

RESUMO

O projeto SPARTACUS trata-se de um sistema de pedido eletrônico de medicamentos e produtos farmacêuticos em geral pela internet.

O SPARTACUS é o primeiro passo para uma distribuidora de medicamentos que deseja oferecer mais uma opção, além do televendas.

O sistema apresentado não se comunica ainda com os outros sistemas existentes no ambiente da distribuidora, como o sistema de gerenciamento bancário e o sistema de gerenciamento, que realiza o controle de contas a pagar, contas a receber, controle de estoque, emissão de nota fiscal, emissão de boleto bancário, etc.

O SPARTACUS foi desenvolvido na plataforma Java, procurando utilizar as melhores práticas do mercado e utilizando um grande número de ferramentas open-source.

SUMÁRIO

AGRADECIMENTOS	I
RESUMO	IV
SUMÁRIO	V
1. INTRODUÇÃO	1
1.1. BREVE HISTÓRICO DO SISTEMA EXISTENTE	2
1.2. PROBLEMAS DIAGNOSTICADOS	2
1.3. SURGIMENTO DA NECESSIDADE DE UM NOVO SISTEMA	2
1.4. USUÁRIOS DO SISTEMA	2
1.5. EMPRESA INTERESSADA	3
2. OBJETIVO	4
3. CRONOGRAMA DE ATIVIDADES	5
4. RECURSOS NECESSÁRIOS AO DESENVOLVIMENTO	6
4.1. RECURSOS DE HARDWARE (1ª fase)	6
4.2. RECURSOS DE HARDWARE (2ª fase)	6
4.3. RECURSOS DE SOFTWARE	6
4.4. FERRAMENTAS	7
4.4.1. ECLIPSE	7
4.4.2. JUDE	7
4.4.3. FIREBIRD	8
4.4.4. POWERDESIGNER	9
4.4.5. TOMCAT	9
4.5. RECURSOS HUMANOS	10
5. PROPOSTA DO NOVO SISTEMA	11
5.1. DESCRIÇÃO DO SISTEMA PROPOSTO	11
5.2. RESULTADOS ESPERADOS	11
5.3. RESTRIÇÕES DO SISTEMA PROPOSTO	12
5.4. RECURSOS NECESSÁRIOS PARA A EXECUÇÃO	12
5.4.1. CLIENTE	12
5.4.2. SERVIDOR	12
5.5. RECURSOS HUMANOS	13
5.6. ÁREAS AFETADAS COM O NOVO SISTEMA	13
6. METODOLOGIA ADOTADA	14
6.1. EMBASAMENTO TEÓRICO	14
6.1.1. SOFTWARE COMERCIAL	14
6.1.2. ARQUITETURA DO SOFTWARE	16
6.1.3. ORIENTAÇÃO A OBJETOS	18
6.1.4. OBJETO-RELACIONAL	19
6.1.5. PADRÕES DE PROJETO	21
6.2. TECNOLOGIA UTILIZADA	23
6.2.1. JAVA	23
6.2.2. J2EE	24
6.2.3. HTTP	27
6.2.4. SERVLET	27
6.2.5. FRAMEWORK	29
6.2.6. STRUTS	30
6.2.7. TILES	32

6.2.8. FOLHAS DE ESTILO	33
6.2.9. JAVASCRIPT	34
6.2.10. XML	35
6.2.11. UML	37
6.3. PADRÕES UTILIZADOS	40
6.3.1. MVC	40
6.3.2. VALUE OBJECT – OBJETOS DE VALOR	40
6.3.3. BUSINESS OBJECT – OBJETOS DE NEGÓCIOS	41
6.3.4. DATA ACCESS OBJECT – OBJETO DE ACESSO A DADOS	41
7. DOCUMENTAÇÃO DA ANÁLISE	42
7.1. REGRAS DE NEGÓCIO	42
7.2. MODELO CONCEITUAL DE DADOS	43
7.3. MODELO FÍSICO DE DADOS	44
7.4. CASOS DE USO	45
7.4.1. CLIENTE	46
7.4.2. ADMINISTRADOR	47
7.4.3. SISTEMA DE GERENCIAMENTO DA DISTRIBUIDORA	48
7.4.4. SISTEMA DE GERENCIAMENTO BANCÁRIO	49
7.5. DIAGRAMA DE CLASSES	50
7.5.1. ANÁLISE	51
7.5.2. PROJETO	52
7.6. DIAGRAMAS DE SEQUÊNCIA	53
7.6.1 EFETUAR CADASTROS	54
7.6.2 CONSULTAR PRODUTOS	55
7.6.3 CONSULTAR PAGAMENTOS	56
7.6.4 LISTAR PEDIDOS EFETUADOS	57
7.6.5 LISTAR PAGAMENTOS EFETUADOS	58
7.6.6 REALIZAR PEDIDO	59
7.7. DIAGRAMA DE ATIVIDADE	60
8. MODELO FÍSICO	61
8.1. DIAGRAMA HIERÁRQUICO DE MÓDULOS (MENUS)	62
8.1.1 MENU CLIENTE	62
8.1.2 MENU ADMINISTRADOR	62
8.2. DESCRIÇÃO DAS INTERFACES (TELAS)	63
8.2.1 LOGIN	64
8.2.2 PRINCIPAL	65
8.2.3 CADASTROS	66
8.2.4 PESQUISAS	74
8.3. DESCRIÇÃO DOS RELATÓRIOS	84
8.4. DESCRIÇÃO ALGORÍTMICA DAS OPERAÇÕES	85
8.4.1 INCLUIR REGISTROS	86
8.4.2 ALTERAR OU EXCLUIR REGISTROS	87
8.4.3 RECUPERAR REGISTROS (PESQUISAR)	88
8.4.4 GERAR PAGAMENTO	89
8.4.5 MONTAR PEDIDO	90
9. DICIONÁRIO DE DADOS	91
9.1. LISTA DE DOMÍNIOS	91
9.2. LISTA DE ITENS DE DADOS	91
9.3. LISTA DE ENTIDADES	94

9.4.	LISTA DE ATRIBUTOS DA ENTIDADE CIDADE	94
9.5.	LISTA DE IDENTIFICADORES DA ENTIDADE CIDADE	95
9.6.	LISTA DE ATRIBUTOS DA ENTIDADE CONDIÇÃO	95
9.7.	LISTA DE IDENTIFICADORES DA ENTIDADE CONDIÇÃO	95
9.8.	LISTA DE ATRIBUTOS DA ENTIDADE EMPRESA	95
9.9.	LISTA DE IDENTIFICADORES DA ENTIDADE EMPRESA	96
9.10.	LISTA DE ATRIBUTOS DA ENTIDADE ESTADO	96
9.11.	LISTA DE IDENTIFICADORES DA ENTIDADE ESTADO	96
9.12.	LISTA DE ATRIBUTOS DA ENTIDADE ITEM	96
9.13.	LISTA DE IDENTIFICADORES DA ENTIDADE ITEM	96
9.14.	LISTA DE IDENTIFICADORES DA ENTIDADE LOTE	97
9.15.	LISTA DE IDENTIFICADORES DA ENTIDADE LOTE	97
9.16.	LISTA DE ATRIBUTOS DA ENTIDADE PAGAMENTO	97
9.17.	LISTA DE IDENTIFICADORES DA ENTIDADE PAGAMENTO	97
9.18.	LISTA DE ATRIBUTOS DA ENTIDADE PEDIDO	98
9.19.	LISTA DE IDENTIFICADORES DA ENTIDADE PEDIDO	98
9.20.	LISTA DE ATRIBUTOS DA ENTIDADE PRODUTO	98
9.21.	LISTA DE IDENTIFICADORES DA ENTIDADE PRODUTO	98
9.22.	LISTA DE ATRIBUTOS DA ENTIDADE PROMOÇÃO	98
9.23.	LISTA DE IDENTIFICADORES DA ENTIDADE PROMOÇÃO	98
9.24.	LISTA DE ATRIBUTOS DA ENTIDADE REGIÃO	99
9.25.	LISTA DE IDENTIFICADORES DA ENTIDADE REGIÃO	99
9.26.	LISTA DE ATRIBUTOS DA ENTIDADE TIPO	99
9.27.	LISTA DE IDENTIFICADORES DA ENTIDADE TIPO	99
9.28.	LISTA DE ATRIBUTOS DA ENTIDADE UNIDADE	99
9.29.	LISTA DE IDENTIFICADORES DA ENTIDADE UNIDADE	99
9.30.	LISTA DE RELACIONAMENTOS	100
10.	CONSIDERAÇÕES FINAIS	101
11.	REFERÊNCIAS BIBLIOGRÁFICAS	102

1. INTRODUÇÃO

A internet ainda é jovem e vulnerável. Portanto, a sua história não é muito longa. A web iniciou quando tudo consistia apenas em páginas estáticas. A menos que você seja uma criança-prodígio de seis anos de idade lendo este livro porque este mais interessado em programação web Java do que em PlayStation 2, é bem provável que você tenha passado por um período em que um web site nada mais era do que páginas HTML (HyperText Markup Language – linguagem de marcação de hipertexto). No início, um web site tinha no máximo uma página e, com mais frequência do que o contrário era chamado de home page (página pessoal).

Os termos “Internet application” (aplicativo de Internet) ou “web applications” (aplicativo da web) foram cunhados quando foi introduzido o conteúdo dinâmico. Vagamente interpretado, um aplicativo web é um web site cujos conteúdos são gerados dinamicamente antes de serem enviados ao browser. Primeiro você deve entender como a Internet trabalha, antes de aprender como trabalha um aplicativo web.

Quando você navega na Internet, basicamente solicita determinado arquivo localizado em um computador em especial no local que você especifica no Uniform Resource Locator (URL – localizador de recurso uniforme). O computador, onde o arquivo esta armazenado é chamado na Internet que solicite arquivos que ele hospeda. Já que você nunca sabe quando um usuário visitará e usará o seu aplicativo web, o seu servidor web precisa estar ativo e em execução o tempo todo.

Quando você clica ou digita em um URL na caixa Location (localização) ou Address (endereço) de seu browser, acontecem as seguintes coisas:

- ❖ O browser cliente estabelece uma conexão TCP/IP (Transmission Control Protocol/Internet Protocol – protocolo de transmissão/protocolo de Internet) com o servidor;
- ❖ O browser envia uma solicitação ao servidor;
- ❖ O servidor envia uma resposta ao cliente;
- ❖ O servidor fecha a conexão;

Note que, depois de enviar a página solicitada ao browser, o servidor sempre fecha a conexão, tendo ou não o usuário solicitado outras páginas do servidor.

(KURNIAWAN, 2002)

1.1.

BREVE HISTÓRICO DO SISTEMA EXISTENTE

Há alguns anos a forma de adquirir medicamentos para a farmácia só era possível através do vendedor. Esse anotava o pedido que seria entregue à distribuidora juntamente com os outros pedidos regionais. Esse processo era demorado, pois a mercadoria solicitada chegaria ao cliente somente alguns dias depois.

Buscando sempre oferecer aos clientes um atendimento mais rápido, as distribuidoras de medicamentos acompanharam a evolução natural. Com a expansão da telefonia, como o surgimento do fax, os pedidos eram enviados através do mesmo. Criou-se também o sistema de televendas, que obviamente eram pedidos feitos pelo telefone, normalmente com a ligação paga pela distribuidora.

A fim de reduzir custos e melhorar o atendimento, surge o pedido eletrônico.

1.2. PROBLEMAS DIAGNOSTICADOS

No atual sistema são facilmente diagnosticados vários problemas:

- ❖ Gastos excessivos com telefone em bate-papo informal;
- ❖ Erro de comunicação entre o comprador e o serviço de televendas, causa de várias devoluções de mercadoria;
- ❖ Despesas com o vendedor em viagens, o que inclui hospedagem e alimentação;

1.3. SURGIMENTO DA NECESSIDADE DE UM NOVO SISTEMA

Com o desejo de atender cada vez melhor e mais rápido, com uma menor margem de erros e facilitando a vida do comprador, surgiu a necessidade de um novo sistema.

Esse deve funcionar de tal forma que transporte virtualmente o comprador para dentro da distribuidora, sem ao menos sair da frente de seu computador.

1.4. USUÁRIOS DO SISTEMA

O sistema poderá ser utilizado por distribuidoras, laboratórios e

estabelecimentos que lidam com medicamentos e produtos farmacêuticos em geral.

1.5. EMPRESA INTERESSADA

O projeto SPARTACUS é uma pesquisa financiada pelo Laboratório Farmacêutico Elofar, através do seu representante no noroeste de Minas: Raimundo Nonato dos Santos.

Esse projeto busca descobrir novas alternativas de mercado para o laboratório distribuir os seus produtos, com maiores facilidades e comodidades para os seus clientes.

2.

OBJETIVO

Desenvolvimento de um sistema de venda de medicamentos e produtos farmacêuticos em geral para o ambiente web, suprimindo as necessidades de uma distribuidora.

Além disso, o projeto pretende reduzir o tempo de desenvolvimento mantendo a qualidade; utilizar ferramentas produtivas que são padrões de mercado e que são implementações de metodologias confiáveis.

3.

CRONOGRAMA DE ATIVIDADES

<i>Atividades</i>	<i>ago/05</i>	<i>set/05</i>	<i>out/05</i>	<i>nov/05</i>	<i>dez/05</i>
Análise					
Projeto					
Implementação					
Testes					
Apresentação					
Documentação					

4.

RECURSOS NECESSÁRIOS AO DESENVOLVIMENTO

4.1. RECURSOS DE HARDWARE (1ª fase)

- ❖ Processador: Pentium II com 300 MHZ;
- ❖ Memória: 128 MB;
- ❖ HD: 4 GB;
- ❖ Gravadora: HP 8100;
- ❖ Impressora: HP 692C;
- ❖ Internet à Cabo com 256 KBPS;

4.2. RECURSOS DE HARDWARE (2ª fase)

- ❖ Processador: Pentium IV com 3 GHZ;
- ❖ Memória: 512 MB;
- ❖ HD: 80 GB;
- ❖ Gravadora: LG HL-DT-ST DVD-RAM GSA-4160B;

4.3. RECURSOS DE SOFTWARE

- ❖ Windows 98 2ª Edição (1ª fase) - <http://www.microsoft.com>;
- ❖ Windows XP Professional (2ª fase) - <http://www.microsoft.com>;
- ❖ Java 2 Platform Standard Edition Development Kit 1.4 - <http://java.sun.com>;
- ❖ PowerDesigner 8.0 - <http://www.sybase.com/products/developmentintegration/powerdesigner>;
- ❖ Jude Community Edition - <http://www.esm.jp/jude-web>;
- ❖ Firebird 1.5 - <http://www.firebirdsql.org>;
- ❖ IBExpert 2005 - <http://www.ibexpert.com>;
- ❖ Tomcat 5.0 - <http://tomcat.apache.org>;
- ❖ Eclipse 3.0 - <http://www.eclipse.org>;
- ❖ Omondo UML (Plugin do Eclipse) - <http://www.omondo.com>;
- ❖ Lombok (Plugin do Eclipse) - <http://www.objectlearn.com>;

- ❖ Macromedia Studio MX - <http://www.macromedia.com>;
- ❖ Microsoft Office 2000 (1ª fase) - <http://www.microsoft.com>;
- ❖ Microsoft Office 2003 (2ª fase) - <http://www.microsoft.com>;

4.4. FERRAMENTAS

4.4.1. ECLIPSE

Ao programar em Java, o programador deve fazer uso de diversas ferramentas de desenvolvimento a fim de aumentar a sua produtividade. Todo projeto bem sucedido começa pela escolha de boas ferramentas.

O projeto Eclipse é um consórcio de mais de 50 grandes empresas da área (como IBM, Borland, Intel, etc.) que visa a criação de ferramentas gratuitas de desenvolvimento para Java. Um dos seus resultados é o ambiente de desenvolvimento Eclipse que vem se tornando referência no mercado por ser poderoso, flexível, fácil de usar e gratuito. Com o Eclipse é possível desenvolver toda a natureza de sistemas, até o voltado para a web. Conta ainda, com o suporte para o desenvolvimento em equipe.

Graças a tecnologia de plugins, o Eclipse permite personalizar o ambiente do desenvolvedor de acordo com o projeto que está sendo desenvolvido, seja ele, um simples projeto com páginas HTML estáticas, até aplicações com o uso de EJB, frameworks diversos ou J2ME. A versão básica encontrada na web já contém alguns plugins, como Ant e CVS. Além disso, a tecnologia de plugins possibilita a criação de seus próprios plugins.

4.4.2. JUDE

JUDE ou Java and UML Developer Environment é uma das ferramentas grátis para UML mais poderosas disponíveis atualmente. Com características que não são encontradas em outras ferramentas grátis, como adicionar métodos no diagrama de seqüência e a alteração se refletir no diagrama de classes.

O JUDE não possui um apelo gráfico tão bom quanto o Poseidon, ou tantas funcionalidades como Rational Rose, Together e Magic Drawn. Sua performance

impressiona, principalmente tratando-se de uma ferramenta 100% Java/Swing, desmistificando que Swing é lento.

4.4.3. FIREBIRD

O Firebird é um banco de dados, Cliente/Servidor Open Source, compatível com o padrãoSQL-Ansi-92, originado a partir do código do Interbase 6. Hoje, entre suas duas versões – Classic e SuperServer, ele roda em mais de 10 plataformas/sistemas operacionais, e vem conquistando cada vez mais espaço no disputado mundo dos servidores SQL.

Em julho de 2000, a Borland abriu o código fonte do Interbase 6 tornando-o Free e Open Source. No entanto, entre o anúncio da abertura do código e a consolidação do processo, a comunidade de usuários passou por um período de incertezas, o que acabou gerando um certo descontentamento pelo modo como a Borland estava conduzindo o processo de abertura de código. Quando a promessa finalmente se tornou realidade, um grupo de pessoas resolveu criar um novo banco de dados, baseado no código do Interbase 6, que foi chamado de Firebird. Entre as pessoas estavam nomes influentes da comunidade Interbase, como por exemplo Ann Harrison, que é conhecida como “mãe do Interbase”, sendo uma das pessoas que mais entendem da sua arquitetura interna e que, por sinal, é casada com Jim Starkey, o criador e idealizador do Interbase.

Ao mesmo tempo em que abriu o código, a Borland anunciou que continuaria desenvolvendo uma versão comercial do Interbase, e que essas novas versões não seriam gratuitas, nem teriam seu código aberto. Isso trouxe mais desconforto no relacionamento entre a comunidade Firebird e a Borland. De lá para cá, a Borland lançou as versões 6.5 e 7.x do Interbase (ambas comerciais) e não fez qualquer tipo de atualização no código do Interbase 6, estando o mesmo estagnado há vários anos. Em compensação, o Firebird ganhou novas versões, e com elas uma legião de usuários que em sua maioria já utilizavam o Interbase 6.0, e acabaram migrando para o Firebird com a garantia de usar um produtos que está em pleno desenvolvimento e que será sempre Free e Open Source.

O Firebird não traz uma ferramenta gráfica de administração padrão (o

Interbase traz o IBConsole), estando disponível apenas os utilitários de linha de comando. No entanto existem inúmeras ferramentas para download, sendo que algumas são gratuitas. A decisão de não trazer uma ferramenta padrão foi tomada para não prejudicar os fabricantes das ferramentas já existentes e permitir que o usuário escolha a que mais lhe agrada. Uma das ferramentas mais completas é o IBExpert (www.ibexpert.com), que possui uma versão Free (com algumas limitações) e uma versão shareware completa. (CANTU, 2005)

4.4.4. POWERDESIGNER

O PowerDesigner é uma ferramenta da Sybase para modelagem e soluções para empresas que necessitam construir ou refazer aplicações de forma rápida e consistente. Para isto, o PowerDesigner possui componentes para diversas atividades: modelagem de processos de negócios, modelagem de dados, modelagem de sistemas orientados a objetos, armazenamento e controle de modelos desenvolvidos. Para cada um desses componentes, existe ainda uma série de funcionalidades agregadas. Entretanto, nos restringiremos a discutir as características envolvidas com a modelagem e projeto de banco de dados.

Sabe-se que um projeto de banco de dados é composto de três fases: modelo conceitual, modelo físico e criação de script (código com os comandos DDL) responsável por criar toda a estrutura necessária no sistema gerenciador de banco de dados (SGBD). (MOSCARDINI)

4.4.5. TOMCAT

Atualmente, estão disponíveis vários contentores servlets. O mais popular – e aquele reconhecido como o contentor servlet/JSP oficial – é Tomcat. Originalmente projetado pela Sun Microsystems, o código fonte Tomcat foi entregue a Apache Software Foundation, em outubro de 1999. Nesse novo lar, Tomcat foi incluído como parte do projeto Jakarta, um dos projetos da Apache Software Foundation. O trabalho pelo processo Apache, Sun e outras empresas – com a ajuda de programadores voluntários de todo o mundo – transformou Tomcat em uma implementação de referência servlet de classe mundial. Dois meses depois da entrega, Tomcat versão 3.0

foi lançado. Ele teve vários lançamentos 3.x até que a versão 3.3 fosse apresentada.

O sucessor da versão 3.3 é a versão atual, versão 4.0. O contentor servlet (Catalina) é baseado em uma arquitetura completamente nova, e foi desenvolvido desde o início para flexibilidade e desempenho. A versão 4.0 implementa as especificações Servlet 2.3 e JSP 1.2.

O próprio Tomcat é um servidor web. Isso significa que você pode usar Tomcat para solicitar serviços HTTP em servlets, assim como arquivos estáticos (HTML, arquivos de imagem e assim por diante). No entanto, na prática, como ele é mais rápido em solicitações não servlets, não JSP, normalmente Tomcat é usado como um módulo, com um outro serviço web mais robusto, como o servidor web Apache ou Microsoft Internet Information Server. Apenas solicitações para servlets ou páginas JSP são passadas para o Tomcat.

Para escrever um servlet, você precisa de pelo menos a versão 1.2 do Java Development Kit. Se você não tiver carregado um, pode fazer o download do JDK 1.2 de <http://java.sun.com/j2se>. A referência de implementação para ambos, servlets e JSP, não está incluída em J2EE, mas está incluída em Tomcat. Tomcat é simplesmente escrito em Java. (KURNIAWAN, 2002)

4.5. RECURSOS HUMANOS

Para o desenvolvimento do SPARTACUS foi necessário um técnico em informática com conhecimentos gerais em desenvolvimento de sistemas.

5.

PROPOSTA DO NOVO SISTEMA

5.1. DESCRIÇÃO DO SISTEMA PROPOSTO

O sistema propõe uma realização da venda de medicamentos e produtos farmacêuticos em geral, com maiores facilidades e rapidez para a distribuidora quanto para o cliente da mesma:

- ❖ Os principais clientes do nosso sistema podem ser as farmácias de todo Brasil;
- ❖ Os nossos fornecedores serão os laboratórios, maiorias multinacionais com sede no país.
- ❖ A distribuidora é conveniada com várias transportadoras que fazem à entrega do pedido;
- ❖ Os pagamentos são feitos somente através do boleto bancário;
- ❖ Na distribuidora temos um software do banco que recebe os pagamentos via web (SGB). O SGB gera um arquivo que deverá ser importado pelo nosso sistema;
- ❖ Temos também um sistema de gerenciamento da distribuidora (SGD) que recebe a notificação do pedido vindo do nosso sistema;
- ❖ O chão de fábrica recebe o pedido do SGD e faz a montagem física;
- ❖ A nota fiscal e o boleto bancário são emitidos pelo SGD e são enviados juntos com a mercadoria pela transportadora conveniada para o cliente;
- ❖ Essa comunicação entre o SGB, o SGD e o SPARTACUS são as restrições do nosso sistema, pelo menos por enquanto;

5.2. RESULTADOS ESPERADOS

- ❖ Redução dos custos com telefone;
- ❖ Redução da taxa de devolução de mercadorias;
- ❖ Possibilita um melhor controle de estoque;
- ❖ Melhor gerenciamento de contas a receber;
- ❖ Melhor gerenciamento de compras a serem realizadas;

A maioria dos resultados esperados pode ser gerada de forma indireta.

5.3. RESTRIÇÕES DO SISTEMA PROPOSTO

- ❖ O sistema não controlará as contas a pagar da distribuidora, nem qualquer outra atividade de gerenciamento da mesma.
- ❖ A comunicação com o sistema de gerenciamento da distribuidora e com o sistema de recebimento bancário são as atuais restrições do SPARTACUS.

5.4. RECURSOS NECESSÁRIOS PARA A EXECUÇÃO

5.4.1. CLIENTE

HARDWARE IDEAL

- ❖ Processador: Pentium II com 300 MHZ;
- ❖ Memória: 128 MB;
- ❖ HD: 4 GB;
- ❖ Internet á Cabo com 128 KBPS;

HARDWARE MÍNIMO

- ❖ Processador: Pentium II com 233 MHZ;
- ❖ Memória: 32 MB;
- ❖ HD: 2 GB;
- ❖ Internet discada com um Modem com 56 KBPS;

SOFTWARE

Necessário um navegador de internet, de preferência, o Microsoft Internet Explorer 6.0, devido a incompatibilidade da tecnologia JavaScript com outros navegadores. Além disso, é necessária a instalação da máquina virtual Java, de preferência, a JRE 1.4.2.

5.4.2. SERVIDOR

HARDWARE IDEAL

- ❖ Processador: DUAL XEON 3.06 GHz Cache 1 MB;
- ❖ Memória: 2 GB DDR;
- ❖ HD: 2 x 72 GB SCSI - RAID 1 *;

HARDWARE MÍNIMO

- ❖ Processador: PENTIUM IV 2.4 GHz;
- ❖ Memória: 512 MB DDR;
- ❖ HD: 2 x 80 GB IDE 1 *;

* São discos em modo de espelhamento para tolerância à falhas.

Para maiores detalhes:

http://www.locawebidc.com.br/assinaturas/servidor_dedicado.asp

SOFTWARE

Necessário a instalação de um servidor web (de preferência o Apache Tomcat 5.0) e do kit de desenvolvimento Java (de preferência o JSDK 1.4.2).

5.5. RECURSOS HUMANOS

Para administração do sistema é necessário no mínimo uma pessoa com conhecimento básico em informática.

5.6. ÁREAS AFETADAS COM O NOVO SISTEMA

O SPARTACUS abrange várias áreas do ramo de medicamentos, principalmente as farmácias de todo o Brasil.

Como o sistema realiza a venda de produtos, todos os interessados na comercialização deste são afetados, mesmo que indiretamente, como os laboratórios, por exemplo.

6.

METODOLOGIA ADOTADA

6.1. EMBASAMENTO TEÓRICO

6.1.1. SOFTWARE COMERCIAL

O termo comercial refere-se a uma organização de indivíduos ou entidades, presumivelmente trabalhando juntos para conseguir alguns objetos comuns. As organizações têm todas as formas e tamanhos, grandes e pequenas, com e sem fins lucrativos, governamentais e não-governamentais.

Porém, há chances de que, quando alguém use o termo comercial, quer dizer uma grande organização com fins lucrativos, como a Intel, General Motors, Wal-Mart, Bank of América ou eBay.

As empresas geralmente têm algumas necessidades comuns, como compartilhar e processar informações, o gerenciamento do componente ativo e controle, o planejamento de recursos, o gerenciamento do cliente, a proteção do conhecimento comercial, etc. O termo software é usado para referir coletivamente a todo o software envolvido em suportar esses elementos comuns de uma empresa.

O software comercial pode consistir em diversas partes distintas atualmente; mas as empresas perceberam gradualmente que há uma forte necessidade de que seus diversos sistemas se integrem bem e se aproveitem onde for apropriado para o máximo benefício da empresa. O B2B e o B2C são bons exemplos de tal integração e aproveitamento.

Algumas das maneiras em potencial de uma empresa esperar aproveitar o software comercial integrado são dadas a seguir:

- ❖ Integrando seu suporte ao cliente e o conhecimento do produto doméstico, uma empresa poderia fornecer serviços novos e melhores serviços para seus clientes através da web.
- ❖ Ligando sua máquina de marketing ao mundo on-line, uma empresa poderia atingir um público muito maior on-line.
- ❖ Ligando seu gerenciamento de vendas e inventário, uma empresa poderia ser capaz de planejar canais de vendas Web específicas e com custo

menor para atingir um segmento do mercado não controlado.

❖ Fornecendo um front-end para um dos serviços usados por seus funcionários, como o sistema de pedido de suprimentos do escritório interno e ligando-o ao sistema da contabilidade, a empresa poderia diminuir o custo geral e melhorar a eficiência do funcionário.

❖ A disponibilidade do sistema do RH da empresa on-line poderia ser usado como uma maneira de dar aos funcionários mais controle sobre a sua saúde, 401(k) opções e reduzir os custos administrativos em geral para a empresa.

❖ Automatizando uma de suas operações com muitos recursos humanos e tornando-as disponíveis a qualquer momento e em qualquer lugar, uma empresa poderia fornecer um serviço melhor para seus clientes enquanto reduz os custos operacionais gerais.

As empresas bem-sucedidas tendem a crescer no tamanho, contratar mais pessoas, ter mais clientes e mais acessos do site Web, ter vendas e rendimentos maiores, adicionar mais locais, etc. Para suportar esse crescimento, o software comercial tem de ser dimensionável em termos de aceitar uma empresa maior e suas operações.

As empresas encontram limites quando crescem. Um limite comum é a incapacidade do hardware do computador em aumentar para colocar mais pessoas no mesmo local físico ou mesmo geográficas. Assim, o desafio da distribuição entra em cena. Diversas máquinas físicas resolvem as necessidades de processamento, mas introduzem o desafio do software distribuído. Novos locais em prédios ou geográficos endereçam a necessidade imediata, mas introduzem o desafio de trazer o mesmo nível de serviços a uma empresa localizada em outro lugar.

Conectar os sistemas conectados anteriormente para ganhar as eficiências de aumenta da empresa pode ser um desafio maior. Os sistemas de herança eram geralmente designados com finalidades específicas em mente e não eram concebidos especificamente com a integração com outros sistemas em mente. Por exemplo, o gerenciamento dos recursos humanos talvez tenha sido tratado como uma necessidade distinta sem muita interação com o gerenciamento financeiro, e o gerenciamento de vendas tinha pouca, se alguma, relação com o suporte do cliente. Essa abordagem

desunida para o desenvolvimento do software geralmente resultava em excelentes produtos pontuais sendo comprados para endereçar necessidades específicas, mas comumente resultam em arquiteturas do software difíceis de integrar.

Um desafio afim é a necessidade de lidar com um ambiente com diversos revendedores. Em parte sem evolução e em parte sem necessidade, o software comercial acabou com produtos similares de diversos revendedores usados para a mesma finalidade. Por exemplo, embora a aplicação RH pudesse ser construída em um banco de dados Oracle 8i, a aplicação de suporte do cliente podia contar com o Microsoft SQL Server.

O software comercial também requer em geral algumas capacidades comuns, como serviços de segurança para proteger o conhecimento da empresa, serviços de transação para assegurar a integridade dos dados etc. Cada um requer habilidades específicas e conhecimento. Por exemplo, o devido tratamento da transação requer estratégias para se recuperar das falhas, lidar com as situações com diversos usuários, assegurar a consistência nas transações etc. Do mesmo modo, implementar a segurança pode desmandar uma compreensão dos vários protocolos de segurança e abordagens do gerenciamento da segurança.

Esses são apenas alguns dos desafios comuns que tem de ser endereçados ao lidar com o desenvolvimento do software comercial. (AHMED, 2002)

6.1.2. ARQUITETURA DO SOFTWARE

A maioria das definições de arquitetura de software envolve referências a um ou mais dos seguintes itens:

- ❖ Estrutura estática do software. A estrutura estática refere-se à maneira como os elementos do software se relacionam;
- ❖ Estrutura dinâmica do software, significando as relações que mudam na duração do software e determinam como o software fica quando executado;
- ❖ Composição (ou decomposição) do software. Refere-se ao tio de partes importantes, porém menores, como subsistemas e módulos, que podem fazer parte do software;
- ❖ Componentes e interação entre elas. A camada permite a imposição de

uma ordem específica ou estrutura no software, assim permitindo e/ou impedindo certas relações como supostamente adequadas para o software;

- ❖ Organização das partes físicas do software a serem distribuídas. O código-fonte físico tem de ser organizado nos devidos tipos de unidades distribuíveis, por exemplo, arquivos.jar, .war e .exe, para um uso otimizado;
- ❖ Restrições no software. Os limites naturais ou auto-impostos. Por exemplo, a exigência para o software ser escrito na linguagem Java;
- ❖ Princípios para o software. Ou seja, por que o software se parece assim? É importante porque de uma perspectiva arquitetural, se algo não puder ser explicado, então realmente não faz parte da arquitetura;
- ❖ Estilo que guia o desenvolvimento e a evolução do software;
- ❖ Funcionalidade do software. Em outras palavras, o que faz o software;
- ❖ Conjunto de decisões importantes sobre a organização do sistema do software;
- ❖ Outras considerações como a reutilização, o desempenho, o dimensionamento etc;

A seguinte definição talvez capture melhor a essência da arquitetura do software: A arquitetura do software de um programa ou sistema de informática é a estrutura ou estruturas dos sistemas, que compõem os componentes do software, as propriedades visíveis externamente desses componentes e as relações entre eles.

Toda parte do software já criada tem arquitetura. A arquitetura existe independentemente da construção do software tê-la criado com consciência ou mesmo tenha sabido o que o termo arquitetura do software significava.

Portanto, a questão real não é se seu software precisa de arquitetura mas se você precisa criá-la de uma maneira deliberada.

A lista seguinte contém algumas razões pelas quais é importante se concentrar na arquitetura do software:

- ❖ Uma abordagem organizada para a estrutura do software levará eventualmente a um sistema de software que é frágil e difícil de adicionar porque nenhuma consideração foi dada à necessidade de se adaptar a exigências novas ou alteradas;
- ❖ A decomposição do software em partes menores torna o software mais

fácil de entender, gerenciar, desenvolver e manter. Se devidamente feito, poderá também melhorar muito a reutilização nos projetos;

- ❖ A arquitetura do software ajuda no desenvolvimento do software baseado em componentes;
- ❖ Uma melhor reutilização poderá ser conseguida por meio de uma devida arquitetura. Considere uma linha de produtos que requeria os mesmos serviços básicos com ligeiras variações. Com uma abordagem de camadas, grandes alterações poderão ser necessárias para suportar os diversos produtos;
- ❖ Os limites mal concebidos podem impedir a evolução do software, por exemplo, um limite para ter um sistema monolítico e não distribuído porque os sistemas de software distribuídos são mais difíceis de construir.
- ❖ Falha em compreender e identificar de antemão como o software poderia ser modificado para aceitar mais um usuário e um processamento de dados mais pesado, para fornecer serviços mais novos, aproveitar as novas tecnologias etc., poderá levar a uma situação em que o software tenha de ser reescrito porque a arquitetura original não considerou as necessidades de dimensionamento e de evolução. A disponibilidade e a confiança do sistema do software são muito dependentes do dimensionamento do sistema;
- ❖ Ter uma arquitetura documentada facilita entender e comunicar a intenção e a essência do sistema do software para a equipe de desenvolvimento;
- ❖ A segurança construída para o software, o teste de software, a manutenção e o gerenciamento geral do software são também fortemente influenciados pela arquitetura do sistema de software; (AHMED, 2002)

6.1.3. ORIENTAÇÃO A OBJETOS

Vai trabalhar com Java ou C#? Ligue-se nestes termos de OOP:

- ❖ Objeto: um programa que funciona de forma autônoma e é usado como bloco básico para a montagem de softwares mais complexos. Um objeto contém tanto os dados quanto o código de programa para manipular esses dados. Um exemplo de objeto é um programa que exibe um botão na tela e

define como esse botão se comporta quando se clica nele.

- ❖ Abstração: A habilidade de um programa ignorar os aspectos não essenciais da informação que está manipulando. A abstração possibilita usar um objeto fornecendo apenas os dados necessários para a tarefa a ser realizada.
- ❖ Classe: Protótipo que define características comuns a todos os objetos de um determinado tipo.
- ❖ Encapsulamento: O termo refere-se ao fato de que o objeto é visto como uma caixa-preta. Sabe-se o que ele faz, mas não como isso é feito. O encapsulamento evita que estrutura interna do objeto seja alterada por código externo. Também facilita o reaproveitamento do código, já que não é preciso saber como funciona o objeto para usá-lo. Basta conhecer os métodos e propriedades.
- ❖ Hereditariedade: O fato de uma classe herdar características da superclasse que deu origem a ela, isso traz ganhos de produtividade na programação, pois não é preciso especificar novamente características que já foram definidas.
- ❖ Hierarquia: As maneiras como as classes se relacionam; superclasses englobando subclasses (veja: Superclasse e Subclasse).
- ❖ Instância: Uma ocorrência de um objeto de determinada classe. Exemplo: Relatório Anual é uma instância da classe Documento.
- ❖ Modularidade: Característica de programas que são divididos em blocos autônomos.
- ❖ Método: É uma ação que a classe ou objeto é capaz de realizar. Exemplo: Fechar é um método do objeto Janela.
- ❖ OOP: Programação orientada a objetos. É um modelo de programação em que cada programa é formado por uma coleção de objetos que interagem. A OOP procura dar aos programas, flexibilidade, facilidade de manutenção e possibilidade de reaproveitamento de código.
- ❖ Polimorfismo: A habilidade de um método comporta-se de modo diferente, dependendo do objeto a que é aplicado. Exemplo: suponha que Formatar é um método da classe Documento. Quando aplicado a um objeto da subclasse Planilha, ele vai oferecer opções de formatação diferentes daquelas

exibidas quando o objeto é da subclasse Gráfico.

- ❖ Propriedade: Característica de um objeto que pode ser alterada por uma instrução de programa. Exemplo: Largura, Altura e Cor são propriedades de Janela.
- ❖ Subclasse: Classe derivada de outra classe: Exemplo: Cientifica e Financeira são subclasses de Calculadora.
- ❖ Superclasse: Classe que dá origem à outra classe. Exemplo: Margem é uma superclasse de MargemEsquerda e MargemDireita. (GRECO, 2004)

6.1.4. OBJETO-RELACIONAL

Quase todas as aplicações requerem dados persistentes. Persistência é um dos conceitos fundamentais em desenvolvimento de aplicativo. Se um sistema de informação não preservasse dados inseridos por usuário quando a máquina central fosse desligada, o sistema seria de pequeno uso prático. Quando falamos sobre persistência em Java™, normalmente estamos falando sobre armazenar dados em um banco de dados relacional usando SQL.

Em um aplicativo orientado para objeto, a persistência permite que o objeto sobreviva ao processo que o criou. O estado do objeto pode ser armazenado em um disco e um objeto com o mesmo estado pode ser recriado em algum ponto no futuro.

Apesar de sua história longa (os primeiros documentos de pesquisa foram publicados no início dos anos 80), as condições para ORM usadas através de desenvolvedores variam. Alguns chamam isso de mapeamento objeto/relacional, outros preferem mapeamento de objeto simples. O golpe acentua quando o problema de incompatibilidade acontece quando os dois mundos colidirem.

Em poucas palavras, mapeamento objeto/relacional é automatizado (e transparente) -persistência de objetos em um aplicativo Java™ para as tabelas de um banco de dados relacional, usando metadados que descrevem o mapeamento entre os objetos e o banco de dados. ORM, em essência, trabalha por reversibilidade transformando dados de uma representação para outra.

Isso implica certas penalidades de desempenho. Porém, se ORM for implementado como middleware, haverá muitas oportunidades para uma otimização

que não existiria para uma camada de persistência codificada à mão. Um custo indireto adicional (o tempo de desenvolvimento) é a provisão e o gerenciamento de metadados que governam a transformação. Mas, novamente, o custo é menor do que custos equivalentes mantendo envolvida uma solução codificada a mão. E até mesmo os banco de dados de objeto ODMG-complacentes requerem metadados em nível de classe significativa.

- ❖ A solução de ORM consiste nas quatro partes seguintes:
- ❖ Uma API para executar operações de CRUD básicas em objetos de classes persistentes;
- ❖ A linguagem ou API para especificarem consultas que recorrem a classes e propriedades de classes e propriedades de classes;
- ❖ Uma facilidade para especificar mapeamento de metadados;
- ❖ Uma técnica para a implementação de ORM para integrar com objetos transacionais a fim de executar verificação suja (dirty), ir buscar associações lentas, e outras funções de otimização;

Estamos usando o termo ORM incluindo qualquer camada de persistência onde SQL seja autogerado de uma descrição baseada em metadados. Não estamos incluindo camadas de persistência onde o problema do mapeamento objeto/relacional seja resolvido manualmente por desenvolvedores por códigos manuais em SQL, usando JDBC. Com ORM, o aplicativo interage com as APIs ORM, e a classe de modelo de domínio é resumida do SQL/JDBC subjacente. Dependendo das características ou a implementação particular, o runtime do ORM também pode assumir responsabilidade por assuntos como encerramento otimista e armazenamento, aliviando completamente o aplicativo destas preocupações. (KING, 2005)

6.1.5. PADRÕES DE PROJETO

No período de 1991 a 1994, Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides – conhecidos com “a turma dos quatro” – usou sua experiência conjunta para escrever o livro *Design Patterns, Elements of Reusable Object-Oriented Software*. Este livro descreveu 23 padrões de projeto, cada um fornecendo a solução para um problema projeto de software comum na indústria. O livro agrupa padrões de

projeto em três categorias – padrões de criação de projeto, padrões estruturais de projeto e padrões comportamentais de projeto. Os padrões de criação de projeto descrevem técnicas para instanciar objetos (ou grupo de objetos). Os padrões de projeto permitem organizar as classes e objetos em estruturas maiores. Os padrões comportamentais estruturais de projeto atribuem responsabilidades a objetos.

A turma dos quatro mostrou que os padrões de projeto evoluíram naturalmente ao longo de anos de experiência da indústria. Em seu artigo *Seven Habits of Successful Pattern Writers*, John Vlissides afirma que “a atividade isolada mais importante na escrita de padrões é a reflexão”. Esta afirmação implica que, para criar padrões, os desenvolvedores precisam refletir sobre seus sucessos (e erros) e documentá-los. Os desenvolvedores usam padrões de projeto para capturar e empregar esta experiência coletiva da indústria, o que, no final, os ajuda a não cometer o mesmo erro duas vezes.

Novos padrões de projeto estão sendo criados todo o tempo e sendo apresentados rapidamente para desenvolvedores no mundo inteiro através da Internet.

Ao longo da última década, o setor de engenharia de software fez progressos significativos no campo de padrões de projeto – arquiteturas comprovadas para construir software orientado a objetos flexível e fácil de manter. O uso de padrões de projeto pode reduzir substancialmente a complexidade do processo de projetar. Projetar um sistemas de ATMs é uma tarefa significamente menos formidável se os desenvolvedores usam padrões de projeto. Além disso, o software orientado a objetos bem-projetado permite aos projetistas reutilizar e integrar componentes preexistentes em sistemas futuros. Os padrões de projeto beneficiam os desenvolvedores de sistemas:

- ❖ Ajudando a construir software com arquiteturas comprovadas e a experiência acumulada das empresas;
- ❖ Promovendo a reutilização de projeto em sistemas futuros;
- ❖ Ajudando a identificar erros e armadilhas comuns que ocorrem quando se constroem sistemas;

Ajudando a projetar sistemas de forma independentes da linguagem na qual eles serão finalmente implementados;

Estabelecendo um vocabulário de projeto comum entre desenvolvedores;

Encurtando a fase de projeto em um processo de desenvolvimento de software;

O conceito de usar padrões de projeto para construir sistemas de software originou-se na área de arquitetura. Os arquitetos usam um conjunto de elementos arquiteturais estabelecidos, com arcos e colunas, quando projetam edificações. Projetar com arcos e colunas é uma estratégia comprovada para construir edificações seguras – estes elementos podem ser vistos como padrões para projetos arquitetônicos.

Em software, os padrões de projeto não são classes nem objetos. Em vez disso, os projetistas usam padrões de projeto para construir conjuntos de classes e objetos. Para usar padrões de projetos de maneira eficaz, os projetistas precisam se familiarizar com os padrões mais populares e eficazes usados no setor de engenharia de software. (DEITEL, 2003)

O mais importante sobre os padrões é que eles são soluções aprovadas. Cada catálogo mencionado inclui apenas os padrões que foram considerados úteis por diversos desenvolvedores em vários projetos. Os padrões catalogados também são bem definidos; os autores descrevem cada padrão com muito cuidado e em seu próprio contexto, portanto será fácil aplicar o padrão em suas próprias circunstâncias. Os padrões catalogados também formam um vocabulário comum entre os desenvolvedores. Podemos usar termos como adaptador e fachada durante as análises da construção e comunicar exatamente o que temos em mente. Cada padrão traz consequências documentadas, boas e ruins, e os desenvolvedores podem escolher um padrão, prevenidos e avisados. Como muito cuidado é colocado no catálogo dos padrões, eles também representam um conjunto de melhores práticas para as equipes seguirem.

Os padrões são um mapa, não uma estratégia. Os catálogos geralmente apresentarão algum código-fonte como uma estratégia de exemplo, mas isso simplesmente seria uma possível implementação. Selecionar um padrão não é igual a importar uma classe; você ainda tem que implementar o padrão no contexto de sua aplicação. Os padrões não ajudarão a determinar qual aplicação você deve estar escrevendo – apenas como implementar melhor a aplicação assim que o conjunto de recursos e outras exigências forem determinados. Os padrões ajudam com *o que* e

como, mas não com *por que* ou *quando*. (HUSTED, 2004)

6.2. TECNOLOGIA UTILIZADA

6.2.1. JAVA

Talvez a contribuição mais importante da revolução dos microprocessadores até sua data seja o fato de ela ter possibilitado o desenvolvimento de computadores pessoais, que agora chegam a centenas de milhões em todo o mundo. Os computadores pessoais tiveram um profundo impacto sobre as pessoas e a maneira como as organizações conduzem e gerenciam seus negócios.

Muitas pessoas acreditam que a próxima área importante em que os microprocessadores terão um impacto profundo serão os dispositivos eletrônicos inteligentes destinados ao consumidor final. Reconhecendo isso, a Sun Microsystems financiou uma pesquisa corporativa interna com o codinome Green em 1991. O projeto resultou no desenvolvimento de uma linguagem baseada em C e C++ que seu criador, James Gosling, chamou de Oak (carvalho) em homenagem a uma árvore que dava para a janela do seu escritório na Sun. Descobriu-se mais tarde que já havia uma linguagem de computador chamada Oak. Quando uma equipe da Sun visitou uma cafeteria local, o nome Java (cidade de origem de um tipo de café importado) foi sugerido e pegou.

Mas o projeto Green atravessava algumas dificuldades. O mercado para dispositivos eletrônicos inteligentes destinados ao consumidor final não estava se desenvolvendo tão rapidamente como a Sun tinha previsto. Pior ainda, um contrato importante pelo qual a Sun competia fora concedido a outra empresa. Então, o projeto estava em risco de cancelamento. Por pura sorte, a World Wide Web explodiu em popularidade em 1993 e as pessoas da Sun viram o imediato potencial de utilizar Java para criar páginas da Web como chamado conteúdo dinâmico. Isso deu nova vida ao projeto.

Em maio de 1995, a Sun anunciou Java formalmente em uma conferencia importante. Normalmente, um evento como esse não teria gerado muita atenção. Entretanto, Java gerou interesse imediato na comunidade comercial por causa do

fenomenal interesse pela World Wide Web. Java é agora utilizada para criar páginas da Web com conteúdo interativo e dinâmico, para desenvolver aplicativos corporativos de grande porte, para aprimorar a funcionalidade de servidores da World Wide Web (os computadores que fornecem o conteúdo que vemos em nossos navegadores da Web), fornecerem aplicativos para dispositivos destinados ao consumidor final (como telefones celulares, pagers e assistentes pessoais digitais) e para muitas outras finalidades.

6.2.2. J2EE

O J2EE define uma arquitetura para desenvolver aplicações Java comerciais complexas e distribuídas.

O J2EE foi originalmente anunciado pela Sun Microsystems em meados de 1999 e foi lançado oficialmente no final do ano de 1999. O J2EE, sendo relativamente novo, ainda está desenvolvendo alterações significantes de versão para versão, especialmente na área do Enterprise JavaBeans (EJB).

O J2EE consiste no seguinte:

- ❖ Regras de construção para desenvolver aplicações comerciais usando o J2EE.
- ❖ Uma implementação de referencia para fornecer uma visão operacional do J2EE.
- ❖ Um conjunto de testes de compatibilidade para ser usado por terceiros para verificar a compatibilidade de seus produtos com o J2EE.
- ❖ Várias Application Programming Interface (API) para permitir o acesso genérico para recursos comerciais e infra-estrutura.
- ❖ Tecnologias para simplificar o desenvolvimento Java comercial.

A plataforma baseia-se no mantra Java: “Escreva uma vez, execute em qualquer lugar” através de um grupo de tecnologias e um conjunto de APIs. Elas são suportadas e vinculadas, por sua vez, por três elementos-chave, a saber a implementação de referência, as regras da construção e o conjunto de compatibilidade.

Existem três plataformas Java atualmente:

- ❖ Java 2 Platform, Micro Edition (J2ME): A plataforma para o desenvolvimento do software para dispositivos incorporados como telefones, palm tops, etc.
- ❖ Java 2 Platform, Standard Edition (J2SE): A mais familiar das plataformas Java 2. Também conhecida como Java Development Kit (JDK) e inclui capacidades como applets, JavaBeans, etc.
- ❖ Java 2 Platform, Enterprise Edition (J2EE): A plataforma para desenvolver aplicações comerciais dimensionáveis. É designada para ser usada em conjunto com o J2SE.

O J2EE também oferece algumas vantagens promissoras. Elas incluem recursos que permitem aos desenvolvedores se concentrarem no desenvolvimento da lógica comercial, na implementação do sistema sem um conhecimento detalhado anterior do ambiente de execução e na criação de sistemas que podem ser portados mais facilmente entre as plataformas de hardware e sistemas operacionais (SOs).

O desenvolvimento de software comercial é uma tarefa complexa e pode requerer um grande conhecimento de muitas áreas diferentes. Por exemplo, um esforço de desenvolvimento típico de aplicações comerciais poderia requerer que você fosse familiarizado com as questões de comunicação entre os processos, as questões de segurança, as consultas de acesso específicas do banco de dados etc. O J2EE inclui um suporte predefinido e largamente transparente para esses outros serviços similares. Como resultado, os desenvolvedores são capazes de se concentrar na implementação do código da lógica comercial ao invés do código que suporta a infra-estrutura da aplicação básica.

Talvez uma das maiores vantagens do J2EE seja seu suporte para os componentes. O software baseado em componentes tem numerosas vantagens sobre o desenvolvimento do software personalizado e tradicional.

- ❖ Produtividade mais alta: Menos desenvolvedores podem conseguir mais reunindo uma aplicação a partir de componentes predefinidos e testados previamente em vez de implementar uma solução personalizada a partir do zero.
- ❖ Desenvolvimento rápido: Os componentes existentes podem ser reunidos rapidamente para criar novas aplicações.

- ❖ Qualidade mais alta: Em vez de testar aplicações inteiras, os desenvolvedores da aplicação baseada em componentes podem se concentrar em testar a integração e a funcionalidade geral da aplicação conseguida através de componentes predefinidos.

- ❖ Manutenção mais fácil: Como os componentes são independentes para começar, a manutenção como, por exemplo, as atualizações de componentes individuais, é muito mais fácil e com um custo melhor.

Embora algum nível de componente de software exista, é muito distante do tipo de componente que prevalece nas outras indústrias, como a eletrônica ou de automóveis. Imagine a indústria eletrônica diminuída se cada e todo chip requerido precisasse ser feito à mão para montar um novo dispositivo eletrônico.

O J2EE facilita a criação de componentes de muitas maneiras. Alguns exemplos vêm a seguir:

- ❖ A natureza: “Escreva uma vez, execute em qualquer lugar” do Java torna-o atraente para desenvolver componentes para um conjunto diverso de sistemas de hardware e sistemas operacionais.

- ❖ O J2EE oferece uma abordagem bem desenvolvida para separar os aspectos do desenvolvimento de um componente de suas particularidades das montagens e seus aspectos da montagem de seus detalhes da distribuição. Assim, os componentes de modo independentes podem ser prontamente integrados em novos ambientes e aplicações.

- ❖ O J2EE oferece uma grande faixa de APIs que podem ser usadas para acessar e integrar os produtos fornecidos por revendedores de terceiros de uma maneira uniforme, por exemplo, banco de dados, sistemas de correio, plataforma de mensagens etc.

- ❖ O J2EE oferece componentes especializados que são otimizados para tipos específicos de papéis em uma aplicação comercial. Por exemplo, os componentes comerciais podem ser desenvolvidos em “tipos” diferentes dependendo do que devem fazer.

Os mercados de componentes já começaram a surgir. Um estudo Garner Group previu que, em 2003, 70% de todas as solicitações de todas as aplicações novas seriam construídas a partir de componentes (CBD), com a adoção rápida e o suporte

amplo da indústria, deverá desempenhar um papel de destaque nesta troca para o CBD. (AHMED, 2002)

6.2.3. HTTP

HTTP (protocolo de transferência de hipertexto) é o protocolo que permite aos servidores web e browsers trocar dados pela web. Ele é um protocolo de solicitação e resposta. O cliente solicita um arquivo e o servidor responde à solicitação. HTTP usa conexões TCP confiáveis – por padrão, na porta TCP 80, HTTP foi definido inicialmente em RFC 2068 (Request for Comments – solicitação de comentários). Depois, ele foi refinado em RFC 2616, que pode ser encontrado em <http://www.w3c.org/Protocols>.

Em HTTP é sempre o cliente quem inicia uma transação, estabelecendo uma conexão e enviando uma solicitação HTTP. O servidor não está em posição de contatar um cliente ou fazer uma conexão de retorno ao cliente. Nem o cliente ou o servidor podem encerrar prematuramente uma conexão. Por exemplo, quando usando um browser da web, você pode clicar o botão Stop em seu browser, para interromper o processo de download de um arquivo, fechando efetivamente a conexão HTTP com o servidor web. (KURNIAWAN, 2002)

6.2.4. SERVLET

A tecnologia Servlet é a base do desenvolvimento de aplicativo web usando a linguagem de programação Java. Ela é uma das tecnologias Java mais importante, e é a tecnologia subjacente para uma outra tecnologia Java popular para desenvolvimento de aplicativo: JavaServerPages (JSP). Portanto, entender a tecnologia servlet e sua arquitetura é importante, se você quer ser um desenvolvedor de servlet. Ainda que você planeje desenvolver o seu aplicativo web Java usando apenas páginas JSP, entender a tecnologia servlet ajuda a montar um aplicativo JSP mais eficiente e efetivo.

O objetivo deste capítulo é apresentar a tecnologia servlet e fazer com que você fique confortável com ela, apresentando instruções passo a passo, que lhe permitam montar e executar um aplicativo servlet.

Quando surgiu inicialmente, essa grande coisa, que chamamos de Internet, consistia apenas em conteúdos estáticos, escritos usando Hypertext Markup Language (HTML). Naquela ocasião, qualquer um que pudesse fazer páginas HTML era considerado um especialista em Internet.

No entanto isso não durou muito.

Logo os conteúdos web dinâmicos se tornaram possíveis, através de tecnologia Common Gateway Interface (CGI). CGI permite ao servidor web chamar um programa externo e passar informações de solicitação HTTP àquele programa externo, para processar a solicitação. A resposta do programa externo então é passada de volta ao servidor web, que a encaminha ao browser cliente. Os programas CGI podem ser escritos em qualquer linguagem que possa ser chamada pelo servidor web. Com o tempo, Perl tornou-se a linguagem mais popular para escrever programas CGI.

Entretanto, à medida que a Internet se tornou mais e mais popular, o número de usuários visitando um web site popular aumentou incrivelmente, e ficou claro que CGI tinha falhado em apresentar aplicativos escalonáveis de Internet. A falha em CGI é que cada solicitação de cliente faz o servidor web reproduz um novo processo do programa CGI solicitado. Como sabemos bem, o processo de criação é uma operação cara, que consome muitos ciclos de CPU e memória de computador.

Gradativamente, tecnologias novas e melhores substituirão CGI como a principal tecnologia para desenvolvimento de aplicativo web. O mundo testemunhou as seguintes tecnologias dominar o desenvolvimento da web: ColdFusion, SSJS, PHP, Servlet, JSP, ASP e ASP.NET.

No ano passado, ASP e servlet/JSP eram as principais tecnologias usadas no desenvolvimento de aplicativo web. Com o lançamento do ASP.NET não é difícil prever que essa tecnologia se tornará o principal concorrente de servlet/JSP. ASP (e ASP.NET) e servlet/JSP, cada qual tem seus próprios fãs, e não é fácil prever qual será a vencedora. O resultado mais provável é que nenhuma das duas será a vencedora absoluta no mercado; ao invés, provavelmente as tecnologias correrão lado a lado nos próximos anos.

Servlet (e JSP) oferece os seguintes benefícios, que não estão necessariamente disponíveis em outras tecnologias:

- ❖ Desempenho: O desempenho de servlets é superior a CGI, pois não há

processo de criação para cada solicitação de cliente. Ao invés, cada solicitação é gerenciada pelo processo contendor de servlet. Depois que um servlet termina de processar uma solicitação, ele permanece na memória, aguardando por uma outra solicitação.

❖ Portabilidade: Semelhante a outras tecnologias Java, os aplicativos servlets são portáveis. Você pode movê-los para outros sistemas operacionais sem dificuldades.

❖ Rápido ciclo-desenvolvimento: Como uma tecnologia Java, servlets têm acesso a rica biblioteca Java, que ajuda a apressar o processo de desenvolvimento.

❖ Robustez: Servlets são gerenciados pela Java Virtual Machine. Como tal, você não precisa se preocupar com galha de memória ou coleta de resíduos, o que lhe ajuda a criar aplicativos robustos.

❖ Aceitação difundida: Java é uma tecnologia amplamente aceita. Isso significa que numerosos fabricantes trabalham com tecnologias baseadas em Java. Uma das vantagens dessa aceitação difundida é que você pode encontrar com facilidade, e comprar, componentes, que se ajustem às suas necessidades, o que poupa tempo de desenvolvimento. (KURNIAWAN, 2002)

6.2.5. FRAMEWORK

Um framework é uma aplicação reutilizável e semi-completa que pode ser especializada para produzir aplicações personalizadas. Como as pessoas, as aplicações de software são mais semelhantes que diferentes. Elas são executadas nos mesmos computadores, esperam uma entrada dos mesmos dispositivos, produzem as mesmas exibições e gravam os dados nos mesmos discos rígidos. Os desenvolvedores que trabalham em aplicações eletrônicas convencionais estão acostumados a kits de ferramentas e a ambientes de desenvolvimento que aproveitam a igualdade entre as aplicações. Os frameworks da aplicação se desenvolvem nessa base comum para fornecer aos desenvolvedores um frameworks reutilizável que pode servir como a fundação para seus próprios produtos.

Um framework fornece aos desenvolvedores um conjunto de componentes

estruturais que tem as seguintes características:

- ❖ São conhecidos por funcionarem bem em outras aplicações;
- ❖ Estão prontos para serem usados como o próximo projeto;
- ❖ Pode também ser usados por outras equipes na organização;

Os frameworks são a proposta clássica do tipo construir versus comprar. Se você construir, irá compreendê-lo quando terminar – mas quanto tempo levará antes de poder fazer por si mesmo? Se a comprar, terá que passar pela aprendizagem – e quanto tempo levará antes de poder fazer por si mesmo? Não há nenhuma resposta certa aqui, mas a maioria dos observadores concordaria que frameworks como o Struts fornecem um retorno significativo no investimento em comparação com começar do zero, especialmente para projetos maiores. (HUSTED, 2004)

6.2.6. STRUTS

O Struts é um software de fonte aberta que ajuda os desenvolvedores a construírem aplicações web rápida e facilmente. O Struts conta com tecnologias padrões – como o JavaBeans, servlets Java e JavaServerPages (JSP) – que a maioria dos desenvolvedores sabe como usar. Adotando uma abordagem baseada em padrões e do tipo “preencha as lacunas” para o desenvolvimento do software, o Struts pode aliviar grande parte do trabalho pesado e demorado que vem com todo projeto novo.

O Struts é mantido pela Apache Software Foundation (ASF) como parte do seu projeto Jakarta. Além do Struts, o Jakarta mantém vários produtos de fonte aberta bem-sucedidos, inclusive o Tomcat, Ant e Velocity. O framework é chamado de “Struts” para nos lembrar das bases que mantêm nossas casas, prédios, pontes e, na verdade, nós mesmos quando estamos com pernas de pau. É uma excelente descrição do papel do que o Struts desempenha ao desenvolver aplicações web. Ao elevar as estruturas físicas, os engenheiros da construção usam suportes para fornecer sustentação para cada piso de um prédio. Do mesmo modo, os engenheiros do software usam o Struts para suportar cada camada de uma aplicação comercial.

A base do código inicial do Struts foi desenvolvida entre maio de 2000 e junho de 2001 quando a versão 1.0 foi lançada. Mais de 30 desenvolvedores contribuíram para a distribuição do Struts e outros milhares seguiram as listas de correspondência

do Struts. Sua base de código é gerenciada por uma equipe voluntária de “Membros”. Por volta de 2002, a equipe do Struts incluiu nove Membros ativos.

O primeiro arquiteto e desenvolvedor do framework Struts é Craig. R. McClanahan. Craig é também o primeiro arquiteto do Tomcat 4 e o arquiteto da implementação do Java Web Services Pack. Agora é chefe de especificação da Sun para Java Server Faces (JSR-127) assim com o Web Layer Architect para a plataforma Java 2 Enterprise Edition (J2EE).

O Struts está disponível para o público sem taxas sobre a licença Apache Software License [ASF, License]. Não há nenhuma aquisição ou outros custos recorrentes para usar o software. Diferente de algumas outras licenças de fonte aberta, a Apache Software License é favorável aos negócios. Você pode usar o Struts para criar um projeto comercial e distribuir o binário Struts sem nenhuma burocracia, taxas ou outros problemas. Pode também integrar os componentes Struts em sua própria estrutura exatamente como se eles fossem escritos em casa. Para obter detalhes completos, veja a Apache Software License em www.apache.org/LICENSE.

Grande parte dos utilitários principais do Java e dos frameworks são agora projetos de fonte aberta. Muitos desenvolvedores que trabalham nesses projetos fazem isso como parte de seus trabalhos regulares com empresas como a IBM, Sun Microsystems e Apple. Colaborar de modo aberto nesse tipo de software aproveita o mercado inteiro. Atualmente, muitos componentes de fonte aberta são integrados em produtos comerciais. As empresas então vendem a documentação profissional, com níveis de suporte garantidos e outros serviços valiosos do futuro mercado para seus clientes.

Quando o software está disponível gratuitamente, fica mais fácil para o mercado suportar. O Struts é um excelente exemplo disso. Embora, ainda seja jovem, já foi representado em dezenas de artigos e seminários, sem mencionar livros como o “Struts em ação”.

Muitas equipes de desenvolvedores não gostam de usar um software que não foi “inventado” em casa. Os componentes de fonte aberta fornecem todas as vantagens de escrever o mesmo software em casa, mas não o limitam a uma solução patenteada que apenas sua equipe compreende.

Os frameworks de fonte aberta são uma vitória para todos.

As aplicações web de hoje são componentes críticos da missão da empresa. Como sempre, as equipes de desenvolvimento precisam construir aplicações em tempo recorde, mas tem que construí-las corretamente e de modo que permaneçam.

Os desenvolvedores web do Java já tem utilitários para construir páginas de apresentação como as JavaServer Pages (JSP) e os templates Velocity. Também temos mecanismos para lidar com o banco de dados – JDBC e Enterprise JavaBeans (EJB), por exemplo. Mas o que usamos para reunir esses componentes? Temos o encanamento e uma parede... Do que mais precisamos?

Os desenvolvedores de hoje precisam construir aplicações cheias de recursos que podem ser mantidas com o tempo. Os frameworks de aplicação web, como o Struts, resolvem problemas comuns, portanto, os desenvolvedores podem se concentrar na funcionalidade exclusiva de sua aplicação. Os frameworks são especialmente importantes ao construir as aplicações web, uma vez que o modo como o HTTP e a HTML funcionam torna difícil criar aplicações dinâmicas.

O Struts tira o máximo da API padrão do servlet Java e ainda torna-se um teste compatível informado para os contêineres servlets. O Struts também baseia-se nos padrões comuns da construção “em camadas” para as aplicações. Essa construção ajuda a tornar as aplicações robustas e dimensionáveis.

Uma parte principal da arquitetura Struts é o modo como ela estende o fluxo do ciclo de solicitação-resposta básico do HTTP. O controlador Struts gerencia os caminhos usados por sua aplicação, ajuda a reunir com segurança a entrada dos usuários e pode localizar as mensagens da aplicação – especialmente as mensagens de erro.

O Struts é uma solução eficiente. Não deterá sua aplicação e geralmente libera os recursos que podem ser melhor usados em outro lugar.

Naturalmente, o Struts tem suas falhas. Muitos nomes de classes eram escolhidos com pressa durante o desenvolvimento e podem ser confusos; outras áreas poderiam também ser melhoradas. Apesar de qualquer desvantagem, o Struts é facilmente o framework de aplicação web mais popular disponível hoje. (HUSTED, 2004)

6.2.7. TILES

A utilização é uma preocupação básica na construção das aplicações atuais – e a consistência é um ingrediente básico da utilização. Os usuários desejam ficar concentrados na tarefa em mãos e ficam facilmente chateados com qualquer inconsistência na interface ou layout da tela de uma aplicação.

A consistência não é um desafio pequeno para uma aplicação web dinâmica; é comum que seja codificada manualmente. As ferramentas de layout para as páginas estáticas estão disponíveis para a maioria dos construtores hoje, mas algumas estão disponíveis para as aplicações baseadas em JavaServer Pages.

Pior, a aparência de uma aplicação é geralmente o último detalhe a ser finalizado, portanto mudará com frequência entre as versões – ou até arbitrariamente como parte de um “relançamento” de um site web maior. Isso pode criar um giro aterrorizante de codificação de última hora e teste – mesmo que seja apenas para alterar a cor de fundo ou adicionar um novo link na barra de menus.

Naturalmente, a consequência é mais que um bicho-papão; é um marco de boa construção. Como qualquer outro componente, as páginas web contêm muitos elementos comuns, cabeçalhos, rodapés, menus, etc. Geralmente, são cortados e colados de uma página para a seguinte. Mas, como qualquer componente, os erros são encontrados e os recursos são melhorados, levando a outro giro da “reutilização” do tipo cortar e colar.

Em uma aplicação web, a marca da página é um componente de programação como qualquer outro e deve ser mantido para o mesmo padrão de reutilização.

O framework Tiles usa uma forma mais avançada da ação de inclusão JSP. Em uma aplicação Tiles, o template do segundo plano ou layout geralmente define a posição de um cabeçalho, corpo do menu, conteúdo e rodapé. As outras páginas são então incluídas para preencher cada uma dessas posições. Se o cabeçalho mudar, então apenas esse arquivo do template precisará ser alterado. A alteração é refletida automaticamente em todas as páginas que incluem esse template. Os efeitos das alterações são minimizados – e os bichos-papões acalmados.

Os componentes HTML padrões, como as Cascading Style Sheets (CSSs), também funcionam bem com templates dinâmicos. Uma folha de estilo pode ajudar a

manter os templates consistentes internamente e minimiza os efeitos da alteração. (HUSTED, 2004)

6.2.8. FOLHAS DE ESTILO

As Styles Sheets ou “Folhas de Estilo” para a Web são um conjunto de definições que irão determinar/modificar a forma como as tags HTML serão exibidas pelo browser. Estão aprovadas pelo órgão que regulamenta os padrões de programação para a Web, o W3Construction (www.w3.org).

Para ter a definição completa da recomendação do W3C, você pode acessar o site www.w3.org/Style/Activity, onde você encontrará um documento contendo toda a definição e implementação desta tecnologia.

As Folhas de Estilo foram apresentadas pela primeira vez pelo W3C em 17 de dezembro de 1996, e tiveram a primeira de uma série de revisões apresentadas, em 11 de janeiro de 1999. Esta primeira versão, denominada CSS Nível 1, não teve porém, sua implementação muito difundida logo de início, já que não havia navegadores que dessem suporte à sua utilização.

As Sytles Sheets vieram a ser utilizadas de forma mais efetiva pelos autores de sites Web a partir do lançamento, pela Microsoft, da versão 3.0 do Internet Explorer, que começou a oferecer suporte parcial para este recurso, e sua então concorrente Netscape, no Netscape Navigator 4.0. A partir de então, inclusive, um dos padrões mais utilizados por autores de sites Web, devido á grande versatilidade que estas trazem na manutenção dos mesmos.

Após esta primeira versão, em 12 de maio de 1998 foi apresentada a segunda implementação das Syles Sheets, denominada CSS Nível 2, contando a partir daí com suporte mais extenso e com algumas adições e melhorias bastante significativas em relação à primeira implementação.

Atualmente, encontra-se em fase de construção a especificação 3 do CSS que pode ser encontrada no site do W3C.

É importante dizemos que, apesar de ter se tornado um padrão no desenvolvimento de páginas Web, o que resultou este sucesso não foram às novidades e novos recursos trazidos pelas Folhas de Estilo, pois quase tudo o que se pode fazer

com Folhas de Estilo pode ser feito também com as tags HTML convencionais (existem poucas exceções e uma enorme quantidade de itens que só poderiam ser feitos com inúmeras linhas de código em HTML). Entretanto, há algumas vantagens que fazem das Folhas de Estilo um recurso realmente poderoso:

- ❖ Maior controle do resultado apresentado;
- ❖ Maior versatilidade na formatação;
- ❖ Redução do esforço de criação e manutenção de sites web;
- ❖ Páginas mais velozes; (RUAS, 2003)

6.2.9. JAVASCRIPT

Ao contrário do que muitos são induzidos a acreditar, a linguagem JavaScript não é nenhuma extensão do Java. É muito comum encontrarmos dúvidas sobre o uso de JavaScript em fóruns Java e vice-versa.

O JavaScript foi desenvolvido pela Netscape e, originalmente, se chamava LiveScript. Mais tarde resolveram trocar o nome para mostrar sua proximidade com o Java, ainda que sejam linguagens totalmente distintas. Por ser uma linguagem interpretada, códigos JavaScript são escritos em forma de texto e na linguagem (em inglês) compreensível por nós humanos.

Mais tarde, um interpretador, disponível em todos os navegadores mais populares, transforma essa linguagem humana em linguagem de máquina. Por esta razão, códigos-fontes de programas JavaScript não podem ser ocultos aos olhos do usuário final.

Java é uma linguagem de programação desenvolvida pela Sun Microsystems. Diferente do JavaScript, Java é compilado em bytecodes para mais tarde ser executado com o uso de um interpretador conhecido com Java Virtual Machine (JVM). Uma vez compilado, o código-fonte de um programa Java é impossível de ser entendido, mesmo, visualizando-o como um arquivo texto.

A linguagem JavaScript possibilita realizar muitas tarefas importantes para o enriquecimento de uma página web. Da validação de dados à exibição de mensagens e efeitos dinâmicos, tudo é possível. Nos dias atuais, programadores JavaScript são tão valorizados quanto programadores de linguagens server-side. Experimente visitar os

sites de algumas grandes empresas e portais e veja como os webmasters dessas páginas empregam a linguagem na medida certa para facilitar a navegação e interação com os visitantes.

À medida que nosso conhecimento e domínio sobre uma determinada linguagem de programação aumentam, é possível visualizar soluções para praticamente todos os problemas envolvendo cálculos e processamento de informações. (SILVA, 2003)

6.2.10. XML

XML significa Extensible Markup Language (Linguagem de Marcação Extensível). Linguagens de marcação compreendem todas aquelas que possibilitam a formatação de elementos por meio de tags e atributos como o HTML e XHTML. Por ser extensível, o XML possibilita a criação de elementos, ou seja, você mesmo pode inventar as suas tags. Eis aqui um dos fatores que a tornam a linguagem preferida na transação e armazenamento de dados. Você pode criar uma série de elementos e, por meio de um acordo, todas as pessoas ou empresas que estejam usando os mesmos dados poderão adotar o mesmo vocabulário e assim facilitar o processo de interpretação das informações para os mais variados sistemas envolvidos.

Um dos usos mais difundidos do XML é o armazenamento e transação de dados entre empresas. Daí o termo B2B (Business To Business). Imagine uma dessas empresas usando um sistema desenvolvido em Java outra usando um sistema em ASP ou PHP e assim por diante. O XML nestes casos é usado para armazenar ou transferir os dados necessários em cada transação e, no momento oportuno, estas linguagens fazem novamente a transformação dos dados para as suas devidas aplicações.

Outra maneira bem interativa de suar o XML é a utilização da linguagem para exibir dados diretamente no browser. Desta maneira nós vamos combinar o JavaScript, HTML e CSS para possibilitar ao usuário várias maneiras diferentes de visualizar e manipular as informações recebidas.

Uma das principais diferenças entre o HTML e o XML é que o HTML é usado para formatação e exibição de informações, enquanto o XML é usado para descrever e armazenar essas informações. Vamos ver isso de uma maneira mais clara.

Vamos começar a nossa análise com uma página web que tem como objetivo a exibição de uma lista de preços de produtos de informática. Sem levar em consideração o uso de ASP, CGI, PHP, etc. para a montagem dinâmica do documento, teríamos o webmaster do site escrevendo uma tabela com mais de 400 itens. Imagine o trabalho que isso acarreta. Agora imagine a atualização do site.

Bem, toda esta tarefa poderia ser realizada simplesmente usando um arquivo XML, com os dados da lista de preço e, por meio de técnicas, ligar esse arquivo XML à página HTML e deixar que ele se encarregue de montar a tabela dinamicamente. Faça a alteração dos preços no arquivo XML e a página refletirá essas mudanças automaticamente.

E, com o uso de XSL (chamado agora de XSLT), Xpath, DOM e outras tecnologias mais, o usuário será capaz de classificar os dados de diversas maneiras, filtrar, efetuar cálculos, etc. Isso tudo em uma mesma seção do browser, sem ter que esperar novas solicitações ao servidor. Agora, se você usa uma linguagem que monte as suas páginas de maneira dinâmica, direto do servidor, a combinação desta linguagem com o XML trará novos recursos para a visualização de seus documentos. (SILVA, 2001)

6.2.11. UML

Na década de 80, um número crescente de organizações começou a usar OOP para programar suas aplicações e surgiu a necessidade de um processo adequado para tratar de OOAD. Muitos metodologistas – incluindo Booch, Rumbaugh e Jacobson – produziram e promoveram individualmente processos separados para satisfazer esta necessidade. Cada um destes processos tinha sua própria notação, ou “linguagem” (sob a forma de diagramas gráficos), para comunicar os resultados da análise e projeto.

No início da década de 90, empresas diferentes, e até mesmo divisões diferentes de uma mesma empresa, usavam processos e notações distintas. Além disso, estas empresas queriam usar ferramentas de software que suportassem seus processos particulares. Com tantos processos e notação padronizados.

Em 1994, James Rumbaugh juntou-se a Grady Booch na Rational Software

Corporation e os dois começaram a trabalhar para unificar seus populares processos. Em seguida, juntou-se a eles Ivar Jacobson. Em 1996, o grupo liberou versões preliminares da UML, para a comunidade de engenharia de software e pediu realimentação. Mais ou menos na mesma época, uma organização conhecida como Object Management Group (OMG) solicitou propostas para uma linguagem comum de modelagem. O OMG é uma organização sem fins lucrativos que promove o uso da tecnologia de orientação a objetos, publicando diretrizes e especificações para tecnologias orientadas a objetos. Diversas operações – entre elas HP, IBM, Microsoft, Oracle e Rational Software – já haviam reconhecido a necessidade de uma linguagem de modelagem comum. Estas empresas constituíram a UML Partners em resposta à citação de propostas do OMG. Este consórcio desenvolveu e enviou a versão 1.1 da UML para o OMG. O OMG aceitou a proposta e, em 1997, assumiu a responsabilidade pela manutenção e revisão continuada da UML. Em 2001, o OMG liberou a versão 1.4 da UML e está trabalhando na versão 2.0 (com liberação prevista para 2002).

A Unified Modeling Language é agora o esquema de representação gráfica mais amplamente utilizado para modelagem de sistemas orientados a objetos. Ela certamente unificou os diversos esquemas de notação mais populares. Aqueles que projetam sistemas que usam a linguagem (sob a forma de digramas gráficos) para modelar seus sistemas.

Uma característica atraente da UML é sua flexibilidade. A UML é extensível e independente dos muitos processos da OOAD. Os modeladores UML ficam livres para desenvolver sistemas com diversos processos, mas todos, os desenvolvedores podem agora expressar tais sistemas com um conjunto-padrão de notações.

A tecnologia de orientação a objetos está em toda a parte no setor de software e a UML esta rapidamente ficando assim. (DEITEL, 2003)

A idéia central de usar UML é capturar os detalhes significantes de um sistema de modo que o problema seja claramente compreendido, a arquitetura da solução seja desenvolvida e a implementação escolhida seja claramente identificada e construída.

Uma notação rica para modelar visualmente os sistemas de software facilita este exercício. A UML não só fornece a notação para os blocos de construção básicos, como também fornece maneiras para expressar as relações complexas entre os blocos

de construção básicos.

As relações podem ser estáticas ou dinâmicas por natureza. As relações estáticas giram basicamente em torno dos aspectos estruturais de um sistema. A relação de herança entre um par de classes, as interfaces implementadas por uma classe e a dependência em outra classe são exemplos de relações estáticas.

As relações dinâmicas, por outro lado, estão preocupadas com o comportamento de um sistema e assim existem durante a execução. As mensagens trocadas em um grupo de classes para cumprir alguma responsabilidade e o fluxo de controle em um sistema, por exemplo, são capturados no contexto das relações dinâmicas que existem em um sistema.

Os aspectos estáticos e dinâmicos de um sistema são capturados na forma de diagramas UML. Há vários tipos de diagramas UML. Eles são organizados em áreas focais específicas da modelagem visual chamadas de exibições.

Os seguintes tipos de diagramas são fornecidos pela UML:

- ❖ Diagrama de caso de uso: um diagrama de caso de uso mostra os casos de uso, os atores e suas relações. Os diagramas de caso de uso capturam as exigências precisas para o sistema da perspectiva de um usuário.
- ❖ Diagrama de classe: um diagrama de classe mostra as relações estáticas que existem entre um grupo de classes e interfaces no sistema. Alguns tipos de relação comuns são de herança, de agregação e de dependência.
- ❖ Diagrama de objeto: um diagrama de objeto fornece uma visão instantânea das relações que existem entre as instancias da classe em um dado ponto no tempo. Um diagrama de objeto é útil para capturar e mostrar, de uma maneira estática, as relações complexas e dinâmicas do sistema.
- ❖ Diagrama do gráfico do estado: as máquinas do estado são excelentes para capturar o comportamento dinâmico do sistema. São particularmente aplicáveis nos sistemas reativos baseados em eventos ou objetos nos quais a ordem dos eventos é importante. Os gráficos de estado são úteis para modelar o comportamento das interfaces.
- ❖ Diagrama de atividade: um diagrama de atividade é uma extensão de um diagrama do gráfico do estado e é parecido em conceito com um fluxograma. Um diagrama de atividade permite modelar o comportamento do

sistema em termos de interação ou fluxo de controle entre as atividades distintas ou objetos. Os diagramas de atividade são melhor usados para modelar os fluxos do trabalho e o fluxo nas operações.

Diagrama de interação: os diagramas de interação são usados para modelar o comportamento dinâmico de um sistema. Há dois tipos de diagramas de interação na UML:

- ❖ Diagrama de seqüência: usado para modelar a troca de mensagens entre os objetos em um sistema. Os diagramas de seqüência também capturam a ordem das mensagens trocadas com tempo relativo.
- ❖ Diagrama de colaboração: a troca de mensagens é capturada no contexto das relações estruturais gerais entre os objetos.

Os dois diagramas são equivalentes e é possível converter um no outro facilmente. Os diagramas de interação são comumente usados para modelar o fluxo de controle em um caso de uso e para descrever como os objetos interagem durante a execução de uma operação como, por exemplo, a realização de uma operação de interface.

- ❖ Diagrama de componente: um componente representa a manifestação física de uma parte do sistema, como um arquivo, um executável, etc. Um diagrama do componente mostra as dependências e as relações entre os componentes que compõem um sistema. Um componente mapeia uma ou mais classes, subsistemas, etc.

- ❖ Diagrama de distribuição: um diagrama de distribuição mostra a arquitetura de um sistema de perspectiva dos nós, processadores e relações entre eles. Um ou mais componentes geralmente mapeiam um nó na distribuição. No contexto do J2EE, os diagramas de distribuição são úteis para modelar e desenvolver a arquitetura do sistema distribuído. (AHMED, 2002)

6.3. PADRÕES UTILIZADOS

6.3.1. MVC

Um dos principais padrões que utilizaremos para construção de nossas

aplicações é o padrão MVC ou Modelo-Vista-Controlador (do inglês Model-View-Controller). Este padrão separa três formas distintas de funcionalidades em uma aplicação. O modelo representa a estrutura de dados e operações que atuam nestes dados. Em uma aplicação orientada a objetos, constitui as classes de objetos da aplicação que implementam o que quer que a aplicação tenha que fazer. Visões implementam exclusivamente a lógica de apresentação dos dados em um formato apropriado para os usuários. A mesma informação pode ser apresentada de maneiras diferentes para grupos de usuários com requisitos diferentes. Um controlador traduz ações de usuários (movimento e click de mouse, teclas de atalho, etc) juntamente com os valores de entrada de dados em chamadas à funções específicas no modelo. Além disso, seleciona, baseado nas preferências do usuário e estado do modelo, a vista apropriada. Essencialmente, estas três camadas abstraem: dados e funcionalidades, apresentação e comportamento de uma aplicação.

6.3.2. VALUE OBJECT – OBJETOS DE VALOR

São classes que contêm os dados da aplicação. São classes Java que possuem seus atributos `private` e seus métodos `get` e `set` `public`. Essas classes são idênticas aos `Form's`, mas elas não estendem a classe `org.apache.struts.action.ActionForm`. Na verdade, elas são espelhos das tabelas no nosso banco de dados com os membros da classe com o mesmo tipo de dados dos atributos na tabela. No SPARTACUS estou utilizando também `TO's`, que são `VO's` utilizados para transferência de objetos.

6.3.3. BUSINESS OBJECT – OBJETOS DE NEGÓCIOS

São classes que fazem a comunicação entre as ações e o modelo (`Action's` e `DAO`). Essas classes são espelhos do `DAO`, sem é claro, qualquer tipo de comunicação com o banco de dados. Essa camada recebe um objeto de transferência (`TO`) da camada de Ação e envia-o para a camada de Modelo

6.3.4. DATA ACCESS OBJECT – OBJETO DE ACESSO A DADOS

Centraliza o serviço de persistência de objetos em um pequeno conjunto de classes, evitando, por exemplo, que o código SQL se espalhe pelo código da solução.

São classes que fazem a comunicação com o banco de dados relacional. Essas classes não possuem atributos, no SPARTACUS elas estendem uma classe abstrata de conexão com o banco de dados chamada *com.nonato.spartacus.util.JDBCBase*, recebem como parâmetros objetos de transferência (TO) e retornam um boolean, uma Collection ou um Form.

7.

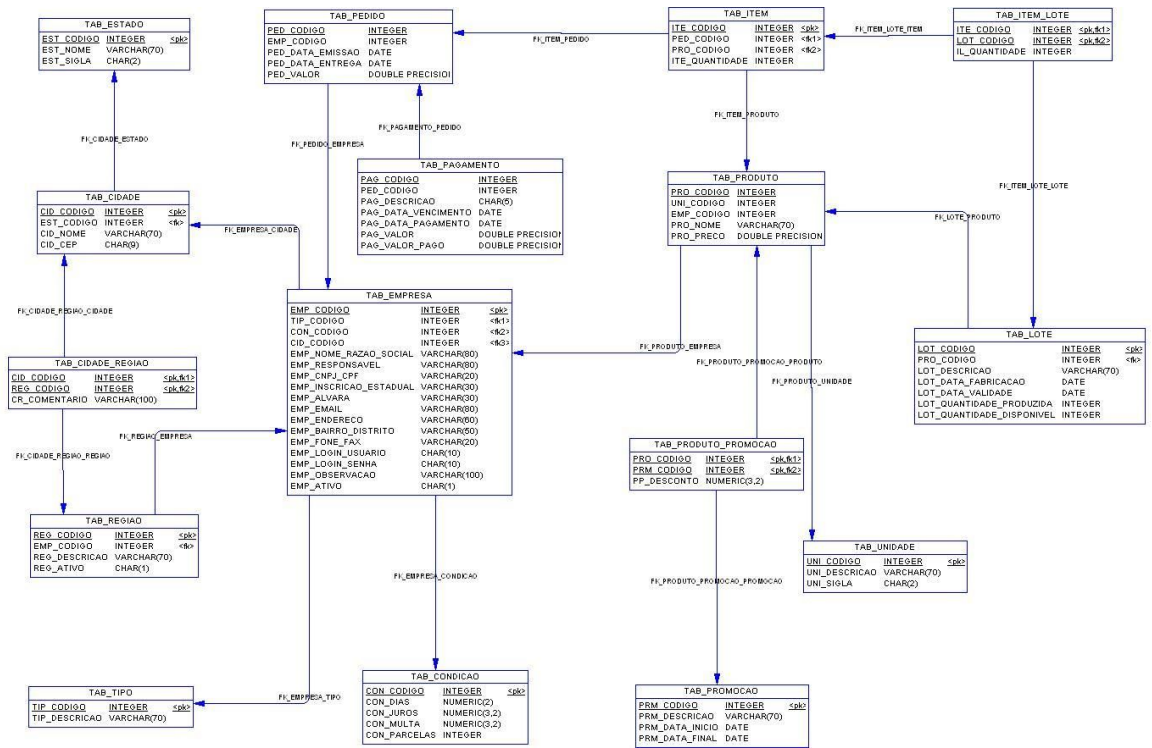
DOCUMENTAÇÃO DA ANÁLISE

7.1. REGRAS DE NEGÓCIO

- ❖ Cidade pode ser de somente um Estado e Estado pode ter nenhuma ou várias Cidades;
- ❖ Cidade pode estar em várias Regiões e Regiões podem ser constituídas por várias Cidades;
- ❖ Empresa pode ser de somente uma Cidade e Cidade pode ter nenhuma ou várias Empresas;
- ❖ Empresa pode ser de somente um Tipo e Tipo pode ter nenhuma ou várias Empresas;
- ❖ Empresa pode ter somente uma Condição e Condição pode ter nenhuma ou várias Empresas;
- ❖ Empresa pode não ter feito nenhum Pedido, ou vários, e Pedido pode ser feito somente por uma Empresa;
- ❖ Pedido tem no mínimo um Item e no máximo vários e o Item pode ser somente de um Pedido;
- ❖ O Item pode ser no máximo de um Produto e o Produto pode ser de vários Itens ou mesmo de nenhum;
- ❖ O Produto podem ter nenhum ou vários Lotes e os Lotes podem ser somente de um Produto;
- ❖ O Produto pode ser fabricado somente por uma Empresa (Laboratório) e a Empresa pode fabricar nenhum ou vários Produtos;
- ❖ O Produto pode ter somente um tipo de Unidade e Unidade pode ter nenhum ou vários Produtos;
- ❖ O Item pode ter nenhum ou vários Lotes e o Lote pode ter nenhum ou vários Itens;
- ❖ O Produto pode estar em nenhuma ou várias Promoções e a Promoção pode ter vários Produtos;

7.2.

MODELO FÍSICO DE DADOS

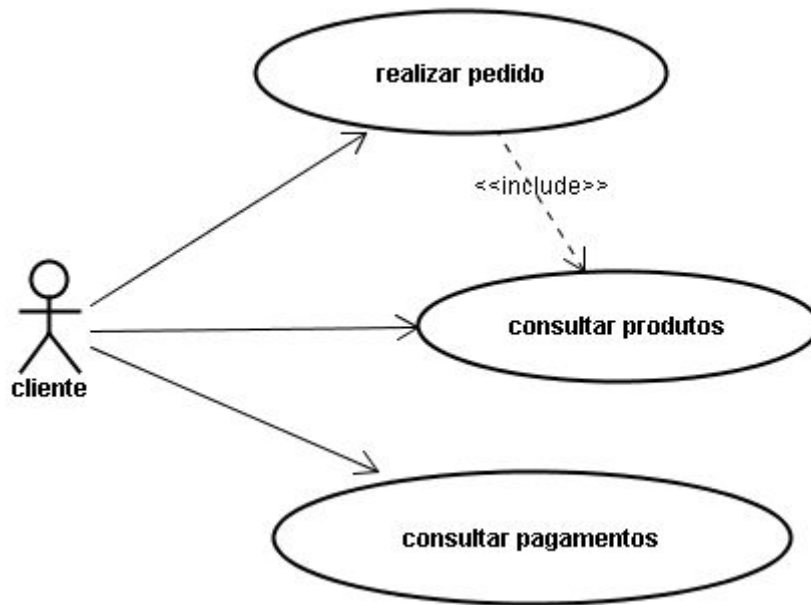


7.4.

CASOS DE USO

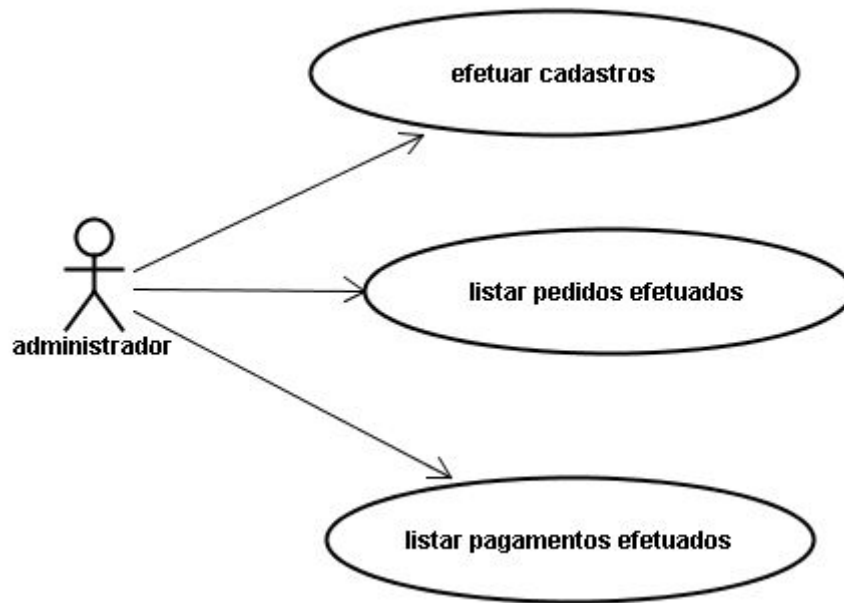
7.4.1.

CLIENTE



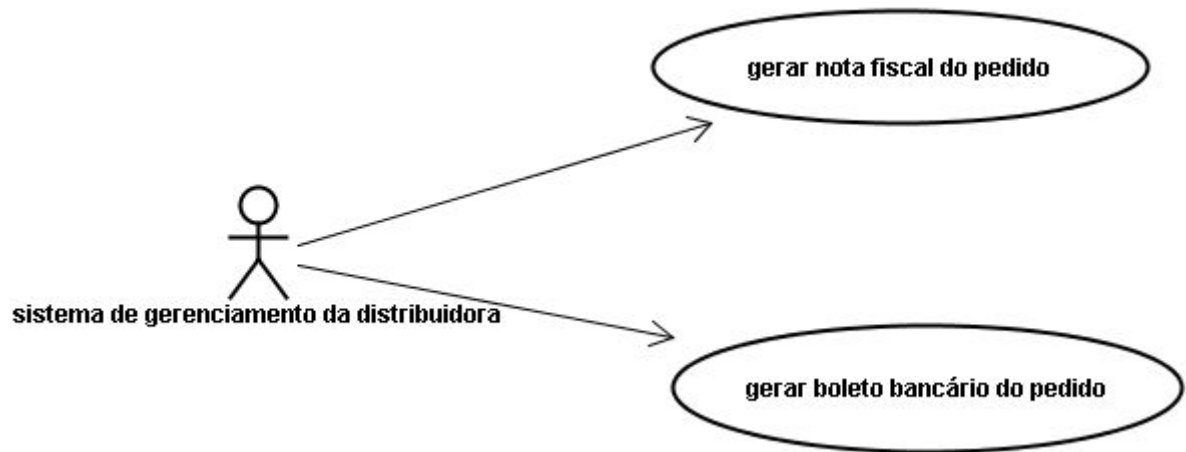
7.4.2.

ADMINISTRADOR



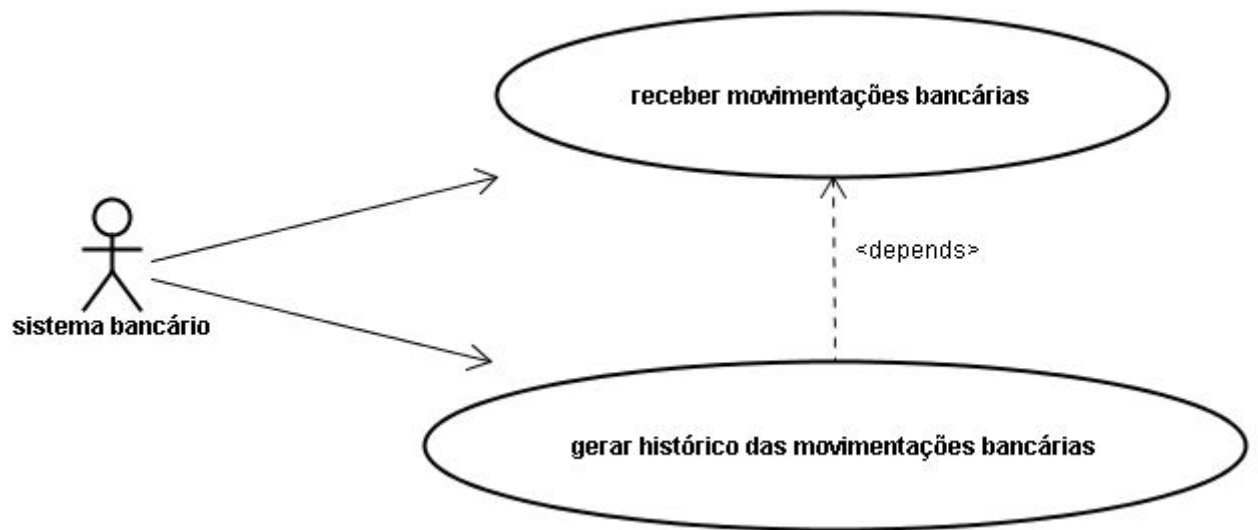
7.4.3.

SISTEMA DE GERENCIAMENTO DA DISTRIBUIDORA



7.4.4.

SISTEMA DE GERENCIAMENTO BANCÁRIO

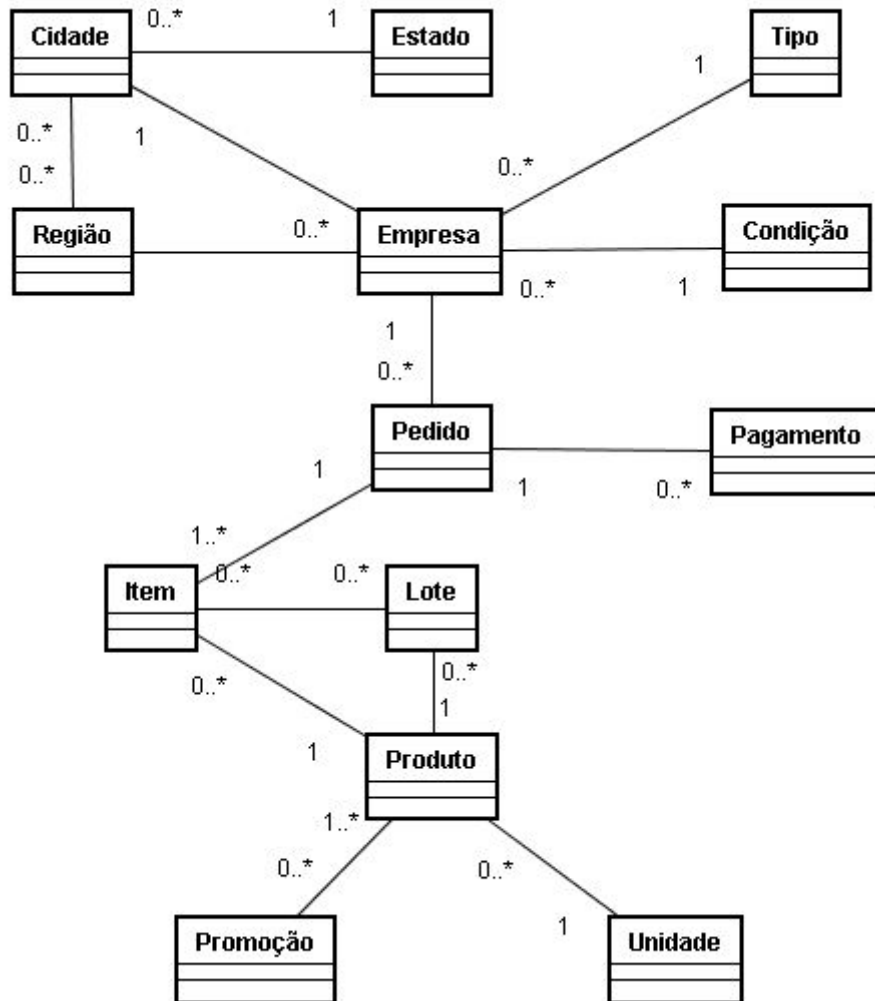


7.5.

DIAGRAMA DE CLASSES

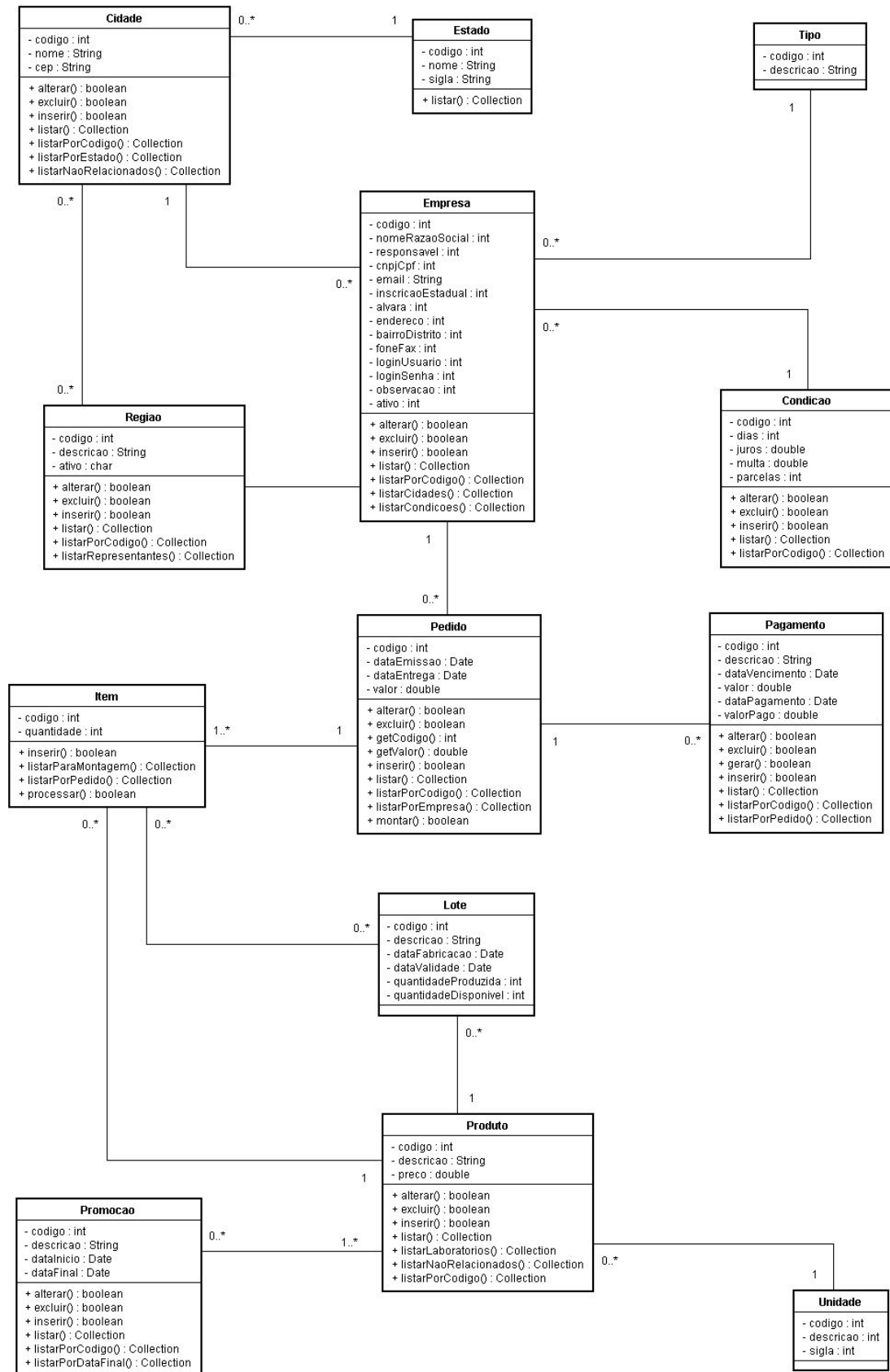
7.5.1.

ANÁLISE



7.5.2.

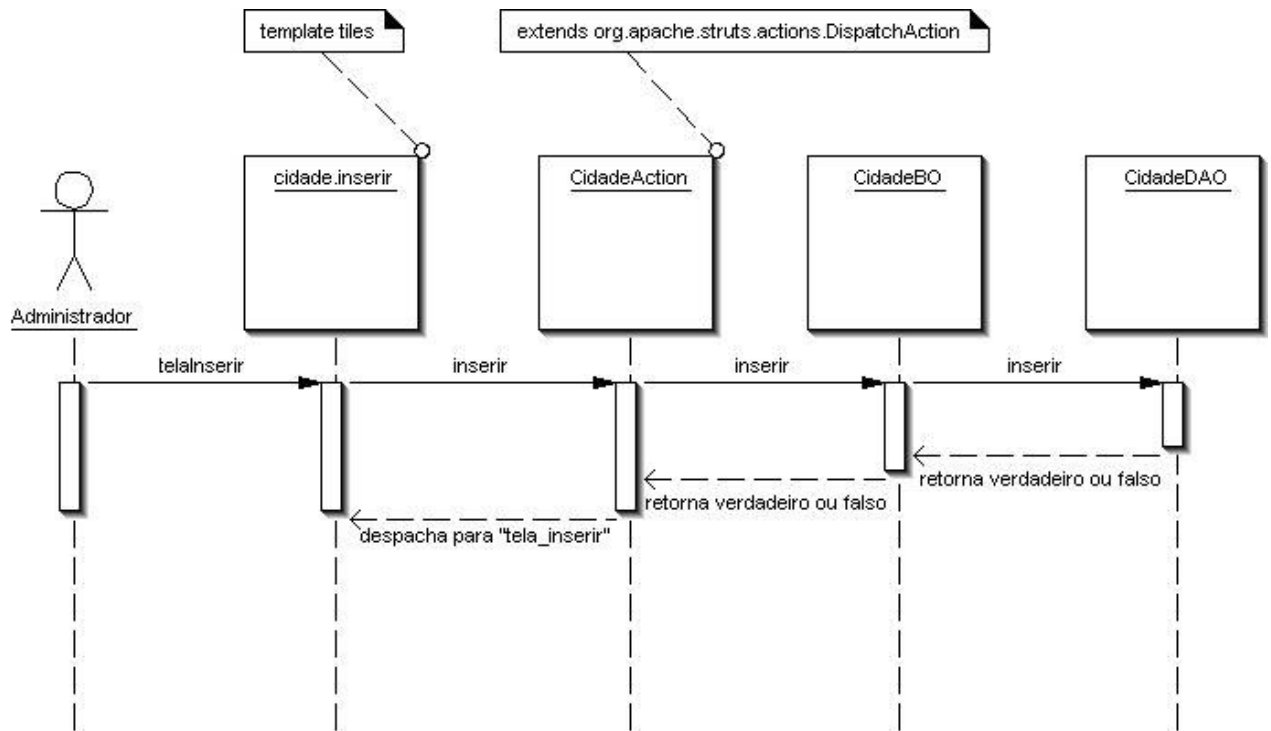
PROJETO

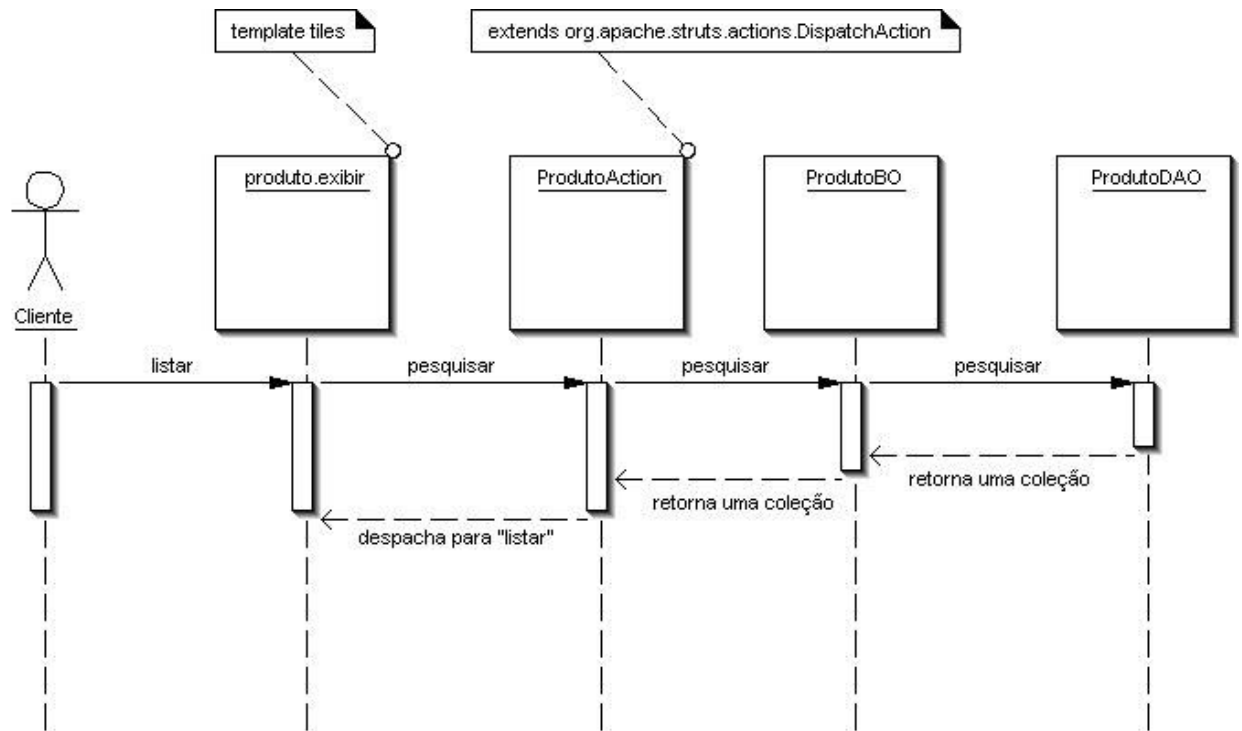


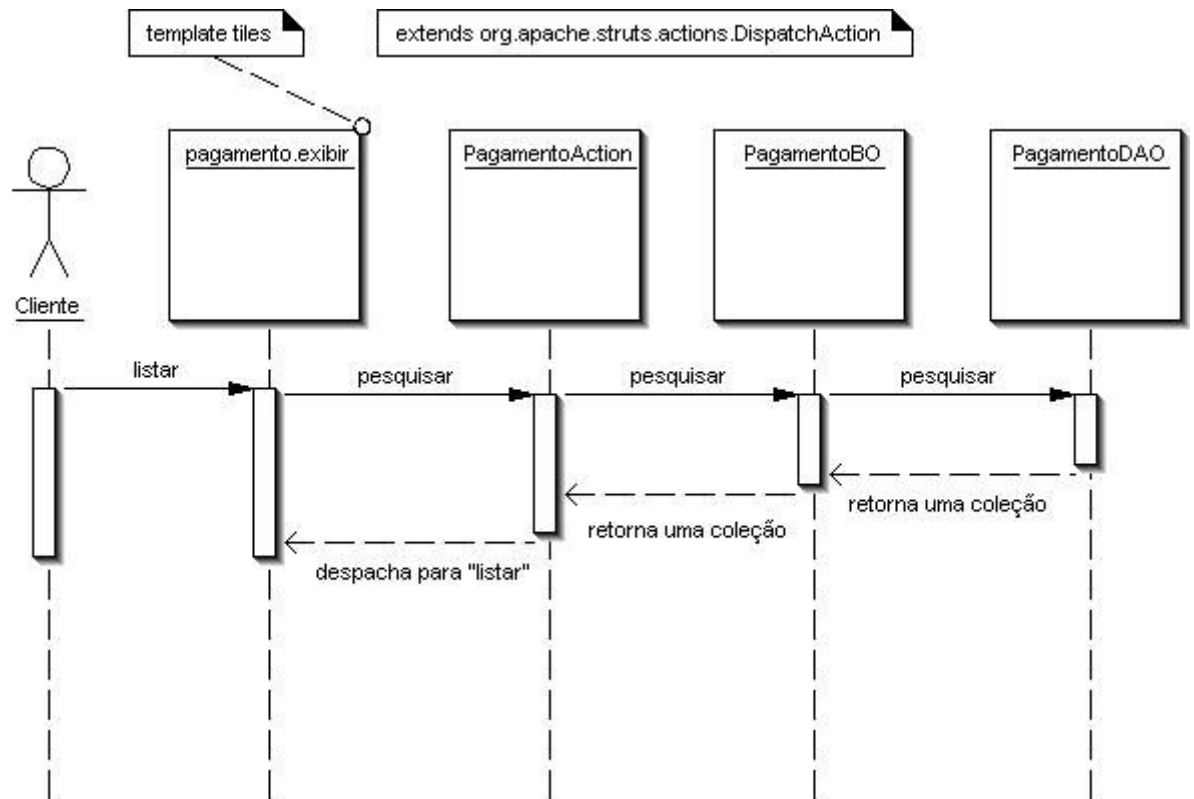
7.6.

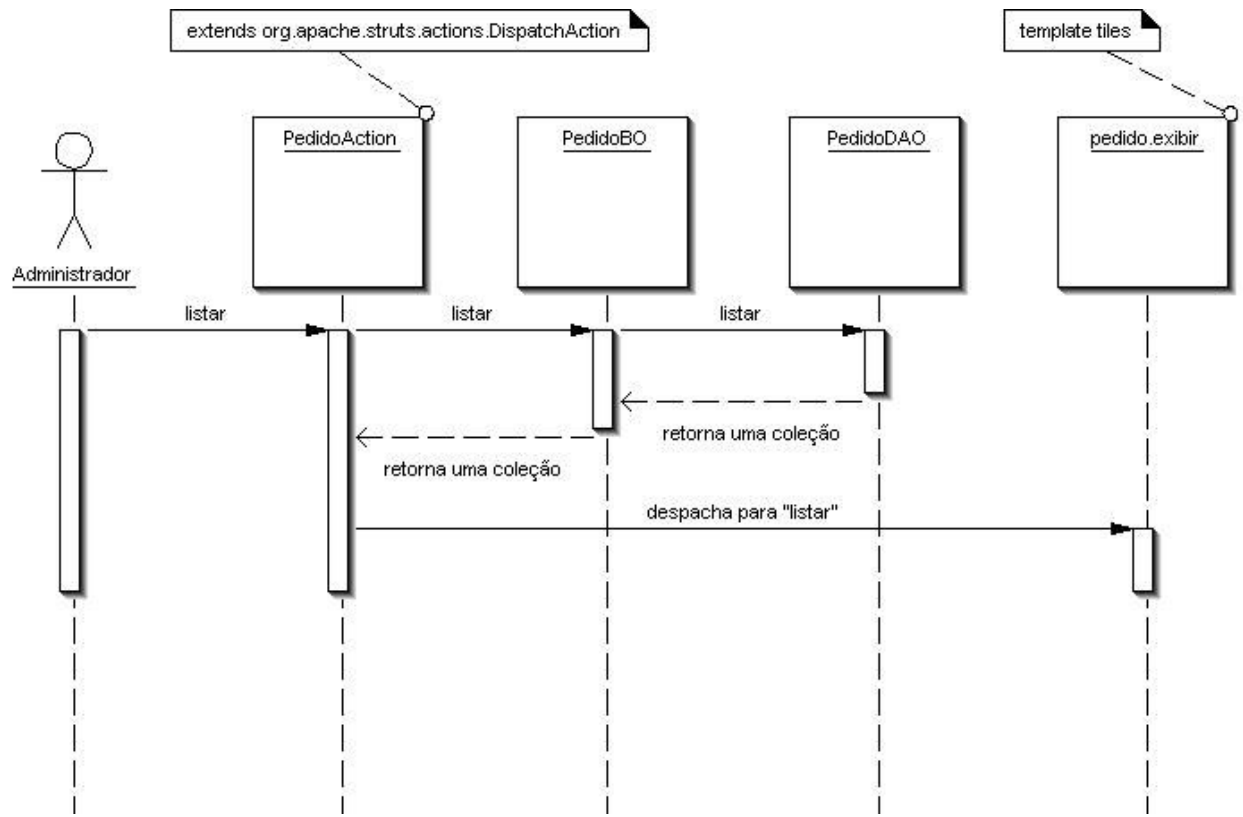
DIAGRAMAS DE SEQUÊNCIA

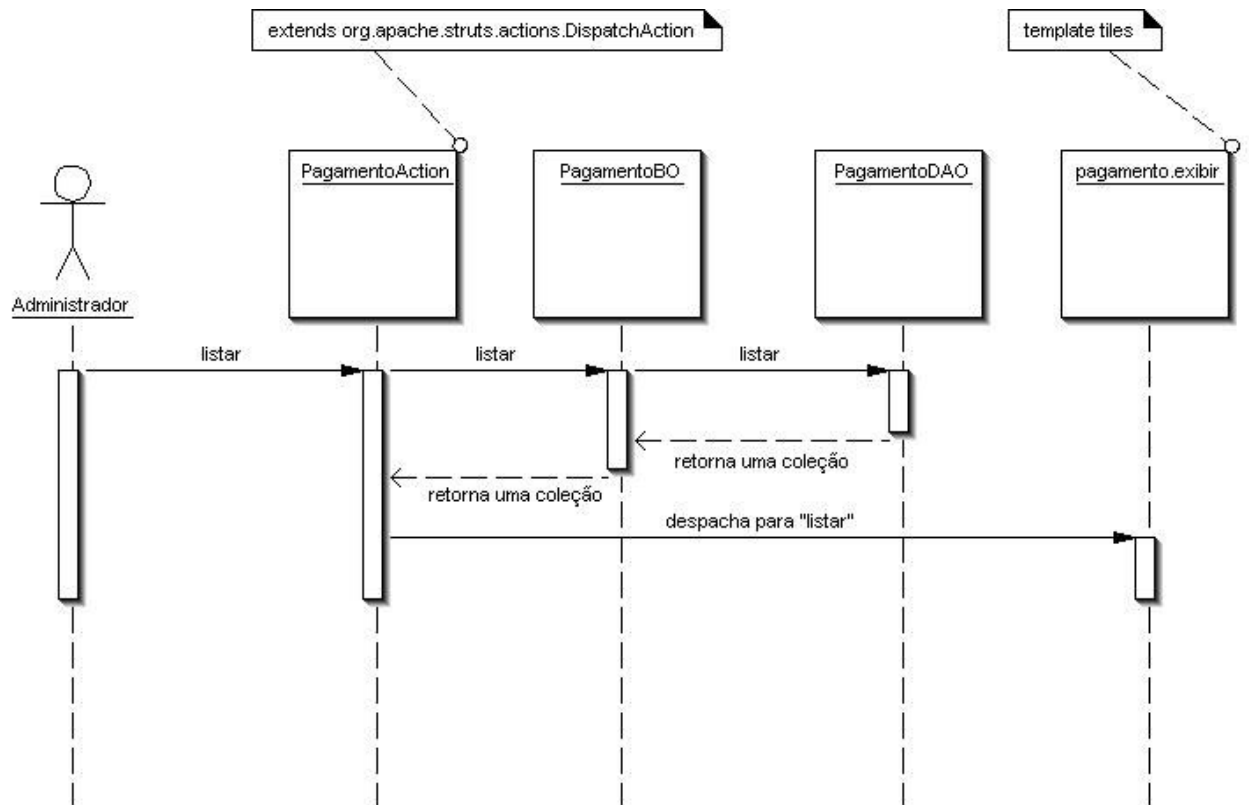
7.6.1

EFETUAR CADASTROS**7.6.2**

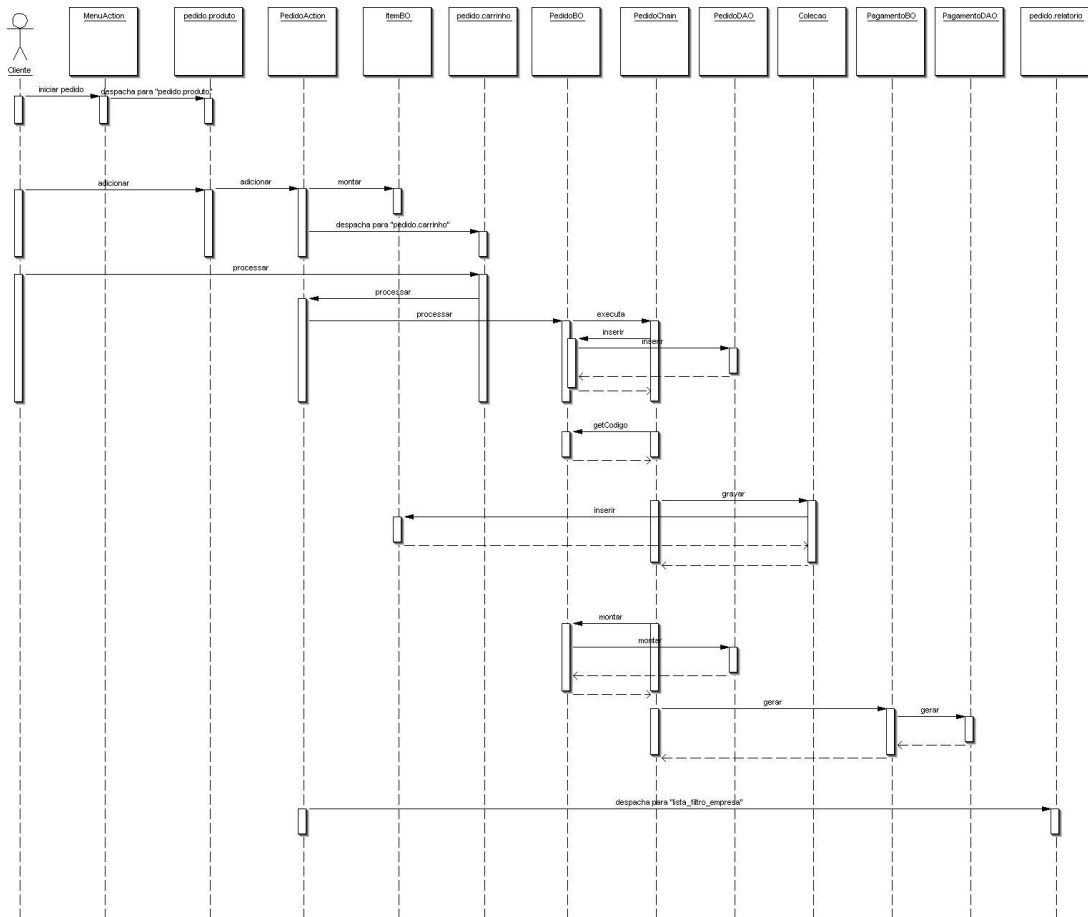
CONSULTAR PRODUTOS**7.6.3**

CONSULTAR PAGAMENTOS**7.6.4**

LISTAR PEDIDOS EFETUADOS**7.6.5**

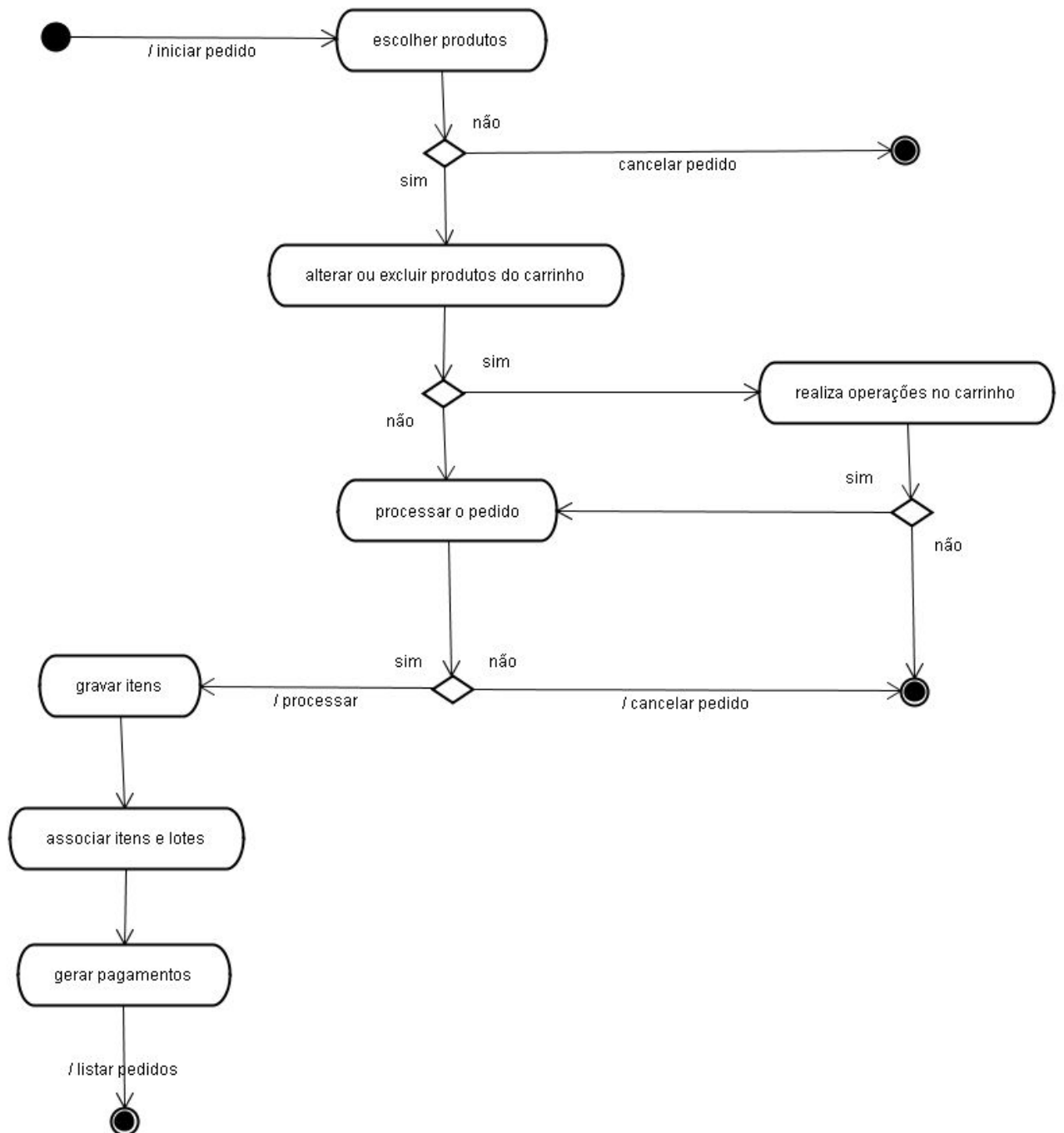
LISTAR PAGAMENTOS EFETUADOS**7.6.6**

REALIZAR PEDIDO



7.7.

DIAGRAMA DE ATIVIDADE



8.

MODELO FÍSICO

8.1.

DIAGRAMA HIERÁRQUICO DE MÓDULOS (MENUS)

8.1.1 MENU CLIENTE

● Iniciar Pedido ● Consultar Produtos ● Listar Promoções ● Listar Pagamentos

8.1.2 MENU ADMINISTRADOR

Cadastros	Pesquisas	Relatórios
Cidades	Cidades	Nota Fiscal
Condições	Condições	
Empresas	Empresas	
Lotes	Lotes	
Produtos	Pagamentos	
Promoções	Produtos	
Regiões	Promoções	
	Regiões	

8.2.

DESCRIÇÃO DAS INTERFACES (TELAS)

8.2.1

LOGIN



Usuário

Senha

Conectar

ELOFAR



8.2.2

PRINCIPAL



SPARTACUS

Cadastros | Pesquisas | Relatórios | Gráficos



ELOFAR

O projeto SPARTACUS é uma pesquisa financiada pelo Laboratório Farmacêutico Elofar, através do seu representante no noroeste de Minas: Raimundo Nonato dos Santos.

Esse projeto busca descobrir novas alternativas de mercado para o laboratório distribuir os seus produtos, com maiores facilidades e comodidades para os seus clientes.

O SPARTACUS é um projeto open-source, desenvolvido na plataforma Java, procurando utilizar as melhores práticas do mercado.

Para maiores informações: org_spartacus@yahoo.com.br

Nonato Consultoria e Sistemas S.A. - Spartacus 1.0 - Enterprise Web Suite
Copyright 2005/2006 - Todos os direitos reservados

11/12/2005



Seja bem vindo, Fábio

Desconectar

8.2.3 CADASTROS

CIDADE



SPARTACUS

Cadastros	Pesquisas	Relatórios	Gráficos
-----------	-----------	------------	----------

Novo	Limpar	Gravar	Alterar	Excluir	Abrir	Imprimir	Sair
------	--------	--------	---------	---------	-------	----------	------

Cadastro de Cidades

Código	Nome	Cep	Estado
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text" value="ACRE"/>

Nonato Consultoria e Sistemas S.A. - Spartacus 1.0 - Enterprise Web Suite Copyright 2005/2006 - Todos os direitos reservados	11/12/2005	 	Seja bem vindo, Fábio	Desconectar
---	------------	---	-----------------------	--------------------

CONDIÇÃO



SPARTACUS

Cadastros	Pesquisas	Relatórios	Gráficos				
Novo	Limpar	Gravar	Alterar	Excluir	Abrir	Imprimir	Sair

Cadastro de Condições

Código	Quantidade de Dias	Valor de Juros (Porcentagem)	Valor da Multa (Porcentagem)	Quantidade de Parcelas


Nonato Consultoria e Sistemas S.A. - Spartacus 1.0 - Enterprise Web Suite
Copyright 2005/2006 - Todos os direitos reservados

11/12/2005

  Seja bem vindo, Fábio

Desconectar



EMPRESA


SPARTACUS

Cadastros | Pesquisas | Relatórios | Gráficos



Novo **Limpar** **Gravar** **Alterar** **Excluir** **Abrir** **Imprimir** **Sair**

Cadastro de Empresas (Clientes, Laboratórios e Representantes)

Código Nome/Razão Social		Responsável	Tipo	CNPJ/CPF
<input type="text"/>		<input type="text"/>	ADMINISTRADOR	<input type="text"/>
Endereço		Bairro/Distrito	Fone/Fax	
<input type="text"/>		<input type="text"/>	<input type="text"/>	
Cidade	Nome	Estado	Cep	
 <input type="text"/>	<input type="text"/>	MG	<input type="text"/>	
Email	Alvará	Inscrição Estadual	Usuário	Senha
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Condição	Quantidade de Dias	Valor de Juros (Porcentagem)	Valor da Multa (Porcentagem)	Quantidade de Parcelas
 <input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Observação				
<input type="text"/>				

Nonato Consultoria e Sistemas S.A. - Spartacus 1.0 - Enterprise Web Suite
Copyright 2005/2006 - Todos os direitos reservados

11/12/2005

  Seja bem vindo, Fábio

Desconectar

LOTE



SPARTACUS

Cadastros	Pesquisas	Relatórios	Gráficos
-----------	-----------	------------	----------

Novo	Limpar	Gravar	Alterar	Excluir	Abrir	Imprimir	Sair
-------------	---------------	---------------	----------------	----------------	--------------	-----------------	-------------

Cadastro de Lotes

Código	Descrição	Data de Fabricação	Data de Validade	Quantidade Produzida	Quantidade Disponível

Produto

Nome

Nonato Consultoria e Sistemas S.A. - Spartacus 1.0 - Enterprise Web Suite
Copyright 2005/2006 - Todos os direitos reservados

11/12/2005

Seja bem vindo, Fábio

Desconectar

PRODUTO



SPARTACUS

Cadastros	Pesquisas	Relatórios	Gráficos
-----------	-----------	------------	----------

Novo	Limpar	Gravar	Alterar	Excluir	Abrir	Imprimir	Sair
------	--------	--------	---------	---------	-------	----------	------

Cadastro de Produtos

Código	Nome	Preço	Unidade	Laboratório	Nome
			CAIXA		

Nonato Consultoria e Sistemas S.A. - Spartacus 1.0 - Enterprise Web Suite
Copyright 2005/2006 - Todos os direitos reservados

11/12/2005

Seja bem vindo, Fábio

Desconectar

PROMOÇÃO


SPARTACUS

Cadastros | Pesquisas | Relatórios | Gráficos

Novo | **Limpar** | **Gravar** | **Alterar** | **Excluir** | **Abrir** | **Imprimir** | **Sair**

Cadastro de Promoções

Código	Descrição	Data do Início	Data do Término
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Nonato Consultoria e Sistemas S.A. - Spartacus 1.0 - Enterprise Web Suite
Copyright 2005/2006 - Todos os direitos reservados

11/12/2005

  Seja bem vindo, Fábio

Desconectar


SPARTACUS

Cadastros | Pesquisas | Relatórios | Gráficos

Adicionar | **Processar** | **Alterar** | **Excluir** | **Abrir** | **Imprimir** | **Sair**

Código	Descrição	Data do Início	Data do Término
1	Semana Maluca	12/12/2005	16/12/2005

Produto	Nome	Preço	Desconto	Valor	Alterar	Excluir
9	Ciprofar 250 mg c/ 14 Comp.	19.0	10.0	17.1		

1

Nonato Consultoria e Sistemas S.A. - Spartacus 1.0 - Enterprise Web Suite
Copyright 2005/2006 - Todos os direitos reservados

11/12/2005

  Seja bem vindo, Fábio

Desconectar

REGIÃO


SPARTACUS

Cadastros | Pesquisas | Relatórios | Gráficos

Novo | Limpar | Gravar | Alterar | Excluir | Abrir | Imprimir | Sair

Cadastro de Regiões

Código	Descrição	Representante	Nome
	<input type="text"/>	<input type="text"/>	<input type="text"/>

Nonato Consultoria e Sistemas S.A. - Spartacus 1.0 - Enterprise Web Suite
Copyright 2005/2006 - Todos os direitos reservados | 11/12/2005 |   | Seja bem vindo, Fábio | **Desconectar**


SPARTACUS

Cadastros | Pesquisas | Relatórios | Gráficos

Adicionar | Processar | Alterar | Excluir | Abrir | Imprimir | Sair

Código	Descrição	Ativo	Representante
1	Noroeste de Minas	Sim	

Código	Nome	Cep	Comentário	Alterar	Excluir
1	Unaí-MG	38610-000			

1

Nonato Consultoria e Sistemas S.A. - Spartacus 1.0 - Enterprise Web Suite
Copyright 2005/2006 - Todos os direitos reservados | 11/12/2005 |   | Seja bem vindo, Fábio | **Desconectar**

8.2.4

PESQUISAS

CIDADE


SPARTACUS

Cadastros | Pesquisas | Relatórios | Gráficos

Novo **Abrir** **Pesquisar** **Imprimir** **Sair**


Código Nome Cep Estado


ACRE

Código	Nome	Cep	Estado	Alterar	Excluir
2	Florianópolis	11111-111	SC		
1	Unai	38610-000	MG		

1

Nonato Consultoria e Sistemas S.A. - Spartacus 1.0 - Enterprise Web Suite
Copyright 2005/2006 - Todos os direitos reservados

11/12/2005   Seja bem vindo, Fábio **Desconectar**


SPARTACUS

Iniciar Pedido **Consultar Produtos** **Listar Promoções** **Listar Pagamentos**

Pesquisar **Imprimir** **Sair**



Código Nome Preço Unidade

CAIXA

Código	Nome	Preço	Unidade	Laboratório
1	Atossion Ped.	0.0	VD	Elofar
2	Atossion Xarope	0.0	VD	Elofar
5	Bacfar 100 ml Susp.	0.0	VD	Elofar
4	Bacfar 50 ml Susp.	0.0	CX	Elofar
3	Bacfar c/ 20 comp.	0.0	CX	Elofar

1 2 3 4 5 6 7

Nonato Consultoria e Sistemas S.A. - Spartacus 1.0 - Enterprise Web Suite
Copyright 2005/2006 - Todos os direitos reservados

11/12/2005   Seja bem vindo, Raimundo **Desconectar**

CONDIÇÃO



Código	Quantidade de Dias	Valor de Juros (Porcentagem)	Valor da Multa (Porcentagem)	Quantidade de Parcelas	Alterar	Excluir
1	30	5.0	0.0	1		

1

Nonato Consultoria e Sistemas S.A. - Spartacus 1.0 - Enterprise Web Suite
Copyright 2005/2006 - Todos os direitos reservados

11/12/2005

Seja bem vindo, Fábio

Desconectar

EMPRESA



Códigos	Pesquisas	Relatórios	Gráficos
<div>Novo Abrir Pesquisar Imprimir Sair</div>			
Código	Nome/Razão Social	Responsável	Tipo CNPJ/CPF
			ADMINISTRADOR <input type="text"/>

Código	Nome/Razão Social	Tipo	Cidade	CNPJ/CPF	Alterar	Excluir
1	Elofar	FORNECEDOR	Florianópolis-SC	11111111111		
3	Farmácia Unai	CLIENTE	Unai-MG	33333333333		
2	Nonato	ADMINISTRADOR	Unai-MG	22222222222		

1

Nonato Consultoria e Sistemas S.A. - Spartacus 1.0 - Enterprise Web Suite
Copyright 2005/2006 - Todos os direitos reservados

11/12/2005

Seja bem vindo, Fábio

Desconectar

LOTE



Códigos	Pesquisas	Relatórios	Gráficos					
<div>Novo Abrir Pesquisar Imprimir Sair</div>								
Código	Descrição	Data de Fabricação	Data de Validade	Produto				
Código	Descrição	Data de Fabricação	Data de Validade	Quantidade Produzida	Quantidade Disponível	Produto	Alterar	Excluir
1	1501	10/12/2005	10/12/2008	1000	980	Ciprofar 250 mg c/ 14 Comp.		

1

Nonato Consultoria e Sistemas S.A. - Spartacus 1.0 - Enterprise Web Suite
Copyright 2005/2006 - Todos os direitos reservados

11/12/2005

Seja bem vindo, Fábio

Desconectar

PAGAMENTO


SPARTACUS

Cadastros | Pesquisas | Relatórios | Gráficos

Novo **Abrir** **Pesquisar** **Imprimir** **Sair**

Código	Descrição	Data de Vencimento	Valor	Data do Pagamento	Valor Pago

Código	Descrição	Data de Vencimento	Valor	Data do Pagamento	Valor Pago	Cliente	Boleto Bancário	Alterar	Excluir
1	1/1	09/01/2006	380.0		0.0	Farmácia Unai			

1

Nonato Consultoria e Sistemas S.A. - Spartacus 1.0 - Enterprise Web Suite
Copyright 2005/2006 - Todos os direitos reservados

11/12/2005

  Seja bem vindo, Fábio

Desconectar


SPARTACUS

☒ Iniciar Pedido ☐ Consultar Produtos ☐ Listar Promoções ☐ Listar Pagamentos

Pesquisar **Imprimir** **Sair**

Código	Descrição	Data de Vencimento	Valor	Data do Pagamento	Valor Pago

Código	Descrição	Data de Vencimento	Valor	Data do Pagamento	Valor Pago	Cliente
1	1/1	09/01/2006	380.0		0.0	Farmácia Unai

Nonato Consultoria e Sistemas S.A. - Spartacus 1.0 - Enterprise Web Suite
Copyright 2005/2006 - Todos os direitos reservados

11/12/2005

  Seja bem vindo, Raimundo

Desconectar

PRODUTO



Códigos	Pesquisas	Relatórios	Gráficos
<div>Novo Abrir Pesquisar Imprimir Sair</div>			
Código	Nome	Preço	Unidade Laboratório
			CAXA

Código	Nome	Preço	Unidade	Laboratório	Alterar	Excluir
1	Atossion Ped.	0.0	VD	Elofar		
2	Atossion Xarope	0.0	VD	Elofar		
5	Bacfar 100 ml Susp.	0.0	VD	Elofar		
4	Bacfar 50 ml Susp.	0.0	CX	Elofar		
3	Bacfar c/ 20 comp.	0.0	CX	Elofar		

1 2 3 4 5 6 7

Nonato Consultoria e Sistemas S.A. - Spartacus 1.0 - Enterprise Web Suite
Copyright 2005/2006 - Todos os direitos reservados

11/12/2005

Seja bem vindo, Fábio

Desconectar

PROMOÇÃO


SPARTACUS

Cadastros | Pesquisas | Relatórios | Gráficos

Novo | Abrir | Pesquisar | Imprimir | Sair

Código Descrição Data do Início Data do Término

Código	Descrição	Data do Início	Data do Término	Alterar	Excluir	Produtos
1	Sexta-Maluca	16/12/2005	16/12/2005			

1

Nonato Consultoria e Sistemas S.A. - Spartacus 1.0 - Enterprise Web Suite
Copyright 2005/2006 - Todos os direitos reservados

11/12/2006   Seja bem vindo, Fábio

Desconectar

REGIÃO



SPARTACUS

Cadastros | Pesquisas | Relatórios | Gráficos

Novo

Abrir

Pesquisar

Imprimir

Sair

Código

Descrição

Representante

--	--	--

Código	Descrição	Ativo	Representante	Alterar	Excluir	Cidades
1	Noroeste de Minas	Sim				

1

Nonato Consultoria e Sistemas S.A. - Spartacus 1.0 - Enterprise Web Suite
Copyright 2005/2006 - Todos os direitos reservados

11/12/2005



Seja bem vindo, Fábio

Desconectar

PEDIDO


SPARTACUS

☐ Iniciar Pedido
 ☐ Consultar Produtos
 ☐ Listar Promoções
 ☐ Listar Pagamentos

Código	Nome	Representante	CNPJ/CPF
3	Farmácia Unai Raimundo Nonato	333333333333	

Código	Nome	Preço	Unidade
			CAIXA

Código	Nome	Preço	Unidade	Quantidade
1	Atossion Ped.	0.0	VD	
2	Atossion Xarope	0.0	VD	
5	Bacfar 100 ml Susp.	0.0	VD	
4	Bacfar 50 ml Susp.	0.0	CX	
3	Bacfar c/ 20 comp.	0.0	CX	

1 2 3 4 5 6 7

Nonato Consultoria e Sistemas S.A. - Spartacus 1.0 - Enterprise Web Suite
 Copyright 2005/2006 - Todos os direitos reservados

11/12/2005

Seja bem vindo, Raimundo


SPARTACUS

☐ Iniciar Pedido
 ☐ Consultar Produtos
 ☐ Listar Promoções
 ☐ Listar Pagamentos

Código	Nome	Representante	CNPJ/CPF
3	Farmácia Unai Raimundo Nonato	333333333333	

Código	Nome	Preço	Quantidade	Quantidade	Excluir
9	Ciprofar 250 mg c/ 14 Comp.	19.0	10	10	<input checked="" type="checkbox"/>
11	Ciprofar 500 mg c/ 14 Comp.	23.0	10	10	<input checked="" type="checkbox"/>

1

Nonato Consultoria e Sistemas S.A. - Spartacus 1.0 - Enterprise Web Suite
 Copyright 2005/2006 - Todos os direitos reservados

11/12/2005

Seja bem vindo, Raimundo


SPARTACUS

☐ Iniciar Pedido
 ☐ Consultar Produtos
 ☐ Listar Promoções
 ☐ Listar Pagamentos

Código	Nome	Representante	CNPJ/CPF
3	Farmácia Unai Raimundo Nonato	333333333333	

Código	Nome	Preço	Quantidade
9	Ciprofar 250 mg c/ 14 Comp.	19.0	10
11	Ciprofar 500 mg c/ 14 Comp.	23.0	10

1

Nonato Consultoria e Sistemas S.A. - Spartacus 1.0 - Enterprise Web Suite
 Copyright 2005/2006 - Todos os direitos reservados

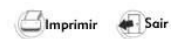
11/12/2005

Seja bem vindo, Raimundo

8.3.

DESCRIÇÃO DOS RELATÓRIOS

PEDIDO



Dados da Empresa

Código	Nome/Razão Social	Responsável	CNP.J/CPF
3	Farmácia Unaf	Raimundo Nonato	33333333333

Dados do Pedido

Código	Data de Emissão	Data da Entrega	Valor
1	10/12/2005		380.0

Dados do Pagamento

Código	Descrição	Data de Vencimento	Valor	Data do Pagamento	Valor Pago
1	1/1	09/01/2006	380.0		0.0

Dados Pedidos

Item	Produto	Nome	Preço	Quantidade Pedida
1	9	Ciprofar 250 mg c/ 14 Comp.	19.0	20

Dados Processados

Item	Produto	Preço	Quantidade Pedida	Lote	Descrição	Quantidade Atendida
1	Ciprofar 250 mg c/ 14 Comp.	19.0	20	1	1501	20

8.4.

DESCRIÇÃO ALGORÍTMICA DAS OPERAÇÕES

8.4.1.

INCLUIR REGISTROS

Chamar tela de inserir;
Preencher os dados;
Capturar dados no controlador;
Converter dados em objetos de transferência;

Chamar objeto de negócios passando um objeto de transferência como parâmetro e recebendo um valor *boolean* (verdadeiro ou falso) como retorno e jogar na variável operação;

Se operação então
 Preenche fila de mensagens
Senão
 Preenche fila de erros

Despacha para a tela de inserir (com a devida fila preenchida);

8.4.2.

ALTERAR OU EXCLUIR REGISTROS

Chamar tela de exibir registros;
Localizar o registro na tabela;
Selecionar o registro;
Despachar para tela de alteração de dados (ou exclusão);
Alterar os dados;
Capturar dados no controlador;
Converter dados em objetos de transferência;

Chamar objeto de negócios passando um objeto de transferência como parâmetro e recebendo um valor *boolean* (verdadeiro ou falso) como retorno e jogar na variável operação;

Se operação então
 Preenche fila de mensagens
Senão
 Preenche fila de erros

Despachar para a tela de alterar (com a devida fila preenchida);

8.4.3.

RECUPERAR REGISTROS (PESQUISAR)

Chama tela de exibir registros;
Preencher os dados (formulário de pesquisa);
Capturar os dados no controlador;
Converter dados em objetos de transferência;

Chama objeto de negócios passando um objeto de transferência como parâmetro e recebendo uma coleção de objetos;

Setar coleção de objetos na requisição;
Despachar para tela de exibir registros;

8.4.4.

GERAR PAGAMENTO

Selecionar código da empresa, a data de emissão e o valor do pedido passando como parâmetro o código do pedido;

Selecionar condição de pagamento da empresa passando como parâmetro o código da empresa;

Selecionar quantidade de dias e parcelas passando como parâmetro o código da condição;

Calcular valor parcelado: dividindo o valor do pedido pela quantidade de parcelas;

Armazenar data do primeiro vencimento;

Enquanto o contador é menor que a quantidade de parcelas faz

Configurar descrição do pagamento;

Gravar registro no banco;

Aumentar data de vencimento;

Incrementar contador;

Fim Enquanto

8.4.5.

MONTAR PEDIDO

Selecionar itens do pedido passando o código do pedido como parâmetro;

Selecionar lote com a data em dia e com o produto disponível passando como parâmetro o código do produto e a data corrente;

Se (quantidade restante maior que zero) então sair;

Se (quantidade restando for menor ou igual a quantidade disponível) então

Quantidade atendida recebe a quantidade restante;

Quantidade restante igual à zero;

Fim Se

Senão Se (quantidade restante maior que quantidade disponível) então

Quantidade atendida recebe quantidade disponível;

Quantidade restante recebe quantidade restante menos a quantidade disponível;

Fim Senão

Gravar dados na tabela Item_Lote;

9.

DICIONÁRIO DE DADOS

9.1. LISTA DE DOMÍNIOS

Nome	Código	Tipo de Dado
dm_codigo	DM_CODIGO	I
dm_nome_razao_social	DM_NOME_RAZAO_SOCIAL	VA80
dm_cnpj_cpf	DM_CNPJ_CPF	VA20
dm_endereco	DM_ENDERECO	VA60
dm_bairro_distrito	DM_BAIRRO_DISTRITO	VA50
dm_fone_fax	DM_FONE_FAX	VA20
dm_data	DM_DATA	D
dm_dinheiro	DM_DINHEIRO	MN
dm_sim_nao	DM_SIM_NAO	A1
dm_descricao	DM_DESCRICAO	VA70
dm_sigla	DM_SIGLA	A2
dm_quantidade	DM_QUANTIDADE	I
dm_cep	DM_CEP	A9
dm_inscricao_estadual	DM_INSCRICAO_ESTADUAL	VA30
dm_alvara	DM_ALVARA	VA30
dm_observacao	DM_OBSERVACAO	VA100
dm_porcentagem	DM_PORCENTAGEM	N3,2
dm_dias	DM_DIAS	N2
dm_pagamento	DM_PAGAMENTO	A5
dm_login	DM_LOGIN	A10
dm_email	DM_EMAIL	VA80

9.2. LISTA DE ITENS DE DADOS

Nome	Código	Tipo de Dado	Tamanho	Precisão	Domínio
est_codigo	EST_CODIGO	I			dm_codigo
est_nome	EST_NOME	VA70	70		dm_descricao
est_sigla	EST_SIGLA	A2	2		dm_sigla
cid_codigo	CID_CODIGO	I			dm_codigo
cid_nome	CID_NOME	VA70	70		dm_descricao
cid_cep	CID_CEP	A9	9		dm_cep
ped_codigo	PED_CODIGO	I			dm_codigo

ped_data_emissao	PED_DATA_EMISSAO	D			dm_data
ped_data_entrega	PED_DATA_ENTREGA	D			dm_data
ped_valor	PED_VALOR	MN			dm_dinheiro
pro_codigo	PRO_CODIGO	I			dm_codigo
pro_nome	PRO_NOME	VA70	70		dm_descricao
uni_codigo	UNI_CODIGO	I			dm_codigo
uni_descricao	UNI_DESCRICAO	VA70	70		dm_descricao
uni_sigla	UNI_SIGLA	A2	2		dm_sigla
lot_codigo	LOT_CODIGO	I			dm_codigo
lot_descricao	LOT_DESCRICAO	VA70	70		dm_descricao
lot_data_fabricacao	LOT_DATA_FABRICACAO	D			dm_data
lot_data_validade	LOT_DATA_VALIDADE	D			dm_data
lot_quantidade_produzida	LOT_QUANTIDADE_PRODUZIDA	I			dm_quantidade
lot_quantidade_disponivel	LOT_QUANTIDADE_DISPONIVEL	I			dm_quantidade
ite_codigo	ITE_CODIGO	I			dm_codigo
ite_quantidade	ITE_QUANTIDADE	I			dm_quantidade
con_codigo	CON_CODIGO	I			dm_codigo
con_dias	CON_DIAS	N2	2		dm_dias
prm_codigo	PRM_CODIGO	I			dm_codigo
prm_data_inicio	PRM_DATA_INICIO	D			dm_data
prm_data_final	PRM_DATA_FINAL	D			dm_data
prm_descricao	PRM_DESCRICAO	VA70	70		dm_descricao
il_quantidade	IL_QUANTIDADE	I			dm_quantidade
emp_codigo	EMP_CODIGO	I			dm_codigo
emp_nome_razao_social	EMP_NOME_RAZAO_SOCIAL	VA80	80		dm_nome_razao_social
emp_cnpj_cpf	EMP_CNPJ_CPF	VA20	20		dm_cnpj_cpf
emp_endereco	EMP_ENDERECO	VA60	60		dm_endereco

emp_bairro_distrito	EMP_BAIRRO_DISTRITO	VA50	50		dm_bairro_distrito
emp_fone_fax	EMP_FONE_FAX	VA20	20		dm_fone_fax
emp_inscricao_estadual	EMP_INSCRICAO_ESTADUAL	VA30	30		dm_inscricao_estadual
emp_alvara	EMP_ALVARA	VA30	30		dm_alvara
emp_observacao	EMP_OBSERVACAO	VA100	100		dm_observacao
tip_codigo	TIP_CODIGO	I			dm_codigo
tip_descricao	TIP_DESCRICAO	VA70	70		dm_descricao
reg_codigo	REG_CODIGO	I			dm_codigo
reg_descricao	REG_DESCRICAO	VA70	70		dm_descricao
con_juros	CON_JUROS	N3,2	3	2	dm_porcentagem
con_multa	CON_MULTA	N3,2	3	2	dm_porcentagem
con_parcelas	CON_PARCELAS	I			dm_quantidade
pag_codigo	PAG_CODIGO	I			dm_codigo
pag_descricao	PAG_DESCRICAO	A5	5		dm_pagamento
pag_data_vencimento	PAG_DATA_VENCIMENTO	D			dm_data
pag_data_pagamento	PAG_DATA_PAGAMENTO	D			dm_data
pag_valor	PAG_VALOR	MN			dm_dinheiro
pag_valor_pago	PAG_VALOR_PAGO	MN			dm_dinheiro
cr_comentario	CR_COMENTARIO	VA100	100		dm_observacao
emp_login_usuario	EMP_LOGIN_USUARIO	A10	10		dm_login
emp_login_senha	EMP_LOGIN_SENHA	A10	10		dm_login
emp_responsavel	EMP_RESPONSAVEL	VA80	80		dm_nome_razao_social
pro_preco	PRO_PRECO	MN			dm_dinheiro
pp_desconto	PP_DESCONTO	N3,2	3	2	dm_porcentagem
reg_ativo	REG_ATIVO	A1	1		dm_sim_nao
emp_ativo	EMP_ATIVO	A1	1		dm_sim_nao

emp_email	EMP_EMAIL	VA80	80		dm_email
-----------	-----------	------	----	--	----------

9.3. LISTA DE ENTIDADES

Nome	Código
cidade	CIDADE
estado	ESTADO
regiao	REGIAO
pedido	PEDIDO
produto	PRODUTO
unidade	UNIDADE
lote	LOTE
item	ITEM
condicao	CONDICAO
promocao	PROMOCAO
empresa	EMPRESA
tipo	TIPO
pagamento	PAGAMENTO

9.4. LISTA DE ATRIBUTOS DA ENTIDADE CIDADE

Nome	Código
cid_codigo	CID_CODIGO
cid_nome	CID_NOME
cid_cep	CID_CEP

9.5. LISTA DE IDENTIFICADORES DA ENTIDADE CIDADE

Nome	Código	Entidade
pk_cidade	PK_CIDADE	cidade

9.6. LISTA DE ATRIBUTOS DA ENTIDADE CONDIÇÃO

Nome	Código
con_codigo	CON_CODIGO
con_dias	CON_DIAS
con_juros	CON_JUROS
con_multa	CON_MULTA

con_parcelas	CON_PARCELAS
--------------	--------------

9.7. LISTA DE IDENTIFICADORES DA ENTIDADE CONDIÇÃO

Nome	Código	Entidade
pk_condicao	PK_CONDICA O	condicao

9.8. LISTA DE ATRIBUTOS DA ENTIDADE EMPRESA

Nome	Código
emp_codigo	EMP_CODIGO
emp_nome_razao_social	EMP_NOME_RAZAO_SOCIAL
emp_responsavel	EMP_RESPONSAVEL
emp_cnpj_cpf	EMP_CNPJ_CPF
emp_inscricao_estadual	EMP_INSCRICAO_ESTADUAL
emp_alvara	EMP_ALVARA
emp_email	EMP_EMAIL
emp_endereco	EMP_ENDERECO
emp_bairro_distrito	EMP_BAIRRO_DISTRITO
emp_fone_fax	EMP_FONE_FAX
emp_login_usuario	EMP_LOGIN_USUARIO
emp_login_senha	EMP_LOGIN_SENHA
emp_observacao	EMP_OBSERVACAO
emp_ativo	EMP_ATIVO

9.9. LISTA DE IDENTIFICADORES DA ENTIDADE EMPRESA

Nome	Código	Entidade
pk_empresa	PK_EMPRESA	empresa

9.10. LISTA DE ATRIBUTOS DA ENTIDADE ESTADO

Nome	Código
est_codigo	EST_CODIGO
est_nome	EST_NOME
est_sigla	EST_SIGLA

9.11. LISTA DE IDENTIFICADORES DA ENTIDADE ESTADO

Nome	Código	Entidade
pk_estado	PK_ESTADO	estado

9.12. LISTA DE ATRIBUTOS DA ENTIDADE ITEM

Nome	Código
ite_codigo	ITE_CODIGO
ite_quantidade	ITE_QUANTIDADE

9.13. LISTA DE IDENTIFICADORES DA ENTIDADE ITEM

Nome	Código	Entidade
pk_item	PK_ITEM	item

9.14. LISTA DE IDENTIFICADORES DA ENTIDADE LOTE

Nome	Código
lot_codigo	LOT_CODIGO
lot_descricao	LOT_DESCRICAO
lot_data_fabricacao	LOT_DATA_FABRICACAO
lot_data_validade	LOT_DATA_VALIDADE
lot_quantidade_produzida	LOT_QUANTIDADE_PRODUZIDA
lot_quantidade_disponivel	LOT_QUANTIDADE_DISPONIVEL

9.15. LISTA DE IDENTIFICADORES DA ENTIDADE LOTE

Nome	Código	Entidade
pk_lote	PK_LOTE	lote

9.16. LISTA DE ATRIBUTOS DA ENTIDADE PAGAMENTO

Nome	Código
pag_codigo	PAG_CODIGO
pag_descricao	PAG_DESCRICAO
pag_data_vencimento	PAG_DATA_VENCIMENTO

pag_data_pagamento	PAG_DATA_PAGAMENTO
pag_valor	PAG_VALOR
pag_valor_pago	PAG_VALOR_PAGO

9.17. LISTA DE IDENTIFICADORES DA ENTIDADE PAGAMENTO

Nome	Código	Entidade
pk_pagamento	PK_PAGAMENTO	pagamento

9.18. LISTA DE ATRIBUTOS DA ENTIDADE PEDIDO

Nome	Código
ped_codigo	PED_CODIGO
ped_data_emissao	PED_DATA_EMISSAO
ped_data_entrega	PED_DATA_ENTREGA
ped_valor	PED_VALOR

9.19. LISTA DE IDENTIFICADORES DA ENTIDADE PEDIDO

Nome	Código	Entidade
pk_pedido	PK_PEDIDO	pedido

9.20. LISTA DE ATRIBUTOS DA ENTIDADE PRODUTO

Nome	Código
pro_codigo	PRO_CODIGO
pro_nome	PRO_NOME
pro_preco	PRO_PRECO

9.21. LISTA DE IDENTIFICADORES DA ENTIDADE PRODUTO

Nome	Código	Entidade
pk_produto	PK_PRODUTO	produto

9.22. LISTA DE ATRIBUTOS DA ENTIDADE PROMOÇÃO

Nome	Código
prm_codigo	PRM_CODIGO
prm_descricao	PRM_DESCRICAO
prm_data_inicio	PRM_DATA_INICIO
prm_data_final	PRM_DATA_FINAL

9.23. LISTA DE IDENTIFICADORES DA ENTIDADE PROMOÇÃO

Nome	Código	Entidade
pk_promocao	PK_PROMOC AO	promocao

9.24. LISTA DE ATRIBUTOS DA ENTIDADE REGIÃO

Nome	Código
reg_codigo	REG_CODIGO
reg_descricao	REG_DESCRICAO
reg_ativo	REG_ATIVO

9.25. LISTA DE IDENTIFICADORES DA ENTIDADE REGIÃO

Nome	Código	Entidade
pk_regiao	PK_REGIAO	regiao

9.26. LISTA DE ATRIBUTOS DA ENTIDADE TIPO

Nome	Código
tip_codigo	TIP_CODIGO
tip_descricao	TIP_DESCRICAO

9.27. LISTA DE IDENTIFICADORES DA ENTIDADE TIPO

Nome	Código	Entidade
pk_tipo	PK_TIPO	tipo

9.28. LISTA DE ATRIBUTOS DA ENTIDADE UNIDADE

Nome	Código
uni_codigo	UNI_CODIGO
uni_descricao	UNI_DESCRICAO
uni_sigla	UNI_SIGLA

9.29. LISTA DE IDENTIFICADORES DA ENTIDADE UNIDADE

Nome	Código	Entidade
pk_unidade	PK_UNIDADE	unidade

9.30. LISTA DE RELACIONAMENTOS

Nome	Código	Entidade 2	Entidade 1
rel_cidade_estado	REL_CIDADE_ESTADO	estado	cidade
rel_pedido_item	REL_PEDIDO_ITEM	item	pedido
rel_produto_unidade	REL_PRODUTO_UNIDADE	unidade	produto
rel_produto_lote	REL_PRODUTO_LOTE	lote	produto
rel_item_produto	REL_ITEM_PRODUTO	produto	item
rel_entidade_tipo	REL_ENTIDADE_TIPO	tipo	empresa
rel_entidade_condicao	REL_ENTIDADE_CONDICAO	condicao	empresa
rel_entidade_pedido	REL_ENTIDADE_PEDIDO	pedido	empresa
rel_entidade_produto	REL_ENTIDADE_PRODUTO	produto	empresa
rel_regiao_entidade	REL_REGIAO_ENTIDADE	empresa	regiao
rel_cidade_entidade	REL_CIDADE_ENTIDADE	empresa	cidade
rel_pedido_pagamento	REL_PEDIDO_PAGAMENTO	pagamento	pedido

10.

CONSIDERAÇÕES FINAIS

Com o desenvolvimento do SPARTACUS pretendemos solucionar o problema das distribuidoras, que trabalham com fax/tele vendas, e desejam oferecer mais uma opção de venda, pelo ambiente web.

11.

REFERÊNCIAS BIBLIOGRÁFICAS

AHMED, Khawar; UMRYSH, Cary. **Desenvolvendo Aplicações Comerciais em Java com J2EE e UML**. Rio de Janeiro: Editora Ciência Moderna Ltda., 2002.

CANTU, Carlos Henrique. **Firebird Essencial**. Rio de Janeiro: Editora Ciência Moderna, 2005.

DEITEL, Harvey e Paul. **Java como Programar**. Porto Alegre: Bookman, 2003.

DUMOULIN, Cedric. **Tiles Advanced Features**. Características Avançadas do Tiles. Disponível em <<http://www.lifl.fr/~dumoulin/tiles/>>. Capturado em Dezembro de 2005.

GOODWILL, James. **Mastering Jakarta Struts**. Indiana, Estados Unidos: Wiley Publishing, Inc., 2002.

GRECO, Maurício. **O Idioma dos Objetos**. InfoExame, julho de 2004.

HUSTED, Ted; DUMOULIN, Cedric; FRANCISCUS, George. **Struts em Ação**. Rio de Janeiro: Editora Ciência Moderna, 2004.

KING, Gavin; BAUER, Christian. **Hibernate em Ação**. Rio de Janeiro: Editora Ciência Moderna, 2005.

KURNIAWAN, Budi. **Java para a Web com Servlet, JSP e EJB**. Rio de Janeiro: Editora Ciência Moderna Ltda., 2002.

MOSCARDINI, Claudete & Rosalva. **Modelagem e Projeto de Banco de Dados com PowerDesigner**. SQL MAGAZINE. Ed. 12, p. 9-13.

RUAS, Nelson da Silva. **Criando Sites Web com Folhas de Estilo**. Florianópolis:

Visual Books Ltda., 2003.

SILVA, Osmar. **JavaScript Avançado: Animação, Interatividade e Desenvolvimento de Aplicativos**. São Paulo: Érica, 2003.

SILVA, Osmar. **XML: Aplicações Práticas**. São Paulo: Érica, 2001.