0)
    {
        CriarDiretorio(PathDataTemp);

        try
        {
            FileStream stream = new
FileStream(PathArquivo<T>(temp),FileMode.OpenOrCreate);
            BinaryFormatter formatter = new BinaryFormatter();
            formatter.Serialize(stream, lista.ToList());
            stream.Close();
        }
        catch (Exception ex)
```

```
        {
            throw new Exception(string.Format(Resources.ERRO_009,
PathArquivo<T>(temp)), ex);
        }
    }
}
```

Figura 4.13 - Trecho de código que gera a base local, arquivo BaseLocal.cs

Dessa forma, o software utiliza os últimos dados locais criados, conforme ilustra a Figura 4.14 - Arquivos da base de dados locais, caso a conexão com a internet apresente qualquer problema.

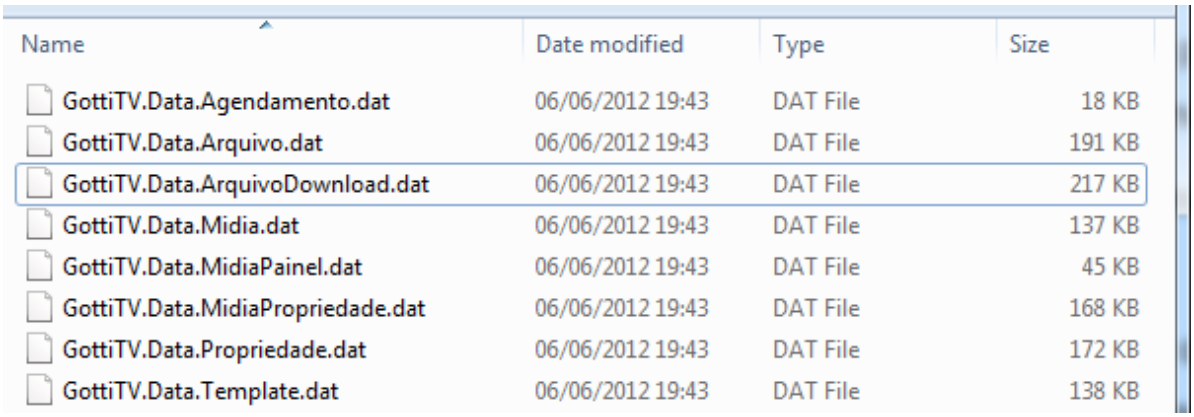


Figura 4.14 - Arquivos da base de dados locais

Nesse caso, em geral, é necessária uma intervenção física para solução do problema, tendo em vista que o acesso remoto também estará indisponível. Porém, mesmo que o material publicitário não esteja 100% atualizado, o painel ainda estará operando.

4.4.3 - Dados estatísticos

Tão importante quanto exibir o conteúdo é gerar relatórios estatísticos da quantidade de exibições do mesmo, o que pode inclusive ser um fator decisivo na renovação de contratos com os clientes.

Portanto, a cada exibição de conteúdo publicitário o software cliente gera um registro em sua base local, informando data, hora e painel que a mídia foi exibida. Esses registros permanecem na base local e são enviados periodicamente para o servidor de banco de dados, para geração de relatórios estatísticos.

4.4.4 - Arquitetura de templates

Levando em consideração que vários tipos de mídias podem ser exibidas, e que cada tipo de mídia pode exigir um *layout* de apresentação distinto, criou-se uma camada exclusiva para a criação de novos *templates*[21], ficando estes em um arquivo dll[22] separado, permitindo atualização apenas desta porção de código.

Esta estrutura de *templates* e dlls fica detalhada no banco de dados, conforme ilustra a Figura 4.15 - Estrutura de templates, no banco de dados.

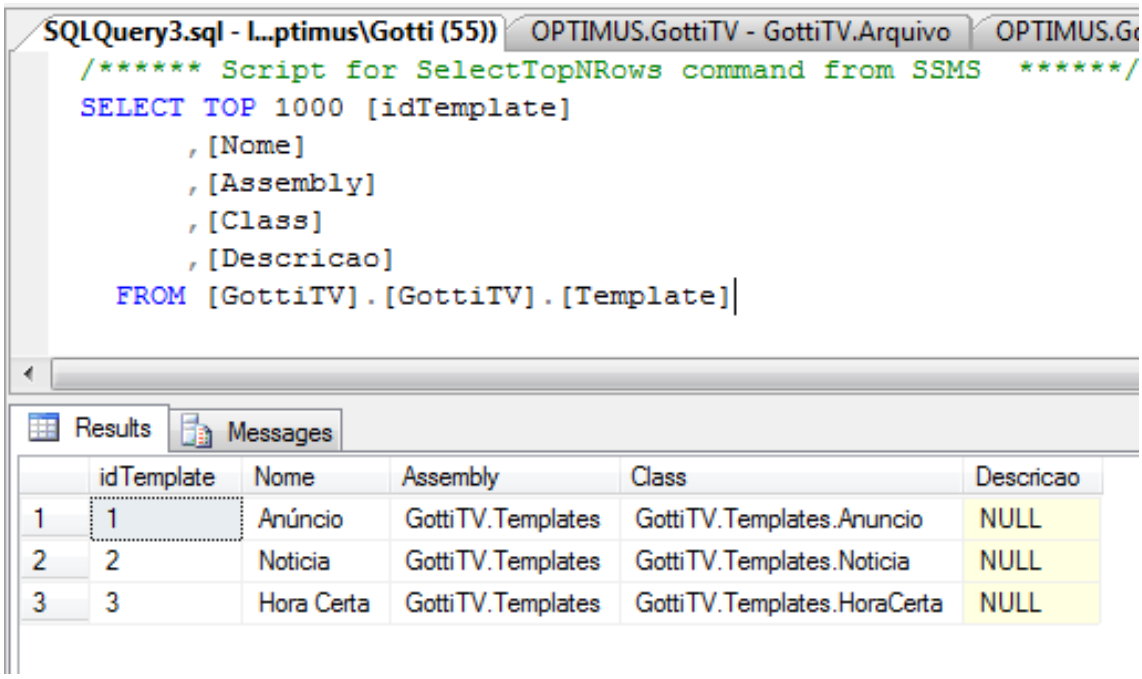


Figura 4.15 - Estrutura de templates, no banco de dados

Cada mídia armazenada é associada a um *template*, que por sua vez está associado a uma dll. Dessa maneira, em tempo de execução, o software cliente utiliza esta a dll que contém o *template* para criar uma instância do *template* e exibir o anúncio conforme suas propriedades, conforme o trecho de código da ilustra Figura 4.16 - Trecho de código que carrega *template*.

[21] Define-se por *template* qualquer documento ou modelo a ser seguido. Neste caso, trata-se de um modelo de layout, onde partes (cabeçalho, rodapé, parte central, etc) irão compor a mídia a ser exibida no painel.

[22] Um arquivo dll pode conter vários tipos de dados, como um driver de impressora, por exemplo. Na plataforma .NET, uma dll é um arquivo que contém a compilação de um código, sendo um dos principais containers de componentes reutilizáveis.

C#

```
public void CarregarAnuncio()
{
    try
    {
        if (MidiasAtuais != null && contadorMidia < MidiasAtuais.Count)
        {
            Midia midiaAtual = MidiasAtuais[contadorMidia];
            if (midiaAtual.MidiasFilhas.Count > 0)
                midiaAtual = midiaAtual.MidiasFilhas.LastOrDefault(n => n.Ativa);

            //Carrega template da dll
            userControl =
(TemplateBase)Assembly.Load(midiaAtual.Template.Assembly).CreateInstance(midiaAtual.Te
mplate.Class);
            userControl.Encerrado += new EventHandler(userControl_Encerrado);

            foreach (MidiaPropriedade item in midiaAtual.MidiaPropriedades)
            {
                if (item.idArquivo.HasValue)
                {
                    string path = string.Concat(Utilities.Configuracoes.PathAnuncios,
item.Arquivo.Extensao, @"\", item.Arquivo.idArquivo.ToString().PadLeft(4, '0'), "-",
item.Arquivo.Nome, ".", item.Arquivo.Extensao);

userControl.GetType().GetProperty(item.Propriedade.NomeReal).SetValue(userControl,
path, null);
                }
                else
                {

userControl.GetType().GetProperty(item.Propriedade.NomeReal).SetValue(userControl,
item.Valor, null);
                }
            }

            //Registra veiculação na base local
            BaseLocal.AdicionarVeiculacao(new Veiculacao()
            {
                idMidia = midiaAtual.idMidia,
                idPainel = Utilities.Configuracoes.IdPainel,
                DtVeiculacao = DateTime.Now
            });

            grdPrincipal.Children.Insert(0, userControl);
            if (grdPrincipal.Children.Count > 1)
                grdPrincipal.Children.RemoveAt(1);
        }
        else
        {
            Logger.Logar(AppResources.ERRO_007);
        }
    }
    catch (Exception ex)
    {
        Logger.Logar(ex);
        //Caso ocorra algum erro em um anúncio, pula pro próximo, evitando que a tela
fica preta e parada
        TrocarAnuncio();
    }
}
```

Figura 4.16 - Trecho de código que carrega *template*, arquivo VideoWindow.xaml.cs

4.4.5 - Configuração

O software cliente precisa de algumas informações de configuração para funcionar corretamente, que ficam armazenadas no arquivo app.config, que é um padrão para aplicações .NET, conforme ilustra a Figura 4.17 - Configuração do sistema, arquivo app.config.

XML

```
<?xml version="1.0"?>
<configuration>
  <appSettings>
    <add key="IdPainel" value="2" />
    <add key="Porta" value="COM1" />
    <add key="HoraDesligar" value="22:00" />
    <add key="ReporteIntervalo" value="5" />
    <add key="PathAnuncios" value="D:\Anuncios" />
  </appSettings>
  <connectionStrings>
    <add name="GottiTV" connectionString="Data Source=localhost;Initial
Catalog=GottiTV;User ID=GottiTV;Password=12345678" />
    <add name="GottiTVContext"
connectionString="metadata=res://*/Modelo.csdl|res://*/Modelo.ssdl|res://*/Modelo.msl;
provider=System.Data.SqlClient;provider connection string=&amp;quot;Data
Source=localhost;Initial Catalog=GottiTV;User
ID=GottiTV;Password=12345678;MultipleActiveResultSets=True&amp;quot;"
providerName="System.Data.EntityClient" />
  </connectionStrings>
</configuration>
```

Figura 4.17 - Configuração do sistema, arquivo app.config

4.4.6 - Interface

Conforme ilustra a Figura 4.18 - Interface do software cliente, a interface do software cliente é simplificada, visto que o acesso a tal software será feito apenas por indivíduos treinados e com capacidade técnica para efetuar manutenção em um painel, nunca um usuário final.

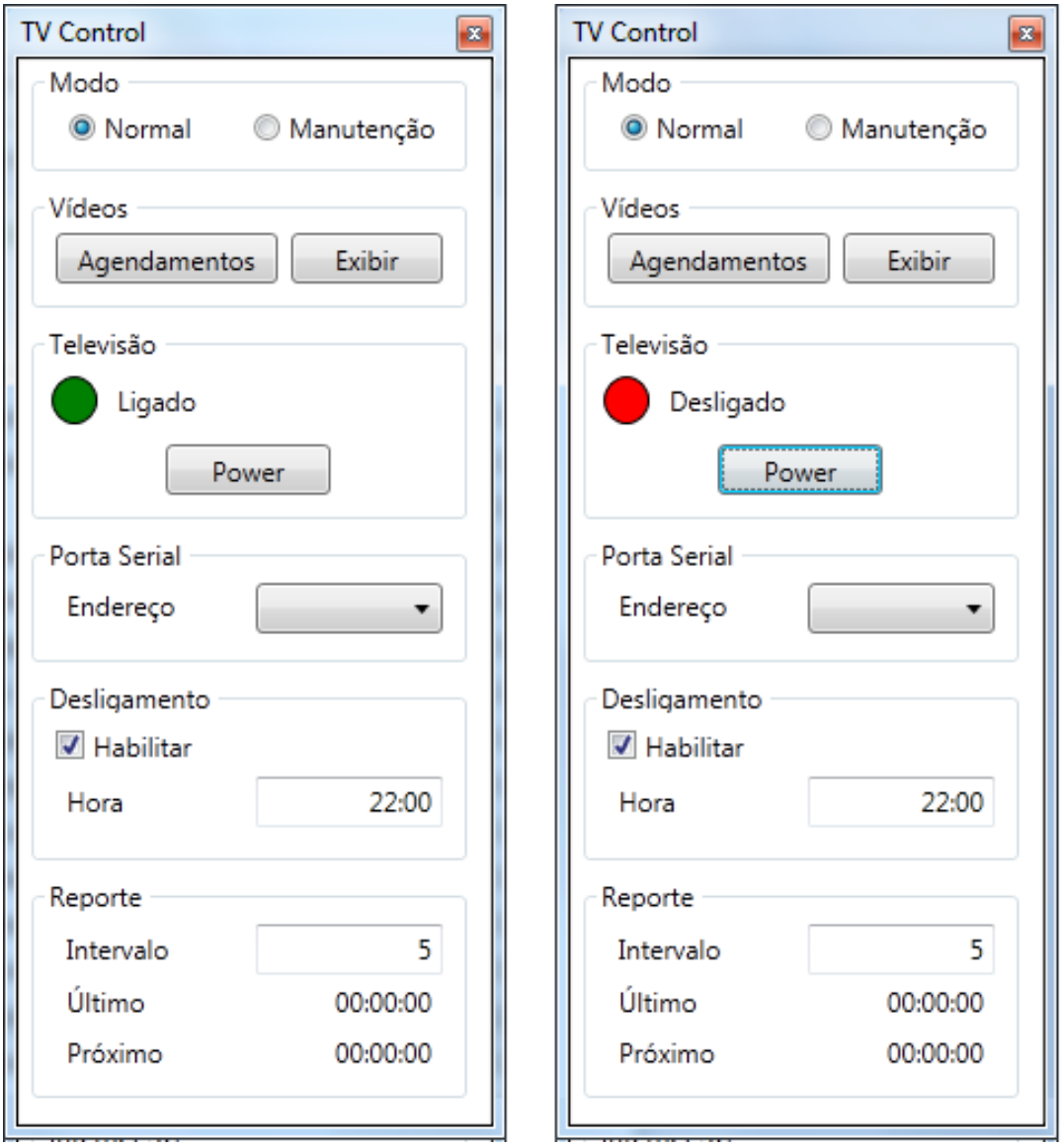


Figura 4.18 - Interface do software cliente

Através desta simples tela do software cliente é possível ligar e desligar a TV, mudar o horário de desligamento automático, colocar o painel em modo de manutenção (TV desligada), alterar o intervalo do envio de reporte, mudar a porta de conexão com a TV, regerar a base local, dentre outras funcionalidades.

4.5 - Software gerenciador (Web)

O software gerenciador é hospedado em um servidor na internet, e tem como responsabilidades:

- Armazenar informações sobre cliente, contratos, pessoas, mídias, templates, veiculações, reportes, entre outros;
- Gerar relatório de veiculações das mídias;
- Gerar relatório de informações sobre o equipamento (status da TV e IP);
- Informar ao software cliente a grade de exibição do painel em questão;
- Disponibilizar o conteúdo publicitário ao software cliente, para realização de *download*;

4.5.1 - Tecnologia

O software cliente foi desenvolvido utilizando tecnologia ASP .NET (Web), proporcionando a solução uma rede de painéis gerenciáveis de qualquer parte do mundo, desde que se esteja conectado a internet.

4.5.2 - Funcionalidades

O diagrama de caso de uso, da Figura 4.19 - Diagrama de casos de uso, descreve as funcionalidades do sistema com suas interações com o mundo exterior. Representa uma visão de alto nível das funcionalidades do sistema, descritas detalhadamente no manual do mesmo, incluído no APÊNDICE deste documento.

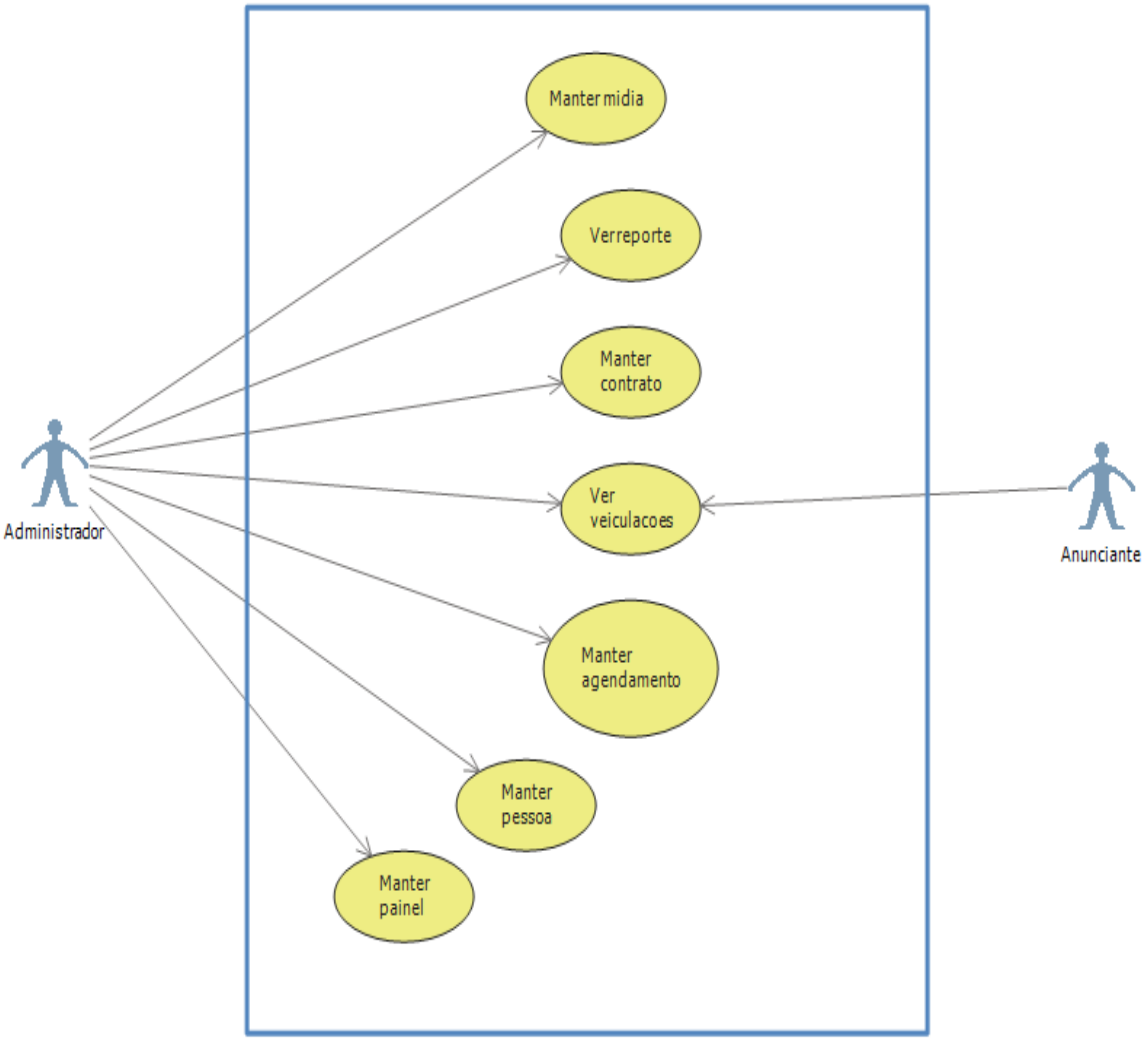


Figura 4.19 - Diagrama de casos de uso

CAPÍTULO 5 - APLICAÇÃO DO MODELO PROPOSTO

5.1 - Apresentação da área de Aplicação do modelo

Como aplicação do modelo proposto, foi implantada na cidade de Uberaba-MG uma empresa de mídia digital OOH que permitiu verificar a eficácia de tal sistema. Observando-se a inexistência do serviço no referido município e a carência de meios atuais e alternativos de mídia voltada ao comércio, utilizou-se a solução proposta para automação de painéis eletrônicos publicitários de alta definição.

5.2 - Descrição da Aplicação do Modelo

O modelo de gestão dos painéis eletrônicos varia de empresa para empresa. Visando exemplificar seu funcionamento enquanto ferramenta publicitária, apresentamos no esquema ilustrado na Figura 5.1 - Exemplo de forma simplificada de contratação e disponibilização das mídias, uma forma simplificada de contratação e disponibilização das mídias.

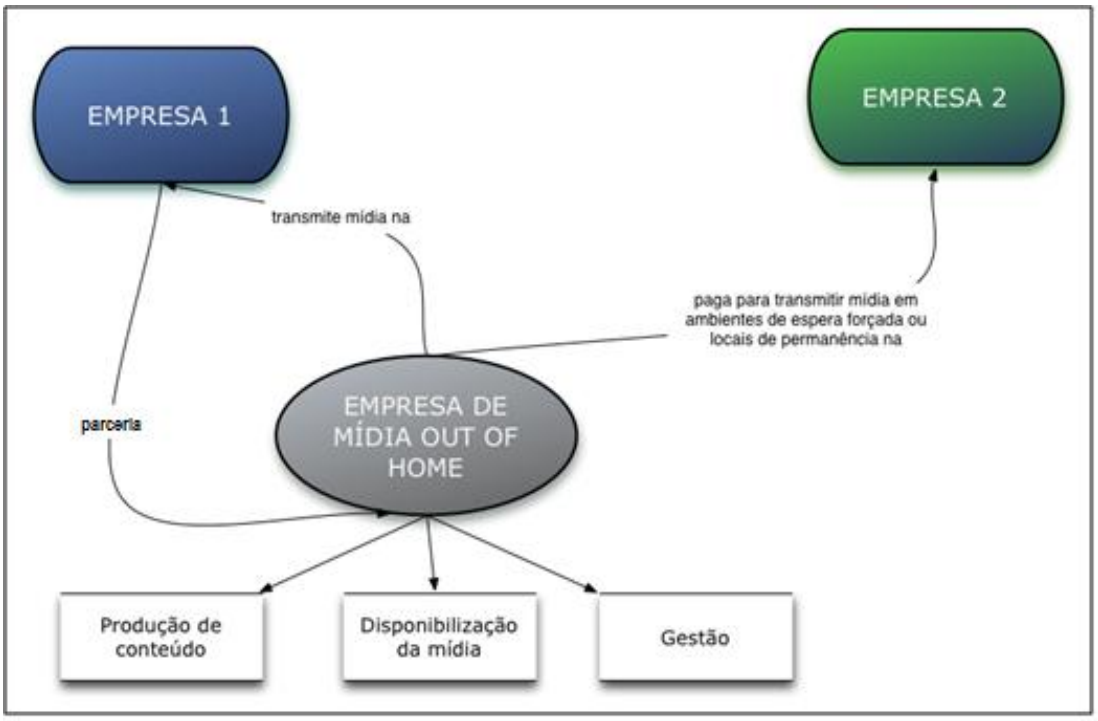


Figura 5.1 - Exemplo de forma simplificada de contratação e disponibilização das mídias

Este é o modelo mais comum de empresas de mídia digital OOH, onde existem parceiros e clientes, sendo os parceiros locais de instalação de painéis, que lucram com uma porcentagem dos anúncios ali veiculados e com a isenção de custos para veiculação de seus próprios anúncios. Já os clientes, são empresas ou pessoas físicas que contratam o serviço da empresa para veicular seus anúncios na rede de painéis instalados em parceiros.

5.3 - Avaliação Global do Modelo

5.3.1 - Parceiros

Em Uberaba – MG, foram firmadas seis parcerias com instalação de seis painéis publicitários em um modelo similar de negócio. Na época, a empresa foi pioneira em mídia digital OOH nesta cidade.

Como pode ser viso na Figura 5.2 - Painel instalado em padaria, na cidade de Uberaba - MG e na Figura 5.3 - Painel instalado em lanchonete, na cidade de Uberaba - MG, a posição horizontal foi escolhida para os painéis, visando aproveitar mídias já existentes, especialmente tendo em vista que as empresas do interior não tem o hábito de realizar campanhas publicitárias frequentemente devido ao custo relativamente elevado. Assim, foi possível reutilizar mídias que inicialmente foram projetadas para televisão na mídia digital OOH.

Figura 5.2 - Painel instalado em padaria, na cidade de Uberaba - MG

Figura 5.3 - Painel instalado em lanchonete, na cidade de Uberaba - MG

5.3.2 - Clientes

Durante um ano de operação a empresa fechou contratos com aproximadamente 40 (quarenta) clientes, tendo recuperado o capital investido para abertura da empresa.

A aceitação foi superior a expectativa, porém o faturamento da empresa ainda era baixo e eram necessários novos investimentos, tanto no quesito humano (mais vendedores, supervisores de vendas, técnicos, etc) quanto em equipamentos (novos painéis, projetores multimídia, mídias diferenciadas para equipamentos em maior escala, etc).

Esse investimento era necessário especialmente tendo em vista a abertura de mais duas empresas no mesmo ramo de atuação, para uma cidade relativamente pequena (300.000 habitantes) onde a cultura de mídia digital OOH ainda precisa de incentivos.

5.3.3 - Eficiência do sistema

Apesar dos percalços enfrentados no caminho, como a dificuldade de conexão com a internet via 3G, e a baixa qualidade do sinal, dificultando a realização dos downloads, o sistema provou sua eficiência, veiculando mídias com confiabilidade.

Em alguns casos, o maior problema foi a rede elétrica, que mostrou-se instável, queimando vários equipamentos, especialmente estabilizadores e alguns *modens* 3G. Em um parceiro específico, foi necessária a substituição por um *no-break*, o que solucionou o problema.

Na Figura 5.4 - Veiculações de mídias em um painel específico, na cidade de Uberaba - MG, é possível visualizar veiculações de mídias em um painel específico, comprovando a eficiência do sistema aqui proposto.

SQLQuery2.sql - s...TV (GottiTV (62))* | SQLQuery1.sql - (...reedon

```
/****** Script for SelectTopNRows command f
SELECT TOP 1000 [idVeiculacao]
      ,[idMidia]
      ,[idPainel]
      ,[DtVeiculacao]
  FROM [GottiTV].[GottiTV].[Veiculacao]
  order by [DtVeiculacao] desc
```

Results | Messages

	idVeiculacao	idMidia	idPainel	DtVeiculacao
1	3542656	162	6	2012-04-03 23:59:37.670
2	3542655	161	6	2012-04-03 23:59:07.537
3	3542654	160	6	2012-04-03 23:58:37.407
4	3542653	2	6	2012-04-03 23:58:22.230
5	3542652	153	6	2012-04-03 23:57:52.057
6	3542651	135	6	2012-04-03 23:57:21.917
7	3542650	134	6	2012-04-03 23:56:51.787
8	3542649	6	6	2012-04-03 23:56:36.627
9	3542648	133	6	2012-04-03 23:56:06.470
10	3542647	132	6	2012-04-03 23:55:36.327
11	3542646	131	6	2012-04-03 23:55:06.247
12	3542645	5	6	2012-04-03 23:55:02.860
13	3542644	92	6	2012-04-03 23:54:32.683
14	3542643	11	6	2012-04-03 23:54:02.517
15	3542642	63	6	2012-04-03 23:53:32.327
16	3542641	7	6	2012-04-03 23:53:16.643
17	3542640	171	6	2012-04-03 23:52:46.467
18	3542639	117	6	2012-04-03 23:52:16.337
19	3542638	72	6	2012-04-03 23:51:46.507
20	3542637	21	6	2012-04-03 23:51:16.267
21	3542636	42	6	2012-04-03 23:50:46.127
22	3542635	45	6	2012-04-03 23:50:15.927
23	3542634	172	6	2012-04-03 23:49:45.427
24	3542633	6	6	2012-04-03 23:49:30.283

Query executed successfully.

Figura 5.4 - Veiculações de mídias em um painel específico, na cidade de Uberaba - MG

CAPÍTULO 6 - CONCLUSÃO

6.1 - Conclusões

O sistema comprovou sua eficiência na prática, no dia a dia empresarial, desempenhando todas as suas funções, como veicular os anúncios, gerar relatórios de reporte dos painéis e relatórios estatísticos de veiculação.

Em muitos casos, os problemas que ocorreram foram muito mais ligados a infraestrutura elétrica e de internet do que ao sistema em si.

Por diversas vezes houve problemas com a conexão com a internet 3G, e os painéis raramente deixavam de exibir conteúdo, mostrando sua capacidade em funcionar *off-line*, como desejado

Foi possível automatizar a publicação de conteúdo e conectar-se remotamente para realização de manutenções inclusive de outra cidade, Brasilia – DF, demonstrando mais uma vez a eficiência do sistema tanto em gerenciar a conexão 3G, quanto na sua função mais importante: exibir conteúdo.

Porém, a atualização de conteúdos de entretenimento diversos, como notícias, horóscopo, previsão do tempo e outros foi manual, o que dispendia muito tempo e por vezes, deixava de ser realizada.

6.2 - Sugestões para Trabalhos Futuros

A integração com conteúdos RSS é de extrema importância em projetos como esse e não foi escopo dessa proposta, portanto essa modificação pode ser realizada sem muito esforço técnico.

A integração com outras marcas de televisores e a utilização de novas conexões, como USB, rede e até mesmo via *wireless* com certeza representaria um avanço considerável neste projeto e pode ser implementada em soluções futuras.

Uma outra sugestão seria integrar o sistema proposto a um sensor GPS podendo fornecer conteúdo publicitário em taxis, ônibus, e até mesmo metrôs, levando em consideração a posição do público no momento. Esse tipo de mídia pode influenciar ainda mais a decisão de compra, visto a proximidade do anunciante com o cliente.

Há ainda, o segmento de vitrines digitais de OOH, que apesar de muito interessante, é pouco explorado no Brasil em geral. Existem películas próprias para retroprojeção onde se pode exibir o conteúdo de um projetor em um vidro de uma vitrine, transformando-a em um ponto turístico de clientes.

Há ainda a possibilidade de se inovar nesse sentido, utilizando sensores como o Kinect da Microsoft, ou sensores *touchscreen*, para que a vitrine possa ser interativa, atraindo ainda mais a atenção do público.

Assim, fica claro que a solução aqui proposta, além de ter sua eficiência comprovada, pode servir de base para vários outros projetos inovadores e rentáveis.

REFERÊNCIAS

ARAÚJO, Anna Gabriela. Eles estão saindo de casa! **Revista Markekting**, São Paulo, no. 451, Ano 43, p. 20-25, Agosto 2010.

KELSEN, Keith. **Unleashing the Power of Digital Signage - Content Strategies For the 5th Screen**. Burlington, Massachusetts, USA: Elsevier Science, 2010.

CURRY, Luiz Fernando. **As Paisagens da Comunicação ao Ar Livre**. NP 03, Publicidade, Propaganda e Marketing, IV Encontro de Núcleos de Pesquisa da Intercom. Porto Alegre, 2004.

FOWLER, Martin. **Patterns of Enterprise Application Architecture**. Boston, Massachusetts, USA: Pearson Education, Inc. , 2003.

SANTANA, Laís Souza Gomes de. **Mídias digitais em espaços públicos: DOOH – digital out of home.** Faculdade de Comunicação, Universidade Federal da Bahia (Monografia de Graduação). Salvador, 2009.

ABDOH. **Associação Brasileira de Mídia Out of Home**, 2008. Disponível em: <http://www.abdoh.com.br/site>. Acesso em: 2 mai. 2012.

DPPA. **Digital Place-based Advertising Association**, 2012. Disponível em: <http://www.dp-aa.org>. Acesso em: 10 mai. 2012.

PIZZOTTI, Ricardo. Produção de Televisão e vídeo, 2012. Disponível em: <http://www.proteve.net/principal.html>. Acesso em: 12 Mai. 2012.

HDMI. **Licensing, LCC**., 2012. Disponível em: <http://www.hdmi.org/about/index.aspx>. Acesso em 25 Mai. 2012

PR NEWSWIRE ASSOCIATION. **Digital Place-based Media Revenue Growth Rate For 2011 Exceeds That Of Overall U.S. Ad Industry By More Than 17:1 Margin**, 2012. Disponível em: <http://www.prnewswire.com/news-releases/digital-place-based-media-revenue-growth-rate-for-2011-exceeds-that-of-overall-us-ad-industry-by-more-than-171-margin-148668545.html>. Acesso em: 1 Jun. 2012.

OUT OF HOME INTERNATIONAL. **Out Of Home Advertising**, 2012. Disponível em: <http://www.oohinternational.co.uk>. Acesso em: 20 Mai. 2012.

MICROSOFT. **ADO.NET Entity Framework At-a-Glance**, [entre 2010 e 2012]. Disponível em: <http://msdn.microsoft.com/en-us/data/aa937709>. Acesso em: 18 Jun. 2012.

MICROSOFT. **Generics (C# Programming Guide),** [entre 2010 e 2012]. Disponível em: <http://msdn.microsoft.com/en-us/library/512aeb7t(v=vs.100).aspx>. Acesso em: 19 Jun. 2012.

SANCHEZ, Fabrício. **Recursos para Internet com .NET – Parte 1**, 2010. Disponível em: <http://www.fabriciosanchez.com.br/2/recursos-para-internet-com-net-parte-1>. Acesso em: 16 Ago. 2010.

TYSON, Jeff. **How Serial Ports Work**, [entre 1998 e 2010]. Disponível em: <http://computer.howstuffworks.com/serial-port.htm/printable>. Acesso em: 5 Jun. 2010. para o conceito de Porta Serial

ALECRIM, Emerson. **Tecnologia HDMI (High-Definition Multimedia Interface)**, 2008. Disponível em: <http://www.infowester.com/hdmi.php>. Acesso em: 29 Mai. 2012. para o conceito de HDMI

WIKIPEDIA. **Ambiente de desenvolvimento integrado**, [entre 2011 e 2012]. Disponível em: <http://pt.wikipedia.org/wiki/Ambiente_de_desenvolvimento_integrado>. Acesso em: 6 Jun. 2012. para o conceito de IDE

MICROSOFT. **Introduction to WPF**, [entre 2010 e 2012]. Disponível em: <http://msdn.microsoft.com/en-us/library/aa970268.aspx>. Acesso em: 10 Abr. 2012

MICROSOFT. **WPF Architecture**, [entre 2010 e 2012]. Disponível em: <http://msdn.microsoft.com/en-us/library/ms750441.aspx>. Acesso em: 10 Abr. 2012

RASMUSSEN, Mark. **Securing .NET Code**, 2006. Disponível em: <http://improve.dk/archive/2006/10/02/securing-dotnet-code-article.aspx>. Acesso em: 10 Jul. 2010.

SUR, Abhishek. **WPF Tutorial: Beginning**, 2010. Disponível em: <http://www.codeproject.com/Articles/140611/WPF-Tutorial-Beginning>. Acesso em 3 Mai. 2012.

APÊNDICE