2BuyNet – Um framework para Instanciação e Administração de Lojas para a Internet

Luidi Xavier Fortunato

Dissertação apresentada ao Departamento de Informática da PUC-Rio como parte dos requisitos para a obtenção do título de Mestre em Engenharia de Software.

Orientador: Carlos José Pereira de Lucena

Departamento de Informática

Pontifícia Universidade Católica do Rio de Janeiro

Abril de 1999

Dedicatória

à minha família

Agradecimentos

- À Capes, pelo apoio financeiro;

- Aos colegas do LES e do Departamento de Informática e da Fundação Padre Leonel Franca;

- A minha família, que sempre me apoiou;

- A aluna de mestrado Viviane Torres, que muito me apoiou no desenvolvimento do software;

1. INTRODUÇÃO 8

1.1 COMÉRCIO ELETRÔNICO 8

1.2 BENEFÍCIOS PARA O CONSUMIDOR 9

1.3 BENEFÍCIOS PARA AS EMPRESAS 10

1.3.1 Distribuição 11

1.3.2 Marketing 11

1.3.3 Serviços 12

1.4 REALIDADE DO COMÉRCIO ELETRÔNICO 13

2. DESCRIÇÃO DO DOMÍNIO DA APLICAÇÃO 15

2.1 GERAÇÃO AUTOMÁTICA 16

2.2 ELEMENTOS DE UM WEB SITE COMERCIAL 17

2.2.1 Carrinho de compras 18

2.2.2 Cadastro de clientes 20

2.2.3 Navegação 21

2.2.4 Administração 22

2.3 FORMAS DE INSTANCIAÇÃO DA LOJA 24

2.4 CUSTOMIZAÇÃO DA INTERFACE 27

2.5 TRABALHOS RELACIONADOS 30

2.5.1 Microsoft Site Server 2.0 Commerce Edition 30

2.5.2 AbleCommerce 31

2.5.3 EZ2-Shop 32

2.5.4 Inex Commerce Court 33

2.5.5 Icat *34*

2.5.6 Viaweb *35*

2.5.7 2BuyNet *36*

3. FRAMEWORK 41

3.1 CONCEITUAÇÃO DE FRAMEWORKS 41

3.2 APLICAÇÃO 2BUYNET VERSUS FRAMEWORK 2BUYNET 42

3.3 MODELO CONCEITUAL DA LOJA 43

4. PROCESSO DE INSTANCIAÇÃO DO FRAMEWORK 52

4.1 COLETA DE DADOS 53

4.2 GERADOR 70

4.2.1 Camada de interface *73*

4.2.2 Processo de geração da interface *74*

4.2.3 Problema do Idioma *75*

4.2.4 Estrutura gerada *77*

4.3 BACKGROUND DAS LOJAS 78

4.3.1 Carrinho de compras *80*

4.3.2 Clientes *81*

4.3.3 Comunicação *82*

4.4 IMPORTAÇÃO/EXPORTAÇÃO DE DADOS 84

4.4.1 Síncrono e assíncrono *86*

5. CONCLUSÃO E TRABALHOS FUTUROS 89

6. APÊNDICE A – LINGUAGEM DE EXTENSÃO DA INTERFACE 91

7. REFERÊNCIAS 105

7.1 SOFTWARES DE APOIO À CRIAÇÃO DE WEB SITES COMERCIAIS 110

Resumo

Essa dissertação apresenta etapas do desenvolvimento de websites destinados ao comércio eletrônico. No primeiro momento, foram estudados os problemas e funcionalidades envolvidos no domínio. O trabalho mostra também um estudo de softwares relacionados à criação de lojas para Internet, como estes se propõem a solucionar os problemas e a implementar as funcionalidades. Como resultado, foi implementado um framework, denomidado 2BuyNet, propondo formas combinadas e novas para a solução dos problemas. A dissertação descreve razões de design sobre o framework, bem como vantagens e desvantagens. Como exercício sobre esse framework foi instanciada uma aplicação destinada à Internet, também denomindada 2BuyNet. Essa aplicação publicada na Internet serve como uma ferramenta para geração e administração de lojas virtuais.

Abstract

This dissertation presents a software development approach for the construction of electronic commerce websites. At first, problems and features related to the domain application are discussed. The dissertation also presents some existing software tools for webstore creation, their proposed solutions and the approach they use to impement their features. Based on these comparasions we have developed a rfamework called 2BuyNet. The framework constitures an original software engineering approach for the development of electronic commerce sites. This dissertation describes the design rationale for the framework structure, as well its pros and cons. As an exercise of the use of the framework an application also called 2BuyNet has been instantiated, using the Web as the interface. This application has been published on the Intenet and works both as a web store generator service and a web store administrative tool.

1. Introdução

O capítulo 1 dessa dissertação trata dos aspectos gerais de comércio eletrônico como a nova forma de negociar trazida pela Internet, as vantagens para os diversos pernosagens do mercado e por último uma descrição da realidade do comércio eletrônico. O capítulo 2 faz um levantamento sobre o domínio do problema, descreve os elementos presentes num web site comercial e apresenta as ferramentas analisadas durante o estudo dessa dissertação. No capítulo 3, é apresentado a solução através da implementação de um framewok, as características desse framework, no caso específico do 2BuyNet, e as características da aplicação 2BuyNet. O capítulo 4 define, em detalhes, as fases framework e as camadas da loja gerada por este. O último capítulo conclui apresentando a experiência adquirida durante o estudo e propõe as futuras áreas de estudo e desenvolvimento do framework.

1.1 Comércio Eletrônico

Com a popularização da Internet, a prática do comércio eletrônico também se popularizou. Muito se discute sobre o seu potencial e o seu impacto nos hábitos dos consumidores. As previsões são sempre muito otimistas e os números envolvidos são astronômicos.

Números e estatísticas indicam que o comércio baseado na Internet é mais eficiente e possivelmente mais efetivo do que o realizado pelos meios tradicionais. Algumas experiências baseadas no comércio online sugerem que o marketing na Web dá um retorno dez vezes maior com um décimo dos gastos em propaganda [10]. Ainda nesse meio, custa

um quarto do preço uma campanha de marketing direto se comparada aos meios tradicionais. Por exemplo, a SunSolve economizou US$ 4 milhões somente em FAQ's, numa reestruturação que realizou para disponibilizar informações na Web.

Analisando as formas atuais de comércio eletrônico [16], identificamos características do processo de comercialização que realmente trazem algum tipo de benefício para uma, ou ambas as partes envolvidas no processo de compra e venda. Essa análise, pela própria imaturidade do comércio eletrônico, é feita com base em casos típicos de lojas na Internet. Essas lojas, excluindo-se alguns serviços, são basicamente versões digitais de catálogos tradicionais. Esses benefícios surgiram não pelo fato da relação de compra ter mudado de alguma forma, mas simplesmente por ela se dar em um meio inovador que permite um alto grau de interatividade. Os benefícios citados adiante não supõe futuras formas de comércio.

1.2 Benefícios para o consumidor

Um importante benefício para o consumidor, no que diz respeito ao comércio através da Web, é o acesso a uma grande quantidade de informação, o que auxilia o processo de decisão sobre uma compra. [10, p.192] A natureza interativa da Web, com suas informações disponibilizadas na forma de hipertexto, permite uma análise aprofundada e não-linear, com mecanismos de busca, totalmente controlados e comandados pelo consumidor.

A capacidade da Web de permitir a coleta e a análise de grandes quantidades de informação possibilita a criação de serviços de comparação e aumentam a velocidade do processo de escolha de um item . Muitas vezes, é possível também, testar um produto pela Web, o que estimula o consumo e aumenta a satisfação dos clientes. Em decorrência da eficiência do meio, é possível refinar o processo de seleção satisfazendo os mais complexos critérios de consumo.

Essas vantagens se estendem ainda mais se projetadas para os consumidores industriais. Para esses, o custo do processo de compra é um fator importante no critério de escolha de um produto. O tempo e o quanto se vai gastar na escolha entre dois produtos também é um quesito de seleção de um produto. Outro benefício para o consumidor, principalmente para o industrial, e que deve ser destacado é a quantidade de forncedores. Normalmente, uma maior oferta de fornecedores permite uma escolha melhor entre as ofertas, com nível elevado de qualidade.

1.3 Benefícios para as empresas

O comércio eletrônico através da Internet também traz uma série de benefícios para as empresas. Da mesma maneira que ocorre com os consumidores, as empresas dispõem, agora, de um meio mais eficiente que transforma as formas conhecidas de interação com os clientes e que traze novas possibilidades.

1.3.1 Distribuição

Um dos benefícios que a Internet traz para as empresas é um eficiente canal de distribuição para algumas categorias de produtos. Esse canal, em função do tipo de produto, pode chegar a ter custo zero na distribuição. Por exemplo, imagine uma empresa de produtos digitais, seus produtos podem ser transferidos pela rede imediatamente e sem nenhum custo adicional tanto para a empresa quanto para o consumidor.

Uma outra característica do comércio pela Web e que se apresenta como uma vantagem para as empresas, é o fato de o processo de compra ser mais conduzido pelo consumidor. Normalmente, numa compra pela Web, o próprio comprador faz o pedido e preenche formulário com as informações necessárias para a compra. Isso permite que a empresa obtenha informações para marketing direto, execute a coleta de dados sobre o comportamento de compras do cliente e obtenha outras informações relevantes. Note que essa prática pode ser questionada sob o ponto de vista legal do uso dessa informação sem autorização prévia do cliente. Esse ângulo da questão está fora do objetivo deste trabalho.

1.3.2 Marketing

Atualmente as empresas, na sua maioria, sub utilizam a Internet. Seus sites são puramente institucionais. Ou seja, elas apenas apresentam informações sobre a empresa, sem acrescentar nenhum serviço ou informação adicional diferente da já disponível nos meios tradicionais. Nestes casos, a Internet é apenas uma duplicação e/ou reunião de todos os elementos já disponíveis nos meios tradicionais. Mas a natureza interativa do meio permite muito mais do que apenas isso, especialmente no que se refere à area de marketing.

Através de serviços simples, como email ou lista de discussão, é possível cativar o cliente e despertar nele um sentimento de fidelidade com a empresa. Serviços de suporte 24 horas também são recursos utilizados. Mesmo através de uma comunicação assíncrona, um suporte 24 horas proporciona ao cliente um sentimento de cuidado com a sua satisfação.

Certamente quando o assunto é marketing, a Internet traz as mais revolucionárias mudanças. Marketing personalizado a um custo que nenhum outro meio pode trazer, pesquisas com clientes de abrangência nacional e internacional e outros métodos podem e devem ser explorados pelas empresas como ferramenta auxiliar ao comércio eletrônico.

1.3.3 Serviços

Serviços oferecidos pela Web têm o sucesso intimamente ligado à própria natureza da plataforma. Alguns serviços prestados por empresas tornam-se infinitamente mais baratos através da Internet. É o caso, por exemplo, da Federal Express [11] que economiza milhões de dólares anualmente em conta telefônica, pois é possível fazer o rastreamento de um pacote pela Internet em vez de ligar para um número gratuito 0800. É também o caso da Dell Computers (vendedor de computadores) que não tem revendedores cadastrados. A única forma de adquirir um computador Dell[1] era ligando para um número telefônico 0800. Hoje em dia, é possível comprar os computadores Dell pela Internet, o que trouxe para a empresa uma economia de milhões, já que o processo de venda de um computador com preço médio de US$ 3,000.00 é demorado, aumentando muito os custos de venda da empresa. Um mesmo computador vendido pela Internet e pelo telefone dá a Dell margens de lucro totalmente diferentes.

[1] http://www.dell.com

Esse são alguns exemplos de como empresas estão reduzindo custos de seus serviços tradicionais. Mas a Internet traz ainda mais um nicho de serviços, os que só podem existir graças à ela. São casos como o eBay[2], uma empresa de leilão pela Internet. A forma como se desenrola um leilão na eBay só é possível graças à Internet. Uma empresa nesse mesmo setor e que também só existe como um serviço para e Internet é a OnSale[3], também provedora de leilões na Internet. Os multimilionários portais, são a prova viva de empresas que são puramente virtuais e que criaram um negócio que só existe na Internet. Não existe o conceito de portal no mundo real, como existe o de leilões. Portais como Yahoo![4], ZAZ[5] e outros são empresas sólidas e que cresceram graças ao crescimento da própria Internet.

1.4 Realidade do comércio eletrônico

Apesar dos benefícios, infelizmente nem todas as previsões sobre o sucesso do comércio eletrônico na Web têm se confirmado e consequentemente os números, apesar de grandes, não são os esperados [20, 21]. A grande revolução na forma de negociar, o grande boom na Internet e a massificação do acesso, parecem ainda presos por algum motivo. Dado esse inesperado cenário real, uma linha de pesquisa nasceu para procurar localizar a raiz do problema. Partindo-se da análise de casos mal sucedidos de negócios na Web, identificaram-se pontos falhos e dificuldades.

---

[2] http://www.ebay.com

[3] http://www.onsale.com

[4] http://www.yahoo.com ou no Brasil http://www.yahoo.com.br

[5] http://www.zaz.com.br

Na maioria dos casos, o processo de desenvolvimento do próprio Web site foi o maior empecilho para o empreendedor. Decisões associadas à plataforma, às inúmeras opções de design e principalmente à velocidade com que qualquer uma dessas alternativas ficam obsoletas, aliado aos custos intrínsecos, levaram muitos à esperar por uma solução mais consolidada, um padrão de mercado, ou que o próprio mercado se decida e eleja os melhores, deixando apenas poucas opções.

As dificuldades em definir uma especificação e implementar as funcionalidades ainda são problemas enfrentados por quem deseja ter uma presença comercial na Internet. Esse motivo é uma das causas da relativamente lenta proliferação de Web Sites comerciais e da adoção vagarosa da Internet como um meio padrão para negócios.

2. Descrição do Domínio da Aplicação

Esse capítulo apresenta uma análise de softwares destinados à criação e manutenção de Web sites comercias para a Internet. O objetivo principal desse estudo de trabalhos relacionados foi fazer o levantamento das características de cada um dos produtos. Entende-se por características não só as funcionalidades que cada programa implementa, mas também as entidades que são criadas para modelar o problema no que se refere à implementação.

A identificação dessas entidades é de extrema importâncias para o sucesso da modelagem de um framework para comércio eletrônico.Conhecendo-se a fundo essas entidades e identificando entidades coincidentes nos diferentes programas, é possível criar um framework tão genérico e poderoso quanto a união de todos os programas que fazem parte do estudo de softwares relacionados [3, 7]. Além disso, as várias entidades usadas em cada software representam visões diferentes para a solução de um mesmo problema. Isto permite uma visão mais abrangente do problema e a apresentação de diversas alternativas para solucioná-lo.

Os seis produtos analisados nesse estudo podem ser divididos em dois grupos bem definidos. Os que representam a categoria ***produto*** e os da categoria ***serviço***. Produto são todos aqueles softwares tradicionais. Ou seja, existe um programa-produto, que deve ser adquirido e instalado numa máquina cliente para desenvolvimento. A forma de cobrança também segue o tradicional, cobra-se pelo software adquirido e dependendo da solução gerada, cobra-se um royalty sobre o uso da solução gerada pelo produto. Na categoria de serviço, um browser conecta-se numa máquina remota onde está publicado um software

que presta o serviço para resolver o problema. Nesse caso, não existe nenhuma compra de software e a forma de pagamento, geralmente, é feita sobre o uso do serviço, ou sobre um aluguel cobrado da solução gerada pelo serviço.

Nesta categoria produto tem-se o Microsoft Site Server 2.0 Commerce Edition, o AbleCommerce, o EZ2-Shop e o Inex Commerce Court. Na categoria serviço tem-se o Viaweb e o Icat.

2.1 Geração automática

Um Web site comercial é uma composição de páginas html, algumas funcionalidades para manipular produtos e pedidos (elementos de um Web site comercial) e um mecanismo para administração desse site, que pode ser outro site ou um programa tipo produto. Fazendo-se uma generalização é possível identificar partes comuns que se repetem em todos os sites.

A forma tradicional de desenvolvimento de um Web site comercial, não prevê nenhuma reutilização de código, uma vez que em momento algum ocorre a preocupação com a identificação dessas partes comuns à todos os sites. Isso pode acarretar um desenvolvimento mais demorado, já que todas as etapas do software precisam que ser re-rescritas (modelo de dados, camada da acesso à dados, camada da aplicação e interface).

Uma outra possibilidade é isolar as partes comuns e as próprias de cada site. Assim, pode-se construir uma máquina que gere as partes comuns automaticamente e de posse das informações específicas para cada caso, complete as partes comuns com as específicas.

A vantagem de uma geração automática é a rapidez no processo de desenvolvimento e a certeza da correção do código gerado. Dependendo da interface da máquina geradora, a operação dessa máquina pode ficar a cargo de um usuário não-especialista, sem haver a necessidade de um especialista em Informática para configurá-la para a geração de uma loja. Uma forma de simplificar a interface dessa máquina é a implementação de um formulário de perguntas (conceito de wizard). Através de um questionário guiado, um usuário não-especialista responde perguntas não-técnicas, sobre o domínio em questão (lojas). No final do processo, todos os pontos específicos para aquele caso particular estarão esclarecidos e a máquina poderá juntar as partes específicas com as genéricas e gerar uma loja configurada para o caso em questão.

A solução da geração automática é adotada em todos os seis casos analisados, tanto para a categoria produto quanto para a categoria serviço. O 2BuyNet também adota a solução de geração automática, combinada com uma série de perguntas guiadas para o preenchimento de informações sobre produtos, departamentos, informações da loja, etc.

2.2 Elementos de um Web site comercial

Os Web site comerciais (lojas) normalmente implementam características comuns [2] entre si. É comum encontrar-se em diferentes sites funcionalidades parecidas, ou muitas vezes iguais, apesar de serem sites desenvolvidos por equipes distintas.

Analisando-se um conjunto de lojas existentes, identificou-se um denominador comum entre elas, ou seja, um conjunto de funcionalidades que praticamente todas oferecem.

2.2.1 Carrinho de compras

A principal função do carrinho de compras é organizar os itens selecionados pelo cliente e prover funcionalidades para manipulá-los. Cada vez que um cliente compra um determinado item, este é adicionado a uma lista (carrinho de compras). Existe todo um conjunto de funções para manipular esse carrinho, como alterar a quantidade, remover um item, limpar todo o carrinho, mostrar o conteúdo de um carrinho e executar um pedido.

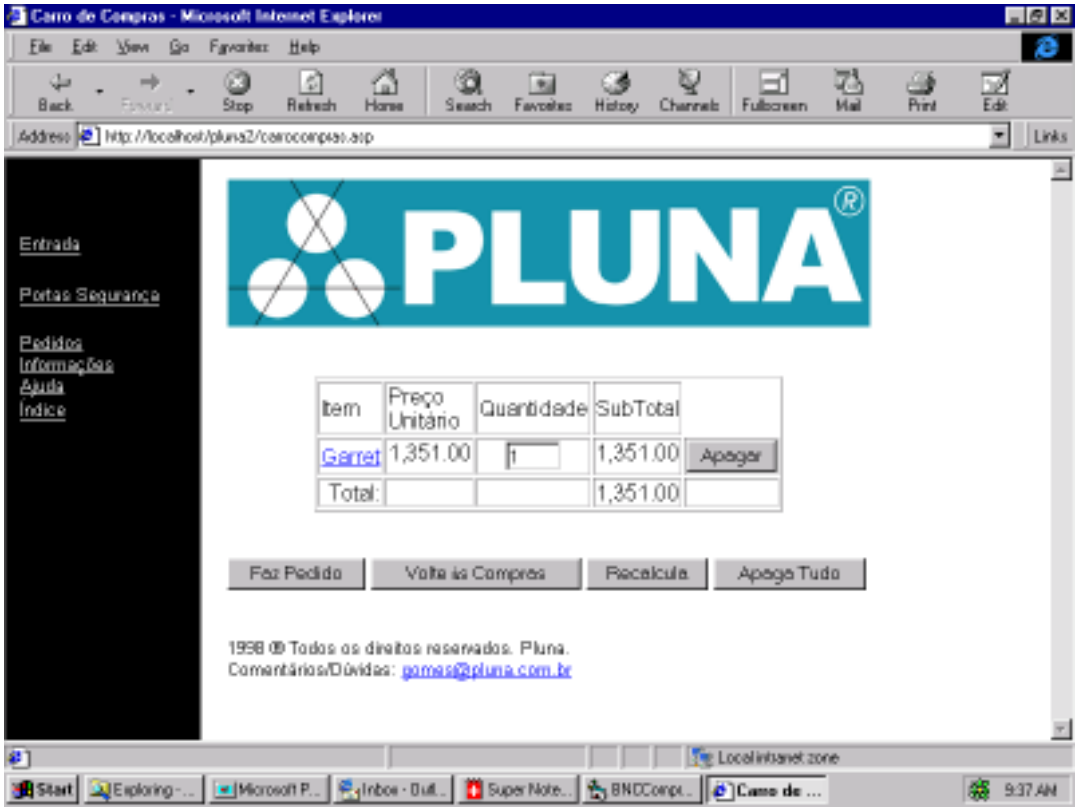


Figura 2.2-1 Carrinho de compras

Como diz o próprio nome, trata-se de uma metáfora para um carrinho de compras do mundo real, onde o cliente navega pelos produtos e ao escolher um que o interessa, adiciona-o ao seu carrinho de compras. No final das compras, ele tem um carrinho repleto de produtos, que no caso virtual ainda podem ser manipulados (aumentar as quantidades ou retirar), e que devem ser passados pelo caixa para o pagamento (na Figura 2.2-1, **Faz Pedido**).

2.2.2 Cadastro de clientes

Um forte apelo dos Web sites comerciais é o seu potencial para marketing personalizado. Na Internet é possível identificar cada pessoa que visita um site.

Através de um esquema que utiliza cookies [23], é possível saber-se se é a primeira vez que um comprador "entra" numa loja ou não. Estendendo-se essa funcionalidade, é possível implementar um tratamento diferenciado para quem a visita pela primeira vez e para clientes habituais.

O cadastro de clientes serve como ferramenta para essa diferenciação. Ao identificar um cliente como um novo visitante, pede-se que ele preencha um formulário com alguns dados pessoais tais como nome, endereço e email. Caso ele o preencha, numa segunda visita à loja, esse visitante já será identificado e a tela principal da loja aparecerá diferente para ele.

Na tela abaixo, como exemplo dessa funcionalidade, existe um link convidando o cliente a fazer um cadastro na loja. Em “você não gostaria de se cadastrar” o cliente é remitido a uma outra tela para preencher um formulário de cadastro.

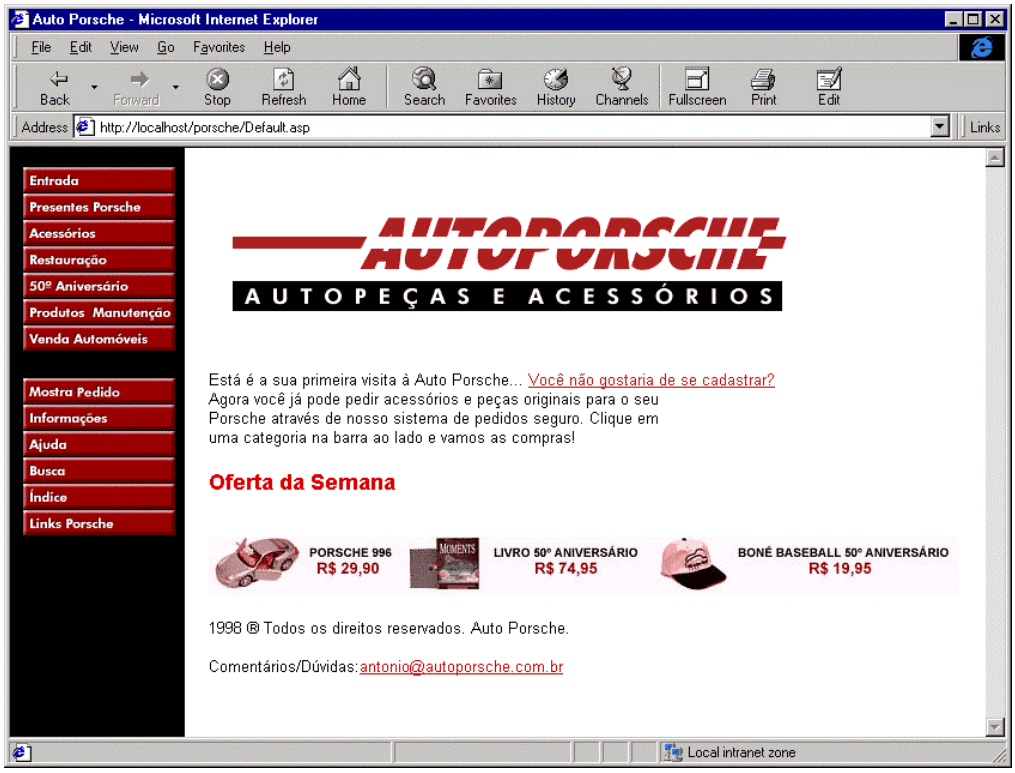


Figura 2.2-2 Tela principal sendo a primeira visita do cliente

2.2.3 Navegação

A navegação é a funcionalidade de um site que interliga suas diversas páginas. Ao especificar a navegação, o desenvolvedor está, ao mesmo tempo, determinando quais páginas estão ligadas entre si e quais não estão. Em função do tamanho de uma loja, a navegação pode se tornar extremamente complexa, exigindo que várias páginas tenham um grande número de links para outras páginas, o que usualmente leva o usuário que está utilizando o site a se perder.

A navegação é um elemento comum entre os Web sites comerciais e a maior parte das máquinas geradoras adotam a solução de uma barra de navegação. Uma barra de

navegação consiste em uma parte da página destinada apenas à links para as outras partes do site. Assim, o usuário acostuma-se a sempre procurar numa determinada posição da tela, as opções de navegação que lhe são oferecidas num determinado ponto do site. Essa barra de navegação pode ser fixa para todo o site, ou seja nunca muda em página nenhuma de todo o site. A barra pode também ser função da página em que se está num determinado momento. Por exemplo, a barra de navegação da página principal pode ser diferente da página de um determinado produto dentro de um departamento.

2.2.4 Administração

A administração de um Web site comercial é a parte encarregada de gerenciar a loja. Normalmente essa visão da loja só é disponível a quem controla a loja, ou seja, ao dono da loja ou as pessoas que trabalham com ele. Um comprador comum não tem essa visão. Para ele, a loja é apenas um conjunto de páginas html contendo os produtos e os demais elementos característicos de um Web site comercial.

A administração de um Web site comercial pode ser uma parte tão grande ou muitas vezes maior que o próprio site da loja. Dentre as funcionalidades da administração podemos destacar:

Cadastro de produtos:

Controla o banco de dados de produtos. Inclui funções tais como inserir novo produto, alterar um produto existente, apagar um produto do banco de dados.

Cadastro de departamentos:

Controla o banco de dados de departamentos. Inclui funções iguais as do cadastro de produtos, só que destinadas aos departamentos.

Controle dos pedidos:

Os pedidos dos clientes são armazenados de alguma forma no banco de dados da loja. Através da funcionalidade de controle de pedidos, é possível consultar esses pedidos para atendê-los.

Controle dos clientes:

O marketing pela Internet é uma questão central para o sucesso de uma loja. Saber quem são os clientes que visitaram uma loja, é de extrema importância para o lojista. Com o controle de clientes é possível controlar essa informação.

Estatísticas:

Novamente uma ferramenta para apoio ao marketing. Informações estatísticas como quais são os produtos que mais vendem, quem são os clientes que mais compram ou quais são as palavras mais procuradas na sua loja, são dados importantes para se traçar um plano de marketing.

Quanto à implementação, existem diversas formas. Algumas vezes, a própria máquina geradora incorpora funções para administrar a loja. A mais comum de todas é a implementação de um outro módulo encarregado da administração. Esse segundo módulo

pode ser um site na Internet (com endereço próprio e publicado apenas para os lojistas) ou um programa (estilo programa-produto) que deve ser instalado pelo lojista e que se encarregará de fazer a administração remota do Web site comercial.

2.3 Formas de Instanciação da loja

Os softwares analisados propõem-se a gerar Web sites comercias automaticamente. Esse tipo de solução leva a uma decisão de design muito comum, onde não existe meio termo ou solução híbrida. Essa escolha de design se refere ao momento no qual a loja será gerada e a forma pela qual essa loja será implementada.

No que se refere ao momento, existem várias soluções possíveis. A categoria produto, normalmente gera um loja localmente, ou seja na própria máquina do cliente. Isso possibilita ao cliente (desenvolvedor) utilizar a loja localmente, avaliar se o que foi gerado realmente corresponde às suas expectativas e somente após isso, utilizar algum mecanismo de publicação para publicar a nova loja na Internet. Note que essa publicação deve seguir algumas regras. A principal delas se refere ao Web server. Dificilmente as lojas geradas podem ser publicadas em um Web server qualquer. Sempre é necessário uma infra-estrutura por parte dos servidores de conteúdo (Web servers). Portanto, no próprio software gerador, muitas vezes, existe uma lista de servidores cadastrados para que o cliente escolha. Uma alternativa é exigir-se uma lista de pré-condições quanto à infra-estrutura e o cliente escolhe o melhor Web server que cumpre tais exigências. Como a geração é feita localmente e só depois é feita a publicação, o cliente tem acesso à todas as páginas html geradas. Por mais que o software tente não deixar transparecer seu mecanismo de

funcionamento, em qualquer solução de geração off-line, é possível em algum nível obter o código fonte das páginas geradas.

Já na categoria serviço, a geração da loja está intimamente ligada à sua publicação. Como não existe nenhum software instalado na máquina do cliente nesse tipo de software, a geração das páginas se dá no servidor onde o próprio serviço está sendo executado. Para que haja uma avaliação da loja por parte do cliente (desenvolvedor) antes que essa seja publicada, o mecanismo de geração/publicação é subdividido em mais uma etapa. Ao gerar uma loja, essa não é publicada automaticamente no endereço definitivo, mas num auxiliar, temporário. Assim, o desenvolvedor, pode analisar a loja gerada e, caso ela esteja de acordo com o especificado, através de um outro mecanismo, ela é copiada para o endereço definitivo. Nesse tipo de mecanismo, apesar de também ser possível analisar a loja antes de publicá-la na Internet, não se tem acesso às páginas geradas. Nenhum dos dois softwares analisados (nessa categoria) permite o download ou visualização do código fonte das páginas geradas. Por esse motivo também, o servidor hospedeiro dessas lojas será o mesmo do software gerador, ou algum outro, escolhido pelo próprio software ou pelo cliente (numa lista de possíveis Web servers disponibilizada pelo programa gerador).

Essas são as duas principais formas de instanciação automática. Note-se que as categorias (produto ou serviço) do software gerador definem como será o processo de geração e publicação das páginas.

Um outro aspecto quanto à forma de instanciação diz respeito ao tipo de código fonte que será gerado para a loja. Um loja virtual necessariamente tem um mecanismo de

persistência a ela associada. Esse mecanismo na maioria das vezes é um banco de dados relacional, responsável por guardar os mais variados tipo de informação, tais como: catálogo de produtos, estrutura dos departamentos dentre outros. E como essa informação será “costurada” com a loja virtual é isto o que define o tipo de código fonte da loja. Esse tipo de código classifica as lojas em duas categorias: as estáticas e as dinâmicas.

Nas lojas estáticas, as informações do banco de dados sempre devem estar disponíveis no momento da geração da loja. Isso porque cada produto terá uma página correspondente em html. Assim, no processo de geração da loja, o banco de dados é percorrido e todas as informações lá disponíveis são processadas para a definição das páginas geradas. Após esse processo, o banco de dados não mais será consultado, pois todas as informações já estarão nas diversas páginas html geradas. Nesse tipo de procedimento, existe uma página para cada produto, uma para cada departamento e assim sucessivamente. Um cliente visitando uma loja, ao abrir uma página de um produto, não estará fazendo nenhum acesso a um banco de dados. Uma vantagem desse método é a velocidade de acesso às páginas pelos browsers. Uma vez que a página é totalmente estática, não tendo nenhum acesso a banco de dados, seu upload pelo browser é extremamente eficiente. Por outro lado, qualquer mudança no catálogo de produtos no banco de dados, não reflete imediatamente na loja. Para que a mudança surta efeito, é preciso acionar novamente o procedimento de geração da loja, para que novas páginas sejam geradas e a mudança possa ser vista.

Na escolha dinâmica o contrário ocorre. Numa loja dinâmica, as páginas estão permanentemente ligadas ao banco de dados. Por exemplo, só existe uma página para todos

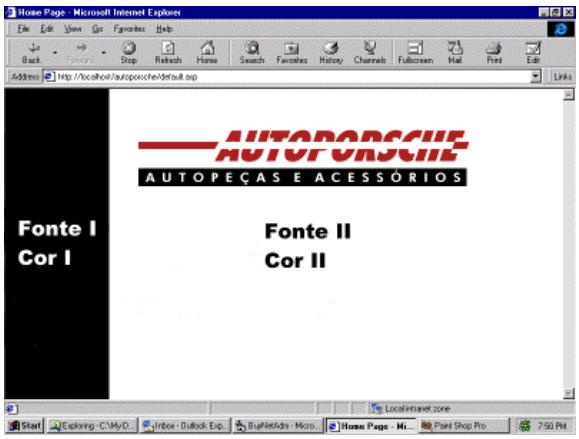
os produtos (espécie de template). Toda vez que um cliente requisita um produto, em tempo de execução, descobre-se qual produto está sendo requisitado. Assim, algum mecanismo busca as informações desse produto (nome, descrição, preço, foto e etc.) no banco de dados e só então, a página template do produto é montada e enviada ao browser cliente. Os benefícios são relativos às mudanças na base de dados, que automaticamente se refletem na loja virtual (sem a necessidade de acionamento do processo de geração da loja). Por outro lado, a solução carrega uma sobrecarga de acesso a banco de dados para cada página requisitada. Numa loja com grande movimentação ou num shopping virtual com um grande número de lojas, isso pode demandar uma infra-estrutura de banco de dados muitas vezes maior que a própria infra-estrutura de servidor de Web.

2.4 Customização da Interface

Um ponto importante nessa categoria de software é a solução adotada para a customização da interface da loja gerada. No processo de desenvolvimento tradicional (sem geração automática), o designer trabalha em conjunto com o desenvolvedor. Dessa forma, o Web site comercial tem a identidade visual que o designer especificar, enquanto o desenvolvedor se encarrega de encaixar os elementos da loja dentro dessa identidade. Ao contrário, num processo de geração automática, a interface também deve ser gerada automaticamente e assim, duas soluções são adotadas para esse problema; os templates ou a customização de cada elemento da interface.

Templates são solução pré-prontas de interface, por exemplo, uma loja com barra de navegação à esquerda, com o background branco e com as letras do tipo Arial, seria um possível template para uma loja..

O template mostrado na figura abaixo, deixa em aberto quatro parâmetros configuráveis. Cor e fonte podem ser escolhidos para as áreas do menu da esquerda e parte central.



Uma com barra de navegação no topo da página, com uma figura como background e com fonte Times Romam, seria um outro template possível.

Através de um menu, com todos essas opções, o cliente escolhe qual a melhor opção para o seu caso e cabe ao software gerador montar as funcionalidades da loja sobre o template escolhido.

Na customização dos elementos, não é oferecido nenhum template, a loja é gerada sempre com a mesma identidade visual. Mas através de um serviço (que pode ser o software da administração) de customização, é possível mudar o atributo de todos os elementos da interface. Por exemplo, existe a possibilidade de mudar o tipo de letra, a imagem ou cor do background, a posição de barra de navegação e assim por diante.

A solução de template é mais simples de ser implementada, pois não é necessário criar um módulo para customização da interface. Geralmente, o grande número de elementos de interface tornam o módulo extremamente extenso e de difícil implementação. Esse excesso de atributos caracteriza uma desvantagem para essa solução, pois o cliente para customizar a identidade visual do Web site comercial, é obrigado a alterar o valor de vários dos atributos da interface, um processo extremamente trabalhoso. E pior ainda, em todos esses softwares analisados, nenhum oferecia uma solução WYSIWYG para essa edição dos atributos. Por isso, para cada atributo mudado, não é possível ver imediatamente o impacto na identidade visual da loja, o que torna o processo de customização excessivamente demorado e uma sucessão de tentativas e erros.

Alguns dos softwares adotam uma solução híbrida, onde num primeiro momento (no questionário) escolhe-se um template, mas após a geração da loja existe um módulo de

customização da interface no qual é possível mudar os valores dos atributos dos elementos da interface. Naturalmente, a solução híbrida traz o melhor dos dois mundos. Apesar de ser a mais trabalhosa do ponto de vista de implementação, pois é necessário implementar as duas soluções, é a mais poderosa para o cliente modificar a identidade visual do Web site.

2.5 Trabalhos Relacionados

Nessa seção veremos vários produtos que implementam máquinas automáticas geradoras de Web sites comercias. Em cada caso, analisaremos as soluções adotadas por cada uma delas sobre os pontos levantados anteriormente. No final, faremos a mesma análise para o 2BuyNet, destacando que esse último adota soluções híbridas em função de ter sido especificado a partir dos produtos analisados.

2.5.1 Microsoft Site Server 2.0 Commerce Edition[6]

Solução da Microsoft para geração automática de Web sites comercias. O Site Server funciona como um produto. O usuário adquire o software, instala no computador cliente e respondendo um questionário que lhe é apresentado através de um wizard, cria uma loja.

[6] http://www.microsoft.com/siteserver/commerce/default.asp

As entidades identificadas nessa solução são: Departamento, Produto, Clientes, Pedidos e Promoções. Dessa forma, um modelo OMT dessa solução pode ser representado da seguinte forma:

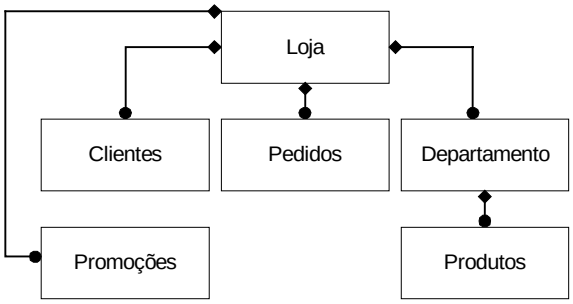


Figura 2.5-1 Diagrama de entidades Site Server

Sob o ponto de vista de cutomização de interface, o site server adota a solução de templates sem o recurso de edição dos atributos posteriormente. Apesar de não haver o recurso de edição dos valores do atributos, o Site Server disponibiliza o código fonte das páginas geradas para o desenvolvedor, permitindo assim qualquer tipo de customização sobre a interface. Como todas as demais soluções, o Site Server não prevê nenhum tipo de parametrização do conteúdo das páginas para uma posterior tradução.

2.5.2 AbleCommerce[7]

Produto de uma empresa especializada em comércio eletrônico, o AbleCommerce se enquadra na categoria de produto. Seu modelo de classes, se assemelha muito com o Site Server da MS.

[7] http://www.ablecommerce.com

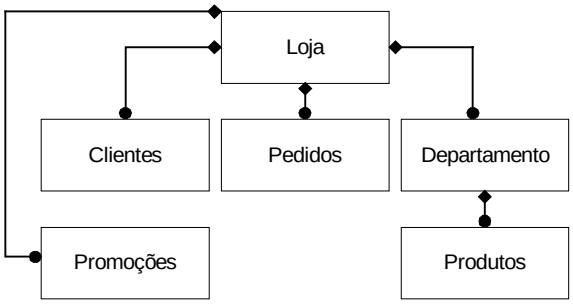


Figura 2.5-2 Diagrama de entidades AbleCommerce

Utilizando uma solução casada com um software chamado ColdFusion, que lhe permite criar páginas dinâmicas, sua solução gera páginas dinâmicas, sempre conectadas a banco de dados. Uma vantagem, pelo fato de ser um produto, é o acesso ao código fonte das páginas geradas para customização da interface, apesar de existir um módulo específico para essa função. A parametrização do conteúdo das páginas não é abordada pela solução, o que inviabiliza qualquer espécie de mecanismo para tradução da loja.

2.5.3 EZ2-Shop[8]

Com modelagem um pouco mais simples que as anteriores, o EZ2-Shop é um produto de um empresa também especializada apenas em comércio eletrônico. Conforme nos demais casos, todas as entidades estão representadas na forma mais tradicional possível: produto, departamentos, clientes e pedidos. A simplificação está na não representação explícita para promoções.

[8] http://www.ez2-shop.com

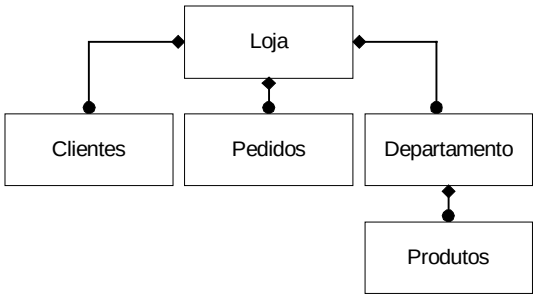


Figura 2.5-3 Diagrama de entidades EZ2-Shop

A interface é totalmente customizável, sendo esse o único mecanismo de alteração. Não existe a opção de escolha por templates da mesma forma como não há infra-estrutura para tradução das lojas por ele geradas.

2.5.4 Inex Commerce Court[9]

Fruto do interesse da Compaq pelo ramo do comércio eletrônico, o Inex é um produto que segue a linha tradicional. Uma entidade nova encontrada na modelagem, o fornecedor, associando dessa forma, produtos a fornecedores. A modelagem também prevê promoções, o que o torna um dos mais completos na representação do domínio de uma loja.

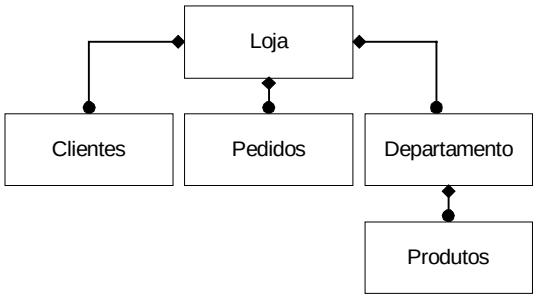


Figura 2.5-4 Diagrama de entidades Inex

[9] http://www.compaq.com/smb/onlineservices/commerce.html

É possível também alterar o valor dos atributos da interface. O Inex está muito próximo ao EZ2-Shop nesse aspecto. Ambos implementam módulos para customização da interface, mas não oferecem o auxílio inicial de um template. Sua forma segue o grupo dos produtos, o que lhe confere todas as características de um software dessa categoria.

2.5.5 Icat[10]

Primeiro na categoria de serviço, o Icat carrega todas as características desse tipo de solução. A interface é customizável através do próprio serviço, havendo mesmo assim, opções de templates. As entidades são apenas as básicas, pedidos, produtos, departamentos e clientes.

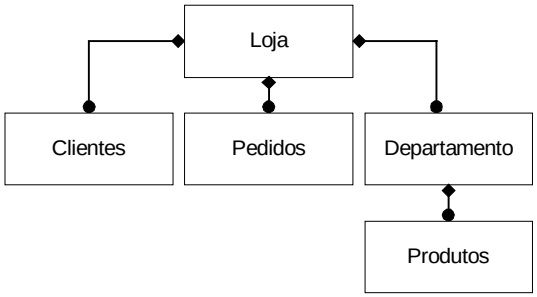


Figura 2.5-5 Diagrama de entidades ICat

Apesar de ser um serviço, também não há opções para tradução da loja. A parte administrativa também é um serviço no qual o cliente (administrador) se conecta através de um browser num endereço diferente do da loja.

[10] http://www.icat.com

2.5.6 Viaweb[11]

A Viaweb foi a pioneira no serviço de geração automática de lojas virtuais e comparativamente com o Icat, oferece mais funcionalidades. Inicialmente, o seu modelo falhava por não oferecer uma modelagem para clientes (o que nas versões mais recentes já foi incorporado). Apesar disso, a Viaweb já oferecia recursos avançados como customização da interface por templates e edição dos valores dos atributos individualmente, upload/download para edição da base de dados off-line, dentre outros.

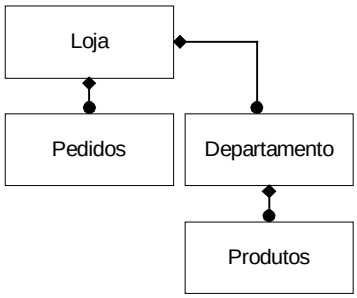


Figura 2.5-6 Diagrama de entidades Viaweb

Uma loja gerada pela Viaweb trabalha de forma estática (assim como o 2BuyNet) e sua publicação também passa por um processo de 2 etapas. Apesar de ser o serviço a mais tempo em operação, a Viaweb ainda não oferece nenhum recurso para tradução do conteúdo das lojas.

[11] http://www.viaweb.com

2.5.7 2BuyNet[12]

O 2BuyNet é um serviço para geração automática de lojas para a Internet desenvolvido no laboratório de Engenharia de Software da Pontifícia Universidade Católica do Rio de Janeiro. No contexto desse estudo de trabalhos relacionados, ele é o produto extraído da análise dessas diversas soluções. Por isso, muitas das decisões de design adotadas nele, são soluções resultantes da conjunção dos outros produtos. Além disso, o 2BuyNet implementa recursos inéditos dos demais produtos, inspirados na necessidade demandada pelas lojas, mas não disponíveis nos outros softwares.

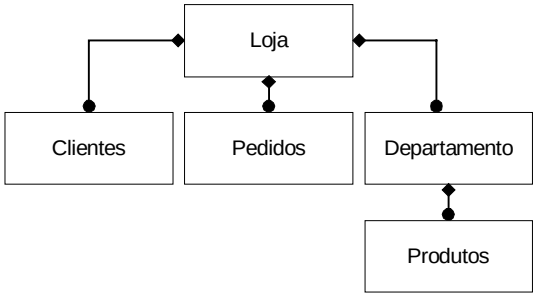


Figura 2.5-7 Diagrama de entidades do 2BuyNet

O 2BuyNet pressupõe uma estrutura hierárquica dividida em departamentos e sub-departamentos para organização de uma loja. Os produtos são totalmente customizáveis, exigindo-se apenas um mínimo de atributos para caracterizar um produto, esses atributos são: nome, descrição, foto e preço. Qualquer especificação adicional pode ser configurado pelo software, que implementa uma meta base de dados para representar tais configurações dos produtos. Um exemplo seria uma loja de vestuário, onde os produtos não teriam apenas nome, descrição, foto e preço, mas também tamanho e cor.

[12] http://www.2buynet.com.br

Esses dois atributos, tamanho e cor, foram definidos pelo cliente (desenvolvedor) e poderá ser atribuído a todos os produtos da loja ou apenas a alguns deles.

As principais entidades previstas pelo 2BuyNet são os produtos, os departamentos, os clientes, os pedidos e o idioma. Seu mecanismo não implementa nada relativo a promoções, mas ao contrário dos demais, indexa todo conteúdo da loja quanto ao idioma. Uma decisão de design semelhante a implementada pelo AulaNet [1, 5]. Assim, é possível traduzir e publicar uma loja sem a necessidade de reconstruí-la.

Por ser um serviço, o 2BuyNet não disponibiliza os fontes das páginas html para o desenvolvedor. No processo de geração de uma loja, esta é gerada e publicada dentro do próprio servidor de Web do 2BuyNet. O processo é dividido em duas etapas, conforme descrito anteriormente, nesse tipo de procedimento. A primeira etapa consiste na geração e publicação temporária e só posteriormente, a publicação na Internet.

Quanto ao tipo de código fonte gerado, o 2BuyNet adota diferentes soluções para diferentes problemas. Em relação ao banco de dados de produtos e departamentos, o 2BuyNet adota uma posição estática, ou seja, existe uma página html para cada produto/departamento descrito no banco de dados. Mas em relação à independência da língua, o 2BuyNet adota uma solução dinâmica. Assim, caso o desenvolvedor tenha traduzido uma loja para uma outra língua diferente da língua mãe, não serão gerados sites diferentes. Existe apenas uma versão do site da loja, mas para cada elemento apresentado

na interface, existe uma chamada ao banco de dados parametrizado pela língua para mostrá-lo.

Em relação a customização da interface o 2BuyNet também adota uma solução mista. Existe um conjunto de templates, para a escolha do cliente (desenvolvedor), mas que são fixos e não podem ter seus atributos alterados posteriormente. A solução é híbrida, pois existe uma API de interface disponível para personalização do site. Nessa API encontra-se TAG's específicas para a criação de uma loja, como por exemplo, <CC>, que indica: "mostre carrinho de compras". Assim, caso um designer deseje desenvolver sua própria identidade visual para uma determinada loja, mas queira utilizar o 2BuyNet, basta fornecer os templates com os TAG's definidos pelo 2BuyNet, e a loja será gerada a partir do design fornecido.

Uma grande vantagem do uso do 2BuyNet para desenvolvimento de uma loja virtual não é apenas o ganho de tempo no desenvolvimento, mas também toda infraestrutura de administração que funciona igualmente para todas as lojas por ele construída. Todos os recursos para geração e administração estão disponíveis no site de administração do 2BuyNet.

Todos os mecanismos para atualização dos bancos de dados, estão presentes tanto on-line quanto off-line. Pelo 2BuyNet é possível, por exemplo, alterar a descrição de um produto através do browser (alteração on-line), como também é possível, fazê-lo off-line. No processo off-line, o cliente (administrador) executa um download (em forma de planilha) do banco de dados que o interessa (nesse caso, de produtos), altera da forma que desejar e depois executa o processo inverso (faz um upload da planilha alterada para o 2BuyNet). Ao receber essa planilha, o programa se encarrega de executar as mudanças no banco de dados. Pelo fato do 2BuyNet não ser dinâmico em relação a alguns banco de dados, nesse caso específico de uma alteração nos produtos, existe a necessidade de uma nova geração da loja.

A aplicação 2BuyNet dentre todas as funcionalidades implementadas, guarda uma característica incomum aos demais softwares relacionados, o fato de ser uma instanciação de um framework. Todos os demais softwares foram desenvolvidos como softwares sem a preocupação de separar interface e aplicação.

3. FrameWork

3.1 Conceituação de Frameworks

O reuso de software tem sido o objetivo da engenharia de software há décadas, buscando uma solução, experiências com frameworks mostraram [27,28] um alto grau de reuso em aplicações construídas a partir de frameworks. No desenvolvimento de softwares complexos o uso de frameworks é cada vez mais comum. Dados os novos requerimentos de qualidade e desenvolvimento de software, o papel dos frameworks no design da arquitetura do software tem evoluido de diferentes maneiras. Com o avanço do paradigma de orientação a objetos, o reuso de componentes maiores tornou-se possível resultando na definição de frameworks orientados a objetos. Um framework orientado a objeto pode ser definido [26] como um conjunto de classes que carregam um design abstrato para solucionar uma família de problemas co-relacionados.

O desenvolvimento de uma aplicação modelada a partir de um framework, sugere fases no processo de construção. Essas fases, comentadas por alguns autores, por exemplo Mattsson [28], são descritas:

Desenvolvimento do framework

Normalmente na fase que consome o maior esforço, o objetivo é elaborar um design reusável e uma implementação relativa ao domínio do problema. Os resultados obtidos dessa fase são uma análise do modelo do domínio e um framework orientado a objeto.

Uso do framework

Também mencionado como **Instanciação do framework** ou **Desenvolvimento da aplicação**. O resultado dessa fase é uma aplicação desenvolvida

reusando o framework. O desenvolvedor deve definir as classes abstratas do framework de acordo com a necessidade da aplicação e escrever classes novas para completar a necessidade da aplicação não prevista pelo framework.

Evolução e Manutenção do framework

Um framework, como qualquer software, está sujeito a mudanças. Causas dessas mudanças podem ser erros notificados pelos usuários das aplicações, identificação de novas abstrações para modelar o domínio, etc.

3.2 Aplicação 2BuyNet versus FrameWork 2BuyNet

2BuyNet é um nome genérico que assume duas versões nessa dissertação. Algumas vezes ele será apresentada como Aplicação 2BuyNet e outras vezes como FrameWork 2BuyNet. O FrameWork 2BuyNet é o framework proposto para criação de lojas virtuais para a Web, suas características e o modelo serão apresentados nessa dissertação. A aplicação 2BuyNet é um serviço publicado na Web, no endereço http://www.2buynet.com.br, que serve como interface para que usuários sem conhecimento de nenhuma linguagem de programação possam utilizar o framework. Pode-se entender a aplicação 2BuyNet de várias formas. Ela é uma instanciação do framework 2BuyNet numa interface Web, ou uma "casca de interface" que roda por trás de uma máquina geradora de lojas para a Internet, que também guia o usuário através das várias etapas do uso do framework até a geração de uma loja.

Por natureza da interface escolhida para a aplicação 2BuyNet e principalmente por causa do público ao qual a ferramenta se destina, não há uma relação um para um nos

recursos disponibilizados na aplicação 2BuyNet. Ou seja, o uso do framework 2BuyNet através da interface 2BuyNet, é mais restritivo. Não são todos os recursos que estão disponíveis. Mas, entre as várias etapas pela qual o framework deve passar até que um loja seja instanciada, o uso pela aplicação é mais curto e mais simples.

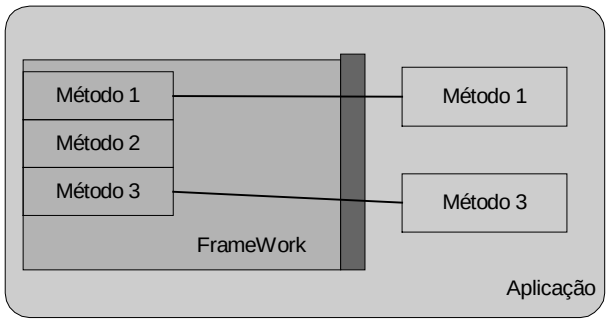


Figura 3.2-1 Aplicação como um “wrapper” e o não mapeamento entre os métodos

3.3 Modelo conceitual da Loja

Uma loja, em última instância, é o resultado final produzido pelo framework. Essa loja é uma aplicação, com arquitetura em três camadas [22,33] (Interface, Aplicação e Persistência), com funcionalidades de uma loja virtual. Como aplicação prática, a camada de interface foi projetada para a plataforma HTML, de forma que a loja virtual gerada pelo framework pudesse ser utilizada na Internet e o framework se tornasse um gerador de lojas para a Internet.

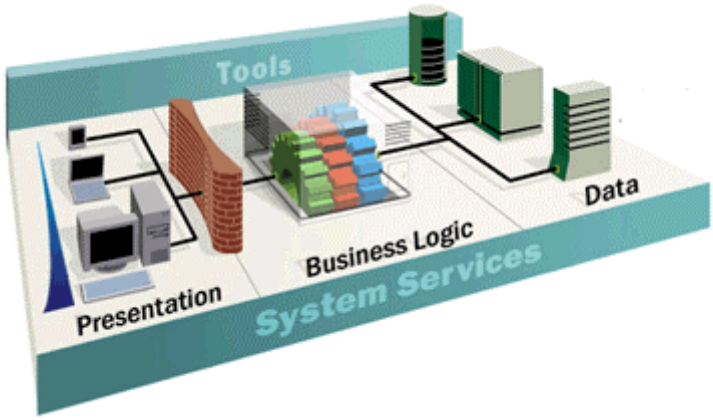


Figura 3.3-1 Arquitetura três camadas. DNA Architecture

A loja tem uma estrutura de dados customizada, em vários níveis, pelo framework. Essa customização acontece tanto na camada da interface (lojas com designs visuais diferentes) quanto na estrutura de banco de dados (meta-banco de dados para representar produtos com diferentes atributos). O framework atua nas três camadas da loja existindo métodos responsáveis tanto por criação da Interface (páginas HTML) quanto por criação da Persistência (tabelas do banco de dados).

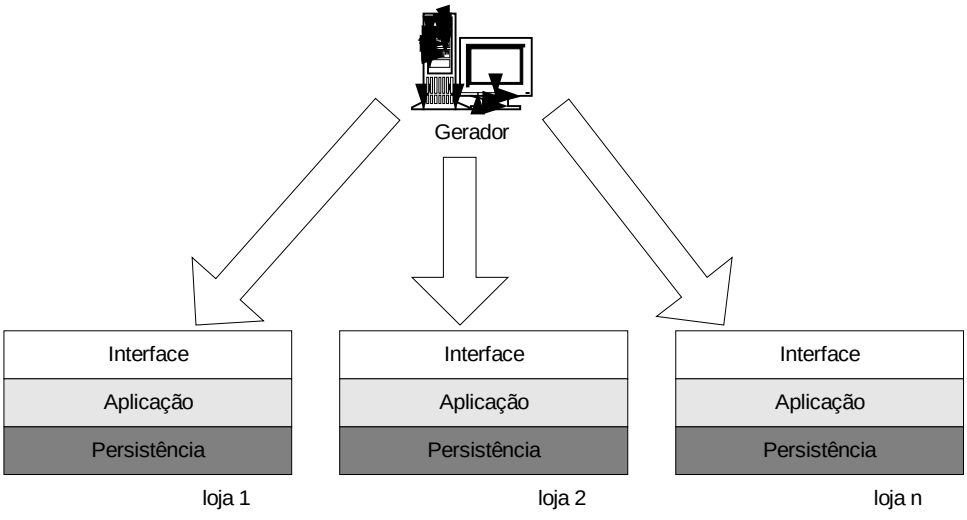


Figura 3.3-2 Geração de várias lojas três camadas

A estrutura de dados da loja comporta um modelo de lojas dividido em departamentos e sub-departamentos. Os produtos são as folhas da árvore. A árvore pode ter qualquer profundidade e qualquer número de ramificações por nó, tanto para as folhas quanto para os demais nós. Dessa forma, qualquer loja adaptável nessa estrutura pode ser criada e administrada pelo framework do 2BuyNet.

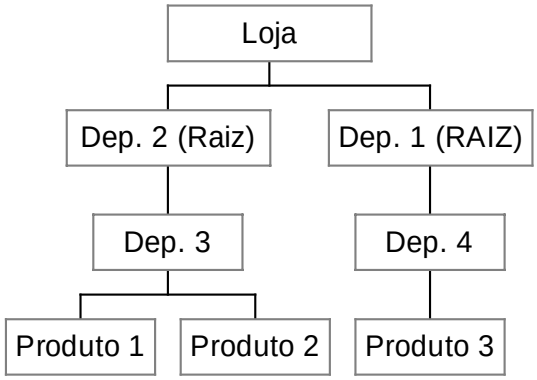


Figura 3.3-3 Estrutura n x m

Podemos tomar, como exemplo, uma loja de Cd-Rom educacionais e de treinamento. Essa loja poderia ter departamentos como Informática e Infantis. O departamento de Informática poderia estar subdividido em Aplicativos, Design e Internet , enquanto o departamento Infantis estaria subdividido em Alfabetização e Percepção Visual. E, finalmente, em cada um desses subdepartamentos estariam os produtos.

Da mesma forma que a estrutura da loja é customizável, a definição de um produto também é. Para que um produto possa ser manipulado pelo framework do 2BuyNet ele deve ter alguns atributos mínimos:

- Nome

  Campo texto descrevendo o nome do produto

- Descrição

  Campo texto com descrição de vendas, características do produtos

- Preço

  Preço do produto

- Peso

  Peso do produto para efeito de cálculo do frete

Qualquer produto que se adeque a esse perfil pode ser tratado pelo framework. Produtos que precisem estender esses atributos também podem ser manipulados. Para criar novos atributos, o framework trata uma entidade denominada característica do produto, posteriormente existem métodos para associar essas características a determinados produtos. Através desse mecanismo, é possível, por exemplo, criar um atributo chamado Tamanho, com valores do tipo Pequeno, Médio e Grande. Criado o atributo ou característica do produto, é necessário ainda, associá-lo a um determinado produto, por exemplo uma camisa. A loja poderia vender uma camisa, e o usuário pode escolher, num mesmo produto, três variantes de tamanho.

Definida a estrutura (departamento e sub-departamentos) e os produtos, o framework se propõe a definir métodos e atributos que permitam manusear esses elementos em um determinado nível de abstração. No caso do 2BuyNet, esse nível de abstração nos permite desvincular dos tipos básicos de qualquer linguagem, bem como suas estruturas de dados; e nos permite tratar os elementos como entidades atômicas. O framework do 2BuyNet trabalha então com abstrações que definem, por exemplo, produtos, departamentos, clientes, etc. Define também ações que os tratam, por exemplo, criarDepartamento, alterarDepartamento, criarProduto, etc.

A esses elementos tratados foi dado o nome de Entidades, cada entidade, define de alguma forma o que o framework entende, dando uma idéia de quais elementos dentro do domínio de loja/comércio o framework 2BuyNet trata. As entidades são respectivamente:

- **Produtos**

  Engloba todas as informações e ações que definem e tratam elementos relacionados a produtos.

- **Departamentos**

  Da mesma forma como Produtos, envolve tudo relacionado a Departamentos.

- **Atributos de Produtos**

  Pode ser considerada uma entidade separada mas num determinado momento se associa a Produtos.

- **Clientes**

  Trata os clientes que passaram pela loja.

- **Pedidos**

  Refere-se aos pedidos recebidos por uma loja até o momento do seu atendimento e processamento.

- **Design**

  Envolve todas as ações para criação e customização da interface de uma loja.

- **Funcionalidades**

  Envolve as funcionalidades que uma loja implementará, por exemplo, processamento de cartões de crédito on-line ou aceitação de boleta bancária como forma de pagamento.

Essas entidades ainda não devem ser entendidas como classes no design do framework. Na verdade, elas são apenas abstrações dos elementos centrais e que de alguma forma o framework 2BuyNet irá tratar. Para exemplificar melhor, um contra-exemplo: uma entidade não tratada pelo framework 2BuyNet é a entidade Produtores. Por isso, durante toda a explicação que se seguirá dividida nos diversos capítulo seguintes, nunca haverá referências a métodos que manipulem produtores ou formas de se associar produtos à

produtores e etc. Para o framework 2BuyNet (pelo menos nessa versão), Produtor é um conceito inexistente.

Veja abaixo uma tabela comparativa entre o framework 2BuyNet e alguns softwares geradores de lojas virtuais:

	Departamento	**Produto**	**Cliente**	**Pedido**	**Promoções**	**Produtores**
Site Server	**X**	**X**	**X**	**X**	**X**	
Inex	**X**	**X**	**X**	**X**		**X**
Ez2Shop	**X**	**X**	**X**	**X**		
Able Commerce	**X**	**X**	**X**	**X**	**X**	
Icat	**X**	**X**	**X**	**X**		
ViaWeb	**X**	**X**		**X**		
2BuyNet	**X**	**X**	**X**	**X**		

A especificação de quais entidades esses softwares tratam, como seus códigos fontes, não são disponibilizados. Foram levantadas empiricamente através do uso dos respectivos programas. As entidades, por sua vez, foram definidas justamente pela repetição de conceitos que de uma forma ou de outra reapareciam nos diversos softwares analisados.

As entidades do framework 2BuyNet, se relacionam de acordo com o diagrama baixo:

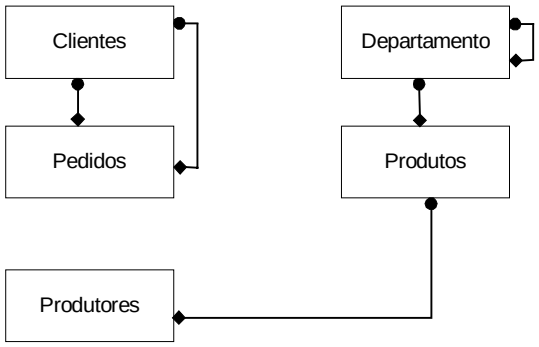


Figura 3.3-4 Relacionamento entre as entidades

Esse diagrama não segue estritamente nenhuma linguagem de modelagem orientado a objeto, porque as entidades tratadas não podem ser mapeados em nenhum dos elementos convencionais das linguagem de modelagem orientadas a objetos.

Uma característica tratada pelo framework que não pode ser encarada como uma entidade é a funcionalidade multilíngue da interface. Pelo fato do framework do 2BuyNet gerar lojas com interface para a Web, uma funcionalidade interessante seria se essa interface fosse parametrizada pela língua. Nesse caso, de alguma forma, se escolhessemos outra língua, toda a interface se transformaria e os textos e figuras passariam a estar escritos no novo idioma escolhido.

Essa multiplicidade de línguas segue a mesma idéia do software AulaNet [1], e não deve ser encarada como uma entidade. A diferença está no fato da língua ser uma característica apenas da interface, uma solução para um problema muito mais no domínio da implementação do que da aplicação.

Seguindo a mesma idéia da multiplicidades de línguas como uma funcionalidade do framework, o 2BuyNet implementa também um mecanismo de importação e exportação de dados. Em se tratando de um framework que faz interface e manipula banco de dados, os métodos do framework não só inserem / alteram / deletam dados, mas também permitem e importação e/ou exportação desse dados de uma forma atômica. Esse mecanismo também deve ser encarado como uma funcionalidade do framework e não como uma entidade do domínio de lojas tratada pelo 2BuyNet.

Essas duas funcionalidades, múltiplos idiomas e importação/exportação de dados, serão tratadas detalhadamente nos capítulos que se seguem. O objetivo principal desse capítulo é apresentar os conceitos principais do framework e caracterizar em que nível da solução do problema o framework atua. Os elementos e entidades serão abordados detalhadamente e separadamente mais adiante.

4. Processo de Instanciação do FrameWork

Nesse capítulo descreveremos as quatro etapas principais do framework 2BuyNet. Como descrito anteriormente, o framework 2BuyNet se propõe a criar uma loja virtual. Esse processo não se dá de uma só vez, mas em partes separadas e em momentos distintos. O framework disponibiliza ferramentas que atuam em todos esses momentos. As partes são:

- Coleta de Dados
- Geração da Loja
- Inteligência e Controle da Loja
- Importação/Exportação de Dados

Um diagrama exemplifica como essas partes interagem. A aplicação 2BuyNet engloba a geração, a coleta e a importação/exportação de dados. As lojas, por sua vez, a medida que são geradas, têm seus bancos de dados próprios, mas compartilham uma camada chamada de Carrinho de Compras, que representa a “inteligência da loja”.

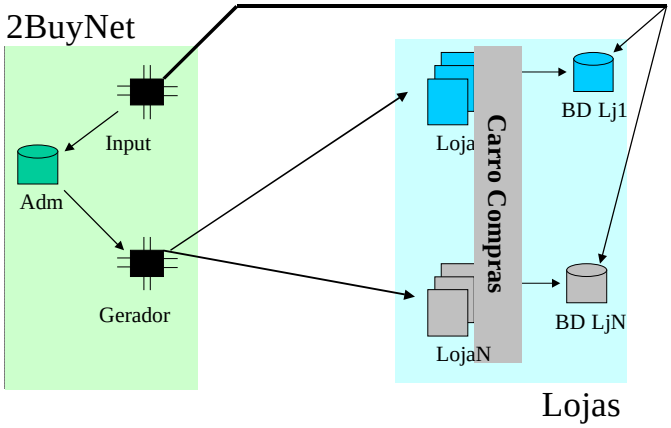


Figura 3.3-1 Relacionamento entre as partes do Framework e da Aplicação

Como resultado da fase de desenvolvimento do framework tem-se um entendimento da necessidade do domínio do problema de comércio eletrônico e como abordá-lo. Desse entendimento, uma proposta de como projetar o framework, seus pontos de flexibilização e suas funcionalidades principais. A divisão do framework em partes é uma proposta de flexibilização e de como num processo de instanciação, uma aplicação deve usar este framework. Pela Figura 3.3-1 com destaque na primeira parte, nota-se uma divisão do framework em duas partes, a coleta e a geração. Essas duas etapas são as partes principais do framework. Carrinho de compras e Importação/Exportação são duas características implementadas pelo framework que trabalham no momento da utilização e manutenção da loja.

4.1 Coleta de Dados

A aplicação 2BuyNet[13] atua principalmente na coleta dos dados para que as outras partes os consultem posteriormente. Esses dados são as informações que dizem respeito ao domínio do problema e devem ser fornecidos por um especialista da área. Os dados são basicamente de 5 tipos diferentes.

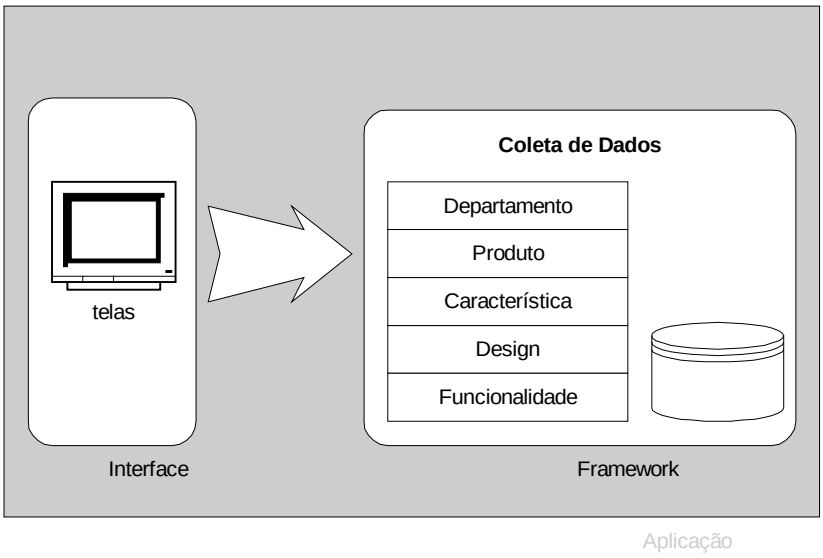


Figura 4.1-1 Diagrama do Coletor de Dados

Dados do Produto

Informam os produtos com os quais a loja trabalha. Alguns de seus atributos fixos são:

Nome

[13] Aplicação 2BuyNet é diferente do framework 2BuyNet. O framework, como o próprio nome diz, serve como instrumento de apoio. A aplicação 2BuyNet é uma aplicação comercial, totalmente construída com o framework 2BuyNet. Pode-se dizer também qua a aplicação 2BuyNet é uma instância do framework 2BuyNet, com uma interface construída para a Web.

Descrição

Peso

Preço

Departamento

Esses são os atributos fixos e relativos ao problema específico do domínio de lojas. Mas em se tratando da implementação de um software comercial, outros problemas devem ser tratados e para isso há reflexos no banco de dados. Veja abaixo a estrutura de dados completa para produtos:

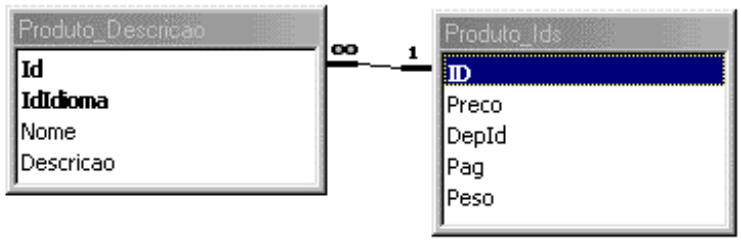


Figura 4.1-2 MER Produtos

No diagrama podemos notar um esquema de dados que se repetirá em praticamente todas as entidades tratadas pelo framework 2BuyNet. Esse esquema se propõe a resolver a questão das multiplicidades de idiomas. Observe-se que a tabela que define os produtos está separada em duas. A primeira denominada de **Produto_Ids** carrega informações sobre o produto que dizem respeito a essência do termo produto, ou seja, informações associadas a produto por si só, independentemente do idioma do país onde esse produto está sendo disponibilizado ou mostrado. Esses atributos são:

Preço

Departamento

Peso

Note que preço poderia ser considerado um atributo que depende do idioma. Dependendo da língua na qual o usuário esteja vendo a loja, o preço variaria. Por exemplo, um site visto em inglês mostraria os preços em dólar, já o mesmo site visto na sua versão em português mostraria os preços em reais.

A decisão sobre preço como um atributo independente de idioma foi tomada em tempo de design. Naquele momento, entendeu-se preço dessa forma, pois as legislações dos países raramente permitem que se venda um produto dentro de um país com preços cotados em outra moeda. Isso não significa que não seja possível exportar. Normalmente, quando um estrangeiro adquire um produto, o pagamento é feito em moeda do país da compra. Posteriormente, ao efetuar o pagamento, algum mecanismo pode ou não converter o tipo de moeda para o do tipo do país no comprador. É dessa forma, por exemplo, que funcionam os cartões de crédito. Ao executar um compra, o cliente paga na moeda do país da compra e posteriormente ao pagar a fatura do seu cartão, as compras efetuadas em outras moedas são convertidas de acordo com algum câmbio legal e pagas na moeda do país do cliente. Essa foi a razão principal pela qual o preço foi considerado um atributo independente de produto.

Os dois únicos atributos que foram considerados como dependentes de idioma foram Descrição e Nome, ambos por descrevem textos sobre os produtos e esses textos só fazem sentido nas suas respectivas línguas.

Um terceiro atributo que dever ser notado está na tabela de Produtos_Ids, o Pag. Ele guarda a informação sobre qual página (arquivo HTML) é a página do produto em questão. Essa informação é necessária na implementação de uma funcionalidade da loja e esse atributo existe apenas por necessidade de implementação.

Em momento algum no design da solução foi adotada um metodologia de criação do banco de dados (modelagem do db) que privilegiasse ou separasse atributos das várias camadas da aplicação. Essa decisão reflete-se nesse atributo Pag. Uma possível regra na modelagem poderia agrupar os atributos níveis ou camadas de acordo com a sua abstração na solução do problema. Certamente, Peso (ou qualquer um outro) e Pag não estão no mesmo nível, e consequentemente deveriam estar em tabelas distintas. Por motivos de simplificação, essa decisão não foi adotada e ambos atributos coexistem na mesma tabela.

Novamente, existe uma separação dos atributos relativos a produto. Esse não direcionamento na definição dos atributos se deve ao fato da multiplicidade de idiomas, que requer uma parametrização a mais.

Dados do Departamento

Com uma estrutura mais simples do que Produtos, os dados sobre departamento guardam informações sobre as subdivisões de uma loja. Uma loja pode ser dividida em vários departamentos, que por sua vez, podem conter outros departamentos, numa estrutura recursiva. Essa estrutura termina numa folha onde um departamento não mais contém sub-departamentos, mas somente produtos.

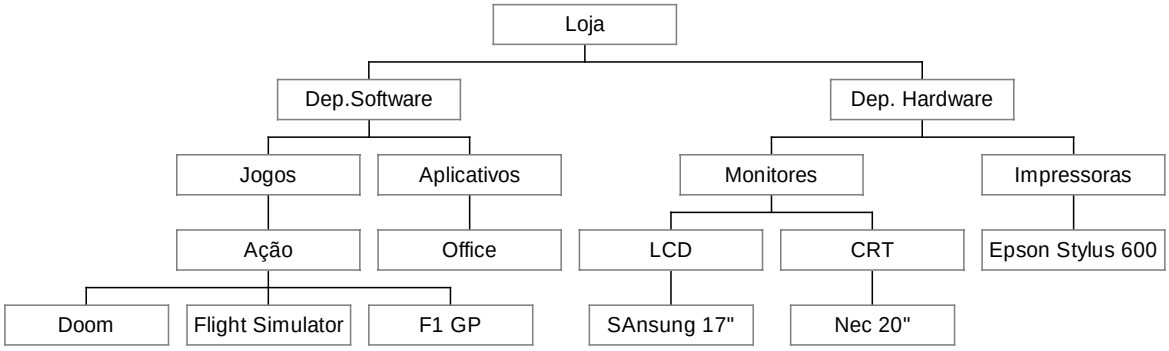


Figura 4.1-3 Estrutura de árvore com vários departamentos

Essa árvore é implementada apenas com a informação, guardada por departamento, de quem é o pai dele (atributo IdFather – Independente de Idioma). Os departamentos que não contém pai, departamentos raiz, têm IdFather iguais a zero (número reservado para a raiz).

De forma semelhante, existe um relacionamento[14] entre produto e departamento. Essa estrutura tem suas limitações, porém atende suficientemente bem a sua finalidade com um grau de complexidade adequado. Note que ela não impede que um departamento

[14] Representado na tabela **Produto_Ids** através do campo **DepId**. Representando qual departamento aquele produto faz parte.

contenha sub-departamentos e produtos, mas impede que um determinado produto tenha apenas uma entrada no banco de dados e pertença a diferentes departamentos.

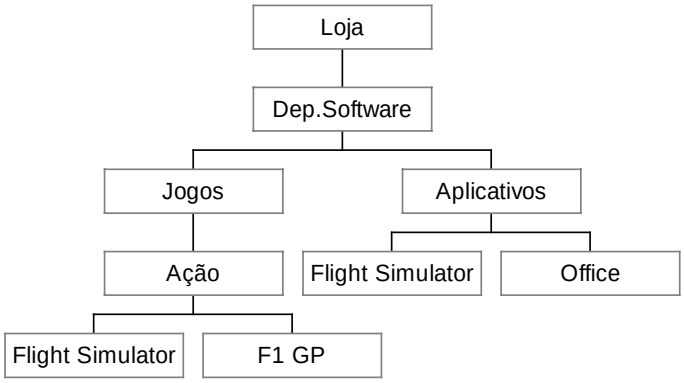
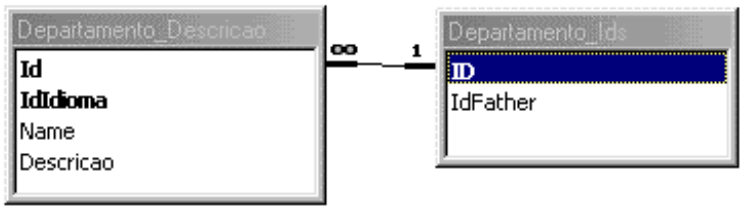
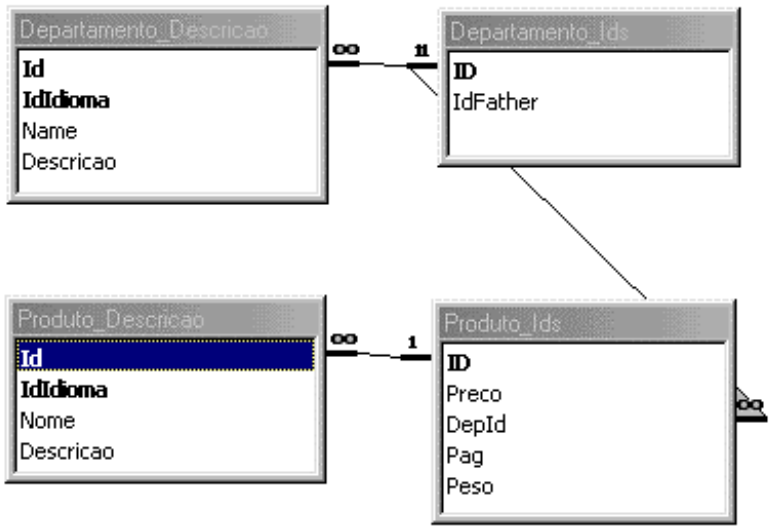


Figura 4.1-4 Estrutura de Departamentos/Produtos impossível

No que diz respeito à multiplicidade de idiomas o problema dos departamentos é resolvido da mesma forma como para os produtos.





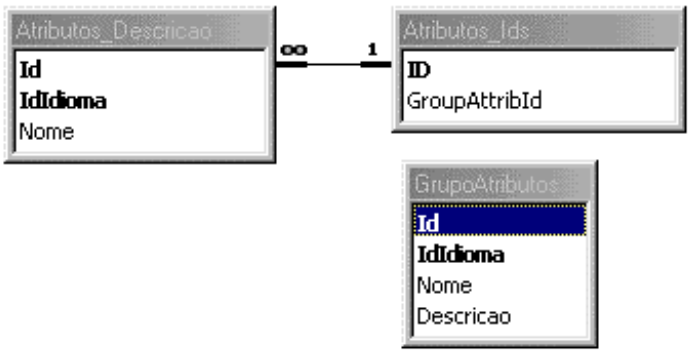
Dados de Características de um Produto (Meta Dados)

Os dados que descrevem os produtos estão representados em suas respectivas tabelas. Esses dados descrevem apenas características mínimas comuns para todos os produtos. Dado a característica aberta do framework 2BuyNet e sua proposição de não restringir nenhum tipo de produto, faz-se necessário que o framework implemente de alguma forma um esquema de meta-banco de dados, ou seja, uma estrutura de banco de dados que descreva os dados e posteriormente guarde os valores desses dados.

No caso do framework do 2BuyNet esse meta banco de dados serve para descrever e guardar características dos produtos. Um exemplo dessa funcionalidade se apresenta numa loja de departamentos. Digamos que uma loja venda produtos com as mais variadas características. Num departamento, vende-se comida enquanto em outro, calçados. Para

descrever o produto comida, as características mínimas[15] projetadas para cada produto já são suficientes, e por isso não precisam de nenhum meta-dado. No caso de calçados, existe o dado tamanho que pode assumir valores variados (ex. 39, 40 e 42). Para esse dado, o framework 2BuyNet propõe a criação de um meta-banco para guardar ambas as informações; a descrição do campo Tamanho e seus valores.

Existem duas entidades para descrever esse meta-banco, o Grupo e o Atributo. Entende-se como grupo, o elemento agregador de valores, no nosso exemplo anterior, Tamanho. Atributo são os diversos valores que um elemento de grupo pode receber, ainda no exemplo anterior, 39, 40 e 42. Essas entidades estão representadas pelas tabelas GrupoAtributos, Atributos_Ids e Atributos_Descricao. Conforme o MER abaixo:



Atributos para também ser independe de idioma, segue a risca o esquema da subdivisão de suas tabelas em duas; as que descrevem seus valores dependentemente do idioma e as que carregam valores absolutos, que pertencem apenas ao conceito de

[15] Nome, Descrição, Preço e Peso

atributos. Já a entidade Grupo não contém nenhum atributo absoluto e por isso é possível representá-lo em apenas uma tabela e ainda assim garantir sua representação com multiplicidade de idiomas.

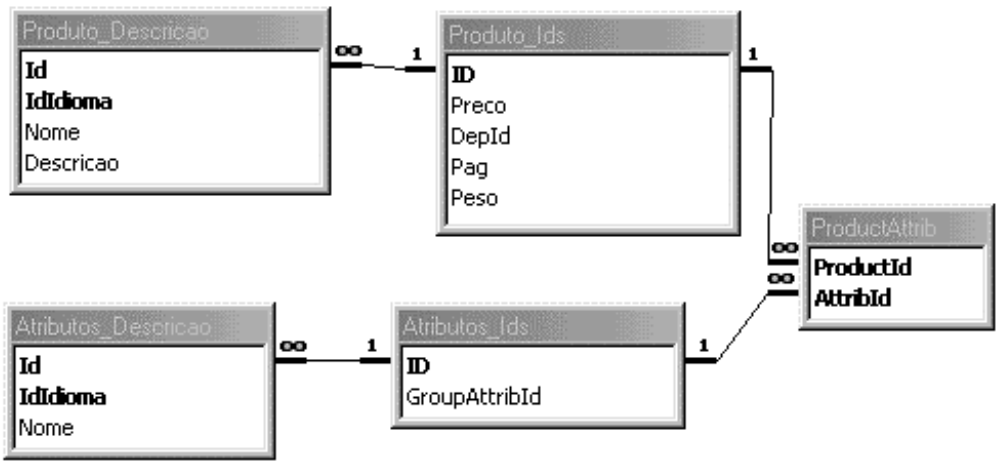
Definida a forma de representação desses dados, ainda resta criar um mecanismo que relacione um produto com nenhuma, uma ou mais características. Em se tratando de um meta-banco de dados, esse relacionamento pode ocorrer em dois níveis. Pode haver um relacionamento entre produto e grupo ou entre produto e atributo. Existem limitações de acordo com a solução escolhida.

A mais simples, mas mais restritiva ocorre caso haja um relacionamento entre produto e grupo. Nesse caso um determinado produto ao se associar com um grupo, carrega todos os valores desse grupo. Por exemplo, imagine uma loja de vestuário. Caso ela venda camisas e sapatos e deseje criar uma característica para produtos chamada tamanho. Todas as suas camisas são do tamanho P, M e G e seus sapatos dos tamanhos 39, 40 e 42. Não será possível criar apenas um característica chamada Tamanho com valores P, M , G, 39, 40 e 42. Ao fazer associação de um sapato com Tamanho, seus possíveis valores seriam P,M,G,39,40 e 42. A solução seria criar uma característica para cada família de produto. Nesse caso, características razoáveis seriam Tamanho_Sapato com valores 39, 40 e 42. Outra característica seria Tamanho_Camisa com valores P, M e G. O que parece bem razoável porque não há o por quê agrupar valores de ordem de grandezas diferentes sob o mesmo grupo.

Nesse caso então, uma associação entre produto e grupo não parece ser restritiva. Imagine-se agora um exemplo mais completo, onde a loja tem camisas também nos tamanhos P,M,G, só que nem todos os modelos de camisas são disponíveis nos três tamanhos. Digamos a camisa A tem no tamanho P e M (pois é mais para o público infantil) enquanto a camisa B, é disponibilizada nos três tamanhos. Da mesma forma que no exemplo anterior, a única saída será criar grupos diferentes, um chamado Tamanho_A e outro Tamanho_B. Numa projeção dessa solução para a realidade das lojas e a diversidade de produtos, constata-se que quase haveria um grupo para cada produto, pois seria quase impossível haver repetição de valores de elementos de grupos para associar com dois ou mais produtos. Uma conseqüência ainda pior do que o grande número de grupos seria o fato do especialista da loja ter que imaginar previamente quais os vários grupos deveriam ser criados dada a restrição no momento da associação. Isso traria um preocupação decorrente da implementação do software no nível de decisão do especialista no domínio.

A segunda forma de associação elimina esse problema, mas é mais complexa de ser implementada. A solução prevê uma associação entre o produto e os valores do atributo. Assim, para solucionar o último exemplo, a camisa A faria uma associação com o valor P e M do grupo Tamanho_Camisa, enquanto a camisa B faria associações com os valores P, M e G também do grupo Tamanho_Camisa.

O framework 2BuyNet adota a segunda solução como forma de associação entre características de produtos e produtos. O diagrama Modelo Entidade e Relacionamento dessas entidades é apresentado a seguir:



Vale ressaltar que esse relacionamento independe de idioma. Um produto tem suas características independentemente do idioma apresentado. Por isso, as tabelas envolvidas não apresentam nenhum campo envolvendo idioma.

Dados de Design

O modelo de customização da interface do framework 2BuyNet é dividido em duas formas: Templates e design próprio. Por suas abordagens diferentes na forma de coleta de dados, suas respectivas representações no banco de dados são também diferentes.

A solução Design Próprio se utiliza da linguagem de extensão de TAG HTML's do framework 2BuyNet para criar a interface. Por isso, toda a persistência dos dados necessários para criar a loja já está contida de uma certa forma nos TAG estendidos dentro de cada arquivo de geração da loja. Não é necessário então nenhum representação no banco de dados para guardar valores ou parâmetros quando se adota essa solução para

customização da interface da loja (a não ser, um indicação de que a solução escolhida é a de Design Próprio[16])

Ao contrário, na solução Template, toda identidade visual da loja já está definida a menos de alguns parâmetros que são, propositadamente, deixados em aberto para não restringir tanto o template. Esses parâmetros em aberto necessitam de uma reflexão persistente no banco de dados do framework 2BuyNet. Os valores guardados nesse banco de dados são sempre valores de elementos de interface[17] que devem ser especificados por algum especialista em Design no momento da geração da loja. A estrutura que guarda esses valores deve ser abrangente o suficiente para futuramente poder guardar qualquer valor parametrizado de qualquer Templete a ser criado. A dificuldade de montar essa estrutura é simplificada quando restringimos o tipo de interface criada como interface Web. Uma análise dos atributos dos TAG HTML são uma amostra de todos os possíveis valores que poderão ser guardados pela estrutura. Uma proposta para essa estrutura de persistência de valores parametrizados pela interface é:

Tabela: **TemplateValor**

IdTemplate	Number
Marcador	Text
Valor	Text

[16] Mas esse valor, na verdade, é guardado como Característica da loja. O que será descrito em seguida.

[17] Exemplos típicos seriam, tipo de fonte, cor da fonte, tamanho da fonte, alinhamento de um parágrafo e etc.

IdTemplate identifica a qual template um marcador pertence. Valor guarda o valor desse marcador. A aplicação 2BuyNet disponibiliza na sua interface a escolha das duas soluções[18] e dentro da solução Templates, ela permite a escolha de um Tempate.

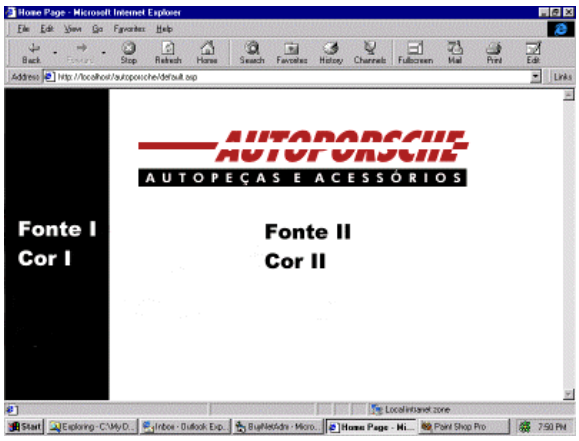


Figura 4.1-5 Template exemplo

Ao escolher o Template, note-se que há quatro parâmetros para customização relativos aos tipos e cores das fontes em duas partes distintas das páginas que formarão a loja.

Escolhidos esses valores pelo usuários, os mesmos serão guardados no banco de dados da seguinte forma:

IdTemplate	Marcador	Valor
1	TEMPI_FON	Arial

[18] Design Próprio ou Templates

1	TEMPI_COR	#000000
1	TEMPI_FON	Arial
1	TEMPI_COR	#FFFFFF

O framework 2BuyNet não prevê nenhum mecanismo simples para criação de novos Templates. Caso um usuário do framework deseje usá-lo, independentemente de seus conhecimentos, não encontrará nenhum método ou atributo para cadastrar e/ou manusear Templates. A única forma de criar novos templates é através de herança de classes e recompilação do framework. Esse trabalho fica a cargo dos desenvolvedores de framework e não é possível nenhum usuário executar tal tarefa. Funcionalidades para esse fim, estão sendo levantadas para as futuras versões do framework 2BuyNet.

Dados de Características da Loja

Os dados sobre Características da loja são informações quanto ao funcionamento da loja. Algumas decisões sobre como a loja funcionará são parametrizadas da mesma forma como os parâmetros da interface. A estrutura de dados para guardar esses valores deve ser tão abrangente e aberta quanto a dos Templates pois, da mesma forma, todos os possíveis valores ainda não foram imaginados. A estrutura muito se assemelha com a dos Templates:

Tabela: **Característica**

Caracteristica	Text
Valor	Memo

Seus possíveis valores estão diretamente relacionados às funcionalidades das lojas que o framework 2BuyNet suporta. Um exemplo seria o pagamento off-line para cartões de crédito. Essa informação fica armazenada como:

Caracteristica	Valor
CARTAO_NU	2
CARTAO1	VISA
CARTAO2	AMEX

Informa-se, assim, que a loja não só aceita cartões de crédito, mas que eles são apenas do tipo American Express e Visa.

Uma outra funcionalidade da loja é o envio de emails como resposta à chegada de algum pedido. Os endereços das pessoas que devem receber esses emails são também características da loja que são guardados da seguinte forma:

Caracteristica	Valor
PEDIDO_AVISO_EMAIL_TO	2
PEDIDO_AVISO_EMAIL_TO1	Luidi@les.inf.puc-rio.br
PEDIDO_AVISO_EMAIL_TO2	Luidi@domain.com.br

Da mesma forma, a solução da forma de customização da Interface fica armazenada como o número do Template que foi escolhido:

Caracteristica	Valor
Template	1

De forma mais complexa que apenas herança de algumas classes, a implementação de funcionalidades na loja gerada, também não é cadastrada ou manipulada por nenhum método público do framework 2BuyNet. Essas funcionalidades estão intimamente ligadas à geração de código da loja que pode se dar de diferentes maneiras. Por exemplo, uma próxima funcionalidade da uma loja gerada com o framework 2BuyNet poderá ser compatível com a linguagem de descrição de conteúdo, o XML. Tal funcionalidade é simples de ser representada. Apenas uma pergunta ao usuário do sistema sobre se ele deseja ou não que sua loja tenha tais características. Mas as ações que devem ser tomadas para que tais características sejam realizadas são de difícil parametrização, dada as variadas possibilidades. Portanto, acrescer o framework 2BuyNet com métodos para criar novas funcionalidades e posteriormente cadastrar suas ações, não faz parte dos planos de generalização do framework. Novas funcionalidades sempre estarão ligadas ao centro (core) do framework.

Concluindo, esses são os cinco tipos básicos de dados que são colhidos pela aplicação 2BuyNet. Pelo fato da aplicação 2BuyNet ser mais restritiva do que o framework 2BuyNet, alguns outros parâmetros do framework não são colhidos através da interface da aplicação. Eles são apenas inseridos como valores constantes e não há como o usuário da aplicação 2BuyNet alterá-los.

Esses atributos estão representados no banco de dados e da mesma forma que os demais serão consultados posteriormente em fases diferentes da geração da loja pelo framework 2BuyNet.

4.2 Gerador

O framework 2BuyNet após concluída a fase da coleta dos dados, tem a possibilidade de gerar a loja virtual. Entende-se por gerar, realmente construir todas as páginas que formam a loja virtual. Esse processo pode ocorrer de várias formas, conforme descrito no segundo capítulo dessa dissertação. Mas o objetivo nesse momento é detalhar o funcionamento do modelo adotado pelo framework 2BuyNet.

O processo de geração de um loja diz respeito justamente à leitura dos dados relativos à loja numa base de dados e construção dessas páginas. No caso do framework do 2BuyNet, a interface da loja gerada, é escrita em páginas HTML, pois o objetivo é gerar uma loja para a Internet. Num processo tradicional de geração de código, a geração do código da interface é apenas uma das etapas. Regra de negócios da aplicação e código para geração da camada de persistência também devem ser gerados. No caso das lojas virtuais, existem algumas simplificações que devem ser descritas.

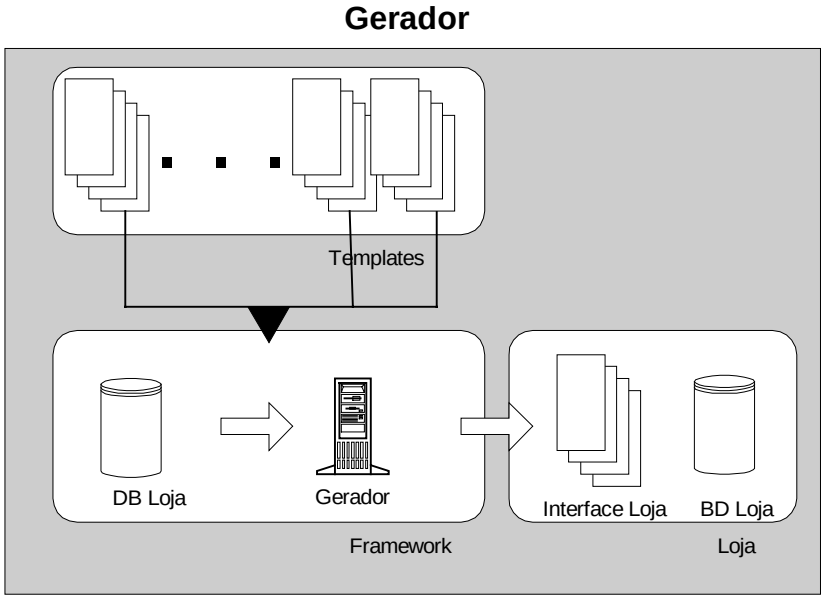


Figura 4.2-1 Diagrama do Gerador

Como no próprio framework 2BuyNet, o design das lojas geradas também seguem a forma de aplicações em três camadas. O processo de geração atua de certa forma nas três camadas, mas de formas diferentes em cada uma delas.

Para a camada da interface, o problema diz respeito principalmente às páginas que se repetem na estrutura hierárquica de uma loja, essas páginas são sempre as dos departamentos e sub-departamentos e as dos produtos. No framework 2BuyNet essas páginas são geradas repetidas vezes, uma para cada valor diferente que essas entidades podem assumir. Dessa forma, um loja com dez produtos dividida em dois departamentos terá no mínimo 12 páginas. Existe um outro aspecto do problema, que diz respeito à customização dessa interface. Detalhes dessas soluções serão apresentadas mais adiante.

Na camada da aplicação (ou regra de negócios), praticamente não existe geração de código. Essa camada é responsável por todas as funcionalidades de uma loja, como por exemplo, manipular carrinho de compras, cálculos de custo de frete, envio de emails e etc. Essa camada, diferentemente da camada de interface, é compartilhada (de certa forma) por todas as lojas. Um design verticalizado para cada loja, pois cada loja forma uma aplicação com três camadas. Mas, ao mesmo tempo, um design horizontal, pois todas as lojas estão montadas sobre uma mesma camada de aplicação.

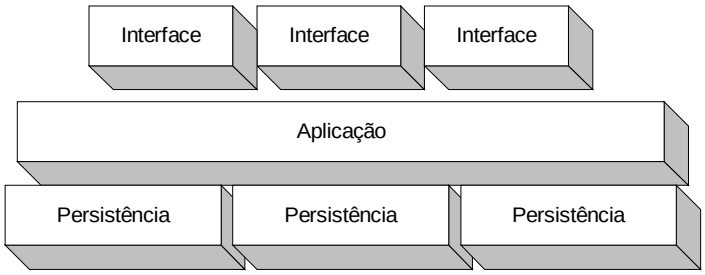


Figura 4.2-2 diagrama das partes comuns e independentes

Dessa forma, não existe geração de código na camada de aplicação. Para que diferentes lojas possam ter regras de negócios diferentes[19], a camada de aplicação consulta a camada de persistência em busca de alguns parâmetros. Valores diferentes para esses parâmetros definem comportamentos diferentes para as lojas. Esse modelo é mais rígido quanto às regras de negócios. As regras já estão pré-estabelecidas e resta ao lojista ajustar alguns parâmetros dessas regras. Um processo de geração de código que atacasse a camada de aplicação e a gerasse de acordo com as necessidades de cada loja seria muito mais abrangente. Por outro lado, deixar a cargo do usuário do framework a definição das regras de negócios, mesmo que numa linguagem de domínio de comércio eletrônico, poderia aumentar muito a complexidade do uso do framework.

Para a camada de persistência, ou seja, todo o controle e armazenamento dos dados de uma loja , foi adotada uma solução híbrida com base nas duas soluções das camadas anteriores. A solução também não permite definições que alterem o código gerado (como

[19] Por exemplo, uma loja deseja enviar um email para o dono da loja toda vez é feito um pedido. Já uma segunda loja deseja mandar um outro email para a empresa de entregas, requisitando a eles que busquem uma mercadoria.

na camada de aplicação), mas existe uma camada para cada loja (como na camada de interface). O fato de não haver a possibilidade de alterar o código gerado traz uma simplificação para processo de geração de código. A simplificação consiste no fato de a estrutura de banco de dados gerado ser constante, independentemente de qual tipo de loja está sendo gerada. Note-se então que há geração de código, toda a camada é gerada para cada loja, mas apenas essa geração é sempre a mesma.

Conforme descrito no capítulo anterior, foi criado um modelo de meta-banco de dados que consegue ser flexível o bastante para representar qualquer tipo de loja, dentro de alguns limites. Sendo assim, o código gerado para criar a camada de persistência permanece sempre o mesmo, simplificando o processo de geração de código. Por exemplo, com a mesma estrutura de banco de dados é possível criar uma loja que venda tanto para artigos de vestuário, quanto para venda de Cd-Rom infantil.

Uma segunda solução[20], seria a regeração da camada de persistência cada vez que houvesse uma mudança na loja. Isso traria vantagens no momento da modelagem do banco de dados, mas por outro lado, traria complicações para o processo de geração de código.

4.2.1 Camada de interface

[20] Mas não a adotada pelo framework 2BuyNet.

As demais camadas, por escolha de design, têm um processo de geração de código bem simples. A camada de interface permite um grau de customização no processo de geração de código que resulta em código gerado totalmente diferente para lojas distintas.

Esse processo de customização é implementado na forma de uma linguagem de programação, uma linguagem simples, derivada do HTML. O gerador de código, para a camada de interface, implementa um compilador simples, com duas etapas; um analisador léxico e um tradutor.

O objetivo da linguagem é estender os TAG's do HTML trazendo comandos relativos ao domínio específico, neste caso, a montagem de uma loja para Web. Os comandos[21] basicamente são:

- Indicam onde informações sobre a loja devem aparecer
- Especificam a navegação
- Executam tarefas da loja

4.2.2 Processo de geração da interface

[21] A descrição completa dos comandos da loja estão no Apêndice A – Linguagem de extensão da Interface

Essa linguagem estendida do HTML é a base do mecanismo dos templates. Conforme foi dito anteriormente, o framework 2BuyNet adota duas soluções para a questão da customização da interface. O usuário do framework pode escolher uma identidade visual pronta que juntamente com sua base de dados, gerará a loja. Ele mesmo define como serão suas páginas, utilizando essa linguagem de extensão.

Tecnicamente essas duas soluções são idênticas. Um template nada mais é que um design feito por alguém e disponibilizado no framework para uso geral, de páginas desenhadas com os comandos estendidos do HTML. Para o gerador não importa se a solução escolhida foi um template ou se é um design próprio. Ambos são tratados da mesma forma.

4.2.3 Problema do Idioma

A funcionalidade de múltiplos idiomas reflete no design da máquina geradora. Da mesma forma como no problema de como gerar a loja[22], existem duas formas de resolver no momento da geração a montagem da loja. Normalmente um site representado em vários idiomas contém algum mecanismo para escolha do idioma desejado. Muitas vezes, na página principal uma barra indica os idiomas disponíveis. A solução mais simples é a replicação do site em várias línguas (solução estática). A escolha de um idioma simplesmente desvia da página principal para a página principal do idioma escolhido. A

[22] Solução estática versus solução Dinâmica

partir dessa página raiz, todo o site está escrito nesse idioma. Essa solução privilegia as máquinas de busca e gera menos acessos a banco de dados, tornando-se mais rápida.

A solução dinâmica supõe que todos os textos ou figuras (passíveis de tradução) não são escritas na página HTML. Existe uma chamada de função que é executada quando a página é carregada. Muitas linguagens implementam essa integração com páginas Web, CGILua [31 32], ASP [30], JavaScript [29] Server Side são algumas delas.

```
<font face=”Arial” size=”2” color=”00FF00”>
        <% getText(“ID_TEXTO”,IdLingua) %>
</font>
```

Esse exemplo demonstra como poderia ser uma dessas chamadas. Note que na função getText existem dois parâmetros, o primeiro indica qual texto queremos resgatar, o segunda em qual idioma esse texto estará. A escolha do idioma, diferentemente da versão estática, não desvia a página principal, mas altera o valor de memória da variável IdLingua. Na versão dinâmica não existe duplicidade do site. Isso resulta em um site mais enxuto, mas pelo fato de cada texto representar um acesso a um servidor de banco de dados, isto significa um site mais lento com muita carga para a máquina servidora.

Cada uma delas têm implicações diferentes para o gerador. Um framework gerador de lojas virtuais poderia até implementar as duas formas. No caso específico do framework 2BuyNet, a forma escolhida foi a estática e não há métodos que representem a forma dinâmica.

4.2.4 Estrutura gerada

O número de páginas e a estrutura de navegação entre as páginas é definida durante o processo de geração. O framework 2BuyNet não admite uma customização da estrutura navegacional. Quais páginas levam a quais páginas e o número total de páginas é sempre fixo, dentro de um certo setor do site. O site de uma loja pode ter sua estrutura dividida em duas partes: Processo de Compra (estrutura fixa) e Catálogo de Compras (estrutura variável). O framework 2BuyNet entende essas duas estruturas distintas. O catálogo de compras é função direta da estrutura de departamentos/sub-departamentos e produtos portanto, variável. O processo de compras é fixo e foi definido com base nos estudo relacionado dos demais softwares e em seus respectivos processos de compra. Em “Electronic Shopping” [2], Gerald Lohse faz uma pesquisa sobre várias características de lojas descrevendo a importância da estrutura do processo de compras. Problemas associados a excesso de perguntas e confirmações, falta de opções e informações podem desestimular o impulso de compra do cliente. O processo de compras das lojas geradas pelo framework 2BuyNet é um dos mais completos e curtos dentre os analisados.

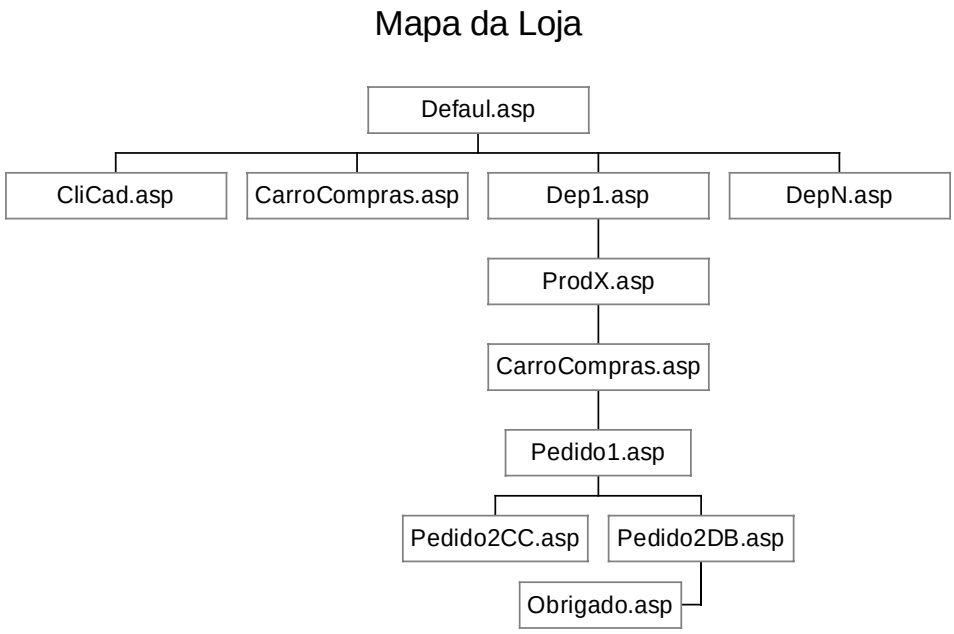


Figura 4.2-3 Estrutura de páginas de uma loja

Note-se pela estrutura que é possível comprar, escolher, comprar e pagar um produto em quatro click's de mouse, ou passando por apenas 4 telas. Dependendo do compromisso com a clareza da informação, o lojista pode comprimir[23] ainda mais esse processo, passando-os para apenas três click's ou páginas.

4.3 Background das lojas

[23] A página carrinho de compras pode se fundir com a de coleta de dados pessoais.

Toda loja é formada por uma coleção de páginas HTML, essa coleção varia em função de fatores tais como: os idiomas, as funcionalidades da loja e etc. De alguma forma, essas páginas HTML precisam implementar uma estrutura de dados e ações para se tornar uma loja. Para o framework 2BuyNet, dentro do seu design de aplicação com três camadas, essa necessidade é representada pela presença da camada de aplicação (ou regra de negócios) no design da loja.

Cada loja é, por si só, uma aplicação de três camadas, sendo a segunda camada a responsável por todas as ações e funcionalidades da loja. Conforme foi descrito no capítulo do gerador, essa segunda camada não é gerada em função de cada loja. O design é feito de forma que a implementação seja a mesma para qualquer loja. Através de parâmetros, cada loja consigue resultados diferentes.

No caso do framework 2BuyNet, essa camada é implementada através de uma DLL, que não é transmitida através da Internet para a máquina cliente, mas sim, instanciada no servidor quando uma loja é carregada. A própria arquitetura das DLL se encarrega de instanciar uma DLL para cada loja. Portanto, ao implementar a DLL não foi necessário prever estruturas de dados e métodos para dar suporte a múltiplas lojas. Cada DLL representa apenas uma única loja.

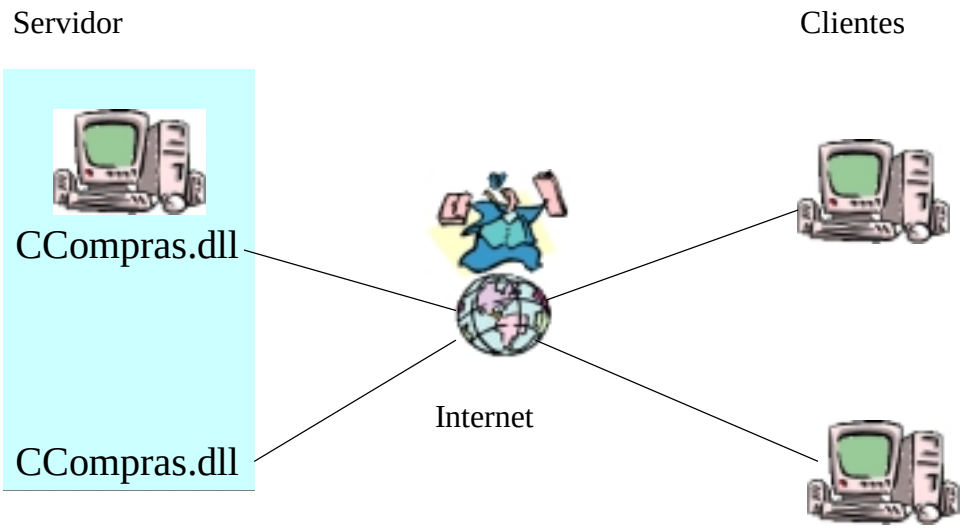


Figura 4.3-1 Esquema de acesso dos clientes ao servidor

As entidades abstratas tratadas, por essa camada, são:

- Carrinho de compras
- Clientes
- Comunicação

4.3.1 Carrinho de compras

O nome carrinho de compras é uma metáfora baseada nos carrinhos de compras dos supermercados. A idéia é implementar uma classe para armazenar itens comprados. O usuário navega pela loja escolhendo os produtos desejados. Toda vez que desejar comprar

um produto, basta adicioná-lo ao carrinho de compras. No final do processo existe um mecanismo para que o usuário veja e pague as mercadorias que estão no seus carrinho.

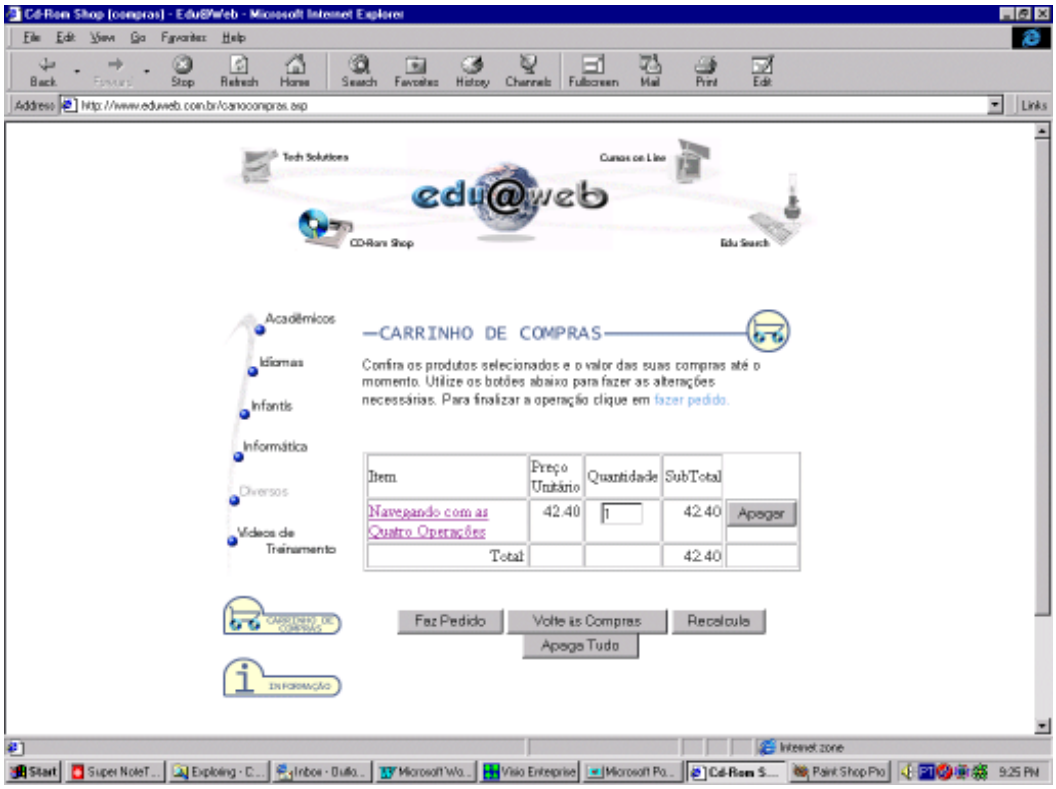


Figura 4.3-2 Carrinho de compras

4.3.2 Clientes

Toda a loja gerada pelo framework 2BuyNet tem a possibilidade de implementar um mecanismo para identificar clientes que retornaram para sua loja. Esse mecanismo funciona através de um sistema de cookies, que marca o browser de uma pessoa. Tempos depois, toda vez que um browser marcado faz uma requisição à uma loja, a loja verifica a marca e recupera toda a informação associada àquela marca. Essas informações podem ser

de qualquer natureza, mas nas lojas do framework 2BuyNet, elas são relativas aos dados de entrega dos produtos comprados, por exemplo: endereço, cep, etc.

A entidade Clientes é responsável por toda a implementação desses recursos. Inserção e alteração das informações de um cliente, recuperação desses dados caso já existam, criação do cookie identificador de clientes e certificação de uma nova visita à loja. Na modelagem do framework 2BuyNet, todas as informações geradas pelo cliente dentro de uma loja são guardadas. Dessa forma, o lojista ganha uma poderosa fonte de informação para traçar o perfil de compra de seus clientes. Todas essas ações também ficam a cargo da entidade Clientes.

4.3.3 Comunicação

Uma funcionalidade muito importante de uma loja na Web é a comunicação. No framework 2BuyNet, as lojas geradas se comunicam através de email. A entidade Comunicação foi modelada de forma a prover métodos para enviar email para as pessoas que fazem parte do processo. Normalmente, no final de um processo de compra, a loja envia um email ao cliente, confirmando a compra. Com os métodos disponibilizados para a loja, é possível mandar emails para qualquer pessoa, como por exemplo, o próprio lojista ou a empresa responsável pela coleta e entrega dos produtos.

As informações necessárias para envio como o endereço, o conteúdo e até mesmo o destinatário, são exemplos de como funciona a parametrização da segunda camada. A

camada disponibiliza apenas o método sendEmail, todos os parâmetro o seu uso, são resgatados do banco de dados através do método getCaracteristica. Dessa forma, com alguns parâmetros fixos, como QUANT_EMAIL, que informa quantos emails devem ser enviados, é possível iterar e resgatar todos os demais valores da base de dados. Esse mecanismo é muito utilizado para resolver os problemas da segunda camada. Outros problemas resolvidos com essa técnica são: cartões de crédito aceitos por uma determinada loja, opções de pagamento de uma loja, formas de entrega que a loja disponibiliza para o cliente, entre outros.

Essas três entidades basicamente descrevem com que elementos a loja interfere. A loja não interfere em produtos, apenas lê suas características no momento em que um cliente o adiciona ao carrinho de compras.

A estrutura completa da loja, incluindo todas as páginas que não têm interface mas que executam comandos e se encaixam na segunda camada são:

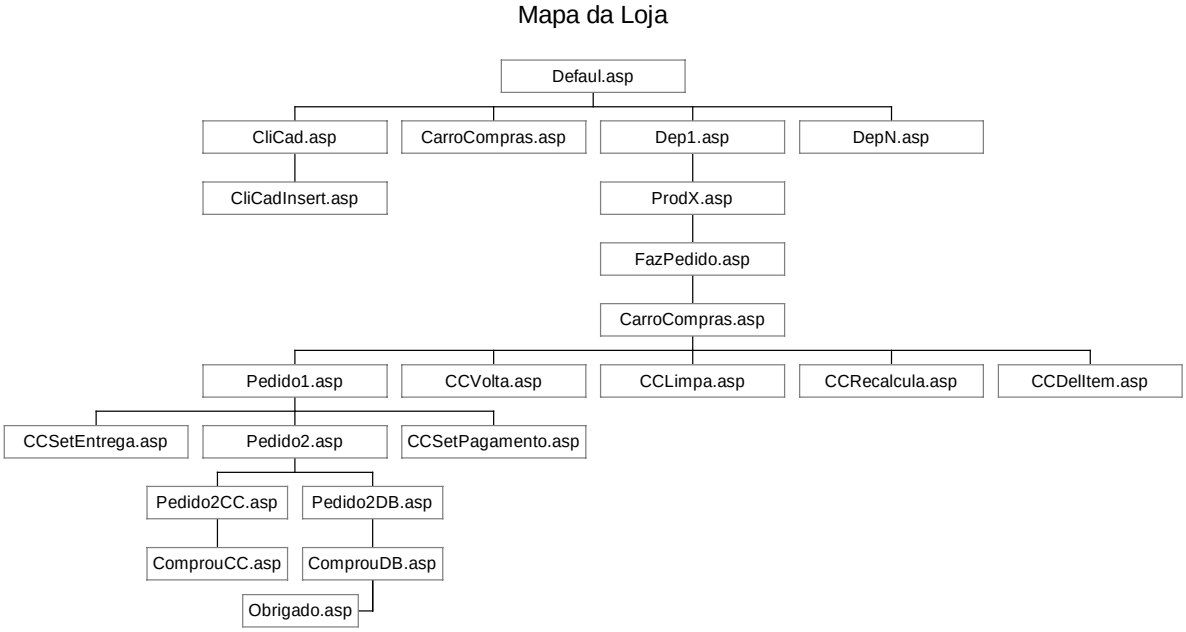


Figura 4.3-3Estrutura da Loja

Por exemplo, note-se o momento em que um cliente adiciona um produto ao carrinho de compras. O cliente está numa página de produto e aperta o botão comprar. Esse botão aciona o *action* de um *form*, que dispara um *submit*. O *action* desse *form* está ligado à página FazPedido.asp, essa página não contém código HTML, ela é simplesmente uma série de comandos e um redirecionamento para a página que mostra o carrinho de compras. A página do carrinho de compras, por sua vez, não sabe quem a chamou. Ela simplesmente desenha o estado atual do carrinho de compras.

4.4 Importação/exportação de dados

O framework 2BuyNet manipula uma série de bancos de dados. Naturalmente o framework contém método para inserir, alterar e apagar elementos de todas as suas tabelas.

Esses métodos seriam suficientes na maioria das aplicações, mas por exemplo, na aplicação 2BuyNet[24], existe um grave problema no momento das atualizações. Por ser uma aplicação para a Web, sua interface roda num cliente enquanto todo o processamento e a base de dados estão no servidor. Atualizações no banco de dados são uma das ações mais freqüentes nesse tipo de aplicação e o tamanho das tabelas costuma ser muito grande nas aplicação de comércio eletrônico. Portanto, a tarefa de atualização das tabelas através de um mecanismo remoto e on-line poderia se tornar um transtorno ou até mesmo ser inviável.

Para isso, o framework 2BuyNet disponibiliza outros métodos para atualização nas tabelas da base de dados. O mecanismo funciona a partir de planilhas eletrônicas. Existem métodos que geram e outros que interpretam o conteúdo das planilhas e atualizam ou descarregam as tabelas. Dessa forma é possível ler e escrever nas tabelas de uma só vez.

[24] Aplicação para Web desenvolvida a partir do framework 2BuyNet

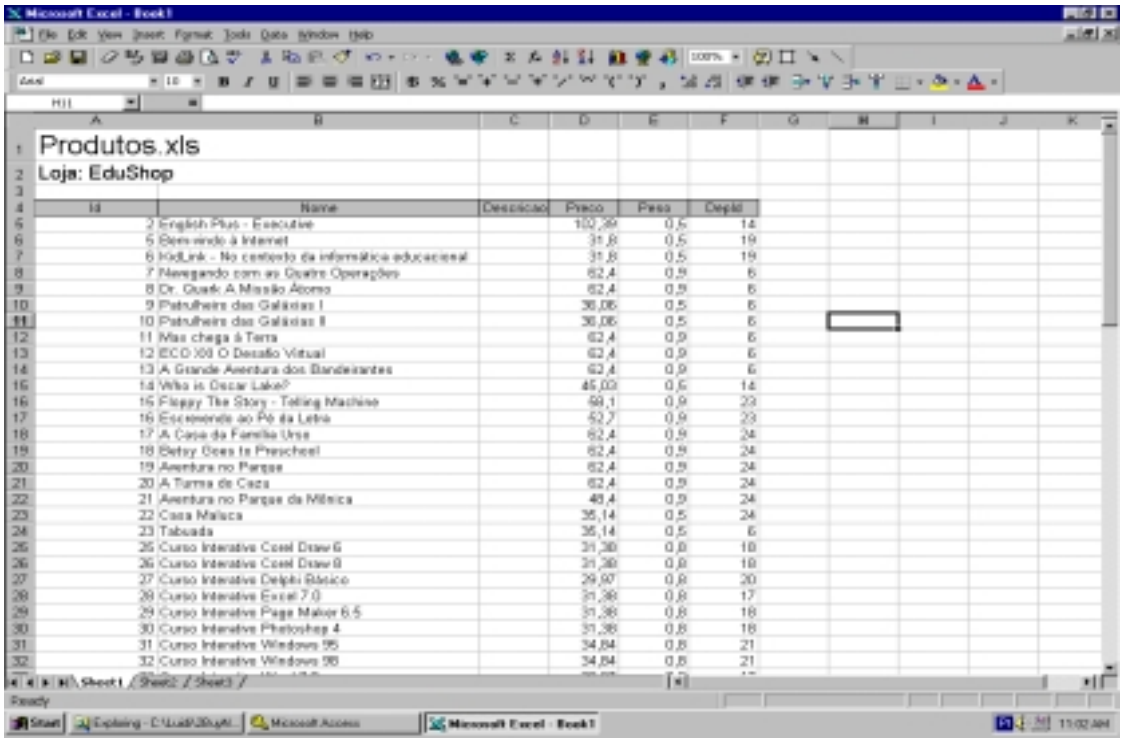


Figura 4.4-1 Exemplo Planilha de Produtos

Casada com essa funcionalidade, existem ainda os métodos para fazer upload e download de um arquivo. Combinando as duas soluções, a aplicação 2BuyNet pode, por exemplo, disponibilizar opções onde o lojista pode fazer um upload de toda a tabela de produtos da sua loja no mundo real, e automaticamente, esse produtos passam a valer na sua loja na Internet. Esse conceito se estende para todas as tabelas, como por exemplo para a tabela de departamentos, características do produto, pedidos e clientes.

4.4.1 Síncrono e assíncrono

A criação desse mecanismo de atualização, trouxe duas formas diferentes de como operar a funcionalidade. Por exemplo, no caso de um download. Digamos que um lojista deseje fazer um download de uma tabela de departamentos. A aplicação 2BuyNet fornece uma opção denominada Donwload Planilha Departamento. Essa opção deve construir a planilha, de forma que o lojista possa fazer o seu download. As duas formas possíveis de executar isso diferem nos momentos em que ocorrem o pedido e a geração real da planilha.

Donwload Síncrono

No download síncrono, o lojista pede a geração da planilha, nesse momento a aplicação fica suspensa enquanto a geração não termina. Após, um link ou um email se encarregam de prover uma forma do lojista resgatar essa planilha de servidor.

O problema dessa solução, na aplicação 2BuyNet, é o tempo de espera da geração da planilha. Um tempo demasiadamente longo pode causar um timeout no browser cliente. Caso isso ocorra, mesmo que a planilha acabe sendo gerada com sucesso, o browser cliente não aceita nenhum envio de dados do servidor, ficando sua tela em branco ao invés de uma mensagem de sucesso contendo um link para a planilha. O controle desse tempo é impossível, ele depende da vários fatores, carga no servidor no momento do pedido, tamanho da planilha e etc.

Download Assíncrono

No download assíncrono, ao fazer um pedido de geração de planilha, o lojista recebe uma mensagem informando que assim que possível sua planilha será gerada. Para esse caso, o pedido é inserido numa fila de pedido, que é atendida de acordo com as

políticas do gerador de planilhas. Tão logo a planilha seja gerada, através de um mecanismo de aviso qualquer, o lojista é comunicado. Esse aviso normalmente é um email que já pode conter a planilha anexada ou um link com o endereço da planilha para um futuro download.

No framework 2BuyNet, o método responsável pela leitura da planilha e atualização de banco de dados não supõe diferenciais, ou seja, a planilha deve conter todo as informações do banco de dados, independentemente se um registro específico tenha mudado ou não. Ao receber uma planilha, o framework 2BuyNet atualiza todo o banco de dados.

5. Conclusão e Trabalhos Futuros

A área de comércio eletrônico continua em evolução e muito se experimenta para descobrir novas formas de comércio. O framework 2BuyNet é uma dessas novas formas. Sua preocupação principal é quanto à redução do custo de desenvolvimento de uma loja, trazendo a possibilidade de se construir uma aplicação customizada a um preço competitivo.

A aplicação 2BuyNet abrange o uso do framework 2BuyNet agregando serviço a ele. Existem soluções para a logística de entrega dos produtos, marketing da loja e criação de interface, entre outros. Adotamos a estratégia da aplicação como serviço, pois traz simplificações no processo de desenvolvimento e manutenção de software. Como descrito, as funcionalidades adotadas seguem conformidade com os softwares do mesmo domínio, muitas vezes combinando solução de vários deles.

Atualmente, tanto o framework quanto a aplicação 2BuyNet já estão implementados. A aplicação pode ser utilizada no endereço http://www.2buynet.com.br. O framework, por sua vez, não está disponível em nenhum endereço.

Como trabalhos futuros, existem duas categorias de estudos: as de médio prazo e as de longo prazo. Num prazo mais longo, vemos o framework 2BuyNet se estendendo e abrangendo novas formas de executar o comércio eletrônico. Os planos buscam também interligar projetos distintos do Laboratório de Engenharia de Software do Departamento de

Informática da Puc-Rio, projetos como o Virtual Market dos alunos Pedro Santos Ripper e Ayrton Maia Neto [12, 13], acompanhar a evolução do projeto do aluno Clécio Guaranys sobre SmartCards visando acrescentar essa funcionalidade ao framework de forma a adicionar valor ao framework e/ou à aplicação 2BuyNet.

A médio prazo, vemos a continuidade e acompanhamento do framework de acordo com a evolução do mercado e das ferramentas concorrentes. Exemplos imediatos de funcionalidades que estão em teste para em breve serem adicionados ao framework 2BuyNet, são a geração de lojas com produtos descritos em XML [14] e classes derivadas para implementação de novas forma de pagamento como SET, entre outros.

Uma tendência também importante a médio prazo é a ampliação da banda passante das conexões na Internet. Em um ano, certamente as taxas de transferência estarão em outro patamar, possibilitando funcionalidades para as lojas que hoje não são viáveis. Seguramente, novas formas de comércio eletrônico serão inventadas.

Quanto ao framework 2BuyNet e à instância da aplicação 2BuyNet, um estudo sobre outras formas de instanciar o framework, gerando diferentes aplicações sobre comércio eletrônico. Como descrito no capítulo 3.1, estudos e análises em concordância com a terceira fase de um framework[25]. De acordo com os resultados obtidos pela aplicação 2BuyNet e com a necessidade do mercado em soluções para comércio eletrônico, reestrutura-se as abstrações do framework 2BuyNet.

[25] Evolução e Manutenção do framework

6. Apêndice A – Linguagem de extensão da Interface

A máquina geradora do framework 2BuyNet contém as característica básicas de um compilador. A linguagem, seguindo o modelo do HTML, contém TAG's específicos com o prefixo 2BN_ para identificar o 2BuyNet.

Existem 6 categorias de TAG's:

Variáveis on line do Cliente

São informações que são ou foram fornecidas pelo cliente. O formato de declaração dessas variáveis é o único que não segue a forma casada de TAG's, mas por características construtivas, a própria forma da linguagem asp.

Lista da variáveis:

Session("entregaNome")	Nome do cliente no formulário de dados pessoais
Session("entregaEnd1")	Primeira parte do endereço do cliente
Session("entregaEnd2")	Segunda parte do endereço, complemento
Session("entregaCidade")	Cidade
Session("entregaEstado")	Estado
Session("entregaCep")	Cep
Session("entregaPais")	País
Session("entregaTel")	Telefone

Session("email") email

Session("entregaFax") Fax

Essas informações podem ser colhidas em dois momentos distintos. Caso o cliente já tenha um cadastro na loja, ao entrar no site, cada uma dessas variáveis estará com seus valores “setados” em função de um mecanismo de *cookies*, que identifica o retorno de um usuário cadastrado e resgata suas informações pessoais (informações para entrega do produto) do banco de dados. Caso seja a primeira vez que o cliente esteja entrando numa loja, ou caso ele tenha optado por não se cadastrar, essas informações serão obrigatoriamente perguntadas na página de coleta de dados pessoais no processo de compras. A partir desse momento, as variáveis passam a conter seus respectivos valores.

Exemplo:

Código fonte

Bem-vindo à Papel Virtual <% Session("entregaNome") %>

No browser

Bem-vindo à Papel Virtual Luidi Fortunato

TAGs Técnicos

Esse tag’s são chamados técnicos, pois são necessários apenas para operacionalizar partes técnicas do funcionamento do site. Eles não têm características que os associe de alguma forma ao domínio de lojas Web.

<2BN_ASP_IDLOJA> </2BN_ASP_IDLOJA>

Nome único de identifica cada loja.

<2BN_ASP_DSN> </2BN_ASP_DSN>

Nome do Data Source Name que identifica a base de dados da loja.

<2BN_ASP_DSNUID> </2BN_ASP_DSNUID>

Login da loja no servidor de banco de dados

<2BN_ASP_DSNPWD> </2BN_ASP_DSNPWD>

Senha da loja no servidor de banco de dados

<2BN_ASP_PGDEFAULT> </2BN_ASP_PGDEFAULT>

Nome da página principal. Normalmente default.asp, mas esse valor pode ser alterado para o caso de lojas que existem dentro de um site maior. Esse é o único tag dessa categoria que sofre influência do idioma. Dependendo do idioma do site gerado, o nome da página default varia.

Exemplo:

Código fonte

<a href=<2BN_ASP_PGDEFAULT>default.asp</2BN_ASP_PGDEFAULT>>

Entrada

No browser

Entrada

(link para a página principal do site)

TAGs Diretos

Os tags diretos são informações simples, parametrizadas na base de dados (por causa das versões em vários idiomas). Tanto os textos como as referências para as imagens são resgatadas com o mesmo mecanismo.

<2BN_ITEM ID='xxxx'> </2BN_ITEM>

Resgata do banco de dados textos. O parâmetro Id, identifica qual texto estará sendo resgatado. Os templates têm áreas reservadas onde aparecerão os textos escritos pelo lojista, por exemplo, texto de boas-vindas.

onde xxxx:

TXT_INFORMACOES	Texto central da página de informações
TXT_FECHADO	Texto central da página quando a loja está fechada
TXT_RODAPE	Texto de rodapé, aparece em todas as páginas
TXT_BOASVINDAS	Texto central da página principal

TXT_CADASTROA Texto do link quando um cliente foi identificado

TXT_CADASTRONA Texto do link quando um cliente não foi identificado

TXT_LINKCADASTRONA Endereço do link quando um cliente não foi identificado

TXT_LINKCADASTROA Endereço do link quando um cliente foi identificado

Exemplo:

Código fonte

<2BN_ITEM ID='TXT_RODAPE'>

</2BN_ITEM>

No browser

1998 ® Todos os direitos reservados. Papiro Virtual.

Comentários/Dúvidas:

papiro@papiroeditora.com.br

<2BN_IMG ID='xxxx'> </2BN_IMG>

Da mesma forma como tag de textos, esse tag resgata o nome de uma figura do banco de dados.

Onde xxxx, representa o nome completo do arquivo, inclusive com a extensão. Ex. Logo.gif

Exemplo:

Código fonte

<img src=

<2BN_IMG ID='logo.gif'></2BN_IMG>

border="0" hspace="0" vspace="0" >

No browser

<img src='logo.gif' border="0" hspace="0" vspace="0" >

<2BN_FPG_CCTIPO> </2BN_FPG_CCTIPO>

Desenha uma lista com todas as opções de cartões de crédito aceitos. Essa informação também depende de idioma e por isso não pode ser escrita direta na página HTML.

Exemplo:

Código fonte

```
<select name="cmbTipo" size="1">
<2BN_FPG_CCTIPO>
</2BN_FPG_CCTIPO>
</select>
```

No browser

```
<select name="cmbTipo" size="1">
<option value="VISA">VISA</option>
<option value="AMEX">AMEX</option>
</select>
```

TAG parâmetro de Template

A funcionalidade de templates pré-definidos, como foi descrito, permite algum tipo de customização. Mesmo num template pronto, é possível variar alguns parâmetros. Esses parâmetros, escolhidos pelo lojista, são armazenados na base de dados, numa tabela especial só para descrever os parâmetros e seus valores. No processo de geração é preciso que exista uma TAG que de certa forma resgate esse parâmetro armazenado no banco de dados e o escreve nas páginas HTML. Esses são os parâmetros de Template.

```
<2BN_TEMPLATE ID=’xxxx’>      </2BN_TEMPLATE>
```

Exemplo:

Código fonte

<2BN_TEMPLATE ID='Temp1'>

<2BN_ITEM ID='TXT_RODAPE'>

</2BN_ITEM>

</2BN_TEMPLATE>

No browser

<FONT FACE="VERDANA" SIZE="2" COLOR="#000000">

1998 ® Todos os direitos reservados. Papiro Virtual.

Comentários/Dúvidas:

papiro@papiroeditora.com.br

</FONT>

TAG de Coleção

O tag de coleção tanto funciona para produtos, quanto para departamentos. Sua função é apenas abrir uma conexão com o banco de dados de forma a fornecer um “handle” interno que será utilizado posteriormente por outros tag’s complementares para resgatarem essas informações uma a uma.

<2BN_LSTDEP_LOOP FATHERID=’id’> </2BN_LSTDEP_LOOP>

No caso de departamentos, a forma de agrupar os dados é pelo departamento pai. Então, por exemplo, caso queiramos montar o menu principal com os departamentos raiz, temos que abrir uma lista de departamentos cujo pai é o zero.

As demais tag que trabalham complementarmente com esse tag de abertura são:

<2BN_LSTDEP ID=’xxxx’></2BN_LSTDEP>

onde xxxx,

IMGPEQ Nome da imagem num tamanho pequeno que representa aquele departamento

PAG O endereço, link

NOME Nome

DESCRI Descrição

ID Id

DEPPAIID Id do departamento Pai

E o tag somador, que disponibiliza o número total de departamentos na coleção.

<2BN_LSTDEP_TOTAL> </2BN_LSTDEP_TOTAL>

Da mesma forma, existe um equivalente para os produtos:

<2BN_LSTPROD ID=’xxx’></2BN_LSTPROD>

Sendo que a chave agregadora, dessa vez, é o Id do departamento onde vários produtos estão inseridos.

Após agrupados os produtos, os valores podem ser resgatados pelos tags:

<2BN_LSTPROD ID=’xxxx’></2BN_LSTPROD>

onde xxxx,

IMGPEQ	Nome da versão pequena da imagem do produto
IMGNOR	Nome da versão normal da imagem do produto
IMGGRD	Nome da versão grande da imagem do produto
PAG	Nome da página do produto
NOME	Nome do produto
DESCRI	Descrição
ID	Id
PRECO	Preço
DEPID	Id do departamento do produto
PESO	Peso

E o tag totalizador:

<2BN_LSTPROD_TOTAL></2BN_LSTPROD_TOTAL>

Que retorna o número de produtos naquela coleção.

TAG de contexto

Existem duas páginas pré-definidas que durante o processo de geração são multiplicadas. São as páginas de departamentos e as páginas de produtos. Pelo fato de ser uma página que se replicará, existem basicamente dois tipos de informação que serão resgatadas do banco de dados. As informações que são buscadas, mas não variam nas páginas replicadas e as que variam. A replicação é um procedimento recursivo que gera

todas as páginas necessárias tanto para produto quanto para departamentos. As informações que não precisam variar em função do contexto, são resolvidas com qualquer um dos tags que foram apresentados anteriormente. Já as que dependem do contexto, e só são referentes a produtos e departamentos, são representadas pelos tags de contexto:

Para departamentos

<2BN_DEP ID='xxx'></2BN_DEP>

onde xxxx,

PAG	link para página
NOME	Nome
DESCRI	Descrição
ID	Id
IDFATHER	Id do departamento do pai
IMGPEQ	Nome da imagem pequena

E para produtos:

<2BN_PROD ID='xxxx'></2BN_PROD>

onde xxxxx,

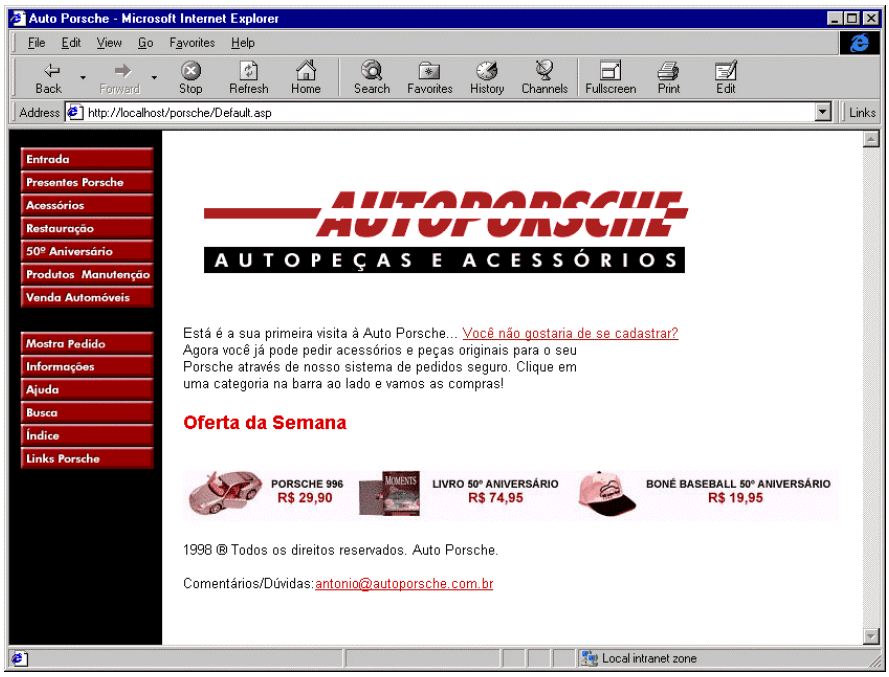
NOME	Nome do produto
DESCRI	Descrição

PAG	link para página
PRECO	Preço do produto
DEPID	Id do departamento do produto
PESO	Peso do produto
IMGPEQ	Nome da imagem pequena do produto
IMGNOR	Nome da imagem
IMGGRD	

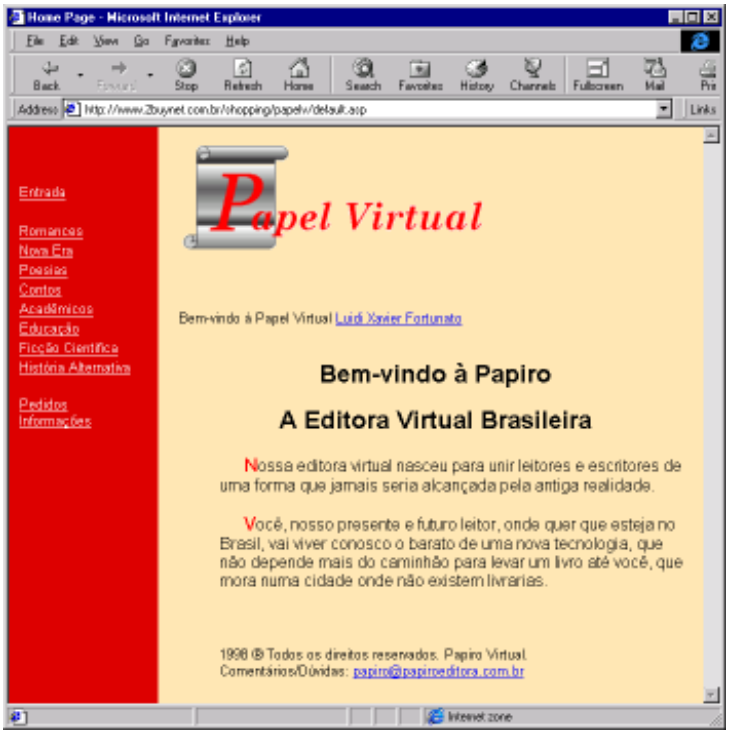
Exemplos de algumas telas escritas com os TAGs da linguagem de extensão de interface

Loja EduWeb de Cd-Roms de educacionais para crianças. Site:

http://www.eduweb.com.br/shop.asp



Loja exemplo do 2BuyNet. Autoporsche, não existe no mundo real.



Loja da Papel Virtual (ex. Papiro Editora). Venda de livros de autores independentes. Site:

http://www.2buynet.com.br/papelv

7. Referências

1. C. Lucena, H. Fuks, R. Milidiu, L. Macedo, N. Santos, C. Laufer, M. Ribeiro, M. Fontoura, R. Noya, S. Crespo, V. Torres, L. Daflon, and L. Lukowiecki (1998), “AulaNet™ - An Environment for the Development and Maintenance of Courses on the Web”, In *Proceedings of the International Conference on Engineering Education 1998,* Rio de Janeiro, Brazil.

2. Gerald L. Lohse e Peter Spiller, 1998 "Electronic Shopping - Designing online stores with effective customer interfaces has a critical influence on traffic and sales", Communications of the ACM, July 1998/Vol. 41. No. 7

3. P. Alencar, D. Cowan, S. Crespo, M. F. Fontoura, and C. J. Lucena, “Using Viewpoints to Derive a Conceptual Model for Web-Based Education Environments”, MCC17/98, Monografias em Ciência da Computação, Departamento de Informática, PUC-Rio, 1998 (also submitted to Journal of Systems and Software).

4. C. Braga, M. F. Fontoura, C. J. Lucena, and L. M. Moura, “Using Domain Specific Languages to Instantiate OO Frameworks”, MCC28/98, Monografias em Ciência da Computação, Departamento de Informática, PUC-Rio, 1998 (also submitted to IEE Proceedings – Software Engineering).

5. S. Crespo, M. F. Fontoura, C. J. Lucena, and L. M. Moura, “ALADIN: An Architecture for Learningware Applications Design and Instantiation”, MCC/98, Monografias em Ciência da Computação, Departamento de Informática, PUC-Rio, 1998 (also submitted to WWW Journal).

6. P. Hudak, “Building Domain-Specific Embedded Languages”, Computing Surveys, 28A(4), ACM, 1996.

7. M. F. Fontoura, E. H Haeusler, and C. J. Lucena, “A Framework Design and Instantiation Method”, , MCC/98, Monografias em Ciência da Computação, Departamento de Informática, PUC-Rio, 1998 (also submitted to ACM TOSEM).

8. The White House, “A Framework for Global Electronic Commerce”, July 1, 1997, http://www.ecommerce.gov/framewrk.htm

9. William J. Clinton, “Presidential Directive on Electronic Commerce”, July 1, 1997, http://www.ecommerce.gov/presiden.htm

10. Philip Kotler, “Marketing Management Analysis, Planning, Implementation and Control”, Ninth Edition, Prentice-Hall, 1997

11. Margo Komenar, “Electronic Marketing”, Wiley Computer Publishing

Evan I. Schwartz, “WEBONOMICS Nine essential principles for ”, 1997

12. Pattie Maes, Robert H. Guttman, and Alexandros G. Moukas, “Agents that Buy and Sell”, Communications of the ACM, March 1999/Vol. 42 No. 3

13. David Wong, Noemi Paciorek, and Dana Moore, “Java-based Mobile Agents”, Communications of the ACM, March 1999/Vol. 42 No. 3

14. Robert J. Glushko, Jay M. Tenenbaum, and Bart Meltzer, “An XML Framework for Agent-based E-commerce”, Communications of the ACM, March 1999/Vol. 42 No. 3

15. Donna L. Hoffman, Thomas P. Novak, and Patrali Chatterjee, “Commercial Scenarios for the Web: Opportunities and Challenges”, JCM Vol. 1, No. 3

16. Rolf T. Wigand and Robert I. Banjamim, “Electronic Commerce: Effects on Electronic Markets”, JCM Vol. 1, No. 3

17. Mitra B. Sarkar, Brian Butler, and Charles Steinfield, , “Intermediaries and Cybermediaries: A Continuing Role for Mediating Players in Electronic Marketplace”, JCM Vol. 1, No. 3

18. Charles Steinfield, Robert Kraut, and Alice Plummer, “The Impact of Electronic Commerce on Buyer-Seller Relationships”, JCM Vol. 1, No. 3

19. John Nouwens and Harry Bouwman, “Living Apart Together in Electronic Commerce: The Use of Information and Communication Technology to Create Network Organizations”, JCM Vol. 1, No. 3

20. “Empresários temem fiasco na Internet”, Jornal do Comércio, 04/02/1999

21. “Comércio eletrônico não decolou ainda”, A Tarde OnLine, 02/06/1999

22. “Windows DNA – Windows Distributed internet Applications Architecture”, Microsoft Corporation White Paper, September 1997

23. MSDN Visual Studio 6.0
Managing Session – ASPDoc Help

24. José Davi Furlan, “Modelagem de Objetos através da UML – análise e desenho orientados a objeto”, Makron Books, 1998

25. Edward Yourdon, “Projetos Virtualmente Impossíveis – Guia completo do Desenvolvedor de software para sobreviver aos projetos virtualmente impossíveis”, Makron Books, 1999

26. Michael Mattsson & Jan Bosch, “Framework Composition: Problems, Causes and Solutions”, University of Karlskrona/Ronneby

27. Michael Mattsson & Jan Bosch, “Frameworks as components: A Classification of Framework Evolution”, University of Karlskrona/Ronneby

28. Michael Mattson, “Object-Oriented Frameworks – A survey of methodological issues”, Department of Computer Science and Business Administration – University College of Karlskrona/Ronneby

29. Javascript, http://www.netscape.com

30. ASP – Active Server Page, http://www.microsoft.com

31. IERUSALIMSCHY, R., FIGUEIREDO, L. H., and CELES, W., “Lua an extensible extension language”, Software Practice and Experience, 1996, 26(6), pp 635-652.

32. IERUSALIMSCHY, R., BORGES, R., and HESTER, A. M.: “CGILua A Multi-Paradigmatic Tool for Creating Dynamic WWW Pages”, SBES’97 (Simpósio Brasileiro de Engenharia de Software), 1997.

33. Frank E. Redmond III, “Building Windows DNA Applications”, Microsoft Corporation Junho 1999. http://msdn.microsoft.com/isapi/msdnlib.idc?theURL=/library/techart/windnadesign.htm

http://www.les.inf.puc-rio.br

Laboratório de Engenharia de Software da Pontifícia Universidade Católica do Rio de Janeiro / Puc-Rio

7.1 Softwares de apoio à criação de Web Sites comerciais

http://www.2buynet.com.br

2BuyNet Puc-Rio/Edu@Web

http://www.microsoft.com/siteserver/commerce/default.asp

Microsoft – Site Server 3.0, Commerce Edition.

http://www.ablecommerce.com/

Able Solutions – Produtora do software AbleCommerce 2.3.

http://www.ez2-shop.com

INTRIX Systems Group, Inc. – Produtora do software EZ2-Shop.

http://www.viaweb.com

Yahoo! Store – Serviço on-line pelo qual um usuario instancia uma loja para Internet. Serviço todo comandado pelo browser. A solução Viaweb foi comprada no ínicio de 1998 pela Yahoo! e adotada como solução para comércio eletrônico.

http://www.compaq.com/smb/onlineservices/commerce.html

Compaq – Serviço / Produto para suporte e criação de Web Sites comerciais.

http://www.icat.com

iCat.com – Serviço On-line para criação de Web Sites comerciais.