Universidade Católica de Brasília
Pró-Reitoria de Pós-Graduação e Pesquisa
Coordenação de Pós Graduação em *Lato Sensu* em Informática
Especialidade de Sistemas Orientados a Objetos

PROJETO FINAL DE PÓS-GRADUAÇÃO DE LATO SENSO EM INFORMÁTICA NA ESPECIALIDADE DE SISTEMAS ORIENTADOS A OBJETOS COM FOCO NA TECNOLOGIA JAVA.

Título: *Cia Nutri Manager - Gerenciador On-Line da Companhia da Nutrição*

Autores: *Sérgio Paulo Rodrigues de Lima*
Dílson Modesto de Mattos

Orientador: *Prof. M.Sc. Fernando Silveira Goulart Júnior*

Brasília – DF
Novembro de 2005

Universidade Católica de Brasília
Pró-Reitoria de Pós-Graduação e Pesquisa
Coordenação de Pós Graduação em *Lato Sensu* em Informática
Especialidade de Sistemas Orientados a Objetos

Título: ***Cia Nutri Manager - Gerenciador On-Line da Companhia da Nutrição***

Autores: ***Sérgio Paulo Rodrigues de Lima***
Dílson Modesto de Mattos

Orientador: ***Prof. M.Sc. Fernando Silveira Goulart Júnior***

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA DA UNIVERSIDADE CATÓLICA DE BRASÍLIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA OBTENÇÃO DO TÍTULO DE PÓS-GRADUAÇÃO EM LATO SENSO EM INFORMÁTICA NA ESPECIALIDADE DE *SISTEMAS ORIENTADOS A OBJETOS COM FOCO NA TECNOLOGIA JAVA* .

Aprovado por:

Presidente - Prof. Marcos Mota do Carmo Costa, PhD. - UCB

Orientador - Prof. Fernando Silveira Goulart Júnior, M.Sc. - UCB

Brasília – DF
Novembro de 2005

"A DEUS Uno e Trino, pelas oportunidades que ELE me deu e pelas pessoas e amigos que ELE me apresentou ao longo da minha vida". - *Sérgio*

"Dedico esta monografia aos meus pais que sempre lutaram bravamente para me dar educação e que com seus recursos e ferramentas ajudaram na formação da índole que hoje possuo e me orgulho de ter e cujos ensinamentos pretendo transferir para os meus filhos". - *Dílson*

AGRADECIMENTOS

"Neste curso, aprendi que cada indivíduo pode ser muito bom em determinadas áreas, mas ainda assim, possui muito que aprender com as experiências profissionais daqueles que o rodeiam. A frase 'sentiu-se um peixe fora d'água' não seria tão distante do meu sentimento ao longo do curso. Muitas barreiras e dificuldades foram encontradas. Muitos colegas que, por falta de tempo nunca tiveram a oportunidade de me auxiliar ao longo do curso. Cada qual possuía seus próprios anseios e suas prioridades. Antes do curso, cheguei a achar que ninguém poderia ter maiores qualidades do que eu; e que mesmo apesar de não possuir uma base sólida, somente o meu esforço seria o suficiente. Infelizmente, não foi assim. Por diversas vezes precisei recorrer a colegas de turma a fim de solicitar uma explicação. Incrível como a vida (e não só o mundo) dá voltas. Desde a primeira série do ensino fundamental, jamais havia precisado de ajuda de qualquer pessoa que fosse. Sempre fui autodidata. Sempre aprendi todas as matérias sozinho. Durante o curso veio a dúvida: 'por que minha metodologia de ensino não está funcionando neste curso'? Porque eu estava trabalhando com algo que não era como as contas de matemática que utilizam regras pré-estabelecidas que sempre dão certo. A análise de sistemas envolve algo que não tem livro que consiga repassar de uma vez para o indivíduo: experiência. Cheguei a passar por situações em que por conta de detalhes (às vezes uma vírgula), não era possível avançar nos estudos. Eu, como autodidata, lia diversas bibliografias e nada. Novas dúvidas surgiam: 'o que está dando errado? O que aconteceu com aquele indivíduo que sempre estava a frente de todos os seus companheiros de turma? Por que não mais conseguia resolver problemas 'fáceis''? Porque faltava experiência. Porque faltava alguém do lado para dizer: 'só está faltando uma vírgula no seu código', ou então: 'você tem que instanciar um objeto antes de chamar o método'. O curso me mostrou que a humildade é uma grande qualidade humana. Ajudar as pessoas sem ganhar nada em troca, é outra. Como em todas as profissões, nem todos os profissionais são exemplos de conduta ou fonte de qualidades, não posso recriminar quem não me ajudou ou se dispôs a fazê-lo. No futuro, com certeza, terei muito mais o que aprender, a cada dia, com todos. Espero que estas lições sirvam não só para mim, mas para todos aqueles que acham, ainda, que por algum motivo, são melhores que os outros. Somos iguais, com qualidades diferentes.

Agradeço a DEUS, que nunca me permitiu, sequer, pensar em desistir. ELE que sempre esteve ao meu lado, principalmente nos momentos de maior dificuldade.

Agradeço aos meus pais, que me educaram e, mesmo com dificuldades, me mantiveram na escola.

Agradeço ao meu pai, "Seu Maninho", pelo exemplo de pessoa que foi para mim em toda a minha vida. Pela sua força de vontade, que de alguma forma, está presente em mim.

Agradeço a minha mãe, "Dna. Deusimar", que suportou toda a minha ignorância ao longo de toda a minha vida estudantil, reclamando dos barulhos dentro de casa, que me impediam de me concentrar nos estudos. Com seu carinho, ela sempre soube lidar com as situações. Ela sabia a importância dos meus estudos para a minha vida.

Agradeço aos meus irmãos, Luiz e **Jules Rimet**, que me deram suporte para chegar aonde cheguei. Agradeço por terem sido exemplos de vida para mim e de me instruírem, por serem pessoas responsáveis e batalhadoras e por jamais terem se envolvido com irregularidades que pudessem me contaminar. Obrigado pelo carinho que eles tiveram comigo ao longo da minha vida.

Agradeço a minha irmã, Lucimar, que me ensinou uma das maiores lições que já tive em minha vida: "neste mundo competitivo, não adianta ser bom, você tem que procurar ser sempre o melhor". Depois de ouvir isto dela, deixei de ser o "aluno bagunceiro" da primeira série, com notas baixas (MM), para ser o aluno que recebeu apenas uma nota MM (em Física, mas o professor teve alguma influência neste resultado) em toda a sua vida escolar, no ensino fundamental e médio. Todas as outras notas, em todas as matérias foram MS ou SS. Agradeço porque com isso, consegui um feito na minha vida escolar, onde na quarta série do ensino fundamental: durante todo o ano letivo, só recebi nota SS, em todas as provas. Sem esta lição, com certeza não teria modificado meu comportamento e não teria tido a presente oportunidade.

Agradeço a minha noiva, Gracielly, que sempre me estimulou a realizar os meus sonhos e sempre esteve ao meu lado, em todos os momentos.

Agradeço, de alguma forma, ao Corpo de Bombeiros Militar do Distrito Federal, que me deu suporte e conhecimento para poder participar deste curso.

Agradeço a todos os meus colegas que me incentivaram a iniciar e terminar o curso. Em especial, ao meu amigo **Otelo**, pela sua incrível disposição em me ajudar.

Agradeço ao meu colega Alexandre, que me indicou o curso.

Agradeço ao "Japa" (Emílio Numazaki) e ao seu irmão Kenta, que por muitas vezes foram incomodados pela minha pessoa e, tiraram muitas dúvidas.

Agradeço ao meu colega Wesley Marques, que nos momentos em que eu estava perdido quanto ao projeto, ele me deu uma direção.

Agradeço ao Professor M.Sc., Orientador, mestre e amigo Fernando Silveira Goulart Júnior que de alguma forma, acreditou que a conclusão do curso e do projeto seria possível. Sem o seu entendimento, com certeza, hoje eu não teria chegado ao final do curso, ou melhor, sequer teria começado. Agradeço pela grande lição que aprendi neste curso e por sua paciência ao longo da orientação.

Agradeço ao Professor PhD Marcos Mota do Carmo Costa que mesmo tendo um conhecimento muito superior ao nosso, de alunos, explicava as dúvidas mais simples. Com certeza, um dos maiores responsáveis pela minha oportunidade de realizar este curso. Agradeço por ter sido, sempre, a pessoa simples que é.

Agradeço ao Professor M.Sc. Rodrigo Bonifácio de Almeida, pela sua paciência e disposição em tentar repassar aquilo que julgava ser o mais importante para os alunos.

Agradeço a Professora M.Sc. Leila de Fátima Carvalho, que muito trabalho teve comigo, para ensinar e tirar minhas dúvidas, principalmente de *Struts* e *EJB*. Agradeço por sua disposição e, por muitas vezes, ter ficado após as aulas tirando minhas dúvidas".

Sérgio Lima

"Agradeço a Deus em primeiro lugar por nos dar a oportunidade de estarmos vivos e realizando os nossos objetivos e sonhos, nos dando forças para vencer todas as dificuldades e chegarmos ao final com sucesso.

Aos meus familiares que souberam entender e apoiar esta minha jornada estando sempre comigo para me auxiliar nos momentos mais difíceis de minha vida e me incentivaram na realização de meus sonhos.

A meu orientador Fernando Goulart pela sua competência, aos professores Marcos Mota e Rodrigo Bonifácio pela sua dedicação em transmitir seus conhecimentos durante todo o curso de Pós-Graduação e a todos os meus amigos de turma, principalmente o Sérgio que faz parte dessa conquista".

Dílson Mattos

"Não são os mais fortes da espécie que sobrevivem, nem os mais inteligentes, mas sim os que respondem melhor às mudanças".

Charles Darwin

RESUMO

O presente trabalho apresenta uma proposta de modernização da empresa Companhia da Nutrição, localizada na Estação Rodoviária de Brasília, onde os dados gerenciais e de frente de loja estarão disponíveis de forma rápida e precisa para todos os usuários do sistema, de acordo com as suas funções. Foi modelado um protótipo de um sistema de gerenciamento de vendas através de uma aplicação *web* para a empresa que aumenta a confiabilidade e a agilidade na troca de informações. O sistema irá fornecer dados que possibilitarão aos gerentes verificar a situação do estoque da empresa, consultar produtos, cadastrar clientes, realizar vendas e transferência de estoque.

Para a realização do protótipo, foram utilizadas ferramentas livres: a linguagem *UML* para a modelagem visual da aplicação; propostas do *eXtreming Programing – XP*, para o desenvolvimento; o ***e-Gen Developer***, como ferramenta, que é um ambiente *RAD (Rapid Application Development)* que proporciona um rápido desenvolvimento de aplicações para *web*, totalmente escrito em Java e baseado no *framework Jakarta Struts*; e banco de dados PostgreSQL, que foi modelado na ferramenta DbWrench, também escrita em Java.

Palavras-chaves: internet, gerenciamento, ferramentas livres.

ABSTRACT

The present work presents a proposal of modernization of the company *"Companhia da Nutrição"* (Company of the Nutrition), located in the Bus Station of Brasilia, where the management data and of store front will be available of fast form and need for all the users the system, in accordance with its functions. Web for the company was shaped an archetype of a system of management of sales through an application who increases the trustworthiness and the agility in the exchange of information. The system will go to supply given that they will make possible to the controlling to verify the situation of the supply of the company, to consult products, to register in cadastre customers, to carry through sales and transference of supply. For the accomplishment of the archetype, free tools had been used: language UML for the visual modeling of the application; proposals of eXtreming Programing - XP, for the development; **e-Gen Developer**, as tool, that is an environment RAD (Rapid Application Development) that it provides a fast development of applications for web, total written in Java and based in framework Jakarta Struts; and data base PostgreSQL, that was shaped in the DbWrench tool, also written in Java.

Word-keys: internet, management, free software.

SUMÁRIO

Pág.

Lista de Abreviaturas 1
Lista de Figuras 3
CAPÍTULO 1 - INTRODUÇÃO 4
1. Introdução 4
1.1. Motivação da Pesquisa 4
1.2. Objetivo Geral 6
1.3. Objetivos Específicos 6
1.4. Fronteiras da Pesquisa 7
1.5. Estrutura do Trabalho 7
CAPÍTULO 2 – NOTAÇÃO E METODOLOGIA 8
2. Introdução 8
2.1. UML – Unified Modeling Language (Linguagem de Modelagem Unificada) 9
2.2. Metodologias de Software ou Processos de Software 12
2.2.1. Metodologias Tradicionais ou Pesadas 13
2.2.2. Metodologias Ágeis 13
2.2.2.1. Extreme Programming - XP 14
CAPÍTULO 3 – TECNOLOGIAS, MODELOS E PADRÕES DE PROJETO 19
3.1. JAVA – Informações Gerais Sobre a Linguagem 19
3.2. J2EE - Java 2 Platform Enterprise Edition 20
3.2.1. - O Modelo de Programação J2EE 21
3.2.2. - Tecnologias utilizadas no J2EE 22
3.2.2.1 - Servlet 22
3.2.2.1.1 - Arquitetura de um Servlet 22
3.2.2.2 - JSP - Java Server Pages 23
3.2.2.3 - Java JDBC - Java DataBase Connectivity 24
3.2.2.3.1 - Acesso ao Banco de Dados 24
3.2.2.4 - JavaBeans 24
3.2.3 - Considerações sobre a J2EE 25
3.2.4 - Tomcat - *Servlet Container* 25
3.3 - XML (Extensible Markup Language) 26
3.4 - Padrões de Projeto (*Design Patterns*) 27
3.4.1 - Pattern MVC - *Model-View-Controller* 28
3.5 - Frameworks 30
3.5.1 - Struts - Apache Struts Framework 31
3.5.1.1 - Motivos para utilizar o Struts Framework 31
3.5.2 - e-GEN Developer – FÁCIL GERAÇÃO 32
3.6 - Banco de Dados PostgreSQL 33
3.7 - DbWrench - Modelagem de Diagramas de Entidade-Relacionamento em Java 34
CAPÍTULO 4 – ANÁLISE E DESENVOLVIMENTO DO PROJETO 36
4.1 – Introdução 36
4.2 – A análise 37
4.2.1 – Análise de Requisitos 38
4.2.2 – Diagrama de Casos de Uso 40
4.2.3 – Diagramas de seqüência 41
4.2.4.– Diagramas de classe 44
4.3 – Projeto 46
4.3.1. – Login 46

4.3.2. – Menu 47
4.3.2.1. – Menu Principal 48
4.3.2.1.1. – Sub-menu Pesquisa Funcionário 48
4.3.2.1.2. – Sub-Menu Pesquisa Produto 49
4.3.2.1.3. – Sub-Menu Pesquisa Cliente 50
4.3.2.1.4. – Sub-Menu Cadastro de Clientes 50
4.3.2.1.5. – Sub-Menu Formulário de Vendas 51
4.3.2.2. – Menu Auxiliar 52
CAPÍTULO 5 - CONCLUSÃO 53
5.1 – Conclusões 53
5.1.1 – Dificuldades 56
5.2 – Restrições 57
5.3 – Trabalhos Futuros 58
5.4 – Considerações finais 59
Referências Bibliográficas 61
Apêndice I – Análise de Requisitos 64
Apêndice II – Documento de Visão do Projeto 70
Apêndice III – *Survey* do Modelo de Casos de Uso 75
Apêndice IV – Caso de Uso 01 – Consulta Produtos 81
Apêndice V – Caso de Uso 02 – Realiza Compra 83
Apêndice VI – Tutorial de Instalação dos Programas 85
Apêndice VII – Modelagem Banco de Dados 101
Apêndice VIII – Glossário 102

Lista de Abreviaturas

API	Application Programming Interface
Applets	Programa Java Executado em um Browser
Backbone	Estrutura Principal
DHTML	Dynamic HTML
EJB	Enterprise Java Beans
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	Secure HyperText Transfer Protocol
IDE	Ambiente Integrado de Desenvolvimento
IRC	Internet Relay Chat
J2EE	Java 2 Enterprise Edition
J2SE	Java 2 Standard Edition
JAAS	Java Authentication and Authorization Service
JavaMail	API Java que fornece uma estrutura de independência de plataforma e protocolos para construção de aplicações com e-mail e mensagens
Java Script	Linguagem de Programação para Páginas da Web
JDBC	Java Database Connectivity
JMS	Java Message Service
JNDI	Java Naming and Directory Interface
JSP	JavaServer Pages
JUnit	Programa Java para testes unitários
MER	Modelo Entidade Relacionamento
MVC	Model View Controller
Perl	The Practical Extraction and Report Language
PL/PgSQL	Procedural Language/PostgreSQL Structured Query Language
Python	Linguagem de programação
RAD	Rapid Application Development
RMI	Remote Method Invocation
Ruby	Linguagem de programação
Servlet	Programa que estende a funcionalidade de um *web server*, gerando conteúdo dinâmico e interagindo com os clientes, utilizando o modelo request/response
SGBD	Gerenciador de Banco de Dados
SHTML	Sever Side Includes HTML
SMTP	Simple Mail Transfer Protocol
SQL	Structured Query Language
SSL	Secure Socket Layer
TagLibraries	Biblioteca de Tags
UCB	Universidade Católica de Brasília
UML	Unified Modelling Language ou Linguagem Unificada de Modelagem
WEB	World Wide Web
WML	Wireless Markup Language

XML	Extensible Markup Language
XSL	Extensible Stylesheet Language

Lista de Figuras

Figura 2.1 – Diagrama Feedback XP 15
Figura 2.2 - Práticas XP 17
Figura 3.1 - Ambiente multi-camada J2EE 21
Figura 3.2 - Modelo de Aplicação Multi-camada 22
Figura 3.3 - MVC em aplicações J2EE Multi-camada 22
Figura 3.4 - MVC Modelo 1 29
Figura 3.5 – MVC Modelo 2 30
Figura 4.1 - Principais casos de uso do sistema 41
Figura 4.2 - Diagrama de Seqüência do Caso de Uso ConsultaProdutos 42
Figura 4.3 - Diagrama de Seqüência do Caso de Uso RealizaCompra 43
Figura 4.4 - VOPC – Consulta Produtos 44
Figura 4.5 - VOPC – Realiza Compras com Ícones 45
Figura 4.6 - VOPC – Realiza Compras 45
Figura 4.7 - Parte do Diagrama de Classes do Projeto 46
Figura 4.8 - Tela de Login do Sistema 47
Figura 4.9 - Tela de Menu Principal 48
Figura 4.10 - Tela de Pesquisa de Funcionário 49
Figura 4.11 - Tela de Pesquisa de Produto 49
Figura 4.12 - Tela de Pesquisa de Clientes 50
Figura 4.13 - Tela de Cadastro de Clientes 51
Figura 4.14 - Calendário dos campos Data 51
Figura 4.15 - Formulário de Vendas 52

CAPÍTULO 1 - INTRODUÇÃO

1. Introdução

A *internet* é um meio de comunicação que facilita a forma como as empresas de pequeno, médio e grande porte vem sendo gerenciadas, aumentando sua competitividade no mercado. Atualmente é possível encontrar a prestação dos mais diversos tipos de serviços na internet, que vão desde correios eletrônicos a resultados de exame de DNA. Evita-se o deslocamento desnecessário do cliente, garante-se qualidade dos produtos ou serviços oferecidos, trazendo comodidade e eficiência como diferenciais de grande valor. Alie-se a tudo isto uma redução de custos e um aumento na quantidade de potenciais compradores, uma vez que as barreiras geográficas são quebradas.

O presente capítulo aborda os motivos que levaram os integrantes do grupo a escolher o tema apresentado e como foi utilizado o conhecimento adquirido em curso na elaboração das opções de solução dos problemas, em especial com foco na tecnologia dominante no curso, a tecnologia JAVA.

1.1. Motivação da Pesquisa

Atualmente, a modernização de empresas é um assunto de grande relevância na sociedade como um todo. Os meios aplicados na execução de tal modernização podem atingir os mais diversos segmentos, em especial, o humano: o aumento do desemprego em virtude de falta de qualificação profissional na área ou do uso de sistemas tecnológicos pode substituir ou transformar a mão de obra envolvida no processo. A inércia de algumas empresas frente à concorrência força tais empresas a fecharem suas portas por falta de condições de competitividade. Os concorrentes investem e garantem as suas sobrevivências no mercado. Neste contexto, a informática tem um papel importante no sentido de melhorar a eficiência do gerenciamento da empresa ou mesmo do atendimento ao cliente. Os computadores são utilizados desde a montagem de um banco de dados de RH dos funcionários da empresa, banco de dados de produtos, até o controle de todas as movimentações de estoque ou vendas.

A informática acelera o serviço e auxilia a ação dos gerentes, trazendo dados precisos para a tomada de decisões.

Nesta situação, os empresários mais bem informados buscam aumentar a produtividade de sua empresa e com isso, gerar maiores lucros. Algumas empresas optam pela instalação de sistemas genéricos que permitem cadastrar seus produtos e lançar suas vendas ou dívidas. Outras preferem providenciar o desenvolvimento de um sistema personalizado às suas necessidades. Mais barato ou com custos elevados, a instalação de sistemas de gerenciamento de empresas vem se tornando cada dia mais populares. Eles são utilizados desde a padaria da esquina até as grandes multinacionais e bancos. Algumas dificuldades geralmente aparecem quando da instalação dos mesmos: "como eu cadastro um produto? Como eu dou baixa de um produto em estoque? Quem pode fazer o que"? Todas estas perguntas estão relacionadas com o treinamento de pessoal da empresa. Não havendo treinamento, por melhor que seja o sistema, muitas de suas funcionalidades não terão qualquer motivo de gasto com implementação, tendo em vista o desconhecimento dos seus usuários sobre a ferramenta.

Um outro aspecto que poderia ser abordado neste tópico é o uso dos sistemas como uma forma de auxiliar o gerenciamento dos proprietários em seus mais diversos negócios. A dificuldade do empresário, que por vezes cuida de uma série de negócios, de estar acompanhando de perto todos os seus investimentos, torna um sistema de gerenciamento remoto uma necessidade urgente. Por exemplo, uma loja de médio porte que abra das 08:00h às 22:00h. O proprietário não teria condições de abrir a loja, atender aos clientes e ainda: negociar com seus fornecedores, pagar contas, realizar parcerias, preparar o projeto de marketing. Quanto maior o aumento no número de vendas, mais difícil se torna o gerenciamento de toda a estrutura de uma forma eficiente e diária. Uma solução muito eficiente seria informatizar todo o processo de vendas, compras, controle de estoque e etc, ou seja, tornar todo o processo de gerenciamento da empresa mais fácil e dinâmico, o que facilitaria a sua tomada de decisões e o tornaria independente de receber informações de seus funcionários sobre as necessidades de produtos em estoque, evitando a falta de produtos em prateleira e, conseqüentemente, o descontentamento ou mesmo a perda de clientes para outras lojas. Neste contexto, o uso de um programa faz com que os funcionários se sintam vigiados, uma vez que o sistema é totalmente impessoal e não permite tentativas de ludibriá-lo. Além disso, o sistema repassa ao empresário, informações e gráficos comparativos das vendas daquele dia com a média diária a ser alcançada.

Diante da real necessidade da empresa na resolução dos seus problemas e do interesse no uso das tecnologias JAVA, pelos integrantes do grupo, surgiu a oportunidade do desenvolvimento do protótipo, com maior motivação, inclusive, por se tratar de algo que seria aplicado diretamente a uma empresa e não se perderia apenas em conceitos e abstrações.

Durante todo o desenvolvimento do trabalho, buscou-se, sempre, o uso de "software livre" visando garantir a concretização de um projeto com o menor custo-benefício em longo prazo, por não utilizar softwares pagos, necessitando de licenças a preços que fogem a realidade das empresas de pequeno e médio porte.

1.2. Objetivo Geral

O objetivo geral deste trabalho é o estudo de tecnologias e *frameworks* J2EE e padrões de projeto voltados para o desenvolvimento de aplicações *web* em J2EE, em especial o uso do e-Gen Developer, como ferramenta RAD (*Rapid Application Development*) que proporciona um rápido desenvolvimento de aplicações para *web*, totalmente escrito em Java e baseado no *framework* Jakarta Struts.

O objetivo é facilitar a gerência da estrutura de comércio varejista de suplementos alimentares e produtos naturais da empresa "Companhia da Nutrição". O sistema visa informatizar o processo de vendas, compras e controle de estoque; ou seja, tornar o processo de gerenciamento da empresa mais fácil e dinâmico, o que facilitaria ao gerente a tomada de decisões e o tornaria independente das informações de seus funcionários sobre as necessidades de produtos em estoque, evitando a falta de produtos em prateleira e, conseqüentemente, o descontentamento ou mesmo a perda de clientes para outras lojas.

1.3. Objetivos Específicos

- Apresentar o e-Gen Developer, como um ambiente RAD (*Rapid Application Development*) baseado no *framework* Jakarta Struts, totalmente escrito em Java;
- Apresentar o *framework* Struts, que já aplica alguns padrões de projeto e facilita a implementação dos conceitos de orientação a objetos, programação multicamadas, e da modelagem utilizando a UML – *Unified Modeling Language*.
- Apresentar as propostas do eXtreming Programing – XP para o desenvolvimento utilizando apenas dois desenvolvedores.

- Apresentar ferramenta DbWrench, escrita em Java, para a modelagem e geração de um banco de dados PostgreSQL.
- Elaborar um protótipo que demonstre algumas vantagens do emprego das tecnologias Java no desenvolvimento de um projeto de gerenciamento on-line de uma loja de suplementos alimentares;

1.4. Fronteiras da Pesquisa

O presente trabalho não apresenta detalhadamente as tecnologias envolvidas, mas descreve os principais aspectos que levaram as mesmas a serem escolhidas para o desenvolvimento do projeto. Assim, pretende-se explorar as principais características das tecnologias a fim de que se viabilize um protótipo, com a possibilidade de novos ajustes e implementações futuras.

Além disso, o desenvolvimento deste trabalho limitou-se ao escopo das necessidades da empresa "Companhia da Nutrição", o que não quer dizer que as mesmas tecnologias não possam ser utilizadas para outros fins ou demais empresas, mesmo as de grande porte.

1.5. Estrutura do Trabalho

Este projeto está dividido em 05 capítulos, que abordam os seguintes aspectos:

O primeiro capítulo trata das informações introdutórias, tais como motivação, objetivos, fronteiras da pesquisa e a estruturação do trabalho.

O segundo capítulo trata da notação e processo utilizados no trabalho, descrevendo a UML e a metodologia ágil eXtreming Programming - XP.

O terceiro capítulo aborda as tecnologias empregadas na construção do protótipo de gerenciamento da empresa.

O quarto capítulo trata da Análise e o Desenvolvimento do Projeto, com as fases descritas pelo eXtreming Programming.

No quinto e último capítulo são descritos a conclusão, objetivos alcançados e considerações finais do trabalho apresentado.

CAPÍTULO 2 – NOTAÇÃO E METODOLOGIA

2. Introdução

O mundo está em constante desenvolvimento. E como não podia deixar de ser, a informática e, mais especificamente, o segmento do desenvolvimento de software esta cada dia mais pressionado pela necessidade de se construir sistemas maiores e mais complexos em um tempo cada vez menor. O emprego das melhores práticas, ferramentas e técnicas é fundamental para garantir a qualidade, os prazos e os custos previstos dos projetos de software. Todas as metodologias visam promover o aprimoramento do desenvolvimento de software. Cada uma a sua maneira, com suas vantagens e desvantagens.

Uma empresa de software bem-sucedida é aquela que fornece software de qualidade e capaz de atender às necessidades dos seus usuários (clientes). A empresa que consiga desenvolver um software com previsão e em um período pré-estabelecido, com utilização eficiente e eficaz de recursos, será considerada uma empresa com um negócio viável. O principal produto de uma equipe de desenvolvimento não são documentos bonitos ou ótimos slogans, mas sim um bom software, capaz de atender as necessidades de seus usuários e respectivos negócios. [Boch et al., 2000].

Da mesma forma que empresas bem-sucedidas contratam mais pessoas, tem mais clientes, mais vendas, etc., o software comercial destas empresas tem de ser dimensionável em termos de suportar o crescimento destas empresas e de suas operações. As empresas encontram limites quando crescem, seja por incapacidade de hardware, incapacidade de um espaço físico que suporte todos os seus clientes e funcionários, entre outros problemas. Assim, conectar sistemas que funcionavam separados anteriormente passou a ser um desafio maior. Os sistemas geralmente eram designados com finalidades específicas e não eram projetados para integração com outros sistemas. Estes são exemplos de desafios comuns ao se lidar com o desenvolvimento de software comercial. [Ahmed et al., 2002]

Neste contexto de dificuldades para expansão dos softwares de empresas bem-sucedidas, é interessante citar uma breve evolução do software comercial. Há pouco tempo, os *mainframes* governavam o mundo e todo software era ligado a essa entidade central, que possuía como vantagens à simplicidade de se lidar com um único sistema para todas as necessidades de processamento, a colocação de todos os recursos, entre outras. Como

desvantagem, significava lidar com limites físicos de dimensionamento, acessos limitados a partir de locais remotos, etc. Um dos aspectos mais problemáticos dos *mainframes*, como uma aplicação de *um nível* era a mistura da apresentação, da lógica de negócios e dos dados em si. A abordagem do cliente-servidor (*dois níveis*) suavizou alguns problemas, movendo a apresentação e alguma lógica de negócios para um nível separado dos dados. A abordagem com *n níveis* tenta conseguir equilibrar os problemas anteriores, separando a apresentação, da lógica comercial e da lógica dos dados. Algumas vantagens da computação com n níveis são: o desenvolvimento mais rápido e com custo menor, o impacto das alterações ser isolado, e as alterações serem mais gerenciáveis. [Ahmed et al., 2002]

Mesmo com a evolução do desenvolvimento de software, segundo o Chãos Report do Standish Group, 1998 [cit. In Ahmed et al. 2002]; "apenas 26% dos projetos de software têm sucesso". Com isso, é possível concluir que a atividade de desenvolvimento de software possui um alto grau de risco, onde a maioria dos projetos que são iniciados, ou não são terminados, ou consomem prazos e orçamentos acima do estipulado no início do projeto.

Segundo [Boch et al., 2000], a modelagem é a parte central de todas as atividades que levam à implementação de um bom software. A construção de modelos visa: comunicar a estrutura e o comportamento desejados do sistema; visualizar e controlar a arquitetura do sistema; compreender melhor o sistema em elaboração, expondo oportunidades de simplificação e reaproveitamento; e gerenciar os riscos. Projetos de softwares mal sucedidos falham por aspectos específicos de cada projeto, já os bem-sucedidos são semelhantes em vários aspectos, onde um dos muitos elementos que contribuem para o sucesso da empresa de software é a utilização da modelagem.

2.1. UML – Unified Modeling Language (Linguagem de Modelagem Unificada)

"A UML (Unifed Modeling Language) é uma linguagem-padrão para a elaboração da estrutura de projetos de software. A UML poderá ser empregada para a visualização, a especificação, a construção e a documentação de artefatos que façam uso de sistemas complexos de software." [Booch et al., 2000, p. 13]. Em um segundo conceito: "A Unified Modeling Language (UML) é uma linguagem gráfica para a modelagem e o desenvolvimento de sistemas de software. Fornece um suporte de modelagem e visualização para todas as fases de desenvolvimento de software, desde a análise das exigências até a especificação, construção e distribuição." [Ahmed et al., 2002, p. 24]

A UML não deve ser confundida com um processo. Ela é apenas uma linguagem, sendo, apenas, uma parte de um método para o desenvolvimento de software. A UML independe do processo, embora seja utilizada em processo orientado a casos de uso, centrado na hierarquia, interativo e incremental [Boch et al., 2000, p. 13]. Ainda segundo Boch et al. (2000), para obter o máximo proveito de UML, será preciso levar em consideração um processo com as seguintes características: orientado a caso de uso (usam os casos de uso como o principal artefato para estabelecer o comportamento desejado do sistema), centrado na hierarquia (a arquitetura do sistema é utilizada como principal artefato para a conceituação, a construção, o gerenciamento e a evolução do sistema em desenvolvimento), interativo (envolve seqüências de versões executáveis) e incremental (envolve integração contínua da arquitetura do sistema para a produção das versões).

A UML tem suas raízes em várias notações baseadas em objetos anteriores. A diferença entre notação e metodologia é fonte comum de confusão. A UML é uma notação que pode ser aplicada usando muitas abordagens diferentes. Estas abordagens são as metodologias. [Ahmed et al., 2002].

"A UML é uma linguagem destinada a: visualizar, especificar, construir e documentar os artefatos de um sistema complexo de software". [Boch et al., 2000, p. 14].

A *visualização* na UML destina-se a padronização de conceitos para que pessoas não envolvidas em um determinado projeto desde o início possam ser adicionadas à equipe e compreendam o projeto e dêem continuidade ao desenvolvimento. O uso de símbolos gráficos com uma semântica bem definida faz com que um desenvolvedor escreva seu modelo, e outros desenvolvedores sejam capazes de interpretá-lo, sem ambigüidades. (Boch et al., 2000).

A *especificação,* na UML, relaciona-se a construção de modelos precisos, sem ambigüidades e completos. [Boch et al., 2000].

A UML como linguagem para *construção*, embora não seja uma linguagem visual de programação, pode ter os seus modelos conectados a várias linguagens de programação, onde seria possível mapear os seus modelos e gerar código a partir do modelo UML para uma linguagem de programação, como Java, por exemplo. [Boch et al., 2000]

A *documentação* na UML é responsável por controlar, medir e comunicar o sistema durante seu desenvolvimento e após sua implantação. [Boch et al., 2000]

"A idéia central ao usar a UML é capturar os detalhes significantes sobre 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." [Ahmed et al., 2002, p. 24].

"A UML não só fornece a notação para os blocos de construção básicos, como também fornece maneiras de expressar as relações complexas entre os blocos de construção básicos". [Ahmed et al., 2002, p. 24].

Segundo [Ahmed et al., 2002] as relações podem ser estáticas ou dinâmicas. As estáticas estão relacionadas com os aspectos estruturais de um sistema, ou seja, relações de herança entre classes, interfaces implementadas por uma classe e dependência em outra classe são exemplos de relações estáticas. As dinâmicas têm relação com o comportamento do sistema e, portanto, existem durante a execução. Um exemplo de relação dinâmica seriam as mensagens trocadas em um grupo de classes para cumprir alguma responsabilidade e o fluxo do controle em um sistema. As relações estáticas ou dinâmicas são capturadas na forma de diagramas UML, que são de vários tipos:

- *Diagrama do caso de uso:* mostra os casos de uso, os atores e suas relações. Capturam as exigências precisas para o sistema de uma perspectiva de um usuário.
- *Diagrama de classe:* mostra relações estáticas que existem entre um grupo de classes e de interfaces no sistema. Exemplos comuns são relações de herança, de agregação e de dependência.
- *Diagrama de objeto:* fornece uma visão instantânea das relações existentes entre as instâncias da classe em um dado ponto no tempo. Este diagrama é útil para capturar e mostrar, de maneira estática, as relações complexas e dinâmicas no sistema.
- *Diagrama do gráfico de estado:* são excelentes para capturar o comportamento dinâmico do sistema. São, em particular, aplicáveis nos sistemas reativos baseados em eventos ou objetos nos quais a ordem dos eventos é importante. Também são úteis para modelar o comportamento das interfaces.
- *Diagrama de atividade:* é uma extensão de um diagrama do gráfico do estado e é parecido em conceito com um fluxograma. Permite modelar o comportamento do sistema em termos de interação ou fluxo do controle entre as atividades distintas ou objetos. São melhor usados para modelar os fluxos do trabalho e o fluxo das operações.
- *Diagrama de interação:* são utilizados para modelagem do comportamento dinâmico de um sistema.

- *Diagrama de seqüência:* usado para modelar a troca de mensagens entre os objetos em um sistema. 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.
- *Diagrama de componente:* um componente representa de forma física, uma parte do sistema, como um arquivo, um executável e etc. Geralmente mapeia uma ou mais classes, subsistemas e etc. Um diagrama de componente mostra as dependências e as relações entre os componentes que compõem um sistema.
- *Diagrama de distribuição:* mostra a arquitetura de um sistema da perspectiva de nós, processadores e relações entre eles. Em J2EE, estes diagramas são úteis para modelar e desenvolver a arquitetura do sistema distribuído.

2.2. Metodologias de Software ou Processos de Software

"Um processo de desenvolvimento de software fornece uma orientação sobre como desenvolvê-lo com sucesso. Tal orientação pode cobrir o espectro inteiro de atividades associadas ao desenvolvimento de software. O processo pode se manifestar na forma de abordagens aprovadas, melhores práticas, regras, técnicas, seqüência etc." [Ahmed, 2002, p.50].

As várias metodologias existentes para o desenvolvimento de software possuem atividades fundamentais comuns a todas elas [Sommerville, 2003]:

- **Especificação de Software:** definição das funcionalidades (requisitos) e das restrições do software. Geralmente é uma fase em que o desenvolvedor conversa com o cliente para definir as características do novo software.
- **Projeto e Implementação de Software:** o software é produzido de acordo com as especificações. Nesta fase são propostos modelos através de diagramas, e estes modelos são implementados em alguma linguagem de programação.
- **Validação de Software:** o software é validado para garantir que todas as funcionalidades especificadas foram implementadas.
- **Evolução de Software:** o software precisa evoluir para continuar sendo útil ao cliente.

Existem várias metodologias de software. É comum algumas organizações adaptarem metodologias às suas realidades, ou mesmo criarem suas próprias. Dentre os vários processos existentes, existem as metodologias tradicionais, que são orientadas a documentação, e as metodologias ágeis, que procuram desenvolver software com o mínimo de documentação.

2.2.1. Metodologias Tradicionais ou Pesadas

As metodologias tradicionais são também chamadas de pesadas ou orientadas a documentação. Essas metodologias surgiram em um contexto de desenvolvimento de software muito diferente do atual, baseado apenas em um *mainframe* e terminais burros [Royce, 1970].

Contra as metodologias tradicionais, segundo [Standish Group, 1995], em uma base de 8380 projetos, apenas 16,2% dos projetos foram entregues respeitando os prazos e os custos e com todas as funcionalidades especificadas. Aproximadamente 31% dos projetos foram cancelados antes de estarem completos e 52,7% foram entregues, entretanto com maiores prazos, maiores custos e uma quantidade inferior de funcionalidades do que as especificadas no início do projeto. Com base apenas nos projetos que não foram finalizados de acordo com prazos e custos especificados, a média de atrasos foi de 222%, e a média de custo foi de 189% a mais do que o previsto.

2.2.2. Metodologias Ágeis

Pensamento dos seguidores das Metodologias Ágeis:

"Estamos descobrindo melhores maneiras para desenvolver software, *fazendo-o e ajudando outros a fazê-lo.* Através deste trabalho nós chegamos a valorizar:

- *Indivíduos e interações* mais que processos e ferramentas
- *Software que funciona* mais que documentação abrangente
- *Colaboração do cliente* mais que negociação contratual
- *Responder à mudança* mais que seguir um plano.

Ou seja, mesmo dando valor aos itens à direita, valorizamos mais os itens à esquerda". [Beck, et al., 2001].

O conhecido "Manifesto para o Desenvolvimento Ágil de Software" foi o resultado de uma reunião entre 17 especialistas (Kent Beck, Mike Beedle, Arie van Bennekum, Alistair

Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas) da comunidade de desenvolvimento, realizada entre os dias 11 e 13 de fevereiro de 2001, em uma estação de esqui nas montanhas de Utah, Estados Unidos.

O conceito do termo "**agilidade**" seria: "s. f. Ligeireza; leveza; desembaraço; presteza de movimentos." [Bueno, 1989]

2.2.2.1. Extreme Programming - XP

"**Extreme Programming** é uma metodologia ágil para equipes pequenas e médias desenvolvendo software com requisitos vagos e em constante mudança".[Beck, 2000]. Extreme Programming (XP) prima pela satisfação do cliente e pela qualidade do software final.

"Extreme Programming é uma disciplina do desenvolvimento do software baseada nos valores de simplicidade, da comunicação, do feedback, e da coragem. Trabalha trazendo a equipe inteira junto na presença de práticas simples, com feedback suficiente para permitir a equipe ver onde estão e de ajustar as práticas a sua situação original". [Jeffries, 2001]

A Extreme Programming assume que o código é tudo o que importa, pois uma quantidade suficiente de pessoas adquiriram a experiência de que o código bem escrito é fácil de compreender e modificar. Os desenvolvedores tradicionais de software acreditam que a documentação é essencial porque muitos desenvolvedores sofreram bastante pela falta de documentação, tornando o código sem manutenção. Visto em suposições, XP é irresponsável e tornará cada vez mais o desenvolvimento não documentável e, portanto, com código de difícil manutenção, enquanto métodos e processos tradicionais desperdiçam o tempo forçando colaboradores a escrever a documentação que ninguém nunca irá ler. [McBreen, 2002]

O primeiro projeto a usar XP foi o C3, da Chrysler. Após anos de fracasso utilizando metodologias tradicionais, com o uso da XP o projeto ficou pronto em pouco mais de um ano [Highsmith et al., 2000].

Segundo [Higtower et. al, 2002] alguns usuários de metodologias mais tradicionais criticaram XP, reivindicando que o mesmo não é um processo real. Ao contrário, XP é uma metodologia extremamente disciplinada que se centra na: constante revisão de código; testes freqüentes; participação do cliente e feedback rápido; refatoramento incessante e refinamento da arquitetura; integração contínua para descobrir problemas no processo do desenvolvimento o mais cedo; e planejamento constante. Kent Beck, criador da metodologia de XP, definiu quatro valores chaves de XP:

- Comunicação (não é limitada por procedimentos formais, usa-se o melhor meio possível; programadores XP comunicam-se com seus clientes e entre eles mesmo, de preferência no "corpo a corpo");
- Simplicidade (solução adotada deve ser sempre a mais simples que alcance os objetivos esperados; deve-se entregar ao cliente apenas o que precisa e quando for necessário);
- Feedback (refere-se a qualidade do código: testes de unidade, programação em pares, posse coletiva; e sobre estado do desenvolvimento: estórias do usuário final, integração contínua, jogo do planejamento; visa garantir que qualquer problema encontrado possa ser corrigido o quanto antes); e
- Coragem (aumento da confiança do programador ajudam-no a melhorar o design do código, retirar código desnecessário, pedir ajuda aos que sabem mais, etc.).

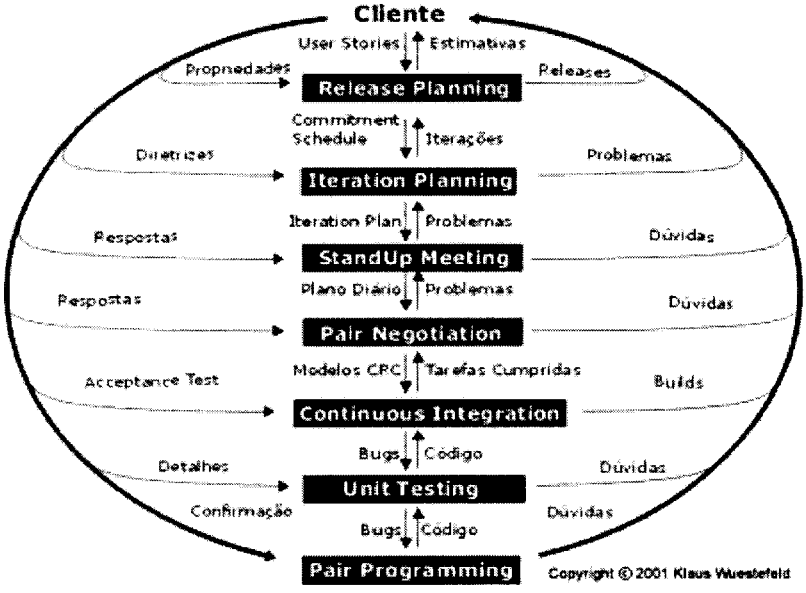


Figura 2.1 - **Diagrama Feedback XP** – (fonte: http://www.xispe.com.br/diagrama.html)

Segundo [Ahmed, 2002], a metodologia proposta por Kent Beck vem merecendo destaque ultimamente, sendo posicionada como um "processo de desenvolvimento de software leve", podendo ser quase construída como um antiprocesso no sentido tradicional. Conforme citado anteriormente, como um dos valores da metodologia, sua principal idéia é manter as coisas o mais simples possível para ter o serviço pronto. Suas atividades são organizadas em torno de quatro tarefas maiores: planejamento, construção, codificação e teste.

Segundo [Higtower et. al, 2002] Beck iterou, em seu livro marco em XP, quatro práticas básicas citadas acima, que são expressas em mais 12 principais práticas, como segue:

- O **Jogo do Planejamento** (*The Planning Game*): uso de uma metodologia rígida e de *design patterns*.
- **Pequenos lançamentos** (*Small Releases*) Um teste -> uma função/método -> dar check-in
- **Metáfora** (*Metaphor*): entendimento comum sobre o funcionamento do sistema. Arquitetura de alto nível. Uso de *Design Patterns*. É obrigatória para poder participar da equipe.
- **Projeto Simples** (*Simple Design*): quanto mais simples, mais ágil.
- **Refatoramento** (*Refactoring*): fazer freqüentemente em passos bem pequenos. Uso de testes automatizados para dar segurança.
- **Teste** (*Testing*): os testes são constantes a fim de corrigir o quanto antes qualquer erro encontrado. Segundo [Ahmed, 2002] o teste primário está centrado no teste da unidade e o teste funcional é ditado pelo cliente para determinar a aceitação do produto de software.
- **Programação em Pares** (*Pair Programming*), ou seja, toda a programação é feita no mínimo em dupla. A troca de pares é freqüente.
- **Propriedade Coletiva** (*Collective Ownership*) Não há papéis técnicos exclusivos (DBA, SQA, CM...), todos são responsáveis e devem ter conhecimento do todo. A qualquer momento, qualquer membro pode alterar o código.
- **Integração Contínua** (*Continuous Integration*): Construção e teste contínuos do projeto. Dá uma marcação no tempo do projeto. No mínimo uma vez por dia.
- **Semana de 40 horas** (*40-hour Week*): trabalhar sempre no horário normal, sem horas extras. Evitar o desgaste excessivo dos desenvolvedores. De preferência, não depender de esforço extraordinário. Permite trabalhar com energia todos os dias: mais produtividade, menos erros, ócio criativo.
- **Cliente no Local** (*On-Site Customer*): o cliente faz parte da equipe de desenvolvimento. Ele interage com a criação do sistema.
- **Padrões de Codificação** (*Coding Standards*): Todo o código-fonte segue convenções de formatação, nomes de identificadores, práticas de programação.

Os padrões são seguidos à risca, sendo exigido o seu seguimento. Comentários somente quando disserem alguma coisa.

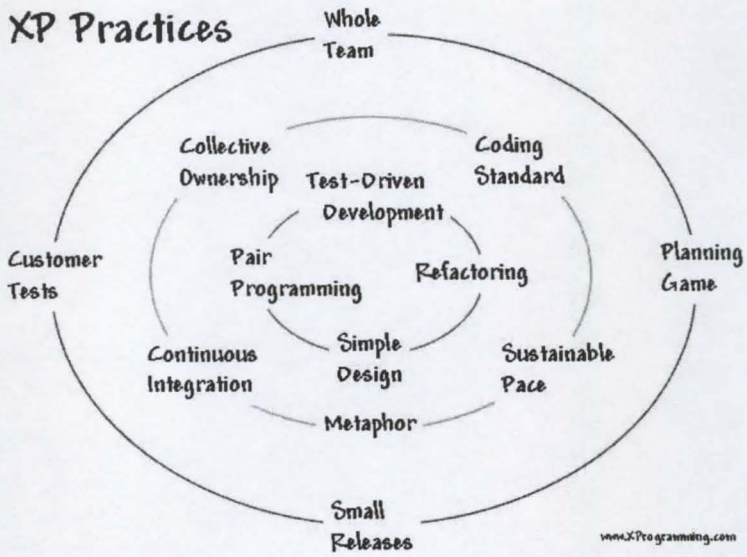


Figura 2.2 - Práticas XP – (fonte: http://xprogramming.com/images/circles.jpg)

Explicando a figura, o círculo exterior das práticas (em vermelho) mantém as pessoas técnicas e os usuários envolvidos em sincronia, já o círculo médio das práticas certifica-se que o trabalho diário dos programadores individuais vem se encontrar com as necessidades e os interesses regulares do negócio. Por último, o círculo interno das práticas (em azul) aplica uma disciplina aos colaboradores individuais para mantê-los focalizados na qualidade quando estiverem envolvidos na programação.

Segundo [Beck, 1999] os contribuidores de um Projeto em XP são membros de um time. Nesta equipe deve constar:

- Cliente ou representante: tem a função de estabelecer os requerimentos do projeto, definir prioridades e direcionar o projeto. É interessante que este membro seja um usuário final que conheça o domínio do problema e suas necessidades. É um dos *principais* membros da equipe. É responsável por decisões que alteram os rumos dos projetos.
- Programadores: são o coração do XP. Eles são os responsáveis por tomar decisões que balanceiam as prioridades. Comparados aos clientes, os programadores sabem *como* programar, os clientes *o que* programar.
- Testadores: auxiliam os clientes a escolher e escrever testes funcionais.
- Tracker: responsável por estimativas e pela coleta das métricas.

- Coach: é responsável pelo projeto como um todo. É responsável pela compreensão da aplicação de uma forma mais detalhada. Orienta a equipe e controla a aplicação do XP, apontando desvios no processo ou, ainda, repassando calma a equipe.
- Consultores ou analistas: ajudam o cliente a definir requerimentos. A ênfase na simplicidade de design reduz a ocorrência dos serviços deste membro, mas eles geralmente ocorrem.
- Big Boss ou Gerente: providencia recursos necessários e coordena as atividades. É um motivador da equipe, fornecendo coragem, confidência e ocasionalmente, insistência aos membros da equipe.

Segundo [Beck, 1999] todo o ciclo de desenvolvimento de código em XP se resume a:

- Programadores programam juntos;
- Desenvolvimento é guiado por testes. O programador primeiro testa e, somente depois codifica. Enquanto todos os testes não rodam, há necessidade de reparos. Quando todos os testes rodam e o programador não pode mais pensar em novos testes, pode-se adicionar a funcionalidade.
- Pares não apenas fazem os testes funcionarem. Eles evoluem o projeto do sistema. Mudanças não são restritas a determinadas áreas. Os pares adicionam valor de análise, projeto, implementação e testes do sistema. Eles adicionam valores sempre que o sistema precisar.
- A integração ocorre logo após o desenvolvimento, incluindo os testes de integração.

CAPÍTULO 3 – TECNOLOGIAS, MODELOS E PADRÕES DE PROJETO

Este capítulo tem como objetivo expor os conceitos das tecnologias, modelos e padrões empregados no presente projeto.

3.1. JAVA – Informações Gerais Sobre a Linguagem

As contribuições de Java como linguagem e ambiente de programação são consenso entre desenvolvedores e pesquisadores. Jamais uma linguagem reuniu rapidamente tantas características favoráveis além do apoio de instituições importantes, estando prestes a se tornar um padrão para o desenvolvimento com objetos. Desde o seu surgimento, Java tornou-se uma escolha atraente para o desenvolvimento de aplicações Internet/Intranet [Hopson ,1998].

A linguagem Java foi projetada para atender às necessidades do desenvolvimento de aplicações em um ambiente distribuído (rede) e heterogêneo. Um dos desafios fundamentais está relacionado com a instalação segura de aplicações que consumam o mínimo de recursos do sistema, possam executar em qualquer plataforma de hardware e software e possam ser estendidas. Dentre as vantagens, podemos citar:

1. Programas Java podem ser executados em todas as plataformas significativas, como Linux, Unix, Microsoft sem necessidade do código-fonte ou recompilação.
2. A uniformidade dos conceitos de Java permite a integração em aplicações de conceitos modernos como componentes, invocação remota, reflexão/introspecção, validação e conectividade a bancos de dados.
3. O potencial de Java se aplica em várias direções, de aplicações convencionais ou distribuídas na Internet a programas embarcados.
4. Java (J2SE, J2EE, etc) é uma arquitetura aberta, extensível, com várias implementações, o que a torna independente do fornecedor.
5. Uma linguagem de programação segura, robusta, rodeada de API's eficazes e completa, um ambiente de desenvolvimento e um ambiente de aplicação;

A linguagem de programação Java é projetada para resolver vários problemas nas modernas práticas de programação. A linguagem Java provê poderosas ferramentas para programadores, facilitando a programação, pois é orientada a objetos. Devido a sua arquitetura neutra, aplicações em Java são ideais para ambientes como a Internet.

No desenvolvimento do projeto, foi utilizada a linguagem Java, por ser uma linguagem moderna, multiplataforma, que oferece um modelo de programação distribuído, simples de programar, robusto e com crescente aceitação comercial.

3.2. J2EE - Java 2 Platform Enterprise Edition

A contínua evolução dos padrões Java conduziu, enfim, à especificação da Plataforma Java para Corporações (lançada pela Sun Microsystems em 1999). Seu lançamento agrega uma série de especificações, cujo objetivo é o de auxiliar os desenvolvedores a alcançar o ideal *WORA("Write Once, Run Anywhere™")* no lado servidor. A especificação desta plataforma consiste de:

J2EE *Platform Specification*: (Especificação da Plataforma) define as APIs Enterprise Java (EJB, JNDI, JDBC, Servlets, JSP, JMS, JavaMail, ...) e suas versões, que devem ser suportadas para garantir mínima qualidade de serviço, compatibilidade, portabilidade e integração.

J2EE *Application Programming Model*: (Modelo de Programação de Aplicação) que visa auxiliar o desenvolvimento de aplicações corporativas multi-camada para a plataforma J2EE. Inclui exemplos e *design patterns* bem sucedidos para corporações.

J2EE *Compatibility Test Suite*: (Conjunto de Testes de Compatibilidade) utilizado por fornecedores de software para verificar se sua implementação da plataforma J2EE é compatível com a especificação J2EE

J2EE *Reference Implementation*: (Implementação de Referência – J2EE SDK) é uma implementação da especificação da plataforma J2EE, que visa demonstrar suas capacidades, bem como prover uma definição operacional da mesma. Está disponível, juntamente com seu código fonte, para livre utilização.

A J2EE proporciona, dessa forma, um ambiente de execução integrado, consistente e

atestado (Fig. 3.1), cujo *backbone* é constituído pelos componentes EJB. Garante assim uma determinada qualidade de serviço, assegurando também portabilidade e interoperabilidade.

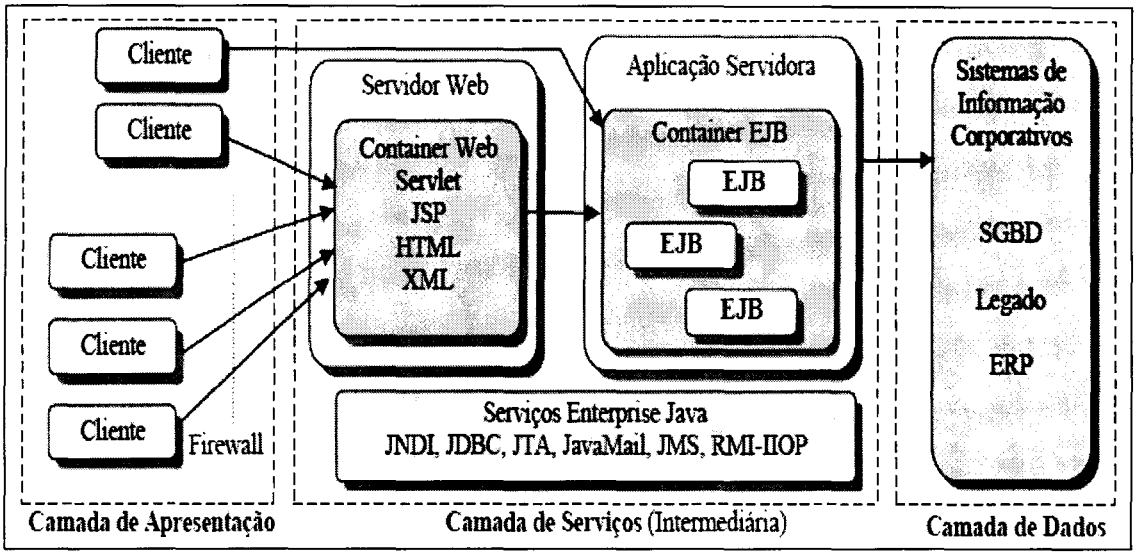


Fig. 3.1 - Ambiente multi-camada J2EE. Fonte: [Sun, 2000]

Uma das principais preocupações das aplicações servidoras se concentra em como prover interoperabilidade e portabilidade, a fim de fornecer uma solução duradoura e de grande alcance. Essas habilidades em particular recebem especial atenção da Plataforma Java para Corporações. Também denominada de Java 2 *Platform Enterprise Edition*, ou J2EE, ela define uma arquitetura Java unificada, centrada em serviços e baseada em componentes, habilitada para aplicações multi-camada, que promove interoperabilidade, portabilidade e ambiente de execução consistente para aplicações corporativas. J2EE define facilidades para composição de aplicação que favorecem o desacoplamento de funcionalidade em componentes lógicos, e encorajam a reutilização de código orientado a componente.

3.2.1. - O Modelo de Programação J2EE

O modelo de programação J2EE fundamenta-se na integração de camadas, ou montagem da aplicação. É assim que o desenvolvimento, a implantação e a reutilização de código orientado a componentes são facilitados. Para tanto, a J2EE considera vários cenários de aplicações – apesar de não haver tendências favorecendo um cenário em detrimento de outro. Dentre os cenários suportados, há destaque para o cenário denominado de Modelo de Aplicação Multi-camada (Fig. 3.2), cuja habilidade maior está em desacoplar o acesso a dados de questões de como realizar interações com os usuários, propiciando escalabilidade.

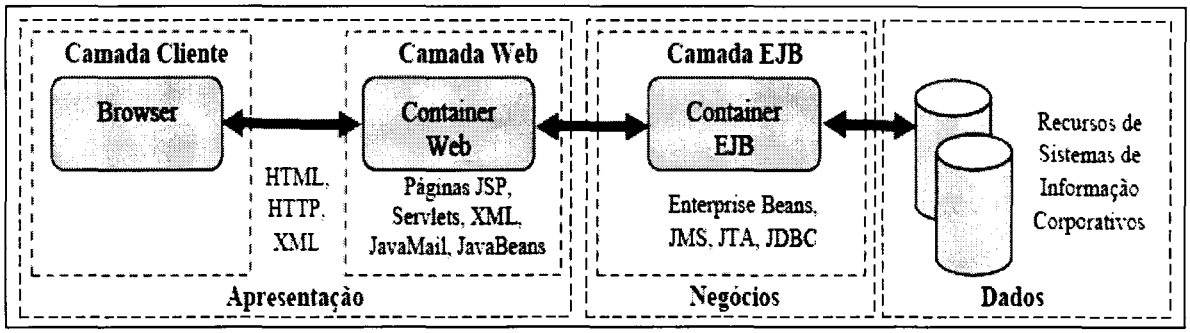


Fig. 3.2 - Modelo de Aplicação Multi-camada. Fonte: [D'Avila, 2003].

O modelo de programação J2EE também adota o padrão MVC (*Model-View-Controller*), que orienta o processo de decompor uma aplicação em componentes lógicos: o *Model* representa as regras de negócio e de dados; o *View* trata da apresentação; e o *Controller* gerencia a interação do usuário com o *View* e as invocações ao *Model*. No modelo de Aplicação Multi-camada da J2EE, o MVC pode ser implementado através de páginas JPS, EJBs e componentes JavaBeans, como ilustrado na Fig. 3.3.

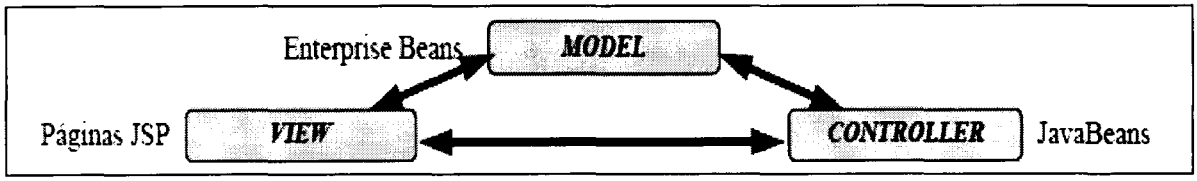


Fig. 3.3 - MVC em aplicações J2EE Multi-camada.[Sun, 2002]

3.2.2. - Tecnologias utilizadas no J2EE

3.2.2.1 - Servlet

Servlet é um programa que estende a funcionalidade de um *web server*, gerando conteúdo dinâmico e interagindo com os clientes, utilizando o modelo request/response. Os servlets não são restritos ao modelo HTTP de request/response, mas esse é o modelo mais comumente utilizado. Ao contrário dos *applets*, os *servlets* não possuem uma interface gráfica, e podem escritos para trabalhar com diversos tipos de protocolo de comunicação, como SMTP, HTTP e IRC [Colla, 2004].

3.2.2.1.1 - Arquitetura de um Servlet

Todos os servlets implementam direta ou indiretamente a interface Servlet. O mais comum é o servlet dar extends na HttpServlet (que implementa a interface Servlet). A interface Servlet fornece métodos para gerenciamento do servlet e sua comunicação com

clientes.

Quando um servlet aceita uma chamada do cliente, ele recebe dois objetos: ServletRequest e ServletResponse. A classe ServletRequest encapsula a comunicação do cliente com o servidor, enquanto a ServletResponse encapsula a comunicação do servidor com o cliente.

A ServletRequest permite que o servlet acesse informações como os nomes dos parâmetros passados pelo cliente, o protocolo usado pelo mesmo e o nome do *host* que fez o chamado ao servidor. Ela também permite que o servlet acesse um *inputstream* - o ServletInputStream - através do qual o servlet recebe dados do cliente.

As subclasses da ServletRequest permitem ao servlet obter dados mais específicos, como informações do cabeçalho HTTP. A ServletResponse fornece ao servlet métodos para responder ao cliente. Ela permite que o servlet defina o tamanho do conteúdo e seu mime type, fornece uma OutputStream - a ServletOutputStream - e também um Writer, através dos quais o servlet poderá enviar respostas ao cliente. As subclasses da ServletResponse fornecem ao servlet mais capacidades específicas em relação ao protocolo, como manipular o cabeçalho HTTP da resposta.[iMasters, 2003]

3.2.2.2 - JSP - Java Server Pages

A tecnologia JavaServer Pages (JSP) é uma tecnologia que permite a criação de páginas dinâmicas. É derivado da linguagem Java e é extremamente poderoso. Permite que os desenvolvedores *web* e designers criem rapidamente e mantenham facilmente páginas *web* dinâmicas, que são muito importantes no mundo dos negócios. Os aplicativos *web* desenvolvidos com JSP são independentes de plataforma. O JSP separa a interface do usuário da geração de conteúdo, permitindo que os designers modifiquem o aspecto geral do design das páginas, sem alterar o conteúdo dinâmico implícito. A tecnologia JSP é uma extensão da tecnologia Java Servlet. Servlets são módulos independentes de plataforma que funcionam do lado do servidor e que se encaixam em no contexto de um servidor *web* e podem ser utilizados para expandir as capacidades deste com mínimo custo de processamento, manutenção e suporte. Diferentemente de outras linguagens de script, os servlets não se prendem a considerações ou modificações de uma plataforma específica, eles são componentes que são baixados, sobre demanda, para a parte do sistema que precisa deles.

O JSP apresenta-se como uma solução profissional, e pode ser considerado o futuro das aplicações WEB: desacopla com facilidade a apresentação da lógica, permite a compilação em tempo real das aplicações (uma vez executado o script pela primeira vez, uma versão compilada se apresenta às requisições seguintes) e integra-se à perfeição no mundo Java. Como este, possibilita acesso a diversos bancos de dados (através de JDBC), é executado em inúmeros sistemas operacionais (virtualmente, o que possuir uma máquina virtual Java) e vem se tornando a opção de fato para o mercado de servidores (gigantes como IBM, Netscape e Oracle têm investido neste nicho suportando JSP), além de possuir alternativas gratuitas ou *Open Source* como o projeto Jakarta/Tomcat.

3.2.2.3 - Java JDBC - Java DataBase Connectivity

JDBC (Java DataBase Connectivity) é a API Java para comunicação com bancos de dados. Ela especifica a interface necessária para conectar a um banco de dados, executar comandos SQL e *queries*, e interpretar resultados. Esta interface é independente do banco de dados e da plataforma. As classes JDBC estão no pacote java.sql, que precisa ser importado para a classe Java que faz uso de alguma classe desta API.

3.2.2.3.1 - Acesso ao Banco de Dados

A API JDBC não pode comunicar-se diretamente com o banco de dados. O *acesso é* fornecido por um *driver* JDBC, que é uma classe específica para o banco de dados e implementa a interface definida pela API. Este *driver* é fornecido pelo vendedor do banco de dados ou por outro provedor de serviço.

3.2.2.4 - JavaBeans

A tecnologia de componentes revolucionou a forma de desenvolver complexos sistemas de informação através da combinação e extensão de blocos reutilizáveis de software. Nessa linha, os JavaBeans (lançados pela Sun Microsystems em 1996) emergiram rapidamente como um importante padrão de componentes para aplicações cliente, com as vantagens de portabilidade e desenvolvimento através de ferramentas visuais. Contudo, os JavaBeans não foram projetados para criar aplicações servidoras. A JVM (Java *Virtual Machine*) possibilita que uma aplicação execute em qualquer sistema operacional – portabilidade WORA (*"Write Once, Run Anywhere™"*) –, porém componentes servidores

precisam de serviços adicionais, não providos diretamente pela JVM, mas sim por uma infraestrutura de sistemas distribuídos [ORFALI, Robert, HARKEY, Dan., 1998]

3.2.3 - Considerações sobre a J2EE

As especificações e definições da plataforma J2EE preenchem diversas lacunas existentes nos atuais ambientes corporativos, principalmente no que diz respeito ao conjunto das características: portabilidade, interoperabilidade, produtividade e consistência de ambiente, dentro de um contexto que permita evolução com escalabilidade e desempenho – propriedades essenciais em aplicações servidoras. E foi esse leque de vantagens altamente promissor que contribuiu para determinar a adoção da J2EE como plataforma de sustentação da proposta deste trabalho.

3.2.4 - Tomcat - *Servlet Container*

O *Tomcat* é um *Servlet Container*, ou seja, é um servidor onde são instaladas *Servlets* para tratar as requisições que o servidor recebe. Foi desenvolvido pela Fundação Apache para permitir a execução de aplicações para *web*. Sua principal característica técnica é estar centrada na linguagem de programação Java, mais especificamente nas tecnologias de Servlets e de Java Server Pages (JSP). Servlet e JSP são duas "formas" de se escrever programas de forma que eles tenham a interface WEB. Ou seja, são páginas HTML mais "poderosas", que ao serem chamadas pelo internauta através do browser executam programas no lado do servidor. O Tomcat é um Servlet Container, ou seja, é um servidor onde são instaladas Servlets para tratar as requisições que o servidor receber. Existem muitos Containers, mas o Tomcat se destaca por ser gratuito, ter suporte (Apache Group) e ser bastante popular.

O Tomcat está escrito em Java e, por isso, necessita que a versão Java 2 Standard Edition (J2SE) esteja instalada no mesmo computador onde ele será executado. No entanto, não basta ter a versão *runtime* de Java instalada, pois o Tomcat necessita compilar (e não apenas executar) programas escritos em Java. O projeto Jakarta da Fundação Apache, do qual o subprojeto Tomcat é o representante mais ilustre, tem como objetivo o desenvolvimento de soluções código aberto baseadas na plataforma Java.

O Tomcat está estável na versão 5, a qual foi utilizada neste projeto. Trata-se de um servidor *web* que disponibiliza páginas JSP, permitindo acesso à banco de dados pela interface JDBC. O Tomcat é o servidor de servlets que é usado na Implementação Referência Oficial para o Java Servlet e tecnologias Java Server Pages. As especificações do Java Servlet e Java Server Pages são desenvolvidas pela Sun sob o Processo da Comunidade Java. O Tomcat é desenvolvido em um ambiente aberto e participativo e liberado sob a Licença de Software Apache. O Tomcat propõe-se a ser um esforço colaborativo dos melhores desenvolvedores pelo mundo.

3.3 - XML (Extensible Markup Language)

Desde meados da última década, temos testemunhado um crescimento espantoso da *web* e, aliado a ele, observado as limitações da linguagem HTML (HyperText Markup Language).

Em resposta à crescente demanda de extensões bem como da necessidade de interoperabilidade, houve um esforço inicial de estender a linguagem HTML provendo ela com folhas de estilo em cascata, mais comumente conhecida como CSS (Cascading Style Sheet). Entretanto, essa solução constituiu-se apenas num paliativo. Como resultado, um esforço coordenado pelo W3C (*WWW Consortium*) foi movido visando oferecer uma nova linguagem que pudesse satisfazer às necessidades de interoperabilidade, escalabilidade e flexibilidade, permitindo-se fácil extensão.

Surge, então, a linguagem XML (Extensible Markup Language) que é caracterizada por prover independência de dados bem como separar conteúdo de apresentação. Um programa em XML compreende a descrição de dados, tornando-se possível seu processamento por uma aplicação. XML tem sido, cada vez mais, utilizada por desenvolvedores de aplicações devido ao suporte que ela oferece tanto a interoperabilidade quanto funcionalidade da *web*. Trata-se de uma linguagem baseada em texto a qual permite qualquer pessoa escrever um código em XML, sendo ele, facilmente, tanto compreensível às pessoas quanto manipulável aos computadores.

O suporte que a XML oferece a separação entre conteúdo e apresentação é um aspecto de suma importância na linguagem. Considere, por exemplo, um repositório de dados. XML poderia ser usada para armazenar e exibir informações estáticas desse repositório.

No entanto, adicionalmente, você ainda é permitido manipular as informações, ou seja, extrair, filtrar, buscar e ordenar informações. Essa característica juntamente com suporte a interoperabilidade constituem fatores determinantes na escolha de tecnologia para desenvolver aplicações orientadas para *web*. Além disso, o 'casamento' da linguagem XML com Java é aspecto chave para o desenvolvimento de aplicações orientadas para web.[Mendes, 2004]

Neste projeto é utilizado arquivos XML, em especial o "web.xml", que prestam um importante serviço ao desenvolvimento, permitindo que atributos de contexto, segurança, tratamento de erros, filters, listeners e outras facilidades sejam declaradas (o quê) ao invés de programadas (como). Os ganhos são muitos: maior qualidade, documentação automática e representação de conhecimento em "campo neutro" (regras em XML facilitam uma futura migração, por estarem "desamarrados" de uma linguagem específica, de repositórios de geradores, CASE ou fornecedores específicos).

Também o Struts evolui no uso de XML com arquivos como "struts-config.xml", "tiles-page.xml", "tiles-menu.xml", "validation.xml" e "validator-rules.xml", sendo que o principal deles, o "struts-config.xml", pode ser estendido pelo usuário para conter atributos próprios.

O e-Gen utiliza o XML com base de armazenamento, uma vez que todas as aplicações desenvolvidas são gravadas em arquivos XML e podem ser recuperadas posteriormente para serem editadas no IDE. A manutenção se torna desta forma extremamente rápida, pois não é necessário acessar diretamente as classes ou as páginas JSP.

3.4 - Padrões de Projeto (*Design Patterns*)

Aplicações empresariais são complexas, isto é um fato. Diariamente desenvolvedores, arquitetos, gerentes de projeto e usuários são forçados a lidar com estruturas de dados complexas, alterações em regras de negócio, mudanças de necessidades dos usuários e novas tecnologias. Naturalmente, os desenvolvedores geralmente têm menos tempo e menos recursos do que precisariam (ou gostariam) para enfrentar tudo isso.

Existem várias formas possíveis de tratar tal complexidade, mas todas podem ser agrupadas em dois princípios: utilizar abordagens comprovadamente funcionais e antecipar necessidades futuras. Em ambos os casos, existem técnicas comuns que transcende um sistema individual. Essas técnicas são conhecidas como *design patterns*. A definição mais

ampla e talvez a mais conhecida foi à citada por Christopher Alexander [Alexander, Christopher, 1997] "um Pattern descreve um problema que se repete várias vezes em um determinado meio, descrevendo o núcleo de sua solução, de modo que esta solução possa ser usada milhares e milhares de vezes".

Esses *patterns* podem ser usados repetidas vezes, facilitando a incorporação da experiência de desenvolvimentos anteriores em novos projetos. *Patterns* também fornecem uma linguagem comum para a discussão de arquitetura de software em um nível mais elevado (apenas dizer que um sistema implementa o *pattern Model-View-Controller)*, por exemplo, pode comunicar uma decisão de projeto muito mais rapidamente e mais precisamente do que uma explicação complexa, assumindo que ambos os lados saibam o que é o padrão *Model-View-Controller*.

Com a utilização cada vez maior da linguagem Java e principalmente de toda a plataforma J2EE os *design patterns* têm se tornado mais populares, uma vez que para construir uma aplicação de grande porte utilizando J2EE é imprescindível a utilização de patterns adequados.

3.4.1 - Pattern MVC - *Model-View-Controller*

Modelo 1 e *Modelo 2* são termos que foram descritos primeiramente em especificação JSP, para projetar maneiras de utilização de JSP e, embora não sejam termos mais utilizados pela especificação J2EE da Sun, são muito usados pela comunidade de desenvolvedores Java.

O modelo 1 consiste em o navegador *web* acessar diretamente as páginas JSP, que por sua vez fazem referência direta aos JavaBeans e às classes de persistência que representariam a camada de negócio da aplicação.

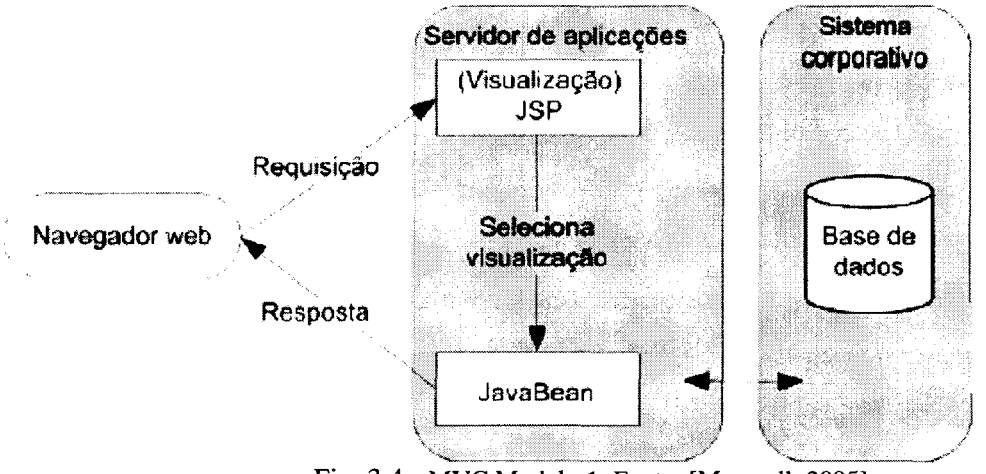


Fig. 3.4 - MVC Modelo 1. Fonte: [Maxwell, 2005]

A próxima visão a ser exibida geralmente é definida através de *links* selecionados ou parâmetros do *request* da página, deixando o controle de fluxo das páginas descentralizado. É uma maneira de desenvolvimento rápida, porém com grandes problemas para manutenibilidade. Utiliza-se o modelo 1 em aplicações pequenas e simples, onde não compensa o *overhead* causado pela modularidade do modelo MVC.

O termo *Modelo 2* era utilizado para designar aplicações JSP onde o navegador *web* acessa primeiramente um servlet de controle, que acessará o negócio do sistema e fará a validação para a decisão do fluxo das visões no sistema. Uma aplicação MVC implementa, na prática, o modelo 2. A idéia do modelo MVC é facilitar as atividades de desenvolvimento de software, utilizando orientação a objetos e dividindo os sistemas em três camadas: modelo (ou camada de negócios), visão (ou apresentação) e controle. Basicamente, esta visão busca diminuir o tempo perdido com atividades repetidas em algumas etapas do desenvolvimento de sistemas semelhantes. Desta forma obtém-se um ganho significativo em produtividade, há uma redução na quantidade de erros durante o processo de desenvolvimento e melhora a manutenibilidade e suporte dos sistemas, incrementando a qualidade do produto final.

Em desenvolvimento *web*, a divisão do modelo MVC é feita da seguinte forma:

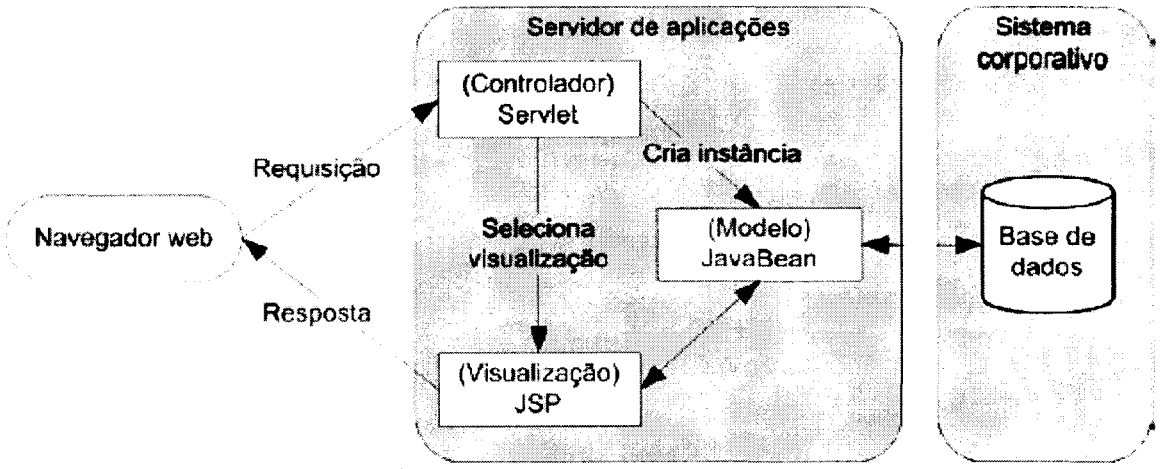


Fig. 3.5 - MVC Modelo 2. Fonte: [Maxwell, 2005]

• **Camada de Controle**: é a camada responsável pelo fluxo, qualidade e segurança das informações no sistema. É a ligação entre as camadas de negócio e apresentação.

• **Camada de Modelo**: são as atividades relativas ao próprio negócio de sistema, como classes de persistência que conversam com a base de dados, "beans" da própria modelagem do sistema e suas relações;

• **Camada de Visão**: são os formulários, relatórios - a interface em geral do sistema com os utilizadores. Elas mostram os resultados gerados na camada de modelo. Geralmente utilizam tecnologias como JSP, HTML ou XSL.

3.5 - Frameworks

Um *framework* provê uma solução para uma família de problemas semelhantes, usando um conjunto de classes e interfaces que mostra como decompor a família de problemas e como objetos dessas classes colaboram para cumprir suas responsabilidades. O conjunto de classes deve ser flexível e extensível para permitir a construção de várias aplicações com pouco esforço, especificando apenas as particularidades de cada aplicação.

Observe que um *framework* é uma aplicação quase completa, mas com pedaços faltando que, ao desenvolver um projeto, o trabalho consiste em prover os pedaços que são específicos para sua aplicação.[Johnson & Roberts, 2004]

Um *framework* captura a funcionalidade comum a várias aplicações que devem ter algo razoavelmente grande em comum, pertencer a um mesmo domínio de problema.

Um *framework* é uma extensão de uma linguagem mediante a uma ou mais hierarquias de classes que implementam uma funcionalidade e que (opcionalmente) podem ser estendidas. A *framework* pode envolver TagLibraries.

Como definição podemos dizer que um *framework* é "Uma arquitetura desenvolvida com o objetivo de se obter a máxima reutilização, representada como um conjunto de classes abstratas e concretas, com grande potencial de especialização". [Mattsson , 1996]

3.5.1 - Struts - Apache Struts Framework

O Struts Framework é um projeto *open source*, originado no Jakarta Project e mantido pela Apache Software Foundation. Foi desenvolvida com o objetivo de fornecer um *framework* para facilitar o desenvolvimento de aplicações para *web*.

É uma implementação do *design pattern* MVC (Model-View-Controller) para aplicações Java com internet. O objetivo do *pattern* MVC é separar de maneira clara a camada de apresentação (*View*) da camada de Negócio (*Model*).

A arquitetura padrão MVC - *Model-View-Controller* (Modelo-Visualização-Controle) é um padrão que define a separação de maneira independente do *Model* (Modelo) que são os Objetos de Negócio, da *View* (Visualização) que compreende a interface com o usuário ou outro sistema e o *Controller* (Controle) que controla o fluxo da aplicação. A tecnologia Struts facilita a implementação de aplicações *web* robustas e internacionalizadas, com controle automatizado de formulários. O mapeamento possibilitado pelo Struts simplifica o processo de desenvolvimento e significa uma grande economia de tempo por parte dos desenvolvedores, além de originar aplicações *web* melhor estruturadas.

3.5.1.1 - Motivos para utilizar o Struts Framework

- Se tornou um padrão de mercado, *no desenvolvimento Java para WEB*;
- Garantia de que alguém (Apache Group) irá manter a *framework* (correção de *bugs* e novos *releases*);
- Integração com a maioria das IDEs de mercado;
- Focar os esforços em regras de negócio;
- Separar a camada de negócio da camada de apresentação;
- Já incorpora diversos *design patterns*;
- Criação de aplicações padronizadas, facilitando a manutenção;
- Criação de aplicações internacionalizadas;

- Possibilidade de gerar a saída de acordo com o dispositivo usado (HTML, SHTML, WML);
- Aumentar a produtividade

Struts implementa muitas boas práticas do J2EE que existem em formas de *design patterns*. Eles são:

- Service to worker;
- Dispatcher View;
- Front Controller;
- View Helper;
- Synchronize Token.

Os dois primeiros *patterns* na verdade são macro *patterns* (*patterns* que são combinações de outros *patterns*). As conseqüências das aplicações desses *patterns* são as seguintes:

- força a modularização e particionamento da aplicação;
- facilita a administração do código;
- facilita alterações no código existente, especialmente manutenções;
- separa a parte de negócio da camada de visão.

3.5.2 - e-GEN Developer – FÁCIL GERAÇÃO

e-Gen Developer [e-Gen, 2005], é um ambiente RAD (*Rapid Application Development*), criado por brasileiros, que proporciona o desenvolvimento extremamente rápido de aplicações para *web*, totalmente escrito em Java e baseado no *framework* Jakarta Struts. Foi criado para superar os desafios fundamentais enfrentados por gerentes de TI: aumentar a produtividade, reduzir os custos de manutenção, e reduzir o tempo de treinamento e adaptação dos colaboradores.

O e-Gen disponibiliza para o desenvolvedor um conjunto de recursos visuais que tornam a criação de uma aplicação uma tarefa rápida. A manutenção das aplicações geradas também compõe um elemento importante dado o dinamismo das empresas e dos mercados.

No e-Gen, o código gerado é padronizado e todas as alterações necessárias podem ser realizadas dentro do ambiente, com acesso ás aplicações geradas anteriormente. Assim, a manutenção de uma aplicação se torna fácil e rápida e pode ser realizada por qualquer membro da equipe. Tudo sendo executado a partir de um navegador.

Nas aplicações baseadas na *web* é utilizado um conjunto de tecnologias (HTML, Java Script, XML, DHTML, APIs Java, JSP, Servlets, SQL, etc) que requerem especialização de cada membro da equipe para finalizar com qualidade suas tarefas. O e-Gen gera as aplicações utilizando as melhores práticas e padrões definidos para as diversas tecnologias.

No desenvolvimento de aplicações, o e-Gen disponibiliza um conjunto de recursos e facilidades que auxiliam no desenvolvimento de projetos *web*, sendo algumas de suas características as seguintes:

O e-Gen é 100% baseado no *framework* Struts, do projeto Jakarta. Ainda hoje o Struts é o *framework* mais utilizado em desenvolvimento de sistemas para *web*.

As aplicações desenvolvidas são gravadas em arquivos XML e podem ser recuperadas posteriormente para serem editadas no IDE. A manutenção se torna desta forma extremamente rápida, pois não é necessário acessar diretamente as classes ou as páginas JSP.

A comunicação entre a aplicação gerada e o banco de dados é feita através de um "*session facade*", possibilitando a mudança do fornecedor de banco de dados sem alteração na aplicação.

Com o e-Gen é possível validar os campos de entrada de dados, tanto do lado do cliente como do lado do servidor.

3.6 - Banco de Dados PostgreSQL

O PostgreSQL é um Sistema Gerenciador de Banco de Dados objeto relacional de código aberto. É caracterizado por sua robustez, confiabilidade, compatibilidade com os padrões SQL92/SQL99 além de ser extremamente flexível e rico em recursos. É considerado objeto relacional por implementar características de um SGBD relacional e características de orientação a objetos, como herança e tipos personalizados. Tem suporte a muitas plataformas dentre elas o Linux, IBM AIX, Sun Solaris e Microsoft Windows.

O banco de dados PostgreSQL nasceu na Universidade de Berkeley, em 1986, como um projeto acadêmico e se encontra na versão 8.0, sendo um projeto mantido pela

comunidade de Software Livre. Pela riqueza de recursos e conformidade com os padrões, ele é um SGBD muito adequado para o estudo universitário do modelo relacional.

Alguns recursos presentes no PostgreSQL em http://www.postgresql.org:

- Sub-consultas;
- Controle de concorrência multi-versão (MVCC);
- Integridade Referencial;
- Funções armazenadas (*Stored Procedures*), que podem ser escritas em várias linguagens de programação (PL/PgSQL, Perl, Python, Ruby, e outras);
- Gatilhos (*Triggers*);
- Tipos definidos pelo usuário;
- Esquemas (*Schemas*);
- Conexões SSL;
- Extensão para GIS (base de dados geo-referenciados).
- O gerenciamento de um servidor de banco de dados PostgreSQL é mais simples que os SGBDs comerciais.

A utilização do banco de dados PostgreSQL é cada vez mais ampla nas empresas que buscam um servidor de banco de dados altamente sofisticado, com alta performance, estável e capacitado para lidar com grandes volumes de dados. O fato de ser um produto Open Source, sem custos de licença, torna o PostgreSQL uma alternativa extremamente atraente para empresas que buscam um custo total de propriedade (TCO) menor para os ativos de TI. Finalmente, este banco de dados já deixou de ser uma curiosidade de técnicos e esta no *datacenter* de grandes empresas do Brasil e do mundo.

Existe uma grande variedade de banco de dados que tratam de persistência, mas para o presente trabalho adotou-se *o PostgreSQL,* por ser líder nas aplicações destinadas a *web*, além de ser o mais maduro dos bancos livres.

3.7 - DbWrench - Modelagem de Diagramas de Entidade-Relacionamento em Java

DbWrench é uma ferramenta gráfica para a manipulação do banco de dados e modelagem de dados, Modelo Entidade-Relacionamento (MER). Foi desenvolvida e otimizada para a utilização com diversos bancos de dados provendo aos seus usuários uma

forma simples e centralizada para a definição dos seus modelos. Também é possível definir os relacionamentos entre tabelas e construir as restrições (*constraints*) associadas a cada relacionamento, sendo criadas automaticamente as chaves estrangeiras nas tabelas relacionadas. Outros recursos importantes são a engenharia reversa e sincronização do modelo com a base de dados. Assim, torna-se fácil à manutenção do seu esquema de banco de dados com o MER definido no DbWrench. Além disto, é possível elaborar consultas SQL de forma gráfica a partir da utilização do "Query Mode". Suas características incluem: uma sintaxe que destaca o SQL *query editor*, o suporte para a maioria dos bancos de dados mais populares hoje em dia, entre eles Mysql, Postgresql, Microsoft SQL Server e Access, um editor gráfico do diagrama de entidade relacionamento (ERD), habilidade de criar, importar e migrar bases de dados. DbWrench é escrito em puro Java permitindo lhe o funcionamento em várias plataformas, tais como Linux, Windows, Macintosh. DbWrench torna-se ideal para ambientes de base de dados heterogêneos. Foi utilizado neste projeto pela facilidade em modelar as tabelas e migração entre bases de dados, como por exemplo, entre Mysql e Postgresql. [Nizana, 2005]

CAPÍTULO 4 – ANÁLISE E DESENVOLVIMENTO DO PROJETO

4.1 – Introdução

Neste capítulo, será abordada a análise e projeto do Cia Nutri *Manager* - Sistema de Gerenciamento On-Line da Companhia da Nutrição. Conforme citado anteriormente, o projeto não visa a completa implementação do sistema, e sim, mostrar com um protótipo, a utilização de tecnologias J2EE para o desenvolvimento de sistemas baseados na internet.

O projeto foi desenvolvido tomando-se com base a Linguagem Unificada de Modelagem – UML e as idéias propostas pela metodologia eXtreme Programming – XP. A UML foi a notação utilizada para a visualização, especificação, construção e documentação do presente projeto, embora tenha sido customizada no que diz respeito aos artefatos gerados. A XP, no seu conjunto básico de doze práticas foi seguida, com alguns pequenos ajustes. A metodologia possui uma característica mais voltada para a programação, e um pouco menos ao controle de projetos. As 12 (doze) práticas principais podem ser citadas como presentes no projeto, com algumas restrições:

1. Uso de uma metodologia rígida e de padrões de projetos: a maioria dos padrões de projeto é implementada pelo próprio e-Gen Developer, automaticamente, sem a necessidade de interferência dos desenvolvedores.

2. Pequenos lançamentos: desenvolvia-se parte do projeto e fazia-se uma verificação.

3. Metáfora: houve durante o projeto o entendimento comum pelos dois desenvolvedores sobre o funcionamento do sistema.

4. Projeto simples: tentou-se fazer com que o projeto ficasse com funcionalidades pouco complexas. Ou seja, tentava-se atingir as funcionalidades da forma mais simples possível.

5. Refatoramento: refere-se a codificação, quando o desenvolvedor observa que o código que gerou precisa passar por uma revisão – não se aplicou ao projeto, tendo em vista que a maioria dos códigos foram produzidos pela ferramenta e-Gen Developer, de forma padronizada e eficiente.

6. Teste: durante todo o desenvolvimento foram feitos testes funcionais visando antecipar os possíveis erros. Entretanto os testes foram realizados pelos próprios desenvolvedores e pelo cliente, após a geração pelo e-Gen. Não foram criadas Classes para

testar o código gerado pelo e-Gen, tendo em vista que tal ação se tornaria inviável, por conta do tempo para a conclusão do projeto.

7. Programação em pares: dois desenvolvedores para o sistema trabalhando em conjunto.

8. Propriedade coletiva: os dois desenvolvedores eram responsáveis pelo projeto, tendo o conhecimento do todo, não havendo papéis técnicos exclusivos. Qualquer um deles poderia alterar os formulários e apresentar novas funcionalidades a qualquer tempo.

9. Integração contínua: construção e testes contínuos que deveriam ser feitos diariamente, mas por dificuldades na aprendizagem da ferramenta e-Gen não foram possíveis de serem realizados de forma ideal: uma vez por dia, desde o início do projeto.

10. Semana de 40 horas: não foi possível cumprir, tendo em vista os desenvolvedores do projeto trabalharem ao longo do dia. Assim, o desenvolvimento era feito exatamente em horários impróprios, segundo a metodologia.

11. Cliente no local: o cliente fez parte da equipe de desenvolvimento, opinando durante a criação do projeto e fazendo testes funcionais ao longo do tempo.

12. Padrões de codificação: a padronização do código fonte foi elaborada, automaticamente, pelo e-Gen Developer. Houve a necessidade de editar código para a implementação do caso de uso "Realiza Compra", devido a uma funcionalidade similar aos populares carrinhos de compras.

A existência de apenas dois desenvolvedores e ser um projeto com tempo de desenvolvimento inferior a 12 (doze) meses, tornou a XP, uma metodologia ideal a ser seguida para a conclusão do projeto.

4.2 – A análise

O conceito do termo "**análise**": "s. f., decomposição de um todo em suas partes *constituintes;* exame de cada parte de um todo". [Bueno, 1989]

Para [Larman, 2001], a Análise enfatiza uma investigação do problema e exigências, em vez de uma solução. O termo "análise" é um termo geral, melhor qualificado, como análise de requerimentos (uma investigação das exigências) ou análise de objeto (uma investigação dos objetos do domínio). Já o "projeto" enfatiza uma solução conceitual que cumpra as exigências, em vez de sua implementação. Por exemplo, uma descrição de um esquema de banco de dados e objetos de um sistema. Finalmente, os projetos podem ser

implementados. Como no caso da "análise", o termo é qualificado melhor, como projeto de objetos ou projeto de banco de dados. Análise e Projeto foram resumidos na frase: *"do the right thing (analysis), and do the thing right (design)"*, que teria uma tradução aproximada: "faça a coisa certa (análise), e faça a coisa direito (projeto)". Fica um entendimento de que a "coisa certa" refere-se a negócio, enquanto a "coisa direito" refere-se a estrutura do software.

Durante a *análise orientada a objeto*, há uma ênfase na pesquisa e descrição de objetos (ou conceitos) no domínio do problema. Já durante o *projeto orientado a objeto*, há ênfase na definição dos objetos de software e como eles colaboram para cumprir as exigências. Enfim, a *análise* enfatiza "o que" o sistema deve fazer, fazendo a investigação do problema e dos requisitos, enquanto o *projeto* enfatiza "como" será descrita uma solução lógica para o problema.

A criação das melhores soluções para o desenvolvimento de software exige o acompanhamento de um processo detalhado para analisar os requisitos do projeto (isto é, determinar *o que* o sistema deve fazer) e desenvolver um projeto que atenda esses requisitos (isto é, decidir *como* o sistema deve faze-lo). Programadores profissionais sabem que a análise e o projeto podem poupar muitas horas ou mesmo evitar uma abordagem de desenvolvimento de sistemas precariamente planejados que têm de ser abandonados no meio do caminho da implementação, gerando desperdício de tempo, dinheiro e esforço. [Deitel, 2005, p. 16].

Assim, uma boa prática exige, principalmente para os grandes projetos, que o desenvolvimento de uma aplicação orientada a objeto deve ser sempre precedida de uma análise orientada a objetos. Nessa fase o trabalho fica mais voltado para a UML e seus diagramas.

4.2.1 – Análise de Requisitos

Iniciou-se o desenvolvimento do software com a criação de uma documentação de análise de requisitos, que foi realizada em conjunto com o cliente, para que fosse possível dar prosseguimento ao projeto. Conforme [Deitel, 2005], um documento de requisitos especifica o propósito geral de um sistema, e o que este deve fazer. Ele é o resultado de um processo detalhado de coleta de requisitos que poderia incluir entrevistas com possíveis usuários e especialistas da área. O detalhamento da Análise dos Requisitos do Sistema encontra-se no

Apêndice I do presente projeto. Os requisitos do sistema poderiam ter sido ajustados, ao longo do período de desenvolvimento, com *stories cards*, aonde o cliente iria especificando suas necessidades ou novas funcionalidades ao longo do tempo. Como não havia experiência no uso de tais meios, os requisitos foram ajustados, ao longo do tempo, em conjunto com o cliente, no próprio documento de análise de requisitos. A metodologia XP não é muito rígida quanto à documentação durante o desenvolvimento do sistema. A documentação gerada serve como base para que todos os membros da equipe de desenvolvimento compreendam o domínio de um problema. Segundo [Jeffries, 2005], o autor faz uma explanação sobre documentação no desenvolvimento XP. Este faz uma crítica à documentação que não esclarece conteúdo, que pode dar margem para um desenvolvedor permanecer com dúvida após sua leitura. Ele considera a conversa um bom mecanismo de esclarecimento de dúvidas de uma forma eficiente e fácil. O autor informa que faz uso de um caderno de anotações, onde anota as mais diversas coisas sobre o projeto, ou mesmo o utiliza para fazer explicações a sua equipe, com desenhos e etc. Quanto ao uso de *stories cards* ele faz uso deles quando seus clientes ou programadores lhe apresentam histórias sobre o funcionamento do sistema. Este uso de *story card* lhe facilita aspectos de planejamento, estimativas e priorizações. Ele, posteriormente, coleta várias informações dos *stories cards* para o seu caderno de anotações. Avaliando *os motivos* e *o que* escrever em um projeto, ele cita que os documentos apresentados aos clientes não devem servir como contratos de negociação, mas como formas de cada um mostrar aquilo que entendeu sobre o funcionamento do sistema. As anotações são utilizadas por ele para aprimorar um tópico, ou para relembrá-los, posteriormente. E apresenta as suas conclusões no artigo, deixando cada desenvolvedor (ou responsável pelo desenvolvimento) com uma liberdade de escolha sobre como seguir:

1. Preferir a conversa face a face para entendimento mútuo e atualização, quando houver necessidades de elaboração.
2. Preferir suas próprias anotações, na sua forma preferida, para que você não perca coisas que você julgou valiosas.
3. Documente mais formalmente para comunicar com pessoas que ficam distantes do projeto devido à distância ou quanto ao tempo. Ele ressalta para que o desenvolvedor espere que isto não funcione muito bem.

4. Documentos de contrato para entendimento somente quando houver real necessidade. Confie nos objetivos comuns, e sua voluntariedade de trabalhar junto e não fazer o seu advogado trabalhar.

4.2.2 – Diagrama de Casos de Uso

Com a finalidade de capturar o que o sistema deve fazer, foi empregada a técnica conhecida como modelagem de caso de uso. Esta modelagem identifica os casos de uso do sistema, onde cada um representa uma capacidade diferente que o sistema fornece para os usuários do sistema. Desta forma, cada Caso de Uso descreve um cenário típico no qual o usuário utiliza o sistema. [Deitel, 2005].

Na UML, os atores são representados por um desenho de traços e os casos de uso são mostrados como elipses. O diagrama de caso de uso tem como objetivo mostrar os tipos de interação entre usuários e o sistema, sem fornecer detalhes. Ele mostra as relações estruturais entre os atores e os casos de uso, não as relações dinâmicas. Esta relação é apresentada por uma associação direcional indicando a fonte da chamada. De acordo com [Ahmed, 2002], algumas perguntas são importantes para a identificação dos atores do sistema: Quem usará esta funcionalidade? Quem está fornecendo ou obtendo as informações? Quem pode mudar as informações? Há qualquer outro sistema que interage com o sistema sendo desenvolvido? Com o auxílio de tais questionamentos foi possível identificar os atores do sistema: Funcionário e Gerente, como usuários do sistema em si; e Consultor Cheque e Sistema Financeiro, como outros sistemas que interagem com o sistema sendo desenvolvido. No presente projeto, foi definido um grupo geral de ator (Funcionário) que foi especializado (Gerente), utilizando a generalização de relacionamentos. Para [Ahmed, 2002], uma maneira de descobrir os casos de uso é tentar identificar o comportamento ou as informações que o ator requer do sistema. Seguindo esta linha de raciocínio, foram identificados os casos de uso do projeto. A Fig. 4.1 mostra os principais casos de uso do sistema, que são chamados pelos atores "Gerente" e "Funcionário". O diagrama de caso de uso, geralmente, pode ser acompanhado por texto que descrevem cada caso de uso de uma forma mais detalhada. Neste projeto, os casos de uso principais do sistema "Pesquisa Produto" e "Realiza Compra" foram descritos de forma mais detalhada, conforme Apêndices IV e V, respectivamente.

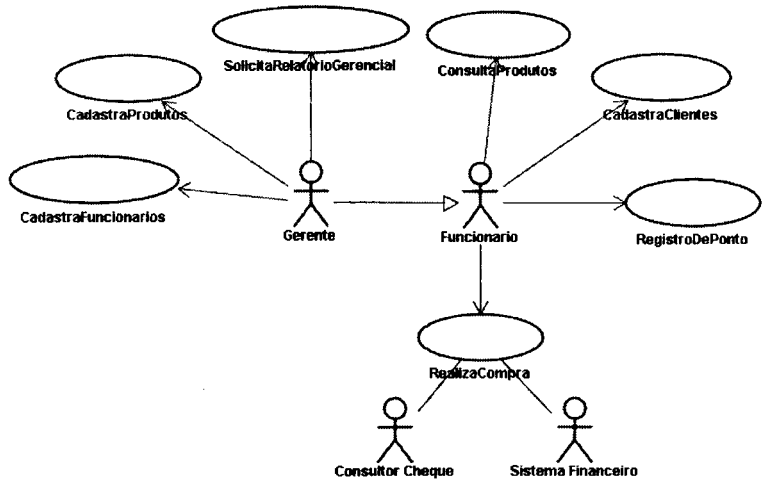


Figura 4.1 – Principais casos de uso do sistema

4.2.3 – Diagramas de seqüência

Passada a criação dos Diagramas de Casos de Uso, de acordo com [Deitel, 2005], na próxima etapa, engenheiros de software (projetistas de sistemas) analisariam o documento de requisitos ou um conjunto de casos de uso e projetariam o sistema antes que os programadores implementassem em uma linguagem de programação específica. Nesta etapa de análise, os projetistas devem produzir uma especificação de alto nível que descreve *o que* o sistema deve fazer.

Para [Ahmed, 2002], um Diagrama de Seqüência é usado para expressar o caso de uso em uma terminologia mais precisa e técnica, sendo feito como uma descrição do caso de uso em termos de interação entre o ator e o sistema. Este diagrama enfatiza a ordem do tempo da interação, onde o eixo vertical representa a dimensão do tempo. Nos diagramas de seqüência das figuras 4.2 e 4.3 há uma descrição textual do caso de uso, de forma seqüencial, em notas. As linhas verticais são usadas para mostrar a duração da interação entre o ator e os objetos do sistema. As setas horizontais indicam a direção da interação. Estas, geralmente, são identificadas com a descrição da ação sobre sua linha. As setas tracejadas indicam uma resposta à requisição feita. A Fig. 4.2 expressa o diagrama de seqüência do caso de uso ConsultaProdutos, enquanto a Fig. 4.3 expressa a interação dinâmica entre os participantes do caso de uso RealizaCompra.

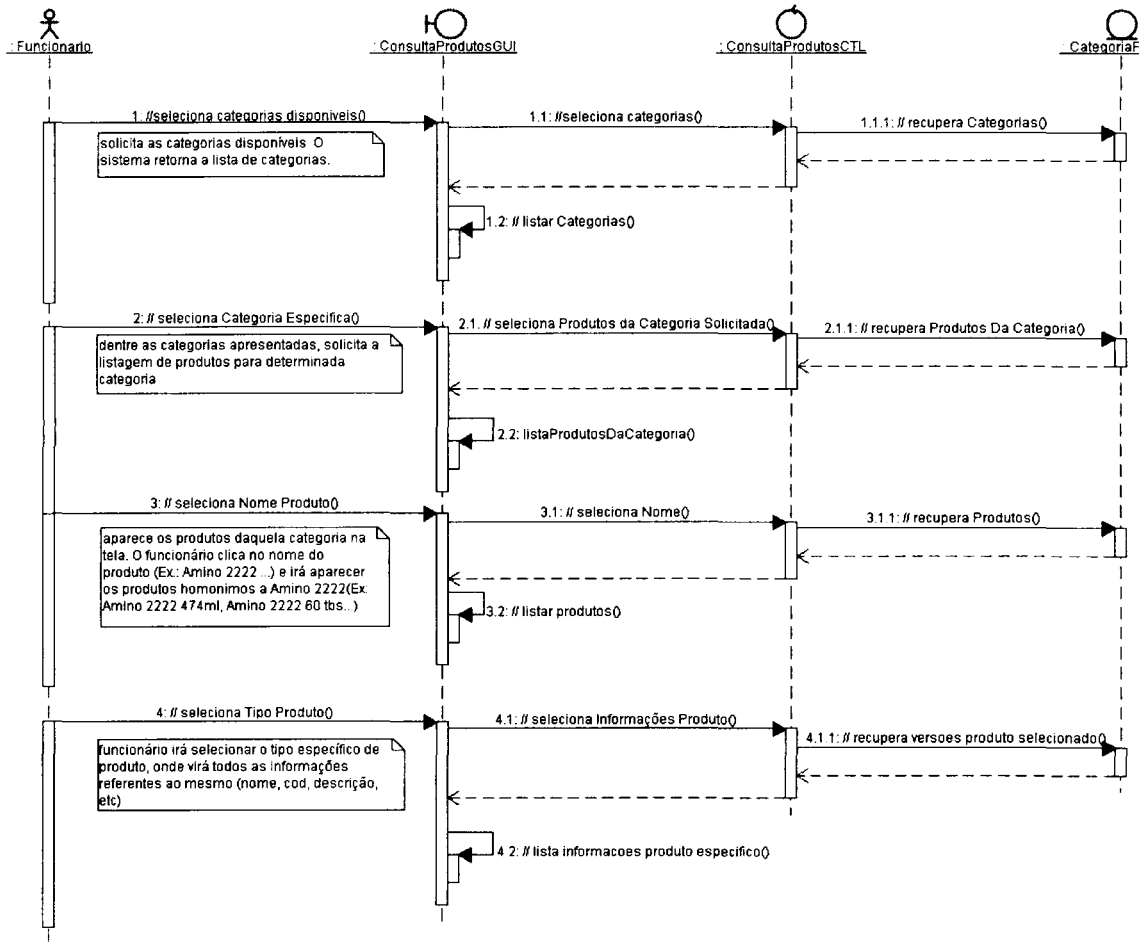


Figura 4.2 – Diagrama de Seqüência do Caso de Uso ConsultaProdutos

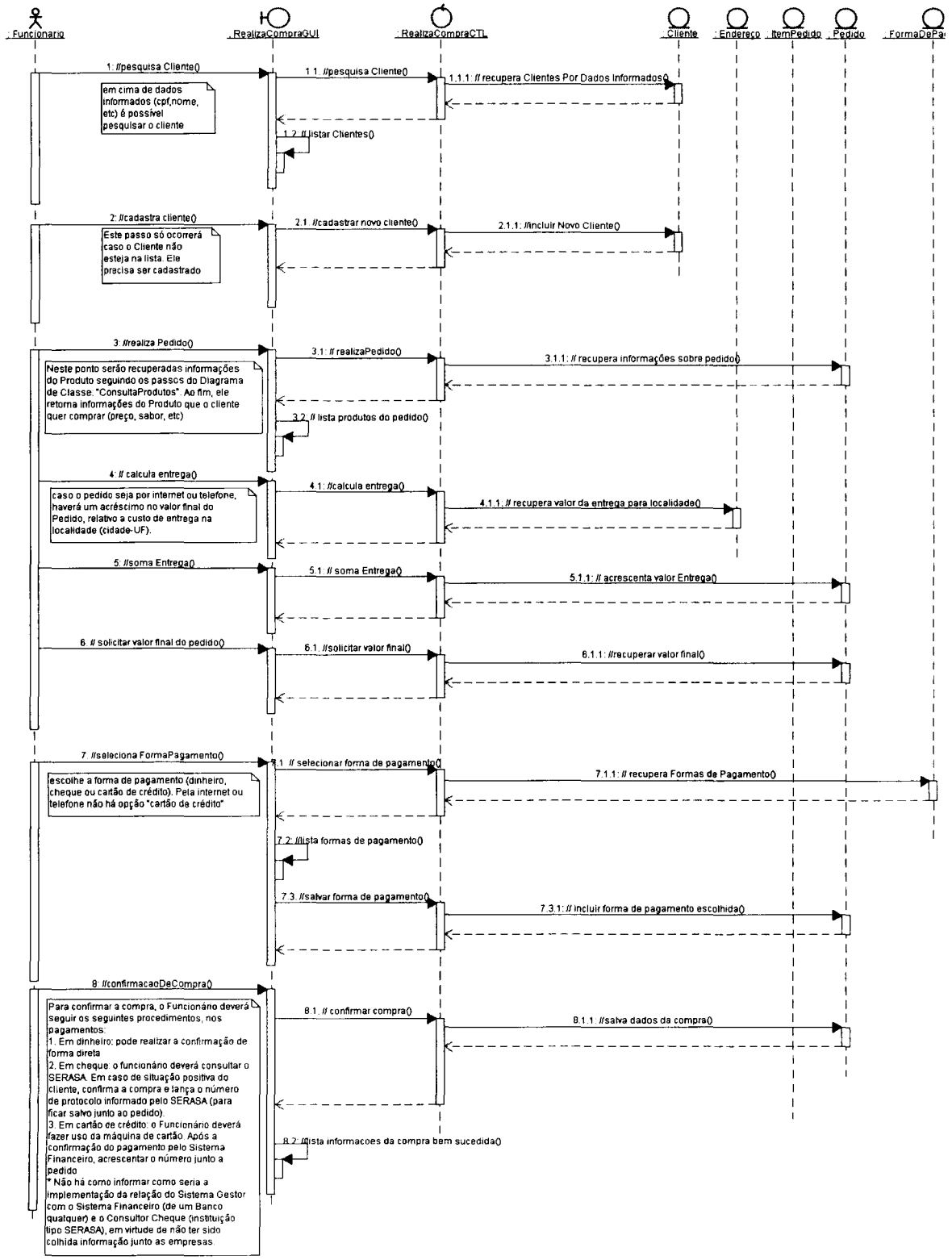


Figura 4.3 – Diagrama de Seqüência do Caso de Uso RealizaCompra

4.2.4.– Diagramas de classe

Para [Booch et al., 2000] um "diagrama de classe é um diagrama que mostra um conjunto de classes, interfaces e colaborações e seus relacionamentos". De acordo com [Ahmed, 2002], o Diagrama de Classe UML é útil para capturar as relações estáticas entre os diferentes elementos estruturais.

As classes que participam dos principais casos de uso foram identificadas e inseridas em diagramas de classes. As figuras 4.4, 4.6 e 4.7 apresentam operações de análise representando as responsabilidades atribuídas à classe. As operações de análise são representadas (inseridas) no terço inferior do retângulo que representa a classe. No terço médio das classes são identificados os atributos da classe, que representam as informações que podem ser solicitadas da classe por outras pessoas ou pela própria classe, para cumprir suas responsabilidades. As informações modeladas como atributos irão requerer um comportamento simples, como as operações *get* ou *set*. Os diagramas de classes das figuras abaixo apresentam, ainda, relações entre as classes, com predominância para as associações. As relações entre as classes apresentam número ou asterisco a fim de indicar a multiplicidade do relacionamento. Os elementos dos diagramas de classes foram dispostos de forma a minimizar o cruzamento de linhas e deixar os itens semanticamente relacionados próximos.

Um único Diagrama de Classe, referido como diagrama Exibição das Classes Participantes (VOPC), é criado para cada caso de uso. Sua finalidade é mostrar, em um único diagrama, todos os aspectos da arquitetura do sistema que são exercidos por um caso de uso específico. No projeto do Sistema de Gerenciamento On-Line da Companhia da Nutrição, foram criados diagramas VOPCs para os casos de uso Consulta Produtos e Realiza Compras, que podem ser observados nas figuras: 4.4, 4.5 e 4.6.

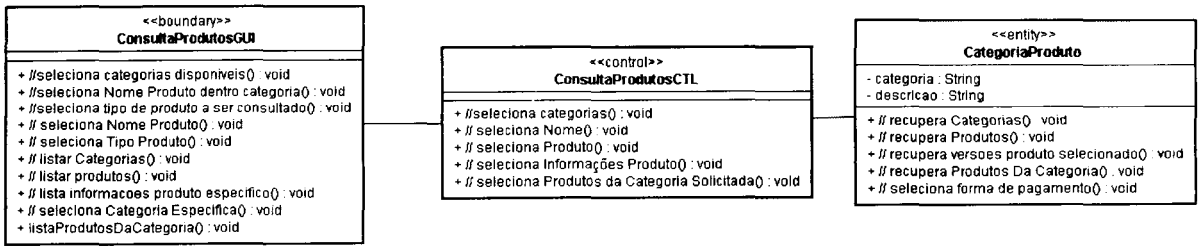


Figura 4.4 - VOPC – Consulta Produtos

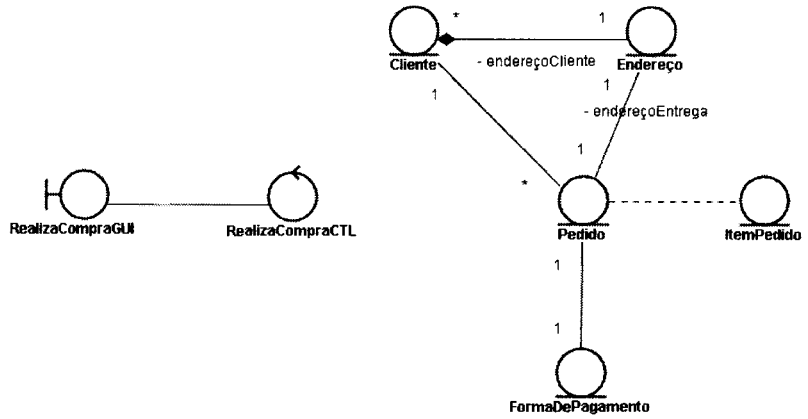


Figura 4.5 - VOPC – Realiza Compras com Ícones

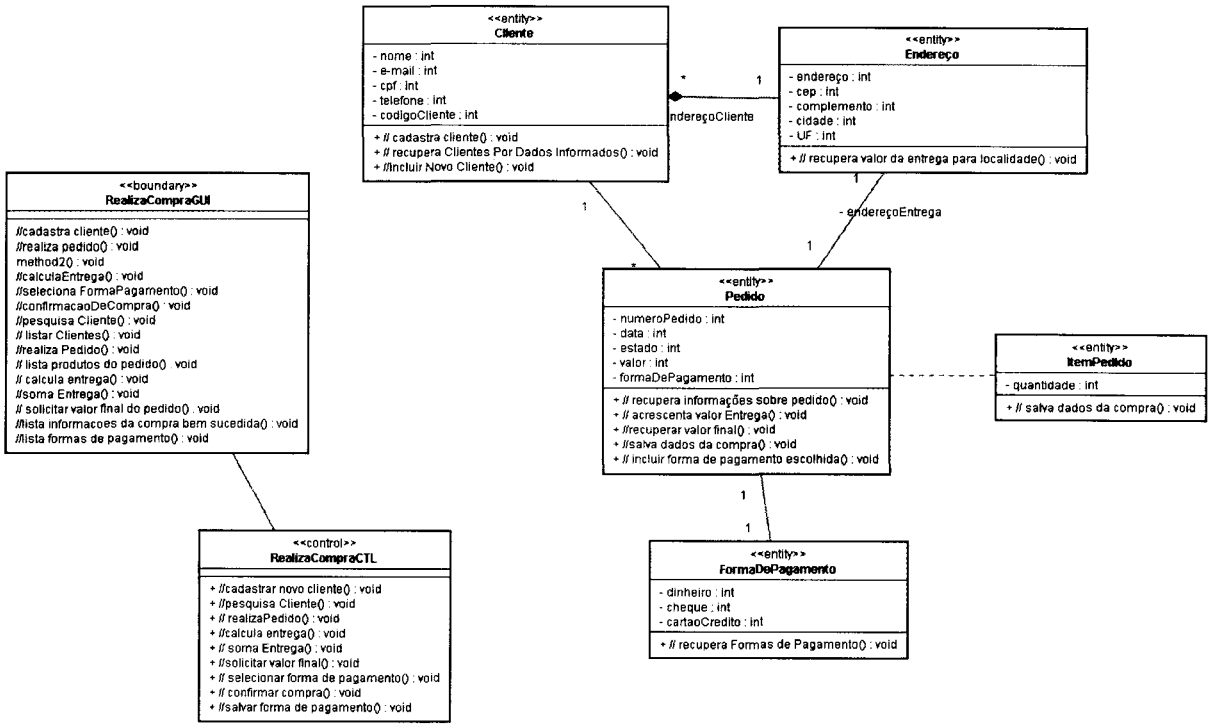


Figura 4.6 - VOPC – Realiza Compras

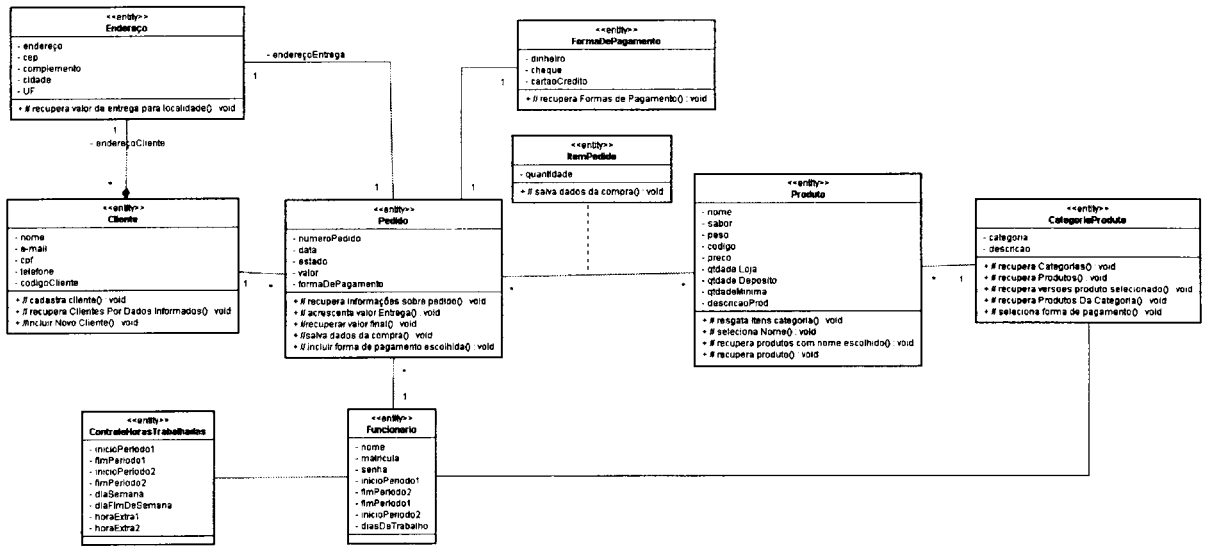


Figura 4.7 – Parte do Diagrama de Classes do Projeto

4.3 – Projeto

As diferenças entre Análise e Projeto já foram abordadas em tópico anterior. Basicamente a Análise refere-se a estudos sobre "o que" o sistema deve fazer, enquanto o projeto relaciona-se a "como" o projeto deve ser desenvolvido.

Com base na documentação elaborada, iniciou-se o desenvolvimento do Sistema de Gerenciamento On-Line da Companhia da Nutrição. É importante esclarecer que o protótipo não visa apresentar exatamente o que será a aplicação na sua conclusão, uma vez que as funcionalidades poderão ser implementadas de diferentes formas, mas atingir um objetivo comum. Além disso, nem todos os casos de uso do sistema serão implementados, principalmente por conta do tempo de desenvolvimento que seria exigido para tal.

A fim de facilitar o entendimento sobre o sistema, seguem algumas nomenclaturas relativas ao mesmo:

Loja é o espaço físico legalmente estabelecido para a venda dos produtos.

Depósito é um local distante da loja e pode ser compreendido com um "escritório" onde são armazenados os produtos, em virtude de não haver espaço suficiente na loja.

4.3.1. – Login

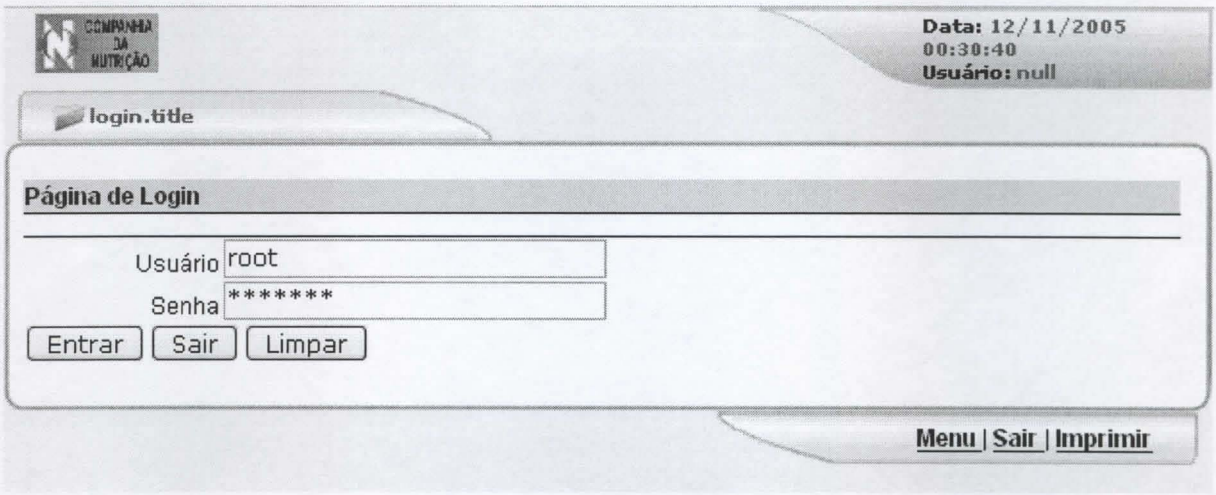


Figura 4.8 - Tela de Login do Sistema

O início do sistema deve contemplar um *login* para os usuários (Gerente ou Vendedor), a fim de diferenciar o restante das funcionalidades disponíveis pela função. O sistema deverá entender o tipo de usuário que o acessa somente pela matrícula do mesmo, viabilizando o acesso ao Menu Principal para cada tipo de usuário. Caso o usuário falhe na inserção da matricula ou da senha, o sistema deverá apresentar informação ao usuário, para que o mesmo tente novamente. A tela inicial terá três botões, para *incluir* os dados, *limpar* os dados ou *sair* da aplicação.

O sistema deverá contemplar autenticação baseada no JAAS (Java Authentication and Authorization Service) para fazer o *login* baseado em informações de um banco de dados. No e-Gen é nativa a autenticação com JAAS, sendo necessário fazer alguns ajustes em alguns arquivos.

A tela possuirá o nome fantasia e a logomarca da empresa, a fim de dar maior identidade visual.

4.3.2. – Menu

O menu será a primeira tela funcional do sistema. O menu principal do sistema será formado por uma pasta (PAINEL DE ADMINISTRAÇÃO) que é dividida em duas subpastas: PRINCIPAL e AUXILIAR. A apresentação será diferente para cada tipo de usuário. Sendo *gerente*, as duas subpastas estarão disponíveis. Sendo um *funcionário*, somente a pasta PRINCIPAL estará disponível. Tal ação visa impedir a viabilidade do funcionário vir a cadastrar ou alterar dados do sistema. O único cadastro que será permitido ao *funcionário* será

o cadastro de "clientes", uma vez que, diariamente, para cada compra, novos clientes comparecem na loja.

Os diversos subitens de cada subpasta funcionarão como links, que direcionarão o sistema para a página solicitada pelo usuário.

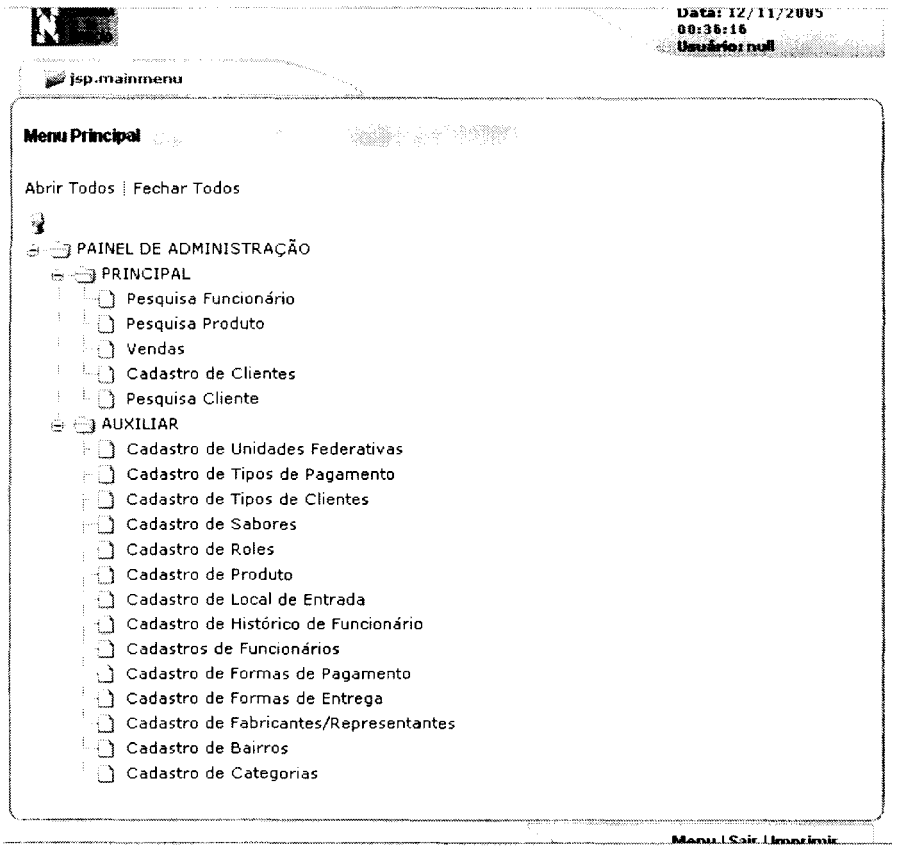


Figura 4.9 - Tela de Menu Principal

4.3.2.1. – Menu Principal

Os itens apresentados a seguir, não necessariamente precisarão se apresentar nesta ordem. Tal apresentação tem motivo didático para melhor elucidação do projeto.

4.3.2.1.1. – Sub-menu Pesquisa Funcionário

O link "Pesquisa Funcionário" deverá encaminhar o usuário para o formulário de pesquisa dos usuários do sistema.

Os formulários de pesquisa deverão, em geral, possuir dois botões: *botão de pesquisa* para submeter uma pesquisa frente aos campos preenchidos pelo usuário no formulário, e um *botão de limpar* os campos preenchidos, para iniciar uma nova pesquisa. Abaixo de cada um

dos formulários de pesquisa existirá um Relatório, com o retorno dos resultados para aquela pesquisa.

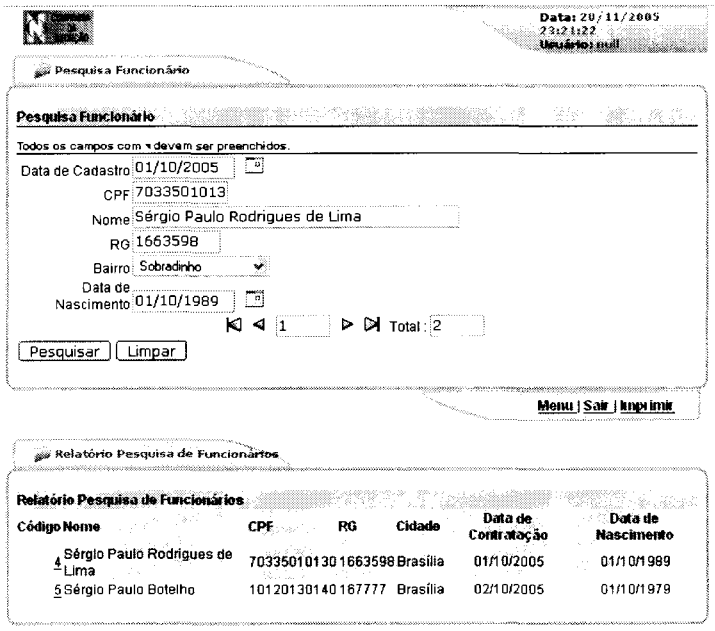


Figura 4.10 - Tela de Pesquisa de Funcionário

4.3.2.1.2. – Sub-Menu Pesquisa Produto

O link "Pesquisa Produto" deverá encaminhar o usuário para o formulário de pesquisa de produtos disponíveis na base de dados.

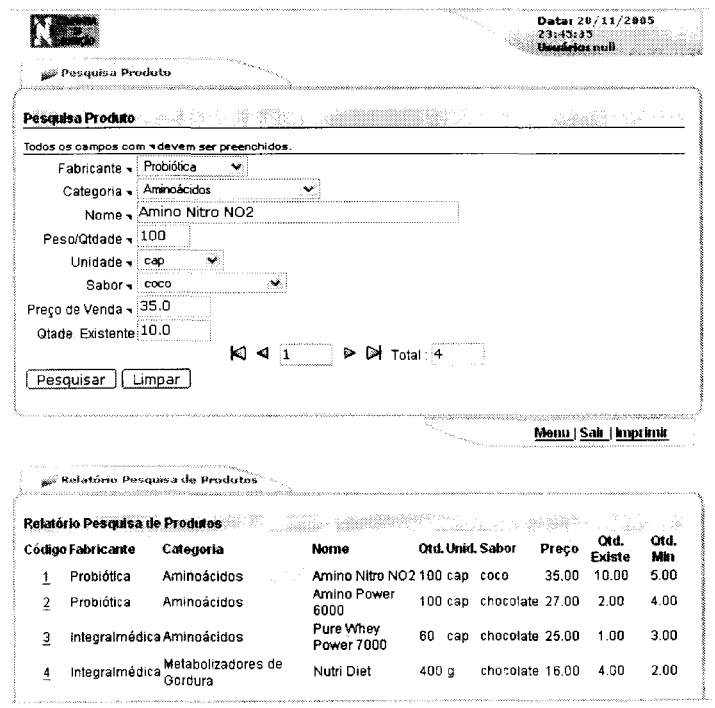


Figura 4.11 - Tela de Pesquisa de Produto

4.3.2.1.3. – Sub-Menu Pesquisa Cliente

O link "Pesquisa Cliente" deverá encaminhar o usuário para o formulário de pesquisa de clientes cadastrados na base de dados.

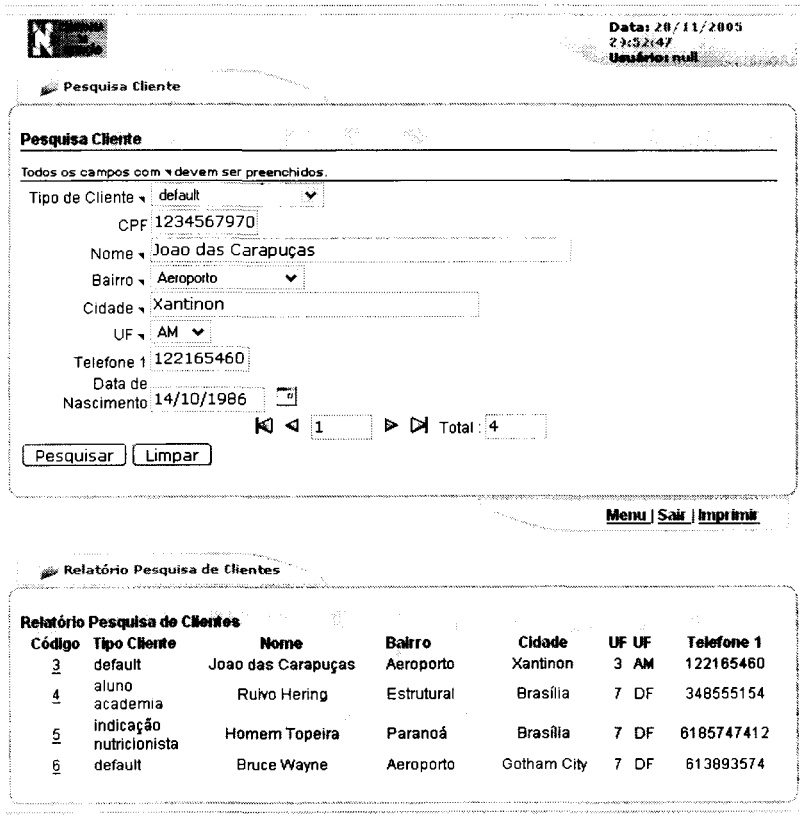


Figura 4.12 - Tela de Pesquisa de Clientes

4.3.2.1.4. – Sub-Menu Cadastro de Clientes

O link "Cadastro de Clientes" deverá encaminhar o usuário para o formulário de cadastro de clientes, a fim de possibilitar o armazenamento dos dados de novos clientes.

O formulário "Cadastro de Clientes" possuirá o campo código auto-incrementável e não editável. Assim, bastará o usuário preencher os dados do cliente e clicar no botão "inserir". O formulário ainda deverá dispor dos botões de *limpar* campos, *pesquisar* clientes, *alterar* dados e *excluir* clientes.

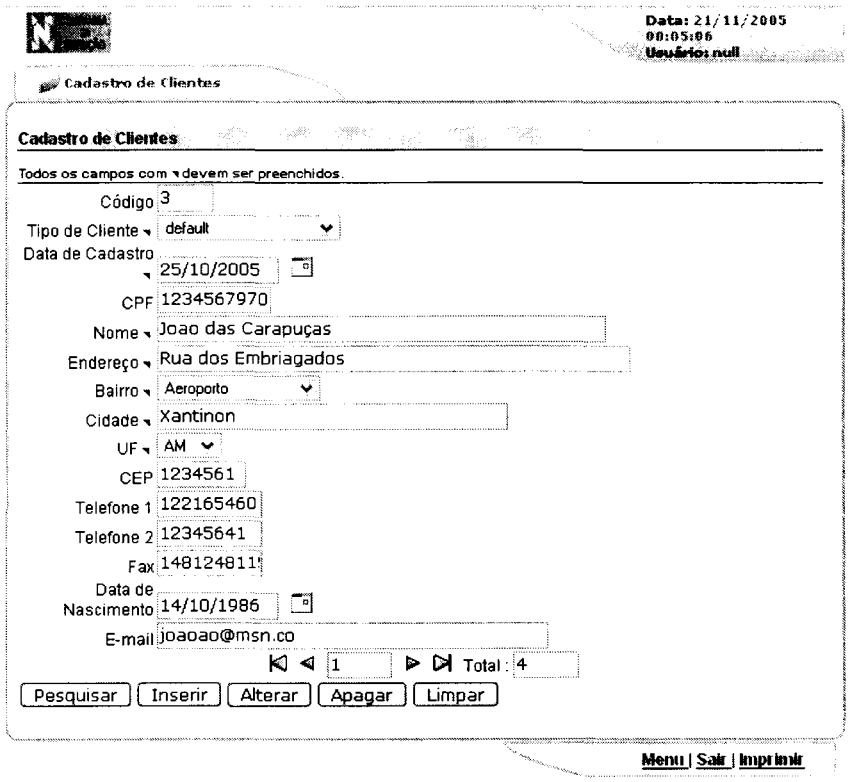


Figura 4.13 - Tela de Cadastro de Clientes

4.3.2.1.5. – Sub-Menu Formulário de Vendas

O formulário de vendas será o formulário que ajustará o caso de uso Realiza Compra. Ele será o responsável por popular os dados necessários para a venda.

Na identificação do cliente, haverá um botão de LOV (*List Of Values*), que possibilitará acessar um "popup" com um formulário de Pesquisa Cliente onde, após a pesquisa, retornarão os dados para os campos de identificação do cliente.

No campo "Data de Venda", haverá um calendário, ao lado do campo, que facilitará a escolha da data, e que é padrão do e-Gen. Ou seja, todos os campos do tipo data possuem tal calendário para facilitar a inserção da data.

Figura 4.14 - Calendário dos campos Data

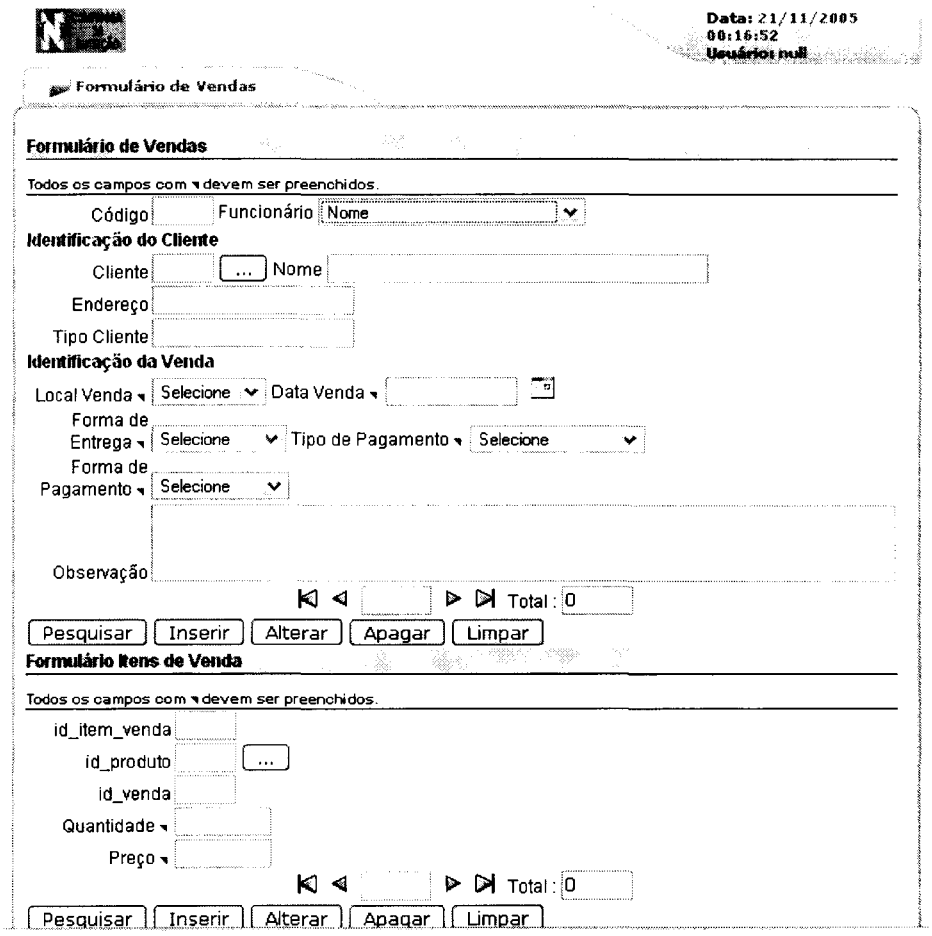


Figura 4.15 - Formulário de Vendas

4.3.2.2. – Menu Auxiliar

O menu auxiliar caracterizar-se-á por uma ter uma série de sub-itens com links para formulários de cadastros diversos. Estes cadastros darão suporte aos campos dos formulários principais. Os mesmos, poderiam ter seus valores acrescentados direto na base de dados, o que iria exigir uma certa especialização do cliente. Visando facilitar alterações futuras, os cadastros do Menu Auxiliar foram dispostos para o usuário "gerente" a fim de facilitar o seu controle sobre o sistema.

CAPÍTULO 5 - CONCLUSÃO

5.1 – Conclusões

As definições do Cia Nutri Manager – Gerenciador On-Line da Companhia da Nutrição foram feitas de forma a atender as necessidades exclusivas da empresa. A participação conjunta dos desenvolvedores e do cliente durante o processo de desenvolvimento foi importante para a criação de um protótipo eficiente.

O uso da notação UML facilitou o entendimento comum dos dois analistas sobre o que o sistema deveria fazer, além de servir como um mapa para orientar o cliente como se daria o desenvolvimento. Assim, foi possível ao cliente, durante todo o projeto, apresentar críticas e sugestões sobre *o que* o sistema deveria fazer, no momento em que as idéias da equipe eram formuladas, informando se as funcionalidades sugeridas pela equipe eram as almejadas pela a empresa.

O uso da metodologia XP foi uma nova experiência para todos os membros da equipe. Os seus conceitos e propostas foram colocados em prática com restrições, conforme citado ao longo do capítulo 4. Dentre as 12 (doze) práticas da metodologia, mereceram destaque:

- Metáfora: o entendimento comum pelos dois desenvolvedores sobre o funcionamento do projeto facilitava a troca de experiências e a solução de determinados problemas;
- Programação em pares: a prática da programação em pares foi eficiente porque os desenvolvedores trocavam experiências no uso do ambiente RAD e-Gen, que não era dominado por nenhum dos dois;
- Cliente no local: o cliente, como usuário final e principal interessado na desenvolvimento do protótipo, contribuiu esclarecendo dúvidas e dando sugestões durante a criação do projeto e fazendo testes funcionais ao longo do tempo. Ele foi importante porque era o responsável por homologar uma funcionalidade, para que a equipe pudesse dar continuidade em outras funcionalidades do sistema;
- Padrões de codificação: a padronização do código fonte foi elaborada, automaticamente, pelo e-Gen Developer. Houve a necessidade de editar código para a implementação do caso de uso "Realiza Compra", devido a uma funcionalidade similar aos populares carrinhos de compras e que não é customizada, automaticamente pelo e-Gen. Ainda assim, o tempo de

desenvolvimento de tal funcionalidade no e-Gen foi extremamente rápido e eficiente.

Uma característica das metodologias ágeis que foi importante para um desenvolvimento mais rápido de protótipo foi a valorização de "*Software que funciona* mais que documentação abrangente" [Beck, et al., 2001]. Assim, o tempo que se perderia em uma documentação detalhada, era utilizado para a correção de erros ou para o desenvolvimento de novas funcionalidades. É importante considerar, também, que em virtude do uso do e-Gen Developer como ferramenta para produção de código, há uma facilidade maior nas futuras manutenções do sistema, devido à padronização da ferramenta.

Após o período de análise, foram pesquisadas as melhores ferramentas que pudessem atender ao desenvolvimento do sistema, chegando-se ao e-Gen Developer como a melhor solução para o desenvolvimento rápido do sistema com qualidade, robustez, facilidade e padronização. Desta forma, a utilização do e-Gen como ferramenta de produtividade contribuiu para:

- redução dos custos de implantação – tanto de treinamento, quanto de compra de licenças;

- a diminuição das chances de erro devido à geração automática do código;

- a geração de projetos de forma padronizada; e

- estar alinhado com desenvolvimento de sistemas voltados para internet que hoje é uma realidade do mercado.

A adoção deste modelo trouxe benefícios para a equipe, possibilitando desenvolver protótipos a serem validados pelo usuário do sistema, reduzindo drasticamente o tempo de desenvolvimento. Outra vantagem é que não foi necessário na equipe de desenvolvimento vários especialistas nas mais diversas áreas que envolvem um sistema voltado para *web*, visto que muito destas especializações já estão encapsuladas na ferramenta, junto a um conjunto de tecnologias (HTML, XML, Java Script, DHTML, APIs Java, JSP, Servlets, SQL, etc.). Assim, o uso do *RAD* e-Gen em conjunto com a metodologia XP, na sua prática *propriedade coletiva* (não há papéis técnicos exclusivos) gerou uma maior velocidade no desenvolvimento do projeto, onde a ferramenta e a metodologia seguiam a mesma filosofia: *software* de qualidade, de forma rápida e padronizada.

Os formulários da aplicação puderam ser divididos em módulos para facilitar o gerenciamento do projeto de desenvolvimento e da aplicação gerada.

O uso do e-Gen, por ser baseado no *framework* Jakarta Struts, agrega todas as características e padrões de projeto deste *framework*.

Quanto à padronização de código, foi possível ser observado, em comparação com o *Code Conventions for the Java™ Programming Language* (Convenções de Código para a linguagem de programação Java) [Sun, 1999] algumas características do código gerado pelo e-Gen, como: geração adequada de sufixos; nome dos arquivos padronizados; comentários de início de código com data de criação e modificação; declaração de pacotes e de *imports* na ordem adequada; uma declaração por linha, com identação adequada; não possui linha em branco entre métodos, mas possui linha em branco entre classes e possui espaços em branco adequados entre caracteres específicos; respeita a convenção de nomes para pacotes, classes, interfaces, métodos, variáveis e constantes; respeita as práticas de programação mais comuns. Como práticas condenáveis segundo as convenções, poderiam ser citadas: tamanho de algumas linhas geradas superior aos 80 (oitenta) caracteres permitidos; separação (partição) adequada em outras linhas, a fim de que não fique tão longa a linha; além do comentário de bloco com a data de criação e modificação do arquivo, não constam outros comentários explicando o código gerado.

Apesar de não ser escopo deste projeto, é interessante ressaltar a integração do e-Gen Developer com a ferramenta de desenvolvimento Eclipse, uma das *IDEs* mais utilizadas para desenvolvimento de projetos em Java, no mundo. Vale ressaltar, também, que a própria RAD foi escrita dentro do ambiente Eclipse. Assim, o Eclipse já reconhece os diretórios onde se encontram os fontes do projeto e o diretório onde são compiladas as classes. Este ambiente é muito utilizado pelos desenvolvedores e-Gen, por facilitar a criação de código para o desenvolvimento de regras de negócio.

A ferramenta DbWrench, ferramenta escrita em Java, foi uma opção eficiente para a modelagem de banco de dados e apresentou como uma das principais vantagens a geração de scripts "limpos", facilitando a criação de banco de dados. É importante ressaltar que outras ferramentas foram testadas, mas todas geravam scripts que implementavam banco de dados inadequados, fazendo o e-Gen não conseguir compilar as classes geradas, dando diversos erros. A pesquisa por uma ferramenta eficiente também demandou tempo da equipe, o que de alguma forma, atrasou o desenvolvimento do projeto.

O protótipo apresenta as vantagens do uso de tecnologia Java, ou seja, ter uma linguagem simples, orientada ao objeto, distribuída, interpretada, robusta, segura, de arquitetura neutra, portável, de alto desempenho, *multithreaded*, e dinâmica.

5.1.1 – Dificuldades

O início do período de desenvolvimento foi marcado por uma grande dificuldade na configuração do RAD e-Gen Developer. Isto porque, para que se consiga que a ferramenta funcione adequadamente, faz-se necessário seguir à risca, sem qualquer personalização, todos os tutoriais propostos no sítio do desenvolvedor. Assim, caso a versão de determinado programa não fosse exatamente o prescrito no sítio, poderia haver algum tipo de incompatibilidade ou necessidade de conhecimento (não disponibilizado no sítio) específico da ferramenta para ajustes e adequação. Alguns destes problemas poderiam ter sido resolvidos com ajustes em alguns arquivos XML. Entretanto, tais ajustes ou adequações não estão prescritos em forma de artigos ou tutoriais, o que dificulta aos iniciantes a permanecerem autodidatas, frente à ferramenta. Com isso, a demora na configuração da ferramenta acabou gerando contra-tempos e desmotivação da equipe de desenvolvimento, uma vez que suas ações ficavam interrompidas por problemas não compreendidos até então, e não presentes em qualquer fonte de consulta. Visando que tais infortúnios não venham a ocorrer com futuros desenvolvedores, foi criado um tutorial de instalação das ferramentas utilizadas pela equipe, que pode ser conferido no Apêndice VI – Tutorial de Instalação dos Programas. Tal tutorial deverá ser seguido à risca, uma vez que alterações no mesmo serão causas de problemas na geração de código ou configuração da ferramenta. De alguma forma, os desenvolvedores acabam tendo uma dependência do ambiente de desenvolvimento, uma vez que o e-Gen gera as aplicações utilizando as melhores práticas e padrões definidos para as diversas tecnologias, mesmo quando o desenvolvedor não as domine completamente. Por outro lado, a inclusão ou substituição de um membro da equipe pode ser realizada de forma rápida e com baixo custo.

O RAD e-Gen Developer possui pouca documentação, principalmente para o desenvolvimento de projetos mais avançados. Não existem livros sobre a ferramenta. As informações possíveis de serem consultadas estão disponíveis nos tutoriais presentes no sítio do desenvolvedor [e-Gen, 2005]. Quaisquer outras formas de suporte técnico somente são possíveis pela lista de discussão do projeto e-Gen, subordinado ao Sou Java, no sítio do Java.Net [e-Gen, 2005]. Assim, ao longo do desenvolvimento diversas dúvidas não foram respondidas ou quando eram respondidas, demoravam. Desta forma, funcionalidades simples demoravam a serem implementadas por falta de experiência na ferramenta.

A falta de uma maior experiência por parte dos integrantes do grupo com a metodologia XP provocou uma necessidade de estudo do processo a fim de tentar adaptá-lo ao projeto, de uma forma eficiente, o que demandou um tempo maior para início efetivo dos trabalhos. Além disso, poucas pessoas conhecidas dos integrantes da equipe tiveram

experiências concretas com a metodologia, o que dificultou, ainda mais a troca de experiências a fim de evitar falhas.

5.2 – Restrições

Alguns casos de uso, dentre os quais: Consultor cheque, Sistema Financeiro, RegistroDePonto e SolicitaRelatorioGerencial não foram implementados em virtude, principalmente, do tempo restante disponível para desenvolvimento. No caso dos dois primeiros, ainda há a necessidade de estudos para adequação de código para que as requisições se adequem ao sistema de empresas terceirizadas.

Apesar da metodologia XP não ser rígida quanto à geração de documentação, dando ênfase à prática de uma codificação limpa e padronizada e a criação de um software de qualidade em detrimento de uma documentação aprimorada; não foi criada documentação, mesmo para manutenção, por dois motivos principais: boa padronização do código criado pelo e-Gen Developer e inviabilidade da equipe ficar revendo código gerado pela ferramenta e gerando documentação a partir daí.

Falta de experiência com o processo e a ferramenta acabou por restringir o desenvolvimento mais rápido do projeto e o término de alguns casos de uso.

Não foram cadastrados funcionários ou clientes reais da loja, para preservar seus dados.

No protótipo não foram cadastrados todos os produtos em todas as categorias, tendo em vista serem mais de 270 (duzentos e setenta) itens, com mais de 04 (quatro) sabores, em média, cada um. Assim, haveria o trabalho de se fazer, aproximadamente, 1000 (mil) cadastros, o que ficou a cargo do cliente, quando o sistema for instalado na loja. Desta forma, o protótipo, após ter seu banco de dados populado, já conseguirá facilitar o gerenciamento da empresa no que tange ao processo de vendas, cadastros em geral e controle de estoque.

O sistema não foi implantado na loja, para testes no local, em virtude da falta de uma infra-estrutura de rede. Os testes foram realizados pelos próprios desenvolvedores e pelo cliente, após a geração pelo e-Gen, somente através de testes funcionais. Não foram criadas classes para testar o código gerado pelo e-Gen, tendo em vista que tal ação se tornaria inviável, por conta do tempo para a conclusão do projeto.

O *layout* externo, mostrado nas figuras do protótipo, para melhorar a aparência da aplicação, não apresentou todas as compatibilidades do layout padrão. Com isso, foi necessário retornar para o *layout* padrão gerado pelo e-Gen, tendo em vista não ter sido alcançado suporte para corrigir falhas apresentadas.

5.3 – Trabalhos Futuros

Alguns aspectos não foram tratados no presente projeto. Eles servem como estímulo para inclusão de novas funcionalidades por outras equipes, incluindo a nossa. Dentre elas, podem ser citadas:

- Conclusão dos módulos previstos na modelagem para o sistema, dentre os quais: o módulo de controle horário dos funcionários e o módulo de relatórios ao gerente, onde poderão ser criados gráficos de produtos mais vendidos, aquisições de clientes por bairros ou por local de atendimento da empresa (loja, telefone ou internet), entre outros, para melhor focar as necessidades dos clientes.
- Implementação da tecnologia J2ME: gerenciamento da empresa e geração de relatórios resumidos por sistemas móveis. Poderiam ser integradas funcionalidades como a emissão de relatórios padronizados ao proprietário, ao fim do dia, a fim de mantê-lo ciente sobre dados importantes de sua empresa.
- Módulos móveis: módulos que possam ser utilizados em PDAs (Assistentes Pessoais Digitais) ou sistemas móveis que envolvam tecnologias como J2ME, WAP, etc. para pontos de vendas fora da empresa. Ou seja, funcionários poderiam fazer uso deste sistema para realizar vendas ou cadastrar novos clientes.
- Segurança: uso de protocolos HTTPS, dificultando, ainda mais, o acesso não autorizado às informações da empresa.
- Testes: o projeto não foi objeto submetido a testes de unidade, realizado por programas específicos para tal, como o JUnit. Isso decorreu também do esgotamento do término do prazo e das dificuldades de se criar testes sobre o código gerado pelo e-Gen.

- Integração com os fornecedores: fornecer interface entre os fornecedores e a empresa (*web service* para baixa de estoque e atualização de novos produtos, arquivos e etc.).
- Integração com os clientes: criação de novo módulo onde o próprio cliente, previamente cadastrado, possa disparar o pedido de compra diretamente de uma página *web*, sem ter a necessidade de ir até o ponto de venda.

5.4 – Considerações finais

O presente projeto foi além de uma grande experiência para os membros envolvidos no desenvolvimento, um grande desafio. O desafio decorreu do fato de iniciar um estudo em cima da implementação de um protótipo que se adequasse às necessidades reais de uma empresa, com diversas exigências para tornar o seu processo de gerenciamento mais adequado ao mundo competitivo atual.

Foi possível observar que a experiência conta muito ao longo do período de desenvolvimento. A falta de informação e experiência no uso das tecnologias, ferramentas e processos selecionados trouxe um grande prejuízo em termos de tempo de desenvolvimento, porque foi necessário aprender para depois iniciar o desenvolvimento. O tempo que se perdeu para instalar um *RAD* como o e-Gen Developer de maneira adequada e, o tempo perdido na seleção de um processo adequado, que fosse voltado para as necessidades da empresa, ao mesmo tempo em que se adequasse à realidade da equipe, composta apenas por dois desenvolvedores, foi crucial para a extrapolação do tempo previsto com o desenvolvimento. Iniciou-se a instalação de uma versão que por conter diversas falhas foi aprimorada no meio do desenvolvimento, o que gerou a necessidade de se instalar a versão mais recente, reinstalar container mais adequado, customizar arquivos XML, e, por fim, refazer toda a aplicação. Enfim, *iniciar* e reiniciar um sistema, sem ter a noção exata de como seria desenvolvido, a partir da análise, trouxe dificuldades que, em conjunto, amadureceram aos dois membros desenvolvedores.

Outra experiência adquirida logo após a criação dos diagramas de classes, da escolha da ferramenta e do processo de desenvolvimento, foi a criação de banco de dados PostgreSQL. A escolha do banco PostgreSQL se deu em virtude de suas diversas características e de ser um banco amplamente utilizado no mercado. É interessante ressaltar

que o e-Gen não limita seu uso a um banco de dados específico, uma vez que ele usa um *driver* JDBC para se comunicar com o banco de dados. Assim, havendo o *driver* adequado, o e-Gen poderá ser integrado a vários bancos de dados presentes no mercado.

Utilizando o e-Gen, foi possível experimentar um aumento significativo da produtividade da equipe de desenvolvimento, o que provocou a redução de custos e do tempo para implementação e manutenção do sistema, facilitamos o gerenciamento dos projetos, através da redução dos requisitos de qualificação individual da equipe. Desta forma, a substituição de um membro da equipe não provocaria um processo de descontinuidade, além de que, o uso do e-Gen permitiu que a equipe ficasse mais focada nas regras de negocio do que em tecnologias.

O envolvimento dos funcionários, direção da empresa e demais colaboradores da empresa Companhia da Nutrição se fez extremamente importante para o desenvolvimento do projeto. É preciso reforçar o trabalho de conscientização e disseminar os benefícios da utilização deste novo modelo de negócio. Ressaltando que esta não é a solução pra todos os problemas da organização, mas um mecanismo que poderá solucionar, de forma eficaz boa parte dos problemas atuais.

Com base na experiência adquirida, não seria falso afirmar que sendo necessário o desenvolvimento de um projeto similar a este, com uma documentação inicial adequada e o cliente ou o futuro usuário do sistema presente junto à equipe de desenvolvimento, com apenas dois membros experientes, não seria necessário mais do que 03 (três) semanas para concluir exatamente o que foi gerado neste projeto, com código padronizado, respeitando padrões de projeto e o que é melhor, com um usuário final satisfeito.

Portanto, o modelo proposto no qual se desenvolve para *web* com o RAD e-Gen Developer, seguindo as melhores práticas adotadas pelo marcado de software se mostrou adequada às necessidades e à estrutura da Cia. da Nutrição. Sendo observadas as condições sobre a implantação, a adoção deste modelo trará grandes resultados para a empresa.

Referências Bibliográficas

[Ahmed, et al., 2002] Ahmed, Khawar Zaman e Umrysh, Cary E. **Desenvolvendo aplicações comerciais em Java com J2EE e UML.** Editora Ciência Moderna Ltda., 2002.

[Alexander, Christopher,1997] Alexander, Christopher. **An Introduction for Object-Oriented Designers** . Disponível em: <http://gee.cs.oswego.edu/dl/ca/ca/ca.html>. Acesso em: 10 -08-2005.

[Beck, 1999] Beck, Kent. **Extreme Programing Explained.** Editora Addison-Wesley,1999.

[Beck, 2000] Beck, Kent. **Extreme Programming Explained.** Editora Addison-Wesley, 2000.

[Beck, et al., 2001] **Manifesto for Agile Software Development**, 2001. Disponível em: < http://www.agilemanifesto.org/> Acesso em 15-08-2005.

[Booch et al., 2000] Booch, Grady; Rumbauch, James; Jacobson, Ivan. **UML, guia do usuário.** Editora Campus, 2000.

[Bueno, 1989] Bueno, Francisco da Silveira. **Mini dicionário da língua portuguesa**. Editora Lisa, 1989.

[Colla, 2004] Colla, Ernesto Coutinho. **Insite, Soluções Internet,** 1998. Disponível em: <http://www.insite.com.br/docs/develop_servlet95.htm> Acesso em 15-3-2005.

[D'Avila, 2003] D'Ávila, Márcio, **Java 2 Platform Enterprise Edition - J2EE,** 2003. Disponível em: <http://www.mhavila.com.br/link/internet/server/javaserver.html>. Acesso em 25-3-2004.

[Deitel, 2005] Deitel, H. M. **Java: como programar.** Editora Pearson Prentice Hall, 2005.

[e-Gen, 2005] E-Gen. **Easy Generation** Disponivel em http://www.egen.com.br Acesso em 15-11-2005.

[Highsmith et al., 2000] Highsmith, J. Orr; K. Cockburn, A. **Extreme Programming.** E-Business Application Delivery, Feb., pp. 4-17, 2000.

[Higtower, et al, 2002] Richard Higtower; Lesiecki Nicolas. **Java Tools for extreme Programming.** Editora John Wiley & Sons,2002.

[Hopson, 1998] Hopson, K.C.; Ingram, Stephen E. **Developing professional Java applets**. Editora Sams.Net, 1998.

[Imasters, 2004] IMASTERS. **Servlet**. Disponível em <http://imasters.com.br/>. Acesso em 19-06-2005.

[Jeffries, 2001] Jeffries, Ron. **What is Extreme Programming?** Disponível em < http://xprogramming.com/xpmag/whatisXP.htm>. Acesso em 19-09-2005.

[Jeffries, 2005] Jeffries, Ron. **A Map is a Document.** Disponível em < http://www.xprogramming.com/xpmag/dbcAMapIsADocument.htm>. Acesso em 25-09-2005.

[Johnson & Roberts, 2004] JOHNSON, Ralph; ROBERTS, Don, **Envolving Frameworks**. Disponível em: <http://jacques.dsc.ufcg.edu.br/cursos/map/html/frame/evolve.html>. Acesso em 10-4-2005.

[Larman, 2001] Larman, Craig. **Applying UML and Patterns: An Introdution to Object-Oriented Analysis and Design and the Unified Process**. Editora Prentice Hall,2001.

[Mattsson, 1996] Mattsson, M. **Object-Oriented Frameworks: A Survey of Methodological Issues.** ,M.Sc. Dissertation, Department of Computer Science and Business Administration,University Collegeof Karlskrona/Ronneby.(1996).

[Maxwell,2005] Maxwell. Disponível em <http://www.maxwell.lambda.ele.puc-rio.br>. Acesso em 05-11-2005.

[McBreen, 2002] McBreen, Pete. **Questioning Extreme Programming.** Editora Addison Wesley, 2002.

[Mendes, 2004] Mendes, Antônio. **Programando com XML**, Editora Campus, 2004.

[Nizana, 2005] Nizana Systems. Disponível em < http://www.dbwrench.com> Acesso em 27 –11- 2005

[Orfali, Rober; Harkey, Dan., 1998] Orfali, Robert; Harkey, Dan. **Client/Server Programming with Java andCORBA.** Editora John Wiley & Sons,1998.

[Royce, 1970] Royce, W.W. **Managing the development of large software systems: concepts and techniques.** Editora IEEE Westcon,1970.

[Sommerville, 2003] Sommerville, I. **Engenharia de Software.** Editora Addison-Wesley. p.592, 2003.

[Spielmann, 2003] Spielmann, Sue. **The Struts Framework - Pratical Guide for Java Programmers**. San Francisco:Morgan Kaufmann Publishers, 2003.

[Standish Group, 1995] CHAOS report, 586 Olde Kings Highway. Dennis, MA 02638, USA, 1995.

[Sun, 1999] Sun Microsystems. **Code Conventions for the Java™ Programming Language.** Disponível em < http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html >. Acesso em 29-11-2005.

[Sun, 2000] Sun Microsystems. **Designing Enterprise Applications with the Java™ 2 Platform, Enterprise Edition**. Disponivel em < http://java.sun.com/j2ee>. Acesso em 15-10-2005.

[Sun, 2002] SUN Microsystems. **Designing Enterprise Applications with the J2EE Platform.** Disponivel em < http://java.sun.com/blueprints/guidelines/designing_enterprise_applications/index.html >Acesso em 15-10-2005.

[Sun, 2005] SUN Microsystems. **Designing Enterprise Applications with the J2EE Platform.** Disponivel em < http://java.sun.com/blueprints/guidelines/designing_enterprise_applications/index.html >Acesso em 15-10-2005.

Apêndice I – Análise de Requisitos

DEFINIÇÃO DO CIA NUTRI MANAGER - SISTEMA DE GERENCIAMENTO ON-LINE DA COMPANHIA DA NUTRIÇÃO

A Companhia da Nutrição é uma empresa especializada na venda de suplementos alimentares e produtos naturais. Esta empresa teve um aumento considerável no número de vendas, nos últimos meses, o que tornou o seu gerenciamento mais complexo. A empresa está localizada na Rodoviária do Plano Piloto, onde circulam, por dia, aproximadamente 600 mil pessoas. A loja abre todos os dias da semana, exceto em feriados santos. O horário de funcionamento da loja, no local, é das 08:00h às 21:00h, exceto aos domingos e feriados, que o horário é das 12:30h às 19:30h. Além das vendas efetuadas no local, a empresa possui um sítio na internet de onde efetua diversas vendas, expõe os seus produtos e formas de pagamento. A empresa também já possui um serviço de entrega, onde um de seus funcionários (motociclista) deixa a loja no momento do pedido e efetua a entrega no endereço solicitado ou despacha a encomenda pelos Correios.

Assim, com o aumento no número de vendas, o gerente deparou-se com uma enorme dificuldade em gerenciar toda esta estrutura, de uma forma eficiente e diária. Principalmente porque ele é o responsável por realizar as parcerias da empresa com os distribuidores, e realizar os pedidos. Atualmente, a gerência é feita por meio de "controles" (estoque, vendas, comissões dos funcionários, cadastro de clientes, etc) impressos em papel, que tem um período de uso (vigência) semanal.

O gerente necessita de um sistema que informatize todo o processo de vendas, compras, controle estoques e etc., e com isso, torne o processo de gerenciamento da empresa mais fácil e dinâmico, facilitando a sua tomada de decisões e o tornando independente quanto ao repasse de informações de seus funcionários quanto às necessidades de produtos em estoque, evitando as suas faltas em prateleira e, conseqüentemente, o descontentamento ou perda de clientes para outras lojas.

A empresa possui um computador Pentium III 850Mhz, com 256Mb de Ram, comprado recentemente, onde irá rodar o sistema.

A empresa "Companhia da Nutrição" tem um número de 04 (quatro) funcionários, sendo que: 02 (dois) são caixas, 01 (um) atendente e 01 (um) atendente e motociclista. O

gerente solicitou que o sistema limite o acesso dos funcionários caixas a determinadas funcionalidades. O sistema deverá possuir um sistema de *login* que interprete o tipo de funcionário por sua matrícula e, a partir daí, permita acesso às telas autorizadas. Assim, os funcionários caixas registrarão um *login* de funcionário e uma senha que lhe dará acesso ao sistema. Além do administrador do Sistema, somente o gerente terá acesso a todas as funcionalidades. Os atendentes não precisam ter acesso ao sistema, uma vez que a responsabilidade pelos valores pagos pelos clientes, bem como o controle de entrada e saída de produtos do estoque da loja (mesmos os produtos entregues pelos motociclistas) é do funcionário caixa que registrou a venda. No caso de compras que necessitem do serviço de entrega pelo motociclista, o funcionário caixa concretiza a venda, somente no retorno do motociclista à loja, quando o mesmo traz a importância paga pelo cliente.

O atendimento no local ocorre da seguinte forma:

1. O cliente, após ter sido atendido pelo funcionário "atendente" é encaminhado ao caixa para efetuar o pagamento do produto. Enquanto isso, o produto que o mesmo solicitou é colocado em uma sacola, onde o atendente aguarda a efetuação da compra (pagamento). O cliente efetua o pagamento podendo escolher a forma de entrega (local, motociclista ou correios), o tipo de pagamento (dinheiro, cheque, cartão de crédito/débito, boleto bancário, transferência bancária e depósito bancário), e a forma de pagamento: à vista, 01 (uma) parcela com prazo, 02 (duas) parcelas e 03 (três) parcelas. O sistema solicita alguns dados do cliente para emissão do cupom fiscal. Nos pagamentos em cheque, há necessidade de consulta ao SERASA a fim de confirmar se existe alguma pendência financeira para o CPF do cliente. O cliente recebe o cupom fiscal do funcionário caixa e apresenta o mesmo ao atendente para a retirada da mercadoria. O funcionário atendente carimba o cupom com "mercadoria entregue", coloca o cupom dentro da sacola e agradece a compra.

2. As compras realizadas via telefone utilizam o mesmo procedimento citado acima, onde o cliente identifica o produto de seu interesse, informa os seus dados, informa o tipo de pagamento dentre os disponíveis. Em caso do pagamento ser em cheque, são solicitados os dados referentes à consulta ao SERASA, bem como a conta e o nº do cheque. É informado ao cliente que no ato da entrega o mesmo deverá apresentar identidade do proprietário do cheque, bem como só será recebido o cheque da conta consultada ao SERASA. Pagamento em cartão de crédito, somente para cartões que possuam numeração em relevo, uma vez que a maquineta da empresa só funciona neste tipo de cartão. O sistema, então, deverá imprimir os dados da compra para que seja providenciada a entrega, pelo

funcionário "motociclista". A conclusão da compra só ocorrerá quando do retorno do funcionário "motociclista" com o valor pago pelo cliente.

3. Nas compras via internet: o sítio da *web* é apenas uma loja virtual expositora, ou seja, não há como o cliente pagar no próprio sítio (por cartão de crédito, por exemplo). Nele o cliente preenche um formulário com dados sobre o tipo de produto que deseja receber, horário de entrega da preferência do cliente e todas as informações coletadas nas compras via telefone. O sistema não precisará ter uma interação com os pedidos feitos via *web*. Desta forma, o cliente efetua o pedido via formulário do sítio e os dados ficam em um e-mail que será consultado por um dos funcionários "caixas". Havendo alguma dúvida, o funcionário caixa entrará em contato com o cliente, para tirar suas dúvidas (exemplo: cliente não informa o sabor do produto). O pedido feito pelo sítio só passará a ser registrado no sistema a partir do momento que o funcionário caixa registrar aquele pedido de compra. Seria como se o cliente da *web* tivesse presente na loja, aguardando a sua vez de ser atendido. Assim, o sistema pode vir a constar de onde ocorreu a compra: "loja" (o cliente atendido na loja), telefone (pedidos via telefone) e sítio (pedidos pela internet). A página da *web* deverá, de preferência, ser gerenciada pelo funcionário "gerente". Ou seja, ao obter um relatório que não consta o produto "Aminoácido Optimum 100 caps" em estoque, o gerente atualizaria a página *web* tornando o produto indisponível, evitando transtornos.

O sistema só acusará o fechamento das compras via telefone ou internet, no momento que o funcionário "motociclista" retornar com os valores pagos em mãos e o entregar ao funcionário "caixa", que registrará o fechamento da venda.

O sistema interage com o sistema financeiro (no caso de cartões de crédito) e com o SERASA (no caso de pagamento em cheque) através da internet rápida instalada na loja (ADSL) ou via linha telefônica.

Havendo qualquer interrupção no sistema de internet, os funcionários poderão faze-lo de maneira convencional, ou seja, no caso dos cartões de crédito através da "maquininha específica" (usa bateria e linha telefônica) e no caso do SERASA através do uso do telefone.

No caso da indisponibilidade das linhas telefônicas, o funcionário poderá utilizar um telefone público ou celular para fazer as consultas ao SERASA. Mesmo o pagamento por cartão de crédito não ficará impossibilitado, uma vez que a empresa possui "maquineta manual" que não precisa de energia, apenas de um telefone para discagem gratuita (0800) e consulta se o cartão do cliente possui limite.

No caso de interrupção no fornecimento de energia elétrica, os funcionários terão que realizar o preenchimento das vendas em formulários específicos, onde constará os dados do cliente, horário de compra, produtos vendidos. As notas fiscais serão emitidas através da forma convencional (bloco de notas).

O sistema deverá ter uma funcionalidade que funcionará com um "Registro de Ponto dos Funcionários". Assim, ao chegarem na empresa, os funcionários requisitarão o "Registrador de Ponto", onde digitarão seu "login" de funcionário e sua senha no computador. Eles repetirão estes procedimentos nos horários de almoço de cada um, bem como ao término do expediente. O ideal é a implementação de um sistema que leia o código de barras da carteira funcional do funcionário, no qual o mesmo passa a ter somente o trabalho de digitar sua senha. Assim, o próprio programa registrará o número de horas trabalhadas, para facilitar o cálculo do salário, bem como o registro de horas extras. Não adiantará o funcionário estender o seu horário de expediente intencionalmente a fim de "ganhar" horas extra, sem ter sido determinado pelo gerente. O programa desconsiderará o período extra que não tenha sido registrado ou confirmado pelo gerente, para fins de cálculo do salário. Ex.: o horário de trabalho do funcionário de nome "João" é das 10:00h às 18:00h. Este funcionário poderia ficar no local até às 20:00h. O sistema registraria o horário que ele estava saindo, mas só iria computar como hora extra após o gerente autorizar. Ou seja, no cálculo final de horas trabalhadas, o sistema acusaria 46h de trabalho semanal. Entretanto, somente o gerente poderia definir se aquelas 02h a mais trabalhadas foram por necessidade da empresa.

O sistema deverá ser capaz de emitir relatórios (vendas, estoque, clientes cadastrados, etc.) ao gerente a todo instante em que for solicitado, facilitando sua visão geral do negócio.

O sistema deverá fazer o *backup* de dados a cada nova venda registrada ou a critério do gerente. Este procedimento destina-se a perda de dados.

O serviço de entrega ocorrerá de forma imediata ou em horários de preferência dos clientes, com cobrança de preços por região. As entregas ocorrerão na ordem do pedido, ou pela localidade. As entregas terão preços diferenciados por região: Plano Piloto, Sudoeste, Cruzeiro e Octogonal (R$ 2,00), Lago Sul, Lago Norte e Sobradinho (R$ 3,00), Guará I e Guará II, Candangolândia, Núcleo Bandeirante, Águas Claras, Vicente Pires e Taguatinga (R$ 4,00). Nestas localidades, nas compras acima de R$ 60,00 as entregas são grátis. Nas demais localidades não há serviço de entrega.

O motociclista é um funcionário atendente, que fica na loja, mas tem a habilidade de andar de moto. Na verdade, é uma atendente que deixa sua função por determinado período para efetuar a entrega.

A maior parte dos produtos será mantida em um depósito em uma localidade diferente da loja. O sistema deverá manter o controle dos produtos que estão disponíveis no depósito e na loja. A partir daí informará, no relatório do gerente, que a quantidade disponível já é limite, ou seja, há necessidade de um novo pedido para reposição.

SITUAÇÕES QUE NÃO INTERAGEM DIRETAMENTE COM O SISTEMA

Por questões de segurança, o gabinete do computador ficará trancado no depósito da loja, em local onde os funcionários não terão acesso. Sendo necessária alguma manutenção, os funcionários solicitarão autorização do gerente para efetuar a abertura do envelope onde ficará a chave do cadeado, para terem acesso ao gabinete.

Na loja, há um ponto de ADSL, onde o sistema financeiro de cartão de crédito ou consulta ao SERASA poderá ser utilizado. Além disso, a comunicação entre os funcionários caixas e o gerente dar-se-á por meio da internet (redução de custos) através de voz por IP (já é feito por um software específico). Poderá ser utilizada, ainda, a comunicação convencional de telefone fixo para fixo.

Por meio de um software específico, as imagens de segurança da loja serão gravadas para o HD da máquina, durante 07 (sete) dias consecutivos, de forma que no oitavo dia, o primeiro dia é deletado automaticamente (evitar ocupar muito espaço do HD com vídeos de segurança). Com o acesso remoto do gerente a máquina, o mesmo poderá efetuar o backup do vídeo de segurança no fim do dia (evitar sobrecarregar o sistema e o tráfego via web). Atualmente a empresa filma em fita VHS os atendimentos ao longo do dia, como uma forma de segurança e como uma forma de monitoramento de seus funcionários (qualificar como anda o atendimento prestado aos seus clientes).

O cliente cadastrado passará a receber as promoções e informativos da loja no seu e-mail ou endereço residencial. A partir do primeiro e-mail o mesmo pode solicitar o cancelamento do envio das mensagens para a sua caixa de e-mail, para não continuar recebendo as mensagens via internet.

O sistema não irá interagir com os fornecedores. Deverão ser estabelecidas quantidades mínimas em estoque a fim de que o gerente possa efetuar novos pedidos.

Os produtos vendidos pela empresa deverão seguir em um documento separado, para fins de conhecimentos pelos analistas.

Brasília-DF, em __ de _____ de 2005.

_____

Analista

_____

Gerente da Empresa

Apêndice II – Documento de Visão do Projeto

1. Introdução

O propósito deste documento é coletar, analisar e definir as necessidades e características do "Cia Nutri Manager - Sistema de Gerenciamento On-Line da Companhia da Nutrição".

O objetivo é facilitar a gerência da estrutura de comércio varejista de suplementos alimentares e produtos naturais da empresa "Companhia da Nutrição". O sistema visa informatizar o processo de vendas, compras, controle de estoque e etc. Ou seja, tornar o processo de gerenciamento da empresa mais fácil e dinâmico, o que facilitaria ao gerente a tomada de decisões e o tornaria independente das informações de seus funcionários sobre as necessidades de produtos em estoque, evitando a falta de produtos em prateleira e, conseqüentemente, o descontentamento ou mesmo a perda de clientes para outras lojas.

Este documento também deve servir como um ponto de controle sobre o escopo, características, necessidades e prazos para o desenvolvimento do sistema.

O desenvolvimento da solução proposta será guiado por um conjunto de características e funcionalidades essenciais, dentre as quais: a utilização de *frameworks* que facilitem a implementação a fim de que se possa focar nas regras de negócio, simplificando o desenvolvimento da aplicação.

2. Sentenças do Problema

O problema da	Pouca informação do gerente
Afeta	-Estoque -Entregas -Falta de atualização do estoque
O impacto disso é	-Insatisfação dos clientes -Redução nas vendas
Uma solução de sucesso permitirá	-Estoque suficiente na loja -Atualização de produtos

O problema da	Demora no atendimento ao cliente

Afeta	-Vendas -Entregas
O impacto disso é	-Insatisfação dos clientes -Redução nas vendas
Uma solução de sucesso permitirá	- Agilidade no atendimento e na entrega do produto

O problema da	Demora na atualização do estoque da loja
Afeta	-Vendas -Entregas
O impacto disso é	-Insatisfação dos clientes -Redução nas vendas
Uma solução de sucesso permitirá	-Estoque suficiente na loja

O problema da	- Falta de linha
Afeta	-Vendas (impede a formalização de pedidos via telefone) -Entregas -Backups -Relatórios ao gerente via internet (remota)
O impacto disso é	-Insatisfação dos clientes -Redução nas vendas -Perda do controle pelo gerente -Perda de dados (informações sobre pedidos e sobre vendas)
Uma solução de sucesso permitirá	-Que os pedidos via telefone e *web* possam ser direcionados para o gerente -Que os relatórios e os "backups" sejam remetidos ao gerente pelo menos duas vezes por dia, a fim de que tenha noção do estoque na loja.

O problema da	-Perda dos dados sobre pedidos, vendas e estoque

Afeta	-Vendas (não sabe o que há no estoque do depósito que fica fora da loja) -Estoque -Entregas -Novos pedidos
O impacto disso é	-Insatisfação dos clientes -Descontrole quanto às necessidades de itens a serem repostos -Redução nas vendas
Uma solução de sucesso permitirá	-Garantir um backup em mais de um local de armazenamento (dois HDs e um disquete, por ex.), efetuado a cada pedido e a cada venda

3. Descrição dos Envolvidos no Projeto

a) Patrocinador do Projeto

Patrocinador	**Descrição**	**Responsabilidade**
Gerente	Proprietário da empresa	Define as necessidades do SGOL

b) Empresa de Desenvolvimento

Empresa	**Descrição**	**Responsabilidade**
SOO/UCB (fictício)	Desenvolvimento de Softwares	Análise e Desenvolvimento do sistema

c) Potenciais Usuários

Usuário	**Representa papel**
Gerente	Proprietário da empresa e funcionário com privilégios no sistema
Funcionários	Funcionários "caixas" que trabalham com o sistema com restrições
Sistema Financeiro	Administradora de cartões de crédito (geralmente tem relação com um banco, que disponibiliza tudo para a empresa)

Consultor de cheque	Instituição tipo SERASA que mantém registro de pessoas com dívidas no comércio

3.1 Necessidades dos Usuários

a) Facilitar o acesso às informações da empresa

O gerente (que também é o Administrador do Sistema) e os funcionários caixas precisam ter disponíveis a qualquer momento os dados relativos ao estoque da loja e do depósito localizado fora da loja, dados sobre as vendas realizadas durante o dia atual ou anteriores, especificado o responsável pela venda, a forma como ela ocorreu (pela loja, pelo telefone ou pela internet), a forma de pagamento utilizada, bem como os clientes cadastrados (no dia e anteriormente), a fim de agilizar o atendimento no caso de clientes que já efetuaram compras na loja.

b) Elaborar relatórios

O sistema deverá emitir relatórios ao gerente, a qualquer momento, com dados sobre a situação do estoque na loja e no depósito, itens de produtos que já atingiram as quantidades mínimas limites, número de vendas por região (cidades satélites do DF e para outros estados), comissões (apenas para vendas na loja) aos funcionários responsáveis pelas vendas (competições internas pelas vendas), atrasos de funcionários, horas extras trabalhadas, etc.

c) Informações específicas dos clientes

O sistema de consulta de cheque exige informações específicas para consulta frente ao SERASA, como CPF, data de nascimento e dados do cheque (agência e banco)

d) Recuperar informações sobre vendas, estoques e clientes cadastrados

A todo instante em que for efetuada uma compra, ou realizado um pedido de entrega, o sistema deverá salvar as novas informações e providenciar ao armazenamento em pelo menos dois discos diferentes, por segurança.

4. Restrições do produto

A definir

5. Aplicações de Padrões

A definir

6. Prioridade

Realizar os registros de vendas, cadastro de produtos, entrada e saída de itens do estoque, cadastro de clientes, tipos de pagamentos, formas de pagamento, local de compra (loja, telefone ou internet) e horário da venda. Além disso, implementar o controle interno de estoque, que acusaria a necessidade de reposição de determinados itens que poderiam estar em falta na loja, mas estarem presentes no depósito. Registrar movimentação de produtos do estoque para a loja, ou da loja para o estoque.

7. Aprovação

Brasília-DF, em __ de _____ de 2005.

_____

Analista

_____

Gerente da Empresa

Apêndice III – *Survey* do Modelo de Casos de Uso

1. Introdução

O propósito deste documento é apresentar o modelo de casos de uso do **Cia Nutri *Manager* - Sistema de Gerenciamento On-Line da Companhia da Nutrição**. Este documento contempla não só os diagramas de casos de uso (formados por atores e casos de uso) utilizados para representar os requisitos funcionais da aplicação, como também o detalhamento dos mesmos. Como o processo é iterativo e incremental, este documento é continuamente atualizado ao longo do ciclo de desenvolvimento.

Referências

Referência	Descrição	Versão
Documento de Visão	Documento contemplando a visão geral do sistema, as necessidades e características do produto. Contempla, ainda, em anexo, os itens de estoque que estarão presentes no sistema.	1.0.0

2. Histórico

Versão	**Modificação em relação à última versão**	**Data**
Criação do documento	Documento contemplando apenas os diagramas de caso de uso e uma descrição, superficial, dos atores e casos de uso	
Priorização dos casos de uso	Priorização dos casos de uso de acordo com os riscos, as necessidades dos usuários e a complexidade de desenvolvimento.	
Detalhamento dos casos de uso	Detalhamento dos casos de uso Consulta Produtos e Realiza Compra.	

3. Atores

a) Gerente: tem a função de Administrador do Sistema. Precisa ter disponível a qualquer momento, assim como os funcionários "caixas", os dados relativos ao estoque da loja e do

depósito localizado fora da loja, dados sobre as vendas realizadas durante o dia atual ou anteriores, especificado o responsável pela venda, onde ela ocorreu (pela loja, pelo telefone ou pela internet), o tipo e a forma de pagamento utilizada, bem como os clientes cadastrados (no dia e anteriormente) a fim de agilizar o atendimento no caso de clientes que já efetuaram compras na loja. O sistema deverá emitir relatórios ao gerente (somente), a qualquer momento, com dados sobre a situação do estoque na loja e no depósito, itens de produtos que já atingiram as quantidades mínimas limites, número de vendas por região (cidades satélites do DF e outros estados), comissões (apenas para vendas na loja) aos funcionários responsáveis pelas vendas (competições internas pelas vendas), atrasos de funcionários, horas extras trabalhadas, etc.

b) Funcionário: assim como o gerente, este ator realiza compras no sistema, cadastra clientes, consulta estoque da loja e do depósito fora da loja, consulta vendas realizadas durante o dia atual, onde está especificado o responsável pela venda, onde ela ocorreu (pela loja, pelo telefone ou pela internet), o tipo e a forma de pagamento utilizada, bem como os clientes cadastrados (no dia e anteriormente), a fim de agilizar o atendimento no caso de clientes que já efetuaram compras na loja.

c) Sistema financeiro: responsável por efetivar as compras via cartão de crédito. Administradora de cartões de crédito (geralmente tem relação com um banco, que disponibiliza tudo para a empresa).

d) Consultor de cheque: efetua as consultas na base de dados da Instituição do tipo "SERASA" e informa a situação do cliente quanto às dívidas no comércio.

4. Caso de Uso do Ator Funcionário

O ator gerente mantém todos os casos de uso de um ator funcionário e ainda possui funcionalidades específicas. Serão omitidos diversos casos de uso de menor prioridade. Como o processo é iterativo e incremental os mesmos poderão ser especificados posteriormente.

Consulta Produtos

Este caso de uso permite que os funcionários (caixa e gerente) consultem os produtos (por categoria, por nome, etc.) disponíveis no estoque da loja e no depósito localizado fora da loja. O item de produto que chegar a quantidade mínima será informado no relatório emitido ao funcionário gerente, a fim de que o mesmo possa efetuar novos pedidos junto aos seus fornecedores. Com este caso de uso, os funcionários saberão os produtos que se encontram disponíveis para venda.

Cadastra Produtos

Caso de uso que permite ao funcionário gerente cadastrar os produtos oferecidos aos clientes, com descrições informativas sobre: preço, número de cápsulas, massa (peso) e o objetivo de cada um (aumento de massa muscular, perda de gordura, etc).

Solicita relatório gerencial

Este caso de uso, exclusivo do funcionário gerente, permite facilitar a gerência da empresa, uma vez que apresentam, em detalhes, informações relativas a situação das vendas (data e horário de compra, produtos comprados, forma de pagamento, etc), do estoque na loja e no depósito, itens de produtos que já atingiram as quantidades mínimas limites, número de vendas por região, comissões (apenas para vendas realizadas na "loja") aos funcionários responsáveis pelas vendas, atrasos de funcionários, horas extras trabalhadas, etc.

Realiza Compra

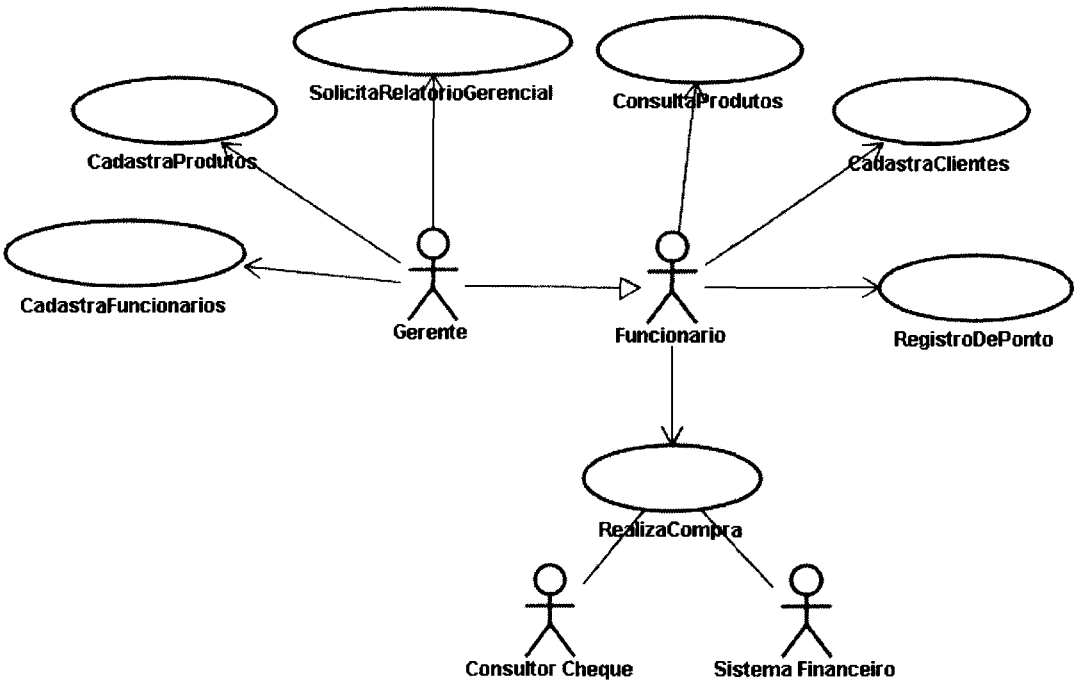
Este caso de uso permite que os funcionários efetuem todos os procedimentos para a compra na loja, pelo telefone ou pelo sítio. Basicamente apresenta a seguinte seqüência de ações: cadastra o cliente, seleciona o produto desejado, seleciona a forma de pagamento, seleciona a taxa de entrega, confirma o pedido, imprime os dados de compra para que o motociclista efetue a entrega, recebe o pagamento pelo produto, fecha o pedido.

Registro de Ponto

Caso de Uso que permite o funcionário usar o sistema para registrar os seus horários de chegada para início de expediente, horário de almoço e término do expediente. Ele não permite que o funcionário altere o horário. O programa pega o horário do sistema.

Cadastra Funcionários

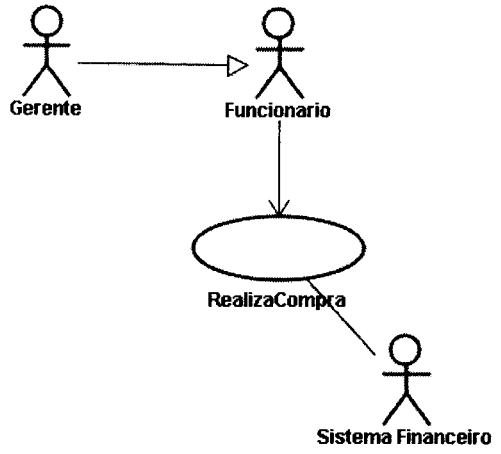
Caso de uso que permite ao funcionário "gerente" cadastrar novos funcionários para que os mesmos tenham acesso ao sistema.



5. Caso de Uso do Ator Sistema Financeiro

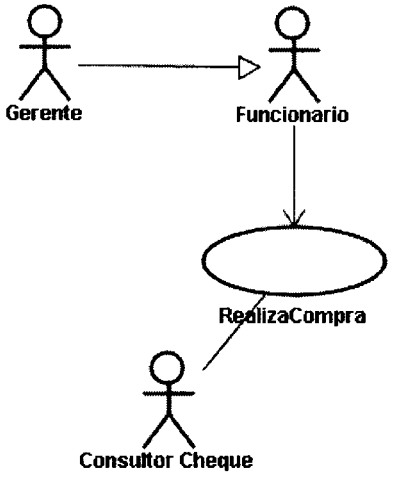
Realiza Compra

Caso de uso que permite efetivar as compras via cartão de crédito.



6. Caso de Uso do Ator Consultor de Cheque

Realiza Compra
Caso de uso que permite efetivar as compras via cheque.



7. Prioridade dos Casos de Uso

Os casos de uso já levantados serão priorizados de acordo com os riscos, a necessidade dos usuários e a complexidade de desenvolvimento. A priorização dos casos de uso serve de entrada para a elaboração do plano de desenvolvimento de software. Serão utilizados os valores "Alto", "Médio" e "Baixo" para qualificar os atributos riscos, necessidade e complexidade. A cada iteração é necessário rever a prioridade dos casos de uso.

Caso de Uso	Risco (3)	Necessidade (2)	Complexidade (1)	Ranking
Solicita Relatório Gerencial	Alto	Alta	Média	
Consulta Produtos	Baixo	Alta	Média	
Cadastra Produtos	Baixo	Alta	Baixa	
Cadastra Clientes	Baixo	Alta	Baixa	
Cadastra Categoria	Baixo	Alta	Baixa	

Cadastra Fabricantes	Baixo	Alta	Baixa	
Cadastra Representantes	Baixo	Alta	Baixa	
Cadastra Funcionários	Baixo	Alta	Baixa	
Cadastra Forma de Pagamento	Baixo	Alta	Baixa	
Cadastra Tipos de Pagamento	Baixo	Alta	Baixa	
Cadastra Formas de Entrega	Baixo	Alta	Baixa	
Cadastra Sabor	Baixo	Alta	Baixa	
Cadastra Unidades Métricas	Baixo	Alta	Baixa	
Registro de Ponto	Baixa	Alta	Média	
Realiza Compra	Alto	Alta	Média	

Apêndice IV – Caso de Uso 01 – Consulta Produtos

1 Breve Descrição

Este caso de uso permite que os funcionários (caixas e gerente) consultem os produtos (por categoria, por nome, etc.) disponíveis no estoque da loja e no depósito localizado fora da loja. O item de produto que chegar a quantidade mínima será informado no relatório emitido ao funcionário gerente, a fim de que o mesmo possa efetuar novo pedido junto aos seus fornecedores. Com este caso de uso, os funcionários saberão os produtos que se encontram disponíveis para venda.

2 Fluxo de eventos

2.1 Fluxo de eventos básico

1 Seleção do Produto

O funcionário pesquisa o produto a ser comprado através de um formulário de pesquisa de produtos. Neste formulário, há diversos campos que servem de filtros para a pesquisa do produto. O funcionário indica a categoria (no campo apropriado) e clica no botão de pesquisa. O sistema retornará os produtos presentes no banco de dados pertencentes àquela categoria, bem como outras informações sobre os itens: nome do produto, quantidade no vasilhame, massa (peso), sabor, quantidade em estoque com referência a cada sabor, separando-se a quantidade do estoque da loja e do estoque do depósito fora da loja. Como o formulário de pesquisa possui outros campos, os mesmos podem ser preenchidos com o fim de filtrar os resultados de acordo com as necessidades do cliente. Ex: o funcionário seleciona a categoria "Aminoácidos"; no campo "Nome do Produto" insere: "Amino 2222" e clica no botão de pesquisa. O sistema apresenta os produtos disponíveis em estoque com aquele nome e pertencentes àquela categoria.

2.2 Fluxo de eventos alternativo

Não se aplica.

3 Requisitos Especiais

Para facilitar a confecção do pedido pelo funcionário gerente, o próprio sistema faz a média de vendas mensais daquele item de produto e sugere um novo pedido, com base na quantidade mensal.

4 Pré-condições

O funcionário deve estar autenticado pelo sistema.

5 Pós-condições

As informações sobre o pedido aos fornecedores poderão ficar armazenadas para fins de controle pelo gerente.

6 Pontos de extensão

Os funcionários têm a opção de solicitar um catálogo impresso dos produtos em estoque.

Apêndice V – Caso de Uso 02 – Realiza Compra

1 Breve Descrição

Este caso de uso permite que os funcionários efetuem todos os procedimentos para efetivar a compra do produto. Basicamente possuirá a seguinte seqüência: cadastra o cliente, pesquisa o produto desejado em formulário específico, retorna os dados do item para o formulário de compras, altera a quantidade dos itens, seleciona o tipo de pagamento, a forma de pagamento, seleciona a forma de entrega, insere a taxa de entrega, confirma o pedido, imprime os dados de compra para que o motociclista efetue a entrega, recebe o pagamento pela compra, fecha o pedido.

2 Fluxo de eventos

2.1 Fluxo de eventos básico

1 Cadastro do cliente

Após o cliente se decidir pela compra, inicia-se os procedimentos para a efetivação da compra. O primeiro passo é o cadastro do cliente no sistema. O funcionário insere o dados do cliente em formulário próprio e cadastra o cliente no sistema. Se o cliente já for cadastrado, o funcionário efetua pesquisa em formulário próprio e, ao clicar no link do cliente, automaticamente, são preenchidos os seus dados no formulário de vendas (nome, telefone, endereço, etc).

2 Dados sobre o pedido do produto

O funcionário solicita que o cliente informe os dados do pedido (tipo de item, sabor, quantidade). O funcionário pesquisa o produto em formulário próprio e retorna os dados do item para o formulário de vendas.

3 Detalhamento do Pedido

O sistema permite atualizar a quantidade do produto de acordo com as necessidades do cliente. Há um campo de soma parcial e, ao final é apresentado o custo total do pedido. Em caso de entregas, é adicionado o custo de entrega pela localidade. Dependendo do tipo de cliente (a critério da empresa) o mesmo poderá obter descontos em suas compras.

4. Forma de Pagamento

O funcionário lança a forma de pagamento (dinheiro, cheque ou cartão de crédito) escolhida pelo cliente. No caso de pedidos via telefone ou internet, as opções serão dinheiro ou cheque.

5. Tipo de Pagamento

O funcionário lança o tipo de pagamento (à vista, a prazo em 1x, a prazo em 2x ou a prazo em 3x) escolhido pelo cliente. No caso de pedidos via telefone ou internet, a opção será sempre à vista.

6. Confirmação de Pedido

O pedido tem confirmação diferente, dependendo da forma de pagamento escolhida. Em caso de dinheiro, a confirmação é imediata. Em caso de pagamento em cheque, é necessário consultar o "Consultor de Cheque". Em caso de pagamento em cartão de crédito, há necessidade do uso do "Sistema Financeiro" para viabilizar a compra.

2.2 Fluxo de eventos alternativo

No caso da impossibilidade de pagamento por cartão de crédito (falta de comunicação ou maquineta quebrada) ou cheque (dívidas do cliente na praça), resta ao próprio usuário apresentar as outras formas de pagamento: "pagamento em dinheiro" ou "transferência eletrônica" ou "boleto bancário". Não havendo interesse pelo cliente, o pedido será cancelado.

3 Requisitos Especiais

Para manter a qualidade do serviço prestado aos clientes, o tempo de resposta das operações de compra via sítio na *web* não pode exceder 15 (quinze) minutos. Ou seja, em quinze minutos após a mensagem aparecer para o funcionário, o mesmo deverá entrar em contato com o cliente confirmando a compra e a forma de pagamento (se for cheque, já deverá ter consultado a situação do cliente no "Consultor de Cheque").

4 Pré-condições

O funcionário deve estar autenticado pelo sistema.

5 Pós-condições

As informações sobre os pedidos a serem entregues são armazenadas até que o funcionário feche a compra, ao receber o valor pago das mãos do funcionário motociclista.

6 Pontos de extensão

O cliente tem a opção de solicitar uma descrição impressa sobre os produtos comprados para ser emitido em conjunto com o pedido.

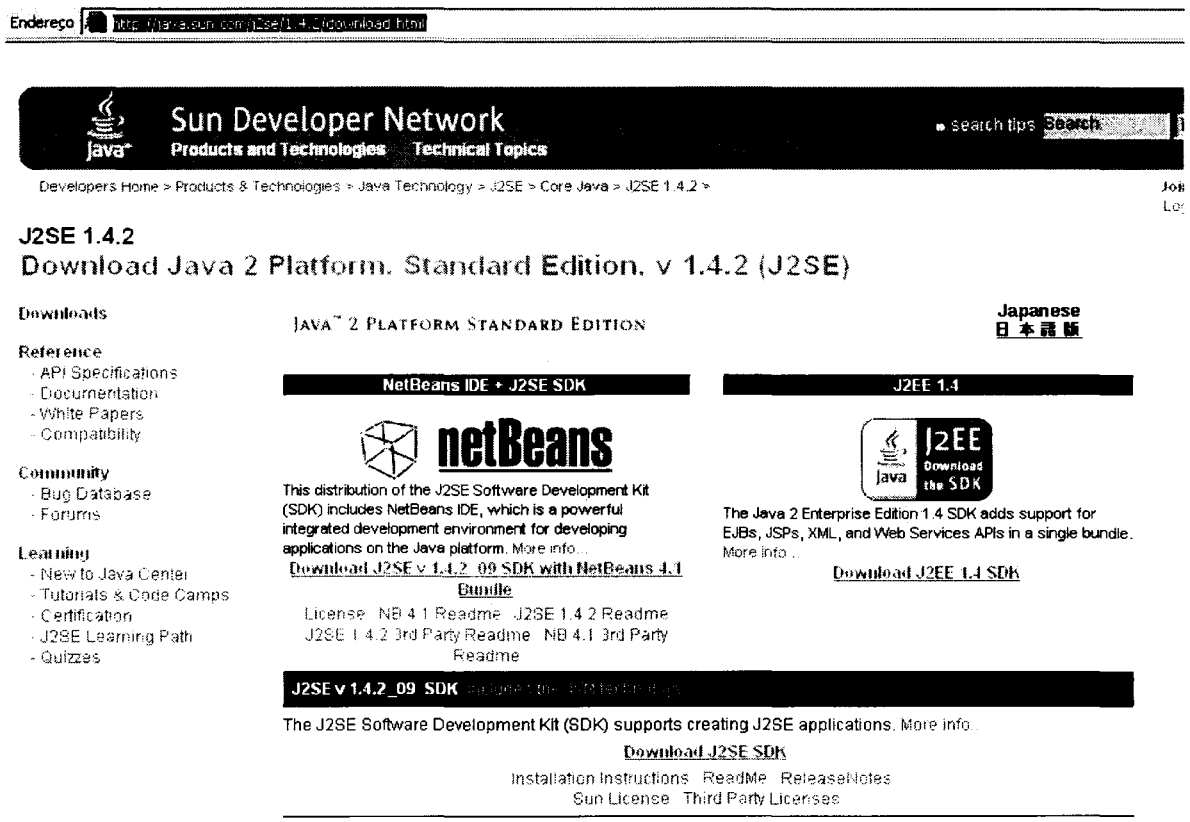
Apêndice VI – Tutorial de Instalação dos Programas

Este tutorial visa repassar, passo a passo, todas as informações sobre a instalação dos programas utilizados no projeto.

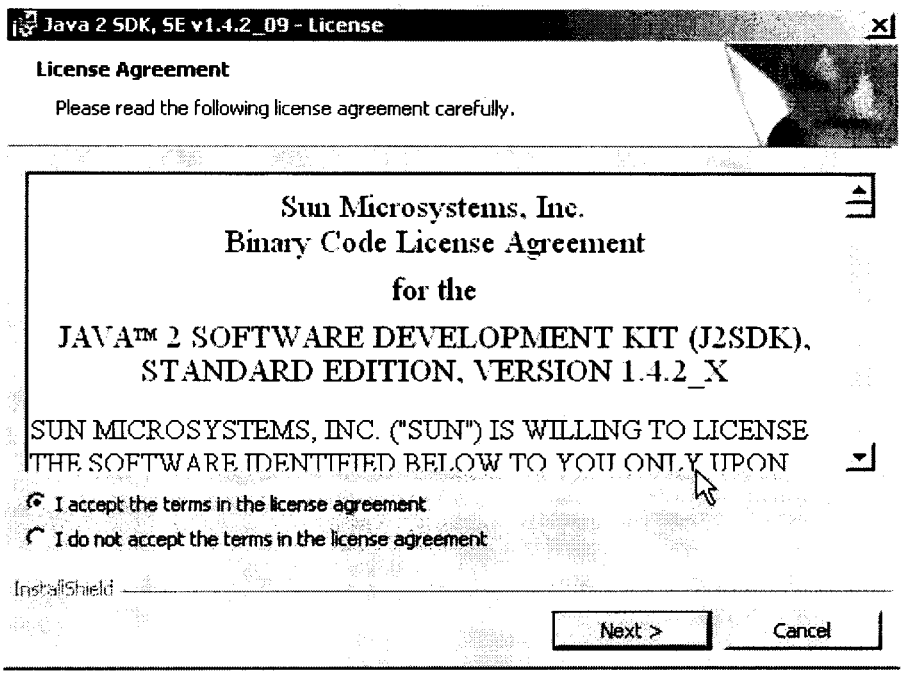
Instalação do J2SE

A instalação do Java é algo que precisa de uma certa atenção, pois 90% dos erros para rodar o e-Gen após sua instalação está relacionado com a instalação do Java. As variáveis de ambiente precisam ter seus caminhos absolutos ajustados para ficarem mais simples.

Em primeiro momento vamos fazer o *download* do J2SE version 1.4.2_09 ou similar (não baixar a versão Java 1.5 devido a conflito no DbWrench) do seguinte link http://java.sun.com/j2se/1.4.2/download.html .

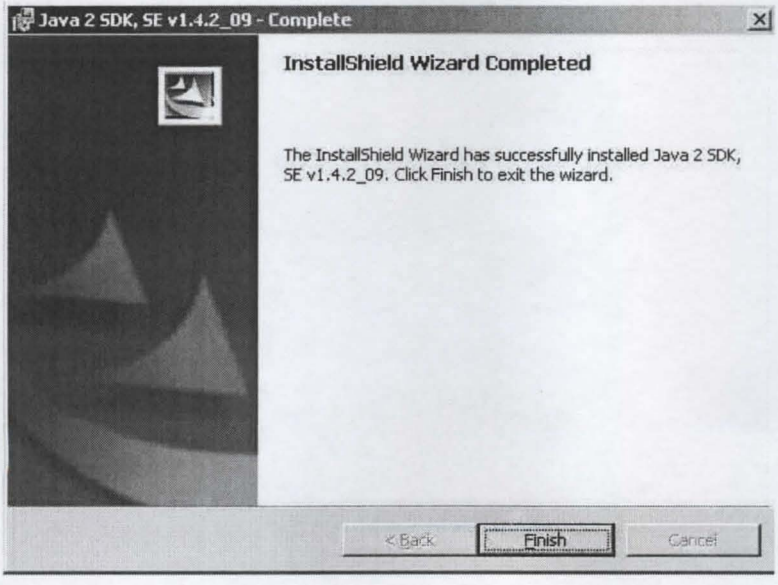


Logo em seguida clique sobre o link **Download J2SE SDK** e na próxima página marque o checkbox "I accept the terms in the license agreement" para aceitar a licença, e em seguida selecione em **Windows Platform - Java(TM) 2 SDK, Standard Edition 1.4.2_09,** o link Windows Offline Installation, Multi-language para iniciar o *download* do Java.



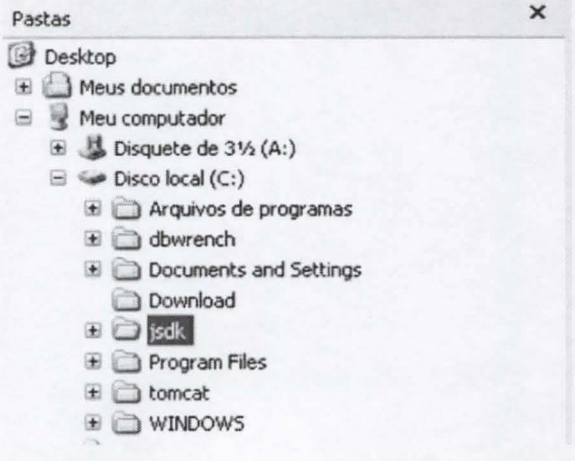
Após ter feito o *download* o próximo passo é a instalação do j2sk. Basta executar o arquivo para iniciar a instalação.

Leia o contrato. Caso aceite, marque conforme imagem acima e Clique em "Next". Na próxima tela, clique novamente em "Next" e "Install" para começar a instalação. É só aguardar a conclusão.

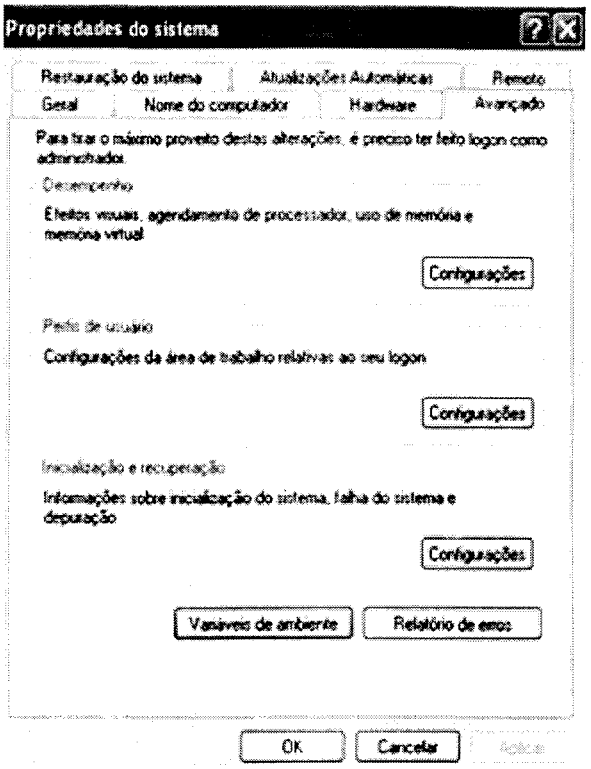


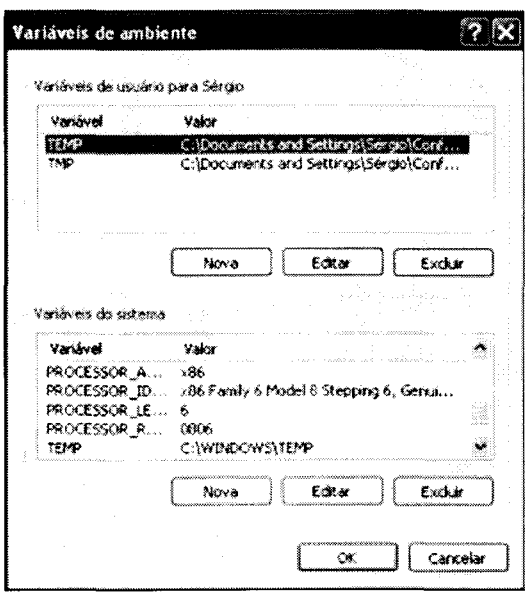
Dê um Finish, em seguida abra o windows explorer você verá que foi criado uma pasta na raiz de C: chamada **j2sdk1.4.2_09**.

Agora renomeie esta pasta de j2sdk1.4.2_09 para **jsdk** ficando assim.

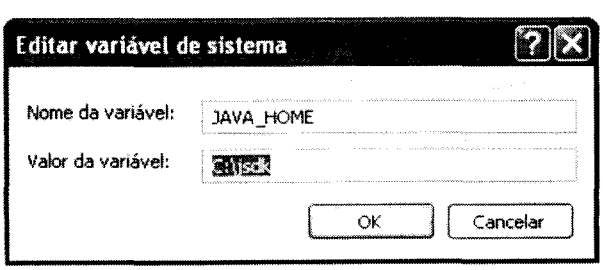


Para configurar agora as variáveis de ambiente, clique com o botão direito sobre o ícone ***Meu Computador*** na sua área de trabalho, escolha **propriedades**, em seguida na janela que abrir clique na aba **Avançado**, e logo em **Variáveis de Ambiente**.



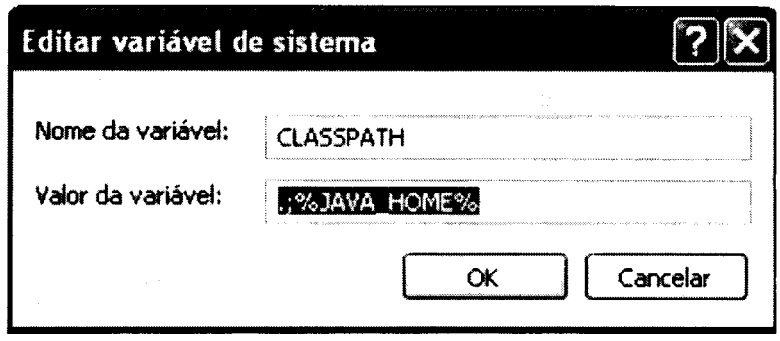


Abaixo *em Variáveis* do sistema clique sobre o botão Nova e vamos adicionar a seguinte variável:

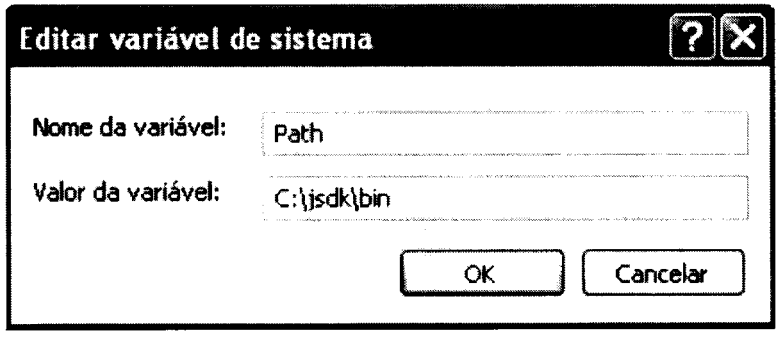


No campo "Nome da variável" adicione em maiúsculo **JAVA_HOME** e o seu valor no campo "Valor da variável" será **c:\jsdk** o caminho da instalação do Java que foi renomeado.

A próxima variável a ser adicionada é a variável **CLASSPATH** com o seguinte valor **.;JAVA_HOME**

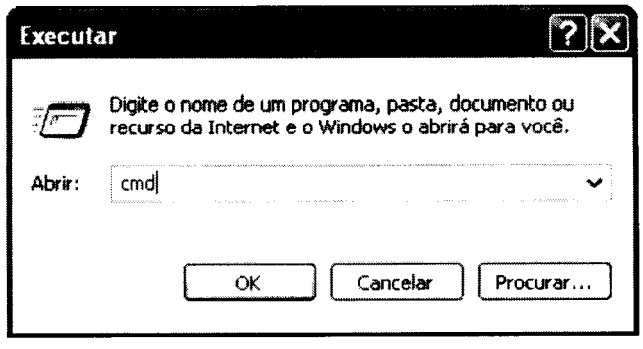


E por último a variável **PATH** com o valor **c:\jsdk\bin**

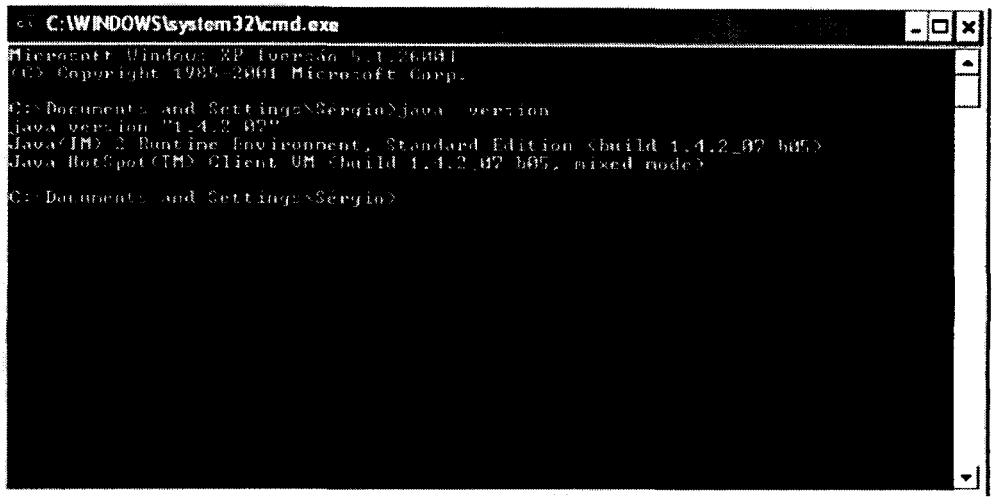


Finalize clicando em "Ok" e, novamente em "Ok" nas janelas abertas. É necessário reiniciar o computador para que as variáveis entrem em funcionamento.

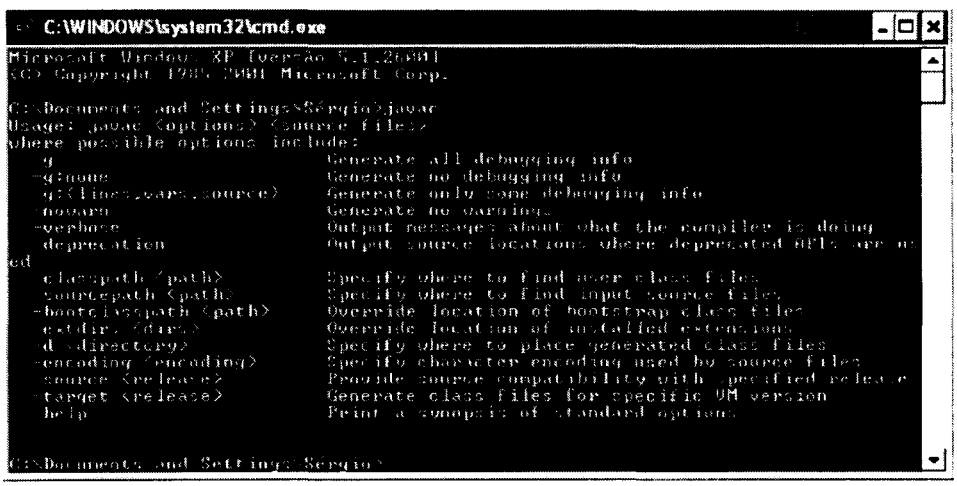
Agora é possível testar a instalação do Java. Para isso, basta abrir o prompt do MSDOS com o seguinte comando **cmd** no Menu executar do windows.



Com o prompt aberto basta digitar **java –version** que deverá retornar a versão do Java instalado.



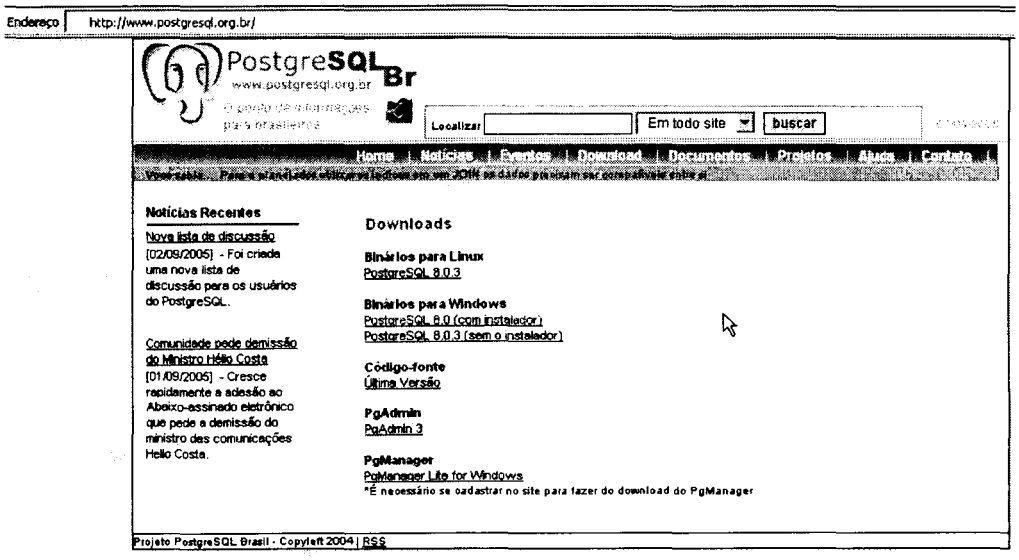
Vamos testar também se o compilador Java esta funcionando corretamente, através do comando **javac**.



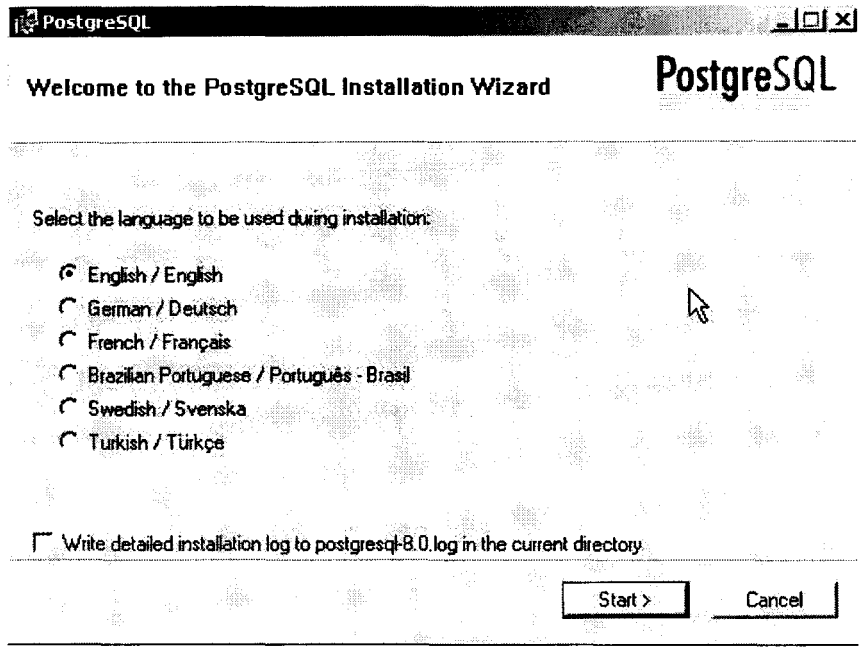
Instalação da máquina virtual Java concluída. Tendo sido executado esses dois comandos e, havendo o perfeito funcionamento, é possível dar seqüência nos próximos tutoriais. Caso não, faz-se necessário rever todos os passos, novamente.

Instalação PostgreSQL 8

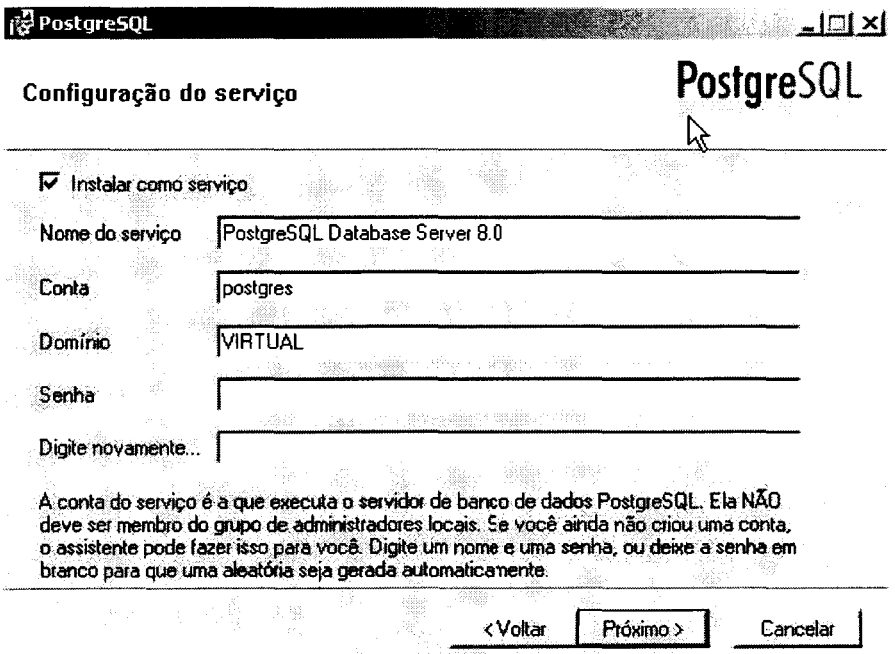
Nesse momento o usuário já terá a máquina virtual Java instalada, o próximo passo é a instalação do banco de dados PostgreSQL, que pode ser baixado no seguinte link http://www.postgresql.org.br/



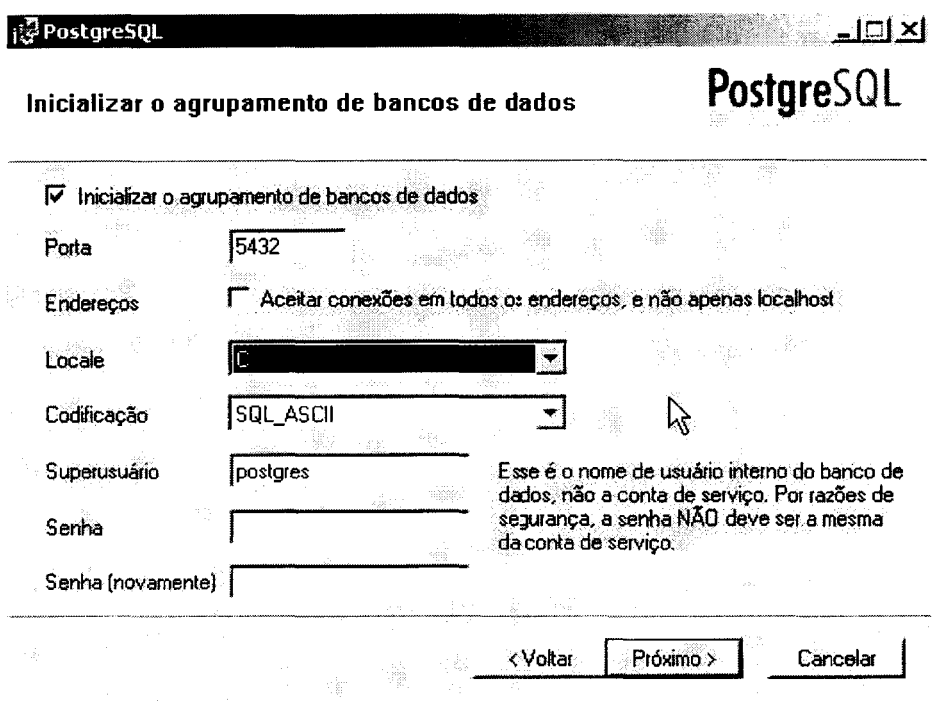
No menu *Download* escolha PostgreSQL 8.0 (com instalador), faça o *download* e extraia o arquivo. Logo após, ao executar o arquivo postgresql-8.0.msi, você verá a seguinte tela:



Escolha o idioma (português) e dê um *Start* > para começar a instalação. Na próxima tela você verá o texto da licença GPL, caso aceite, clique no botão "Próximo" e, mais uma vez "Próximo". A seguir, você verá a tela abaixo:

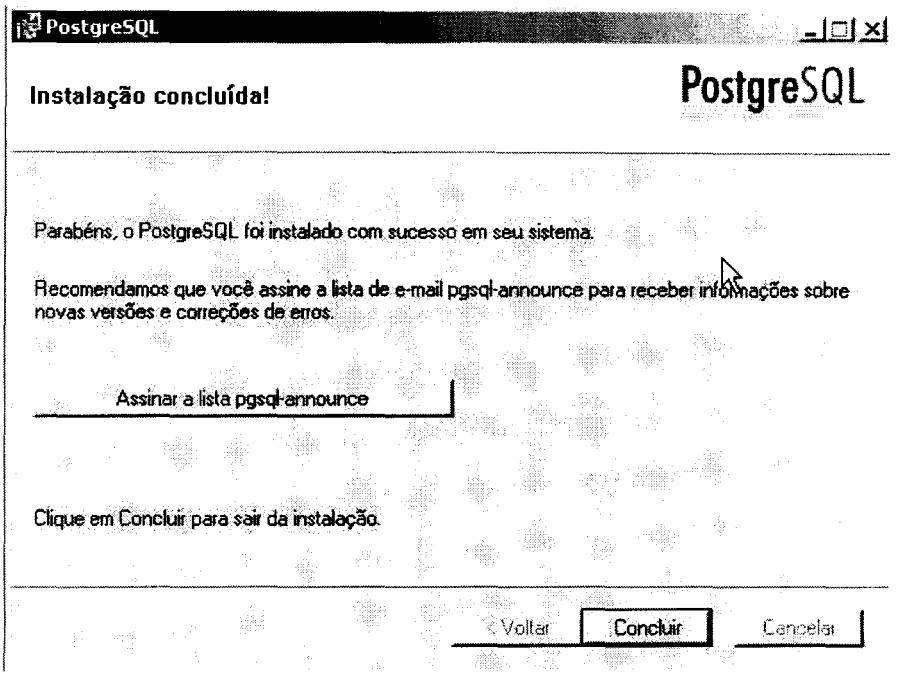


Dê um nome para o serviço somente informativo ou deixe *default*. Em conta coloque o usuário ex: *root* em Domínio deixe também *default*, e em senha digite uma senha Ex: *sorvete*. Digite novamente a senha para confirma-la e mais uma vez clique no botão *Próximo*.

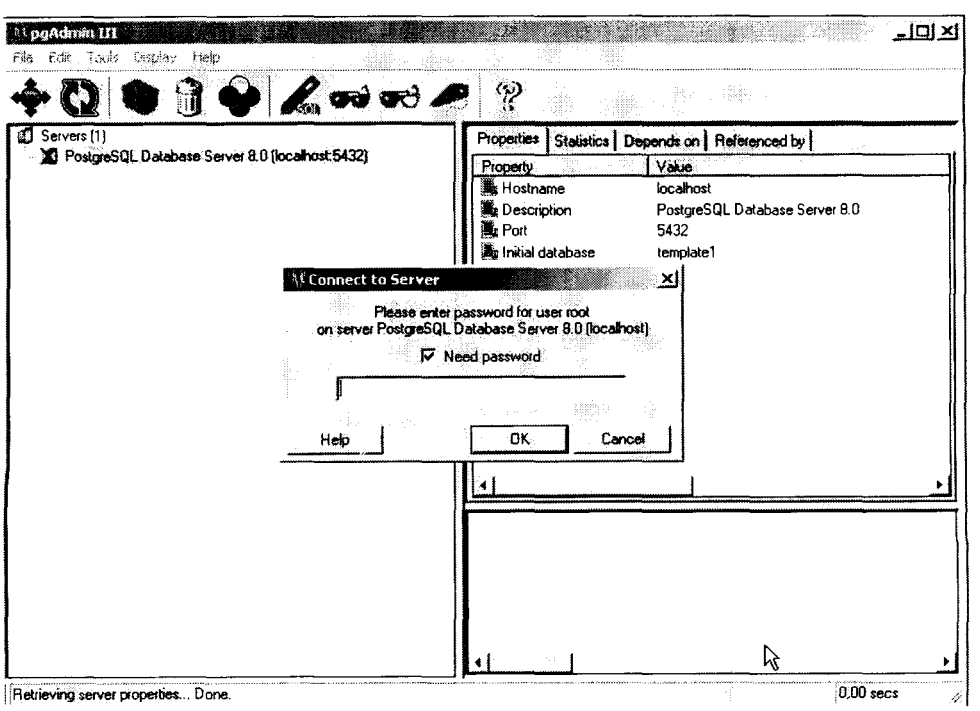


Na tela acima vamos marcar a configuração de "Inicializar o agrupamento de banco de dados". A porta deixamos *default* e Endereços desmarcado. *Locale* e Codificação deixe também com os valores *default*. Em "Superusuário" coloque o usuário root e outra senha para

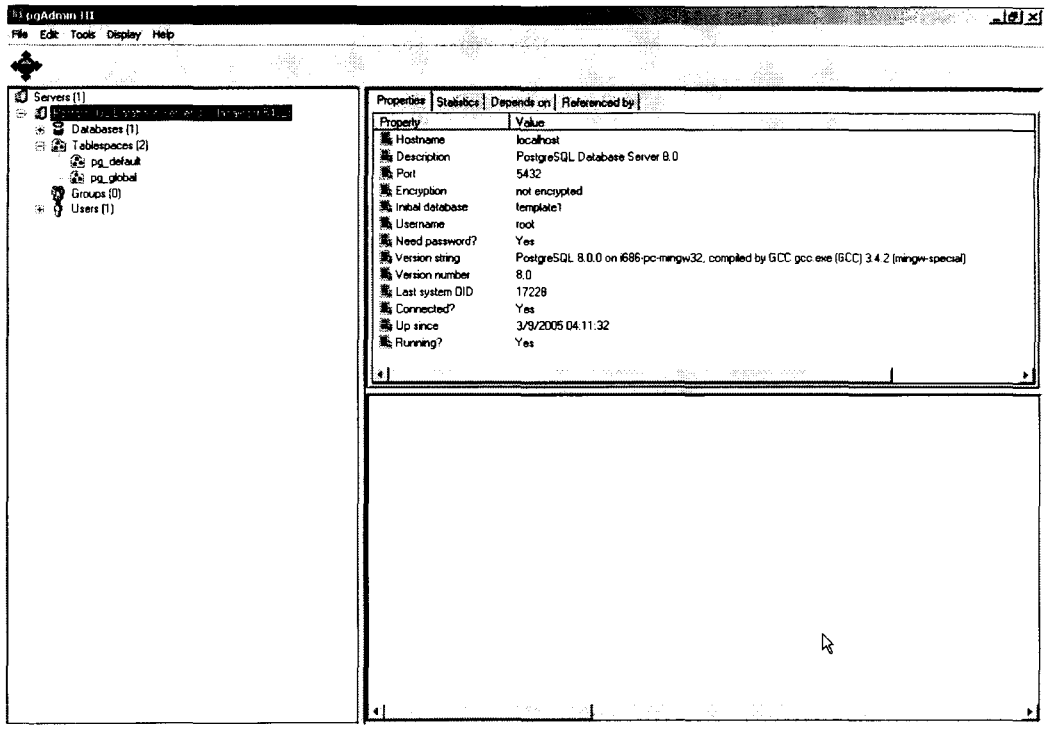
manter uma boa segurança, confirme a senha e dê um clique no botão "Próximo" e "Próximo" mais uma vez. Espere a sua finalização para concluir a instalação.



Agora vamos testar a conexão com o gerenciador de banco de dados, abra o menu iniciar > Programas > PostgreSQL 8.0 > Iniciar serviço. Logo em seguida, siga o mesmo caminho, mas clique no administrador pgAdmin III. Aparecerá a tela exibida abaixo.

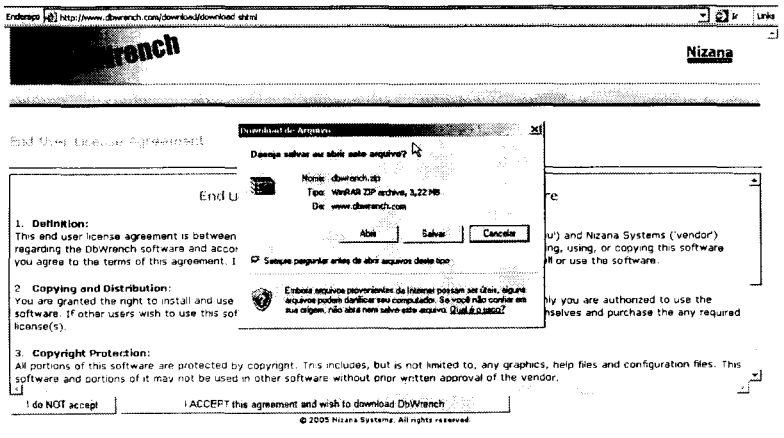


Dê dois cliques no link **PostgreSQL Database Server 8.0(localhost:5432)** será aberto o prompt para digitar a segunda senha que foi escolhida na hora da instalação do banco. Logo você verá a tela seguinte, indicando que houve uma conexão realizada com sucesso. Assim, a instalação do banco está concluída.

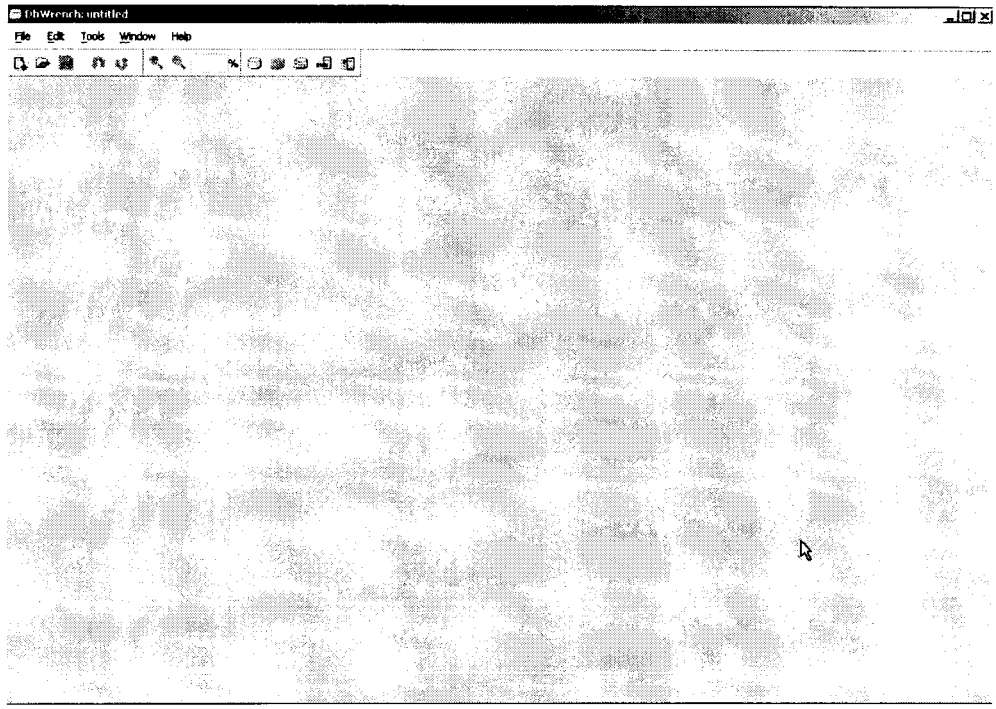


Instalação DbWrench

O próximo programa a ser instalado é a ferramenta DbWrench responsável pela geração do diagrama ER do nosso sistema. Faça o download no seguinte link http://www.dbwrench.com/download/default.shtml clique no menu **Download DbWrench,** leia o contrato da licença de uso da ferramenta e clique em *I ACCEPT* para fazer o *download* do arquivo *zip*.



O DbWrench não exige uma instalação, basta somente extrair os arquivos em uma pasta qualquer, e executar o arquivo de lote dbWrench.bat que nada mais é que o comando Java para executar uma aplicação, então java –jar dbwrencall.jar esse comando abre a aplicação conforme tela abaixo. Passados estes passos, a instalação da ferramenta estará concluída.



Instalação do Tomcat

Conforme visto ao longo do projeto, o Tomcat é um Servlet Container, ou seja, é um servidor onde são instaladas Servlets para tratar as requisições que o servidor receber. Existem muitos Containers, mas o Tomcat foi escolhido por ser gratuito e bastante popular.

Podemos observar que Servlets são bastante semelhantes com CGI. Por que então usar Servlets? Porque servlets possuem as seguintes vantagens sobre CGI:

- Servlets não rodam em um processo separado.
- Servlets são mantidas na memória entre uma requisição e outra.
- Há apenas uma instancia carregada de cada Servlet no container que serve todas as requisições para ela concorrentemente.

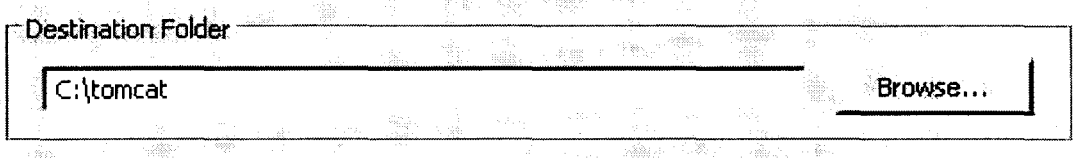
Fazendo *download*

O *download* do Tomcat poderá ser feito pelo seguinte link

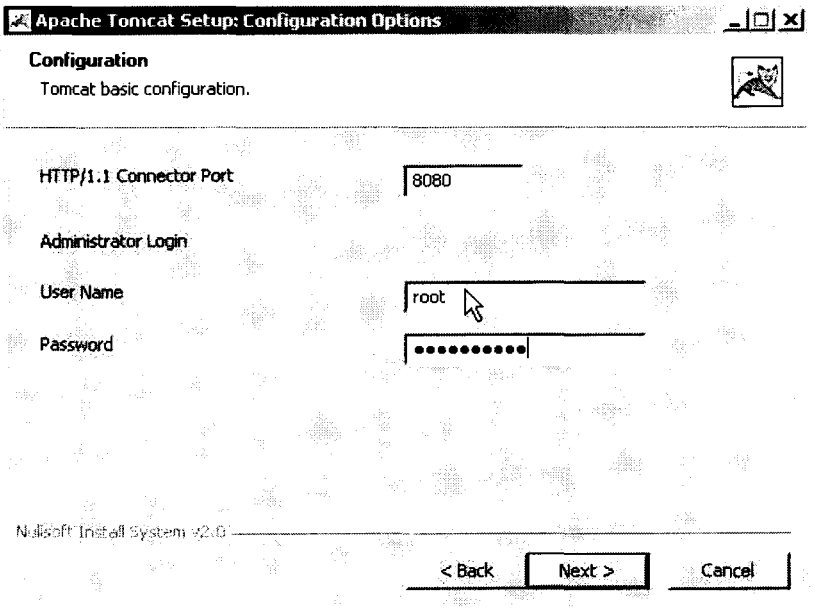
http://jakarta.apache.org/site/downloads/downloads_tomcat-5.cgi role a página até o final e escolha a versão 5.0.28 exe.

- 5.0.28
  - Binary
    - README (contains packaging information)
    - 5.0.28 exe
      - [md5] [pgp]

Depois execute o arquivo jakarta-tomcat-5.0.28.exe para iniciar a instalação. Clique em "Next", leia a licença. Caso aceite, clique em "I agree" e mais uma vez clique em "Next". Na tela a seguir vamos mudar o caminho de instalação do Tomcat de C:\Arquivos de programas\Apache Software Foundation\Tomcat 5.0 para c:\tomcat conforme figura.



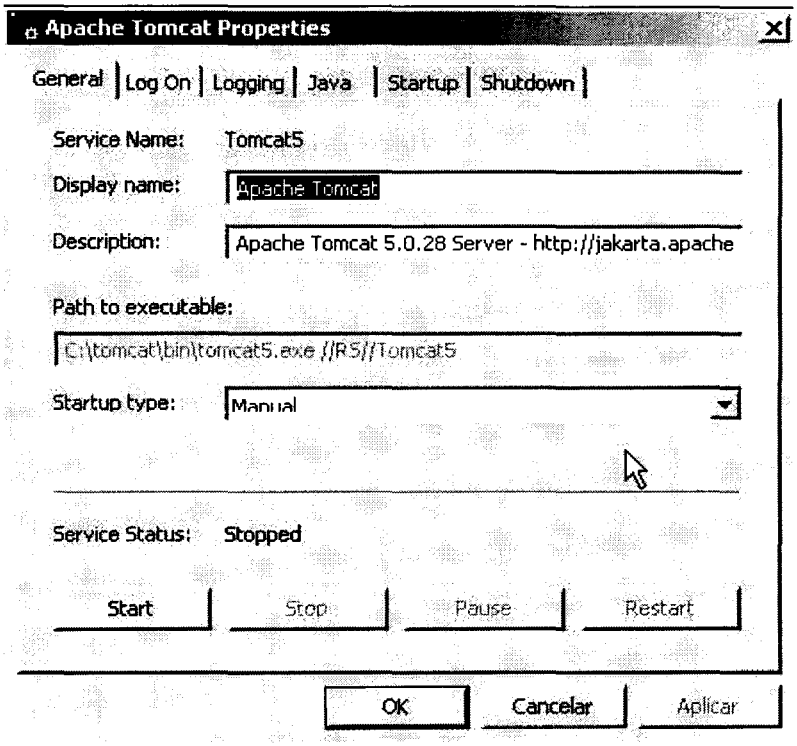
A próxima tela definimos em que porta que o serviço vai rodar, o usuário e senha deverão ser o mesmo criado no postgreSQL, ex: root e senha sorvete.



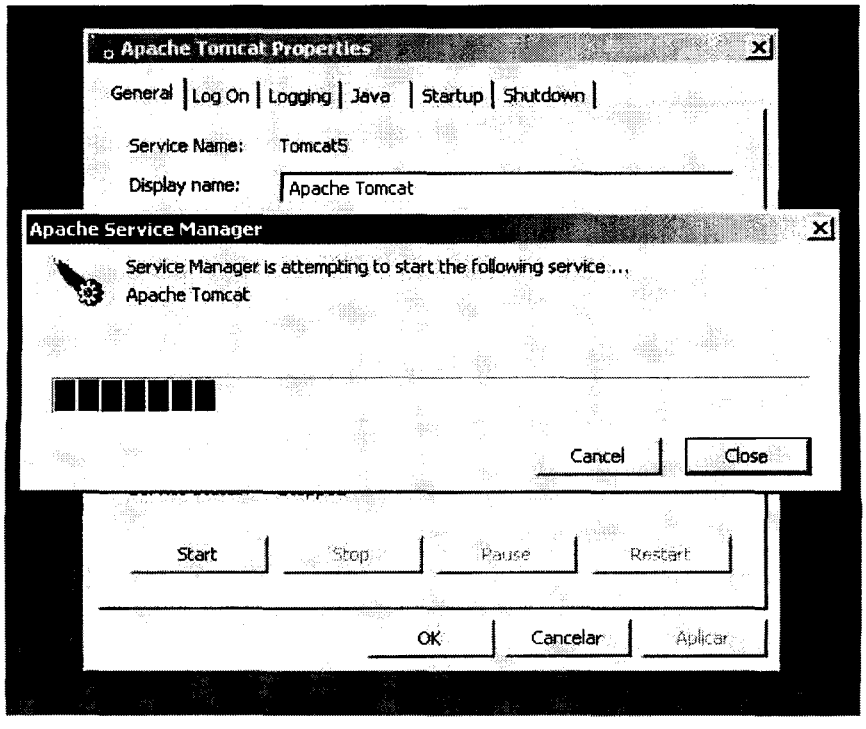
Clique em *next.* Na próxima tela será apresentado, automaticamente, o caminho da nossa máquina virtual: (c:\jsdk). Clique em *Install* para iniciar a cópia dos arquivos.

Ao finalizar a instalação desmarque as opções *Run Apache Tomcat* e *Show readme*, e de um *Finish*.

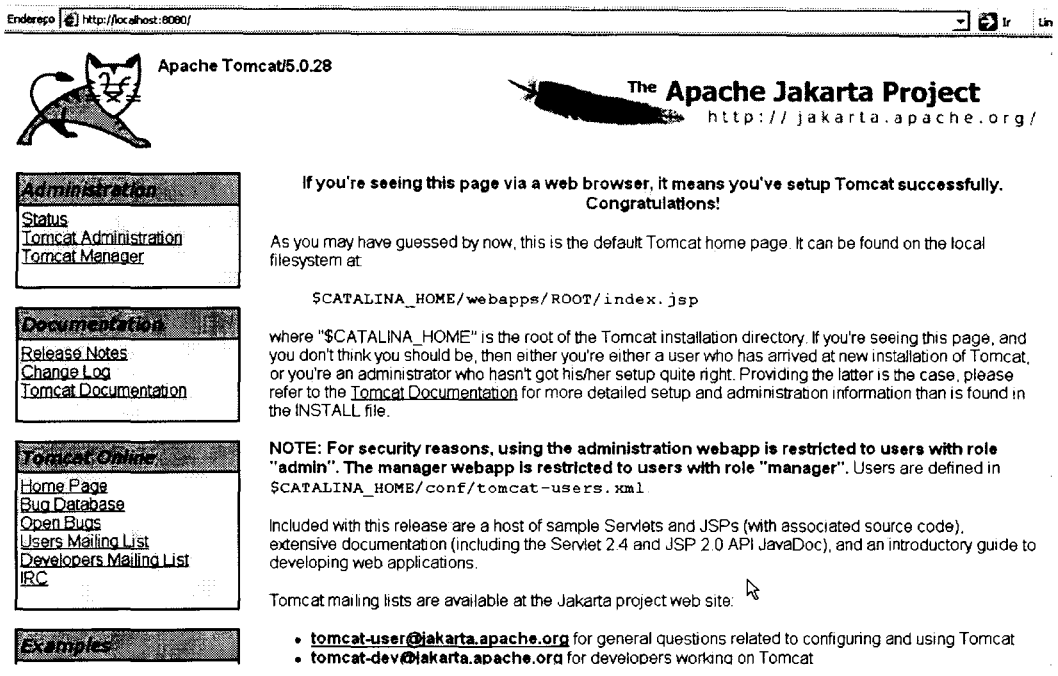
Vamos agora testar se a instalação ocorreu tudo bem, abra o menu iniciar > programs > Apache Tomcat 5.0 > Monitor tomcat, nesse momento ele deverá ter ido para o "systray do Windows", ficando perto do ícone de "relógio", na barra de tarefas. Dê dois cliques sobre ele. Será possível visualizar a tela a seguir:



Clique no botão "Start" para iniciar o serviço.

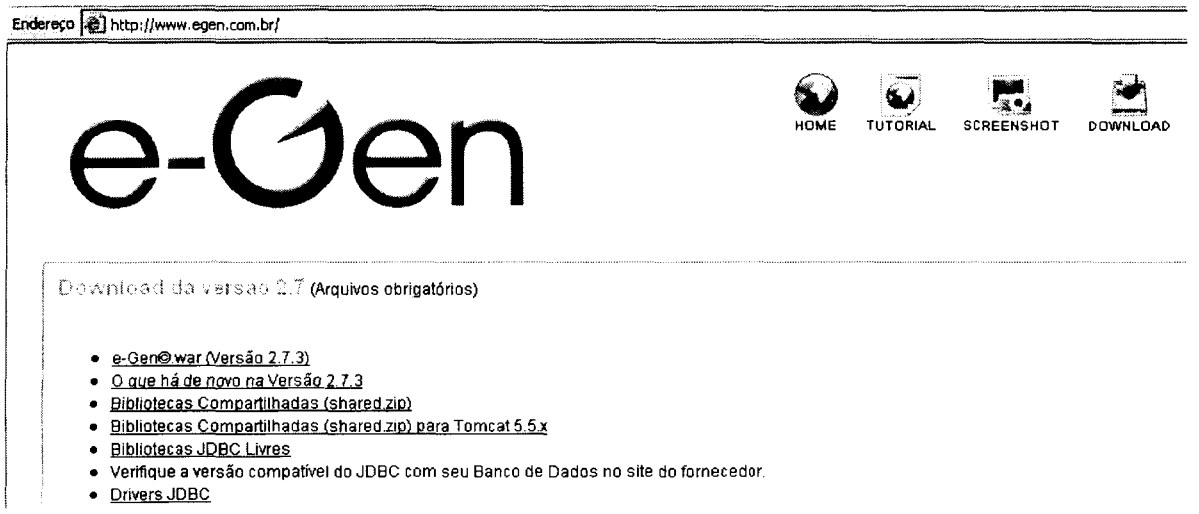


Espere o Tomcat carregar. Logo em seguida, abra a janela do *browser* e digite o seguinte endereço http://localhost:8080 você verá a tela a seguir, indicando que o Tomcat foi instalado com sucesso.



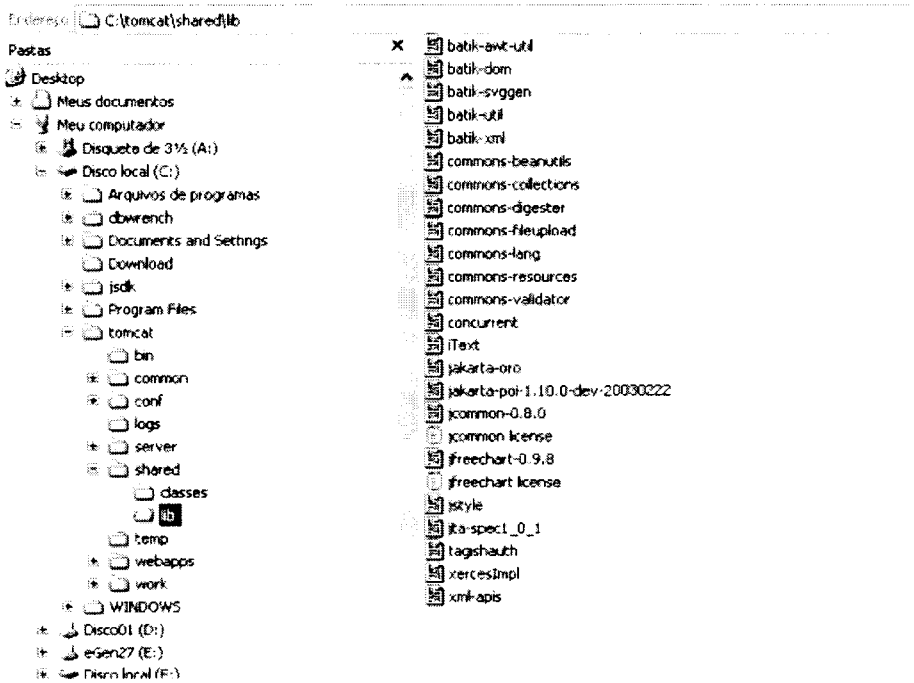
Instalação e-Gen Developer

A instalação do e-Gen não é complexa. Deve-se observar alguns pequenos detalhes para que não ocorram falhas durante o uso da ferramenta. Inicia-se fazendo o *download* dos seguintes arquivos no link do sítio do e-Gen http://www.egen.com.br/pt_br/download.html .

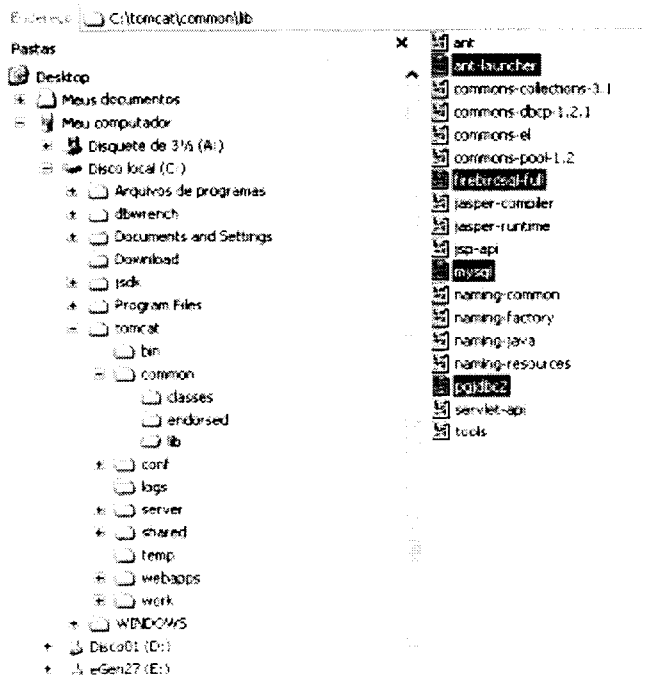


e-Gen©.war (Versão 2.7.5)
Bibliotecas Compartilhadas (shared.zip)
Bibliotecas JDBC Livres

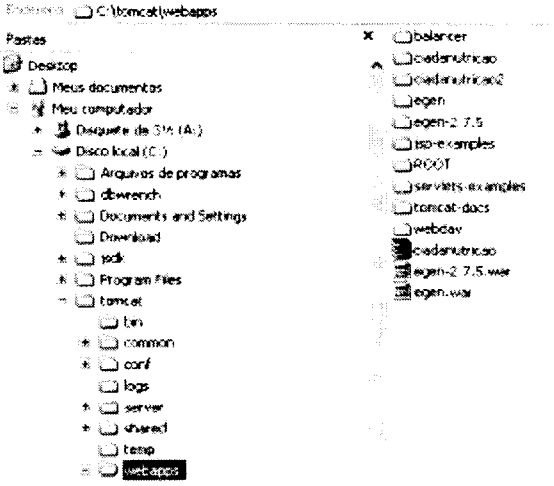
Salve esses arquivos em pasta separadas, em seguida extraia o arquivo shared.zip e copie seu conteúdo para dentro da pasta c:\tomcat\shared\lib conforme figura abaixo.



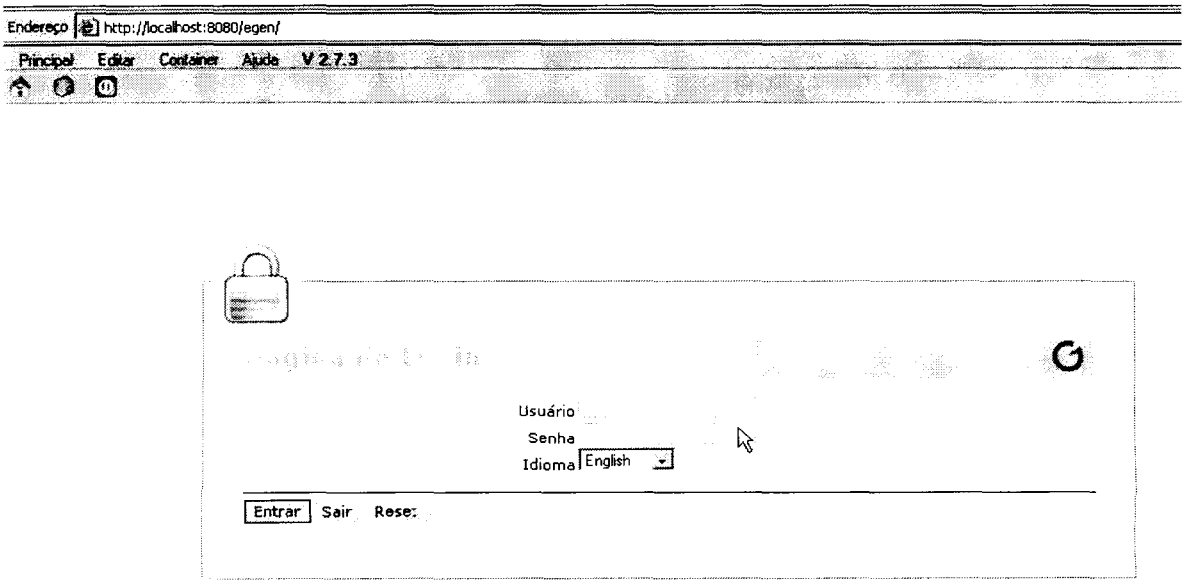
O próximo passo é extrair também o conteúdo do arquivo jdbc.zip e copiá-los para dentro da pasta c:\tomcat\common\lib conforme figura.



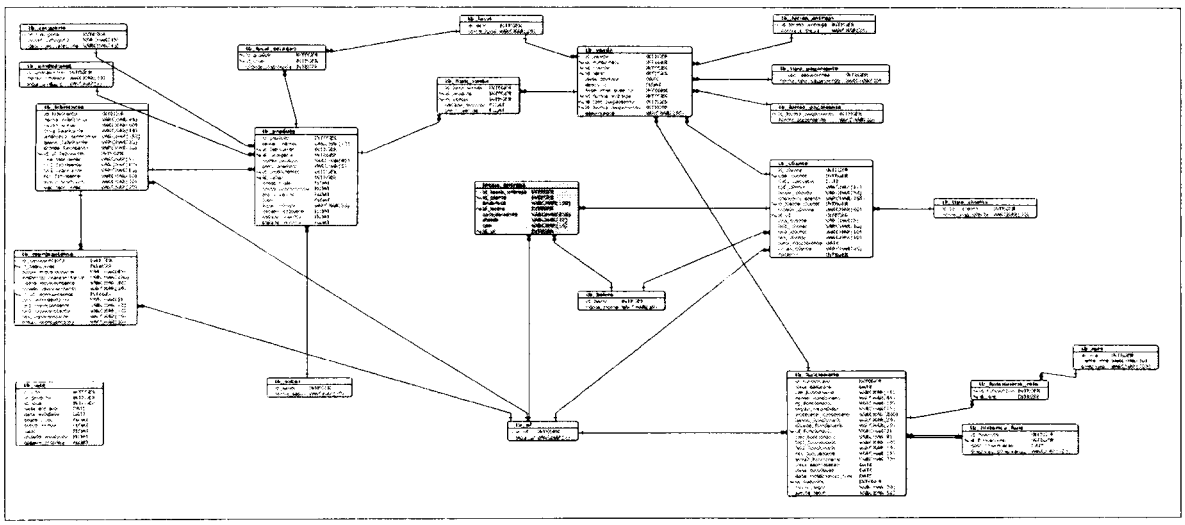
Vamos agora copiar o arquivo egen.war para dentro da pasta c:\tomcat\webapp. Em seguida, iniciaremos serviço do Tomcat pelo seu monitor conforme aprendemos na instalação do Tomcat. Caso o Tomcat já esteja em funcionamento, é necessário parar o serviço (stop) e logo em seguida reiniciá-lo (start). Depois que o serviço estiver funcionando você verá que o Tomcat extraiu o conteúdo do arquivo egen.war e gerou uma pasta chamada egen dentro da pasta webapps conforme figura abaixo.



Agora abra o seu navegador de internet e digite o seguinte endereço http://localhost:8080/egen onde será possível ver a tela a seguir:



Apêndice VII – Modelagem Banco de Dados



Apêndice VIII – Glossário

Propósito do Documento

Este documento tem como objetivo apresentar uma lista dos termos usados no desenvolvimento do Cia Nutri *Manager* - Sistema de Monitoramento On-Line da Companhia da Nutrição. Essa lista de termos é útil para facilitar a comunicação entre usuários, clientes e equipe de desenvolvimento.

Termos do Glossário

FRAMEWORK	Um *framework* de aplicação é uma aplicação reutilizável e semicompleta que pode ser especializada para produzir aplicações personalizadas [Spielmann, 2003]
TEMPLATE	Página WEB que especifica o Layout Padrão
BROWSER	Navegador de páginas na internet
DESIGN PATTERNS	Padrões de Projeto ou *Design Patters* são soluções comuns para problemas comuns de programação. Associado com a programação orientada a objetos os Design Patterns permitem obter as características de reuso e outras qualidades associadas com a orientação a objetos.
MAINFRAMES	Os computadores *mainframe* são máquinas de grande porte, com velocidades de processamento e capacidade de armazenamento bastante elevadas.
STORY CARD	Tipo de cartão utilizado no processo extreme Programming para que os clientes e a equipe de desenvolvimento possam relatar histórias sobre funcionalidades do sistema.
CASO DE USO	A descrição de um conjunto de seqüências de ações, incluindo variantes, que um sistema realiza, fornecendo o resultado observável do valor de um ator.
DIAGRAMA	A apresentação gráfica de um conjunto de elementos, em geral representada como um gráfico conectado de vértices (itens) e arcos (relacionamentos).
IMPLEMENTAÇÃO	Uma realização concreta do contrato declarado por uma interface; a definição de como algo é construído ou computado.
PARÂMETRO	A especificação de uma variável que pode ser alterada, passada ou retornada.
REQUISITO	Uma característica, propriedade ou comportamento desejado de um sistema.
TAREFA	Um caminho único para a execução de um programa; um *thread* ou um processo.