Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

GLENIO LAUTIENE JUSTE

SOFTWARE PARA GESTÃO DE REVENDAS DE SEMENTES HORTALIÇAS

Assis/SP

2014

GLENIO LAUTIENE JUSTE

SOFTWARE PARA GESTÃO DE REVENDAS DE SEMENTES HORTALIÇAS

Projeto de pesquisa apresentado ao Curso de Análise e Desenvolvimento de Sistemas do Instituto Municipal de Ensino Superior de Assis - IMESA e Fundação Educacional do Município de Assis – FEMA, como requisito parcial à obtenção do Certificado de Conclusão.

Orientado: Glenio Lautiene Juste

Orientador: Fernando Cesar de Lima

Assis/SP

2014

FICHA CATALOGRÁFICA

JUSTE, Glenio Lautiene

Software para Gestão de Revendas de Sementes Hortaliças / Glenio Lautiene Juste. Fundação Educacional do Município de Assis, 2014.

51 p.

Orientador: Fernando César de Lima

Trabalho de Conclusão de Curso – Instituto Municipal de Ensino Superior de Assis – IMESA.

1.Orientação a Objetos 2. Primefaces3. Netbeans e Java

CDD: 001.61
Biblioteca da FEMA

SOFTWARE PARA GESTÃO DE REVENDAS DE SEMENTES HORTALIÇAS

GLENIO LAUTIENE JUSTE

Trabalho de Conclusão apresentado ao Curso de Análise e Desenvolvimento de Sistemas do Instituto Municipal de Ensino Superior de Assis - IMESA e Fundação Educacional do Município de Assis – FEMA, como requisito parcial à obtenção do Certificado de Conclusão.

Orientador: _____

Analisador: _____

Assis/SP

2014

DEDICATÓRIA

Dedico este trabalho a Deus e a minha família.

AGRADECIMENTOS

Agradeço primeiramente a Deus por ter me conduzido até o presente momento.
A minha família por me apoiarem neste projeto.
Ao meu orientador Fernando Lima.
Ao meu amigo e professor Frederico Manoel Bertoluci Reis.
Meu amigo de estudos Denis Mendonça Ladeira.

"Se queres paz, te prepara para a guerra, se não queres nada, descansa em paz”
Humberto Gessinger

RESUMO

Este trabalho tem como objetivo levantar toda a documentação para desenvolver um software para a empresa Covale, empresa fundada em 1996, esta que atua no ramo de revenda de sementes de hortaliças.

No início, com muitas dificuldades, a empresa foi se firmando no mercado, porém com o passar dos anos e com uma administração bem informada com as tendências de mercado, a Covale se expandiu e hoje é uma empresa consolidada com cerca de duzentos clientes fixos mais as vendas com clientes não fixos, e com uma movimentação diária em torno de quarenta vendas por dia.

Com a expectativa de crescimento de cinquenta por cento nos próximos dois anos surgiu a necessidade e o interesse de informatizar a empresa Covale, auxiliando assim com dados precisos o administrador nas tomadas de decisões futuras da empresa.

Palavras - Chave: Revenda de hortaliças, software de gestão.

ABSTRACT

This work aims to raise documentation to develop a software for the company Covale, founded in 1996, this company engaged in the resale of vegetable seeds.
Earlier, with many difficulties, the company was increasing in the market, but over the years with an administration well informed with market trends, the Covale expanded and today is an established company with nearly two hundred regular customers more sales with no fixed clients, and with a sales daily around forty sales per day.
With the expected growth of fifty percent over the next two years, the need and interest of computerizing the company Covale emerged, thus assisting with accurate data administrator in making future business decisions.

Keywords: Sale of vegetables, management software.

Lista de Figuras

Lista de Tabelas

SUMÁRIO

1. INTRODUÇÃO

Em um mundo onde a globalização está cada vez mais presente, os processos cá
vez mais automatizados e as barreiras de distância sendo quebrada pelo aume
da popularização da internet, a necessidade da informatização e armazenamento de
dados e informações de cada mercado tornam-se o primeiro passo para a migração de seu negócio para a informática.

A forma de simplificar e melhorar nosso modo de vida motiva o homem a buscar meios e soluções para aprimorar o modo em que vivemos.

Um desses modos foi à informática, que começou a surgir por volta de 2000 A.C. com a invenção do Ábaco. Essa invenção rudimentar foi evoluindo até chegar os dias atuais com o computador.

O computador é uma máquina que processa dado, orientado por um conjunto de instruções e destinada a produzir resultados completos, com um mínimo de intervenção humana (LIMA, 2012).

Nos tempos atuais, quase todos os eventos cotidianos pode ser feito através do computador como fazer compras, fazer pagamentos, sendo que até nos automóveis se emprega muita tecnologia. Praticamente todo o comércio no mundo usa computador.

Seguindo a tendência de que tudo está se informatizando, surgiu o interesse do proprietário da empresa Covale obter um software para a empresa.

Diante da necessidade do proprietário não ter um sistema informatizado, sem nenhum controle adequado de estoque e fluxo de caixa, onde tudo é feito manualmente através da movimentação do dia-a-dia do estabelecimento comercial, surgiu à necessidade de desenvolver um software para a empresa para dar um suporte adequado para o gestor, permitindo assim um controle total e efetivo no ambiente da empresa.

O software irá armazenar dados dos produtos do estoque da empresa e permitir a visualização e impressão de relatórios para facilitar o domínio do usuário à empresa física dando assim um suporte necessário para o gestor gerenciar e auxiliar nas tomadas de decisões quanto aos investimentos futuros da empresa.

1.1. Objetivo do Trabalho

O objetivo do trabalho é desenvolver um software para a empresa Covale que armazenará dados com precisão e segurança e auxiliará o administrador nas tomadas de decisões quanto ao futuro da empresa. O software deverá fornecer à empresa os controles de estoque, vendas e compras, incluindo cadastros de clientes, de fornecedores, de vendedores e de cidades. Finalizando o sistema oferece a opções de relatórios de clientes, fornecedores, estoque, compra e venda.

1.2. Justificativa

A empresa não possui nenhum software, e a necessidade de ter um controle mais efetivo dos dados da empresa e uma rapidez na hora de consultar os dados armazenados em um programa levou a necessidade de o administrador adquirir um software para sua empresa.

1.3. Público Alvo

O software será desenvolvido para a empresa Covale, uma empresa de pequeno porte que atua no ramo de revenda de sementes de hortaliças.

2. ANÁLISE DOS REQUISITOS

A análise de requisitos é um processo repetitivo, que envolve a compreensão do domínio, assim como a coleta, a classificação, a estruturação e validação dos requisitos (Ian Sommerville, 2003).

2.1. Levantamento dos Requisitos

Conforme Burnett (1998), requisitos, simplesmente pode ser definido como "algo que um cliente necessita". Um entendimento completo dos requisitos do software é fundamental para o desenvolvimento do software. Se um software for mal analisado e especificado, não importa o quanto o código está bem escrito, o software desapontará o usuário e poderá trazer problemas ao desenvolvedor.

2.1.1. Formas Adotadas

A forma adotada no levantamento dos requisitos foi primeiramente fazer uma entrevista com o proprietário da empresa, tendo como objetivo compreender o que se esperava do software. Na ocasião, o analista teve a oportunidade de conhecer o empreendimento (espaço físico), ver e analisar os produtos comercializados.
Foram entrevistados também os funcionários, dois no total, que deram suas opiniões e como um sistema informatizado poderia melhorar na rotina diária da empresa.
Foi constatada a ineficiência de armazenamento dos dados da empresa. De uma forma antiga, onde tudo é feito em papel e caneta, não existia uma segurança adequada para armazenar com precisão e principalmente para direcionar as tomadas de decisões do gestor.
Um questionário foi preenchido para dar suporte no levantamento de requisitos. Segue abaixo um pequeno trecho do questionário feito com o gestor:

1) Qual o ramo do seu empreendimento?
Resposta: O foco é a venda de sementes para hortas tanto para clientes quanto para lojas revendedoras e pequenos plantios.

2) Como é feito o controle de dados do empreendimento?

Resposta: É feito através de anotações em papel.
3) As sementes comercializadas têm rastreabilidade?
Resposta: Não.

4) Quais outros produtos comercializados na empresa?
Resposta: Rações e vitaminas para aves, venenos para pragas de hortas e adubo.

5) Quais os problemas gerados por a empresa não ter um sistema informatizado?
Resposta: Muita dificuldade no controle de produtos e estoque, no controle de caixa e na emissão de relatórios.

6) Quais os requisitos necessários em um sistema para atender as necessidades da empresa?
Resposta: O sistema precisa ter um cadastro de clientes, de fornecedor, de estoque e um controle do caixa.

7) O que o senhor espera da implantação de um sistema em seu comércio?
Resposta: Espero que o sistema ofereça toda a segurança necessária para me auxiliar com mais eficiência mantendo o crescimento do meu comércio no mercado competitivo.

Foi usado também o auxílio de pesquisas na Internet e em livros, para encontrar e auxiliar no planejamento do desenvolvimento do software e incrementado alguns protótipos de telas do programa.

2.1.2. Conflitos Encontrados

Os conflitos encontrados são de o cliente não ter uma noção clara do software para a empresa e tenha receio de não conseguir articulá-lo. O software deverá ser web e no cadastro de vendas, podem ocorrer duas vendas para a mesma pessoa, porém, uma pode ser física e outra jurídica.

2.1.3. Propostas de Solução

A proposta de solução é dar um treinamento específico ao cliente e funcionários da empresa e fazer com que o usuário na hora de cadastrar a venda, quando inserido o nome do cliente, o programa exiba uma alternativa no qual o usuário faça a opção entre pessoa física ou pessoa jurídica.

2.2. Validação dos Requisitos

Essa atividade analisa os requisitos quanto a sua adequação, consistência e integridade. Durante esse processo, fatalmente são descobertos erros na documentação de requisitos. Os requisitos devem então ser alterados, a fim de corrigir esses problemas (Ian Sommerville, 2003).

Na validação dos requisitos, depois de certo período de pesquisa, foram feitas algumas telas do sistema e o que cada tela de cadastro deveria conter para um melhor entendimento do gestor e dos funcionários. Para esse desenvolvimento foi utilizada a ferramenta Axure, onde foi feito uma prototipagem do sistema a ser desenvolvido para que o gestor pudesse ter uma melhor noção do software.

2.2.1. Resultados da Revisão Formal

O gestor se mostrou satisfeito com o material provisoriamente apresentado. Apresentados também aos funcionários, estes mostraram-se muitos satisfeitos com o protótipo de telas feitas com o programa Axure.

2.3. Prototipagem

Um protótipo é uma versão inicial de um sistema de software, que é utilizada para mostrar e experimentar opções de projeto e, em geral, para conhecer mais sobre os problemas e suas possíveis soluções (Ian Sommerville, 2003).
Na entrevista com o gestor, foi feito um esboço em papel e caneta de algumas telas do sistema, permitindo assim que o gestor expressasse sua opinião e, em primeiro plano, pudesse ter uma noção de como a interface do sistema ficaria no seu

resultado final.

Abaixo, alguns exemplos do protótipo apresentado ao cliente desenvolvidos no programa Axure RP e uma breve descrição das telas.

Na figura 1 é mostrada a tela inicial do sistema, onde o usuário executa login e senha para acesso ter acesso ao sistema.

Figura 1. Protótipo Tela Login

Em seguida, na figura 2, estando corretas a efetuação do login e senha, aparece uma tela com um menu para que o usuário escolha a opção que deseja:

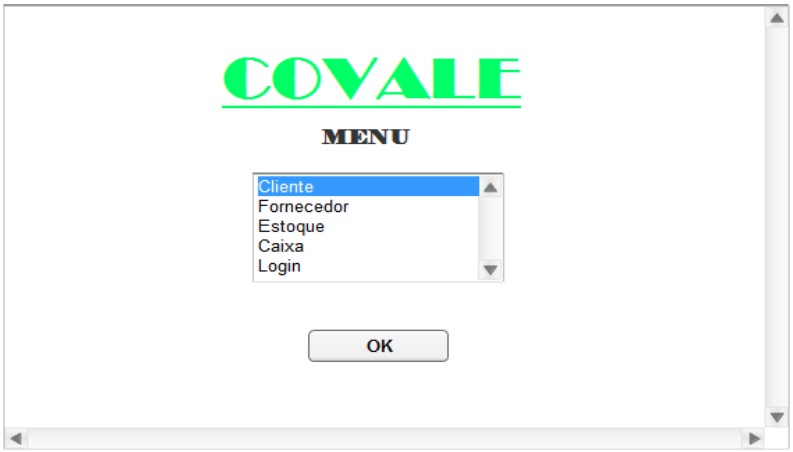

Figura 2. Protótipo Tela Menu

Conforme na figura acima, o usuário opta pela alternativa Cliente. Na figura 3 ilustra o usuário escolhendo a opção por cadastrar um novo cliente.

Figura 3. Protótipo Tela Cadastro Cliente

Na figura 4 ilustra um exemplo do usuário efetuando um cadastro de cliente.

Figura 4. Protótipo Tela Cadastro de Clientes

A figura 5 ilustra a tela final do sistema após o cadastramento de um cliente.

Figura 5. Protótipo Tela Final

2.3.1. Planos de Teste de Aceitação

O plano de teste ocorrerá da seguinte maneira: Será entregue ao cliente uma versão beta do software para ele ir testando e avaliando o programa até chegar a comum acordo com as preferências do cliente.

2.4. Gerenciamento dos Requisitos

O gerenciamento de requisitos é um modelo sistemático para encontrar, documentar, organizar e rastrear os requisitos variáveis de um sistema (WTHREEX, 2014).

2.4.1. Atributos dos Requisitos

Atributos de requisitos capturam informações adicionais, tais como risco, iteração planejada, origem do requisito, sobre cada requisito. Estas informações adicionais são usadas para a gerência do projeto (WTHREEX, 2014).

2.5. Especificações dos Requisitos

Pode-se entender requisito como uma função, restrição ou propriedade que deve ser fornecida, encontrada ou atendida para satisfazer às necessidades do usuário do sistema (MACORATTI.NET, 2014).

O software tem o objetivo de auxiliar no controle financeiro da empresa, para que o gestor tenha dados precisos e auxilie em suas decisões futuras do seu negócio.

O software dará um suporte ao gestor também quanto ao controle de estoque, permitindo que o gestor saiba com precisão o volume de produtos mais vendidos, produtos com prazo de validade a vencer entre outras coisas.

Auxiliará no cadastro de clientes, saber o que determinado cliente mais comprou, o tempo de uma compra e outra e etc.

3. PLANEJAMENTO DO PROJETO

À medida que o projeto avança, contudo, o plano do projeto deve ser revisto, uma vez que problemas podem surgir ou porque se ganha um maior entendimento sobre o problema. Essas revisões do plano de projeto são ditas atividades de acompanhamento do projeto e tipicamente são realizadas nos marcos do projeto (Ricardo de Almeida Falbo, 2005).

3.1 Descrições da Metodologia de Análise

Para a fase de análise do sistema foi utilizado à linguagem UML (Unified Modeling Language). A UML é uma linguagem visual utilizada para modelar softwares baseados no paradigma de orientação a objetos (Gilleanes T. A. Guedes, 2009).
Os objetivos da UML são: especificação, documentação, estruturação para sub-visualização e maior visualização lógica do desenvolvimento completo de um sistema de informação. A UML é um modo de padronizar as formas de modelagem.
A UML 2.0 possui diversos modelos de diagramas como, por exemplo:

– Diagrama de Classes
– Diagrama de Objetos
– Diagrama de Componentes
– Diagrama de Instalação
– Diagrama de Pacotes
– Diagrama de Estrutura
– Diagrama de Caso de Uso
– Diagrama de Estados
– Diagrama de Atividade

Para criar esses diagramas necessitamos de ferramentas específicas para substituir o papel e a caneta.
Modelar um sistema é uma forma eficiente de documentar todas as etapas de desenvolvimento do software, o que proporciona a empresa uma organização de todas essas informações. Para fazer essas modelagens do sistema foi utilizada a ferramenta Astah Community.

3.2 Descrições do Ambiente de Desenvolvimento

3.2.1 Netbeans – O NetBeans foi iniciado em 1996 por dois estudantes tchecos na Universidade de Charles, em Praga, quando a linguagem de programação Java ainda não era tão popular como atualmente. Primeiramente o nome do projeto era Xelfi, em alusão ao Delphi, pois a pretensão deste projeto era ter funcionalidades semelhantes aos IDEs então populares do Delphi que eram mais atrativas por serem ferramentas visuais e mais fáceis de usar, porém com o intuito de ser totalmente desenvolvido em Java.

Em 1999 o projeto já havia evoluído para uma IDE proprietário, com o nome de NetBeans DeveloperX2, nome que veio da idéia de reutilização de componentes que era à base do Java. Nessa época a empresa Sun Microsystems havia desistido de sua IDE Java Workshop e, procurando por novas iniciativas, adquiriu o projeto NetBeans DeveloperX2 incorporando-o a sua linha de softwares.

O NetBeans fornece uma base sólida para a criação de projetos e módulos, possui um grande conjunto de bibliotecas, módulos e APIs (Application Program Interface, um conjunto de rotinas, protocolos e ferramentas para a construção de aplicativos de software) além de uma documentação vasta — inclusive em português — bem organizada. Tais recursos auxiliam o desenvolvedor a escrever seu software de maneira mais rápida.

Atualmente está distribuído em diversos idiomas e isto tem o tornado cada vez mais popular, facilitando o acesso a iniciantes em programação e possibilitado o desenvolvimento de aplicativos multilíngüe.

Como o NetBeans é escrito em Java, é independente de plataforma, funciona em qualquer sistema operacional que suporte a máquina virtual Java (JVM) (OFICINADANET, 2014).

3.3. FERRAMENTAS UTILIZADAS

Neste capítulo será descrito as ferramentas e tecnologias adotadas para o desenvolvimento do software.

3.3.1. Axure RP - é uma wireframing , prototipagem rápida , e especificação de ferramenta de software voltado para aplicações web e desktop. Ele oferece recursos normalmente encontrados em ferramentas de diagramação, como arrastar e soltar colocação, redimensionamento e formatação de widgets. Além disso, ele tem recursos para anotar os widgets e definir interações como a vinculação, de ligação condicional, simulando guia controles, exibir / ocultar elemento etc. Há suporte para simulação de média fidelidade de Rich Internet Applications . O Axure RP pode gerar protótipos HTML e Microsoft Word especificações dos diagramas (APPNOVATION, 2009).

3.3.2. Astah Community - Astah Community é uma ferramenta gratuita voltada para a modelagem de diagramas UML (Unified Modeling Language). Além do Astah Community, existem outras três versões: Astah UML, Astah Professional e Astah Share que disponibilizam outras funcionalidades além da modelagem UML, porém, sua licença é comercial.

A ferramenta Astah Community é conhecida por sua praticidade e simplicidade em elaborar diagramas, como por exemplo: diagramas de classe, caso de uso, sequência, atividade, comunicação, máquina de estado, componentes, implantação, estrutura de composição, objetos e pacotes (CONSULTORIODESOFTWARE, 2014). O Astah Community foi utilizado para desenvolver os diagramas de Caso de Uso, narrativas de Caso de Uso e diagrama de Classe.

3.3.3. DBDesigner - O DBDesigner 4 é uma ferramenta CASE (Computer-Aided Software Engineering) desenvolvida pela empresa Fabulous Force Database Tools. Esta ferramenta é livre e utilizada para a modelagem de dados visual que está disponível sob a licença GNU General Public License (GLP).

O DBDesigner utiliza qualquer banco de dados, entre eles o MySQL, Oracle, MSSQL e PostgreSQL, mas obviamente não se limita apenas a esses. Através dele podemos modelar tabelas de forma gráfica, relacionamentos e muito mais. Utilizando este poderoso ambiente temos a geração da modelagem, do projeto, da implementação e da manutenção integradas em apenas um ambiente. (DEVMEDIA, 2014).

3.3.4. pgAdmin - é projetado para atender às necessidades de todos os usuários, desde escrever consultas SQL simples para o desenvolvimento de bancos de dados complexos. A interface gráfica suporta todas as funcionalidades do PostgreSQL e torna a administração mais fácil. O aplicativo também inclui um editor com destaque de sintaxe SQL, um editor de código do lado do servidor, um SQL / batch / shell agente de agendamento de trabalho, o apoio à Slony-I mecanismo de replicação e muito mais. Conexão com o servidor pode ser feita utilizando soquetes de domínio Unix (em * nix) TCP / IP, ou, e pode ser criptografado SSL para a segurança. Não há drivers adicionais são necessários para se comunicar com o servidor de banco de dados.

O pgAdmin é desenvolvido por uma comunidade de especialistas do PostgreSQL em todo o mundo e está disponível em mais de uma dúzia de idiomas. É um Software Livre liberado sob a licença PostgreSQL (PGADMIN, 2014).

3.3.5. PrimeFaces - PrimeFaces é uma biblioteca de componentes de código aberto para o JSF 2.0 com mais de 100 componentes, permitindo criar interfaces ricas para aplicações web de forma simplificada e eficiente. Ele é considerado muito melhor do que muitas outras bibliotecas de componentes JSF.

O PrimeFaces é um framework da Prime Teknoloji (empresa da Turquia) que oferece um conjunto de componentes ricos para o JavaServer Faces. Seus componentes foram construídos para trabalhar com AJAX por “default”, isto é, não é necessário nenhum esforço extra por parte do desenvolvedor para realização de chamadas assíncronas ao servidor. Além disso, o PrimeFaces permite a aplicação de temas (skins) com o objetivo de mudar a aparência dos componentes de forma simples (WILLIAMGAMERS, 2012).

3.3.6. Apache Tomcat - é uma implementação de software de fonte aberta das tecnologias Java Servlet e JavaServer Pages. As especificações Java Servlet e JavaServer Pages são desenvolvidas sob o Java Community Process . Apache Tomcat é desenvolvido em um ambiente aberto e participativo e liberado sob a Licença Apache. Poderes do Apache Tomcat numerosos em grande escala, aplicações de missão crítica da web através de uma variada gama de indústrias e organizações (TOMCAT, 2014).

3.3.7. JAVA EE - é uma plataforma de programação para servidores na linguagem de programação Java.1 A plataforma fornece uma API e um ambiente de tempo de execução para o desenvolvimento e execução de softwares corporativos, incluindo serviços de rede e web, e outras aplicações de rede de larga escala, multicamadas, escaláveis, confiáveis e seguras. Java EE estende a Java Platform, Standard Edition (Java SE),2 fornecendo uma API para mapeamento objeto-relacional, arquiteturas multicamada e distribuídas e web services. A plataforma incorpora um desenho amplamente baseado em componentes modulares rodando em um servidor de aplicação. Softwares para Java EE são primeiramente desenvolvidos na linguagem de programação Java. A plataforma enfatiza a convenção sobre configuração e anotações para configuração (NETBEANS.ORG, 2014).

3.3.8. JavaServer Faces (JSF) - é um framework MVC baseado em Java para a construção de interfaces de usuário baseadas em componentes para aplicações web. Possui um modelo de programação dirigido a eventos, abstraindo os detalhes da manipulação dos eventos e organização dos componentes, permitindo que o programador se concentre na lógica da aplicação.

Foi formalizada como um padrão através do Java Community Process e faz parte da Java Platform, Enterprise Edition.

JavaServer Faces é baseada em um modelo de desenho de IU (interface de usuário) baseada em componentes, usando arquivos XML chamados de modelos de visão ou Facelets views. Os pedidos são processados pelo FacesServlet, que carrega o modelo de visão adequado, constrói uma árvore de componentes, processa os eventos e apresenta a resposta, normalmente na linguagem HTML, para o cliente. O estado de componentes de interface do usuário e outros objetos de interesse de escopo, é salvo no final de cada pedido em um processo chamado stateSaving (nota: transiente true) e restaurado na próxima criação desta visão. Objetos e estados podem ser salvos ou no cliente ou no servidor (LUCKOW, Décio Heinzelmann; MELO, Alexandre Altair de. Programação Java para a Web, 2010).

3.4. Ferramentas para análise

Para a análise foram usadas as ferramentas Axure RP, Astah Community e o DBDesigner.

O Axure RP foi utilizado para desenvolver alguns protótipos de telas da interface do software para que se pudesse ter um conhecimento inicial do que seria o software.
O Astah Community foi para o desenvolvimento dos diagramas de Caso de Uso e a especificação de Caso de Uso e para elaborar o diagrama de Classe.
O DBDesigner a elaboração do diagrama ER (Entidade-Relacionamento).
Essas ferramentas são gratuitas.

3.5. Banco de Dados

Um Banco de Dados é antes de qualquer coisa uma coleção logicamente coerente de dados com determinada significação intrínseca. Em outras palavras um arquivo contendo uma série de dados de um cliente, um arquivo com dados aleatoriamente gerados e dois arquivos padrão dbf (dBase) que tem uma relação definida entre ambos, não pode ser considerada uma Base de Dados Real.
Um Banco de Dados contém os dados dispostos numa ordem pré-determinada em função de um projeto de sistema, sempre para um propósito muito bem definido.
Um Banco de Dados representará sempre aspectos do Mundo Real. Assim sendo uma Base de Dados (ou Banco de Dados, ou ainda BD) é uma fonte de onde poderemos extrair uma vasta gama de informações derivadas, que possui um nível de interação com eventos como o Mundo Real que representa. A forma mais comum de interação Usuário e Banco de Dados dá-se através de sistemas específicos que por sua vez acessam o volume de informações geralmente através da linguagem SQL. Os Administradores de Banco de Dados (DBA) são responsáveis pelo controle ao acesso aos dados e pela coordenação da utilização do BD. Já os projetistas de Banco de Dados (DBP) são analistas que identificam os dados a serem armazenados em um Banco de Dados e pela forma como este serão representado.
Os Analistas e Programadores de Desenvolvimento criam sistemas que acessam os dados da forma necessária ao Usuário Final, que é aquele que interage diretamente com o Banco de Dados (SURIAN; NICOCHELLI, 2014).

3.5.1. PostgreSQL - é um sistema gerenciador de banco de dados objeto relacional (SGBDOR), desenvolvido como projeto de código aberto.

O PostgreSQL é um dos resultados de uma ampla evolução que se iniciou com o projeto Ingres, desenvolvido na Universidade de Berkeley, Califórnia. O líder do projeto, Michael Stonebraker, um dos pioneiros dos bancos de dados relacionais, deixou a universidade em 1982 para comercializar o Ingres, porém retornou a ela logo em seguida. Após seu retorno a Berkeley, em 1985, Stonebraker começou um projeto pós-Ingres com o objetivo de resolver problemas com o modelo de banco de dados relacional. O principal problema era a incapacidade de o modelo relacional compreender “tipos” (atualmente, chamados de objetos), ou seja, combinações de dados simples que formam uma única unidade.

O projeto resultante, chamado Postgres, era orientado a introduzir a menor quantidade possível de funcionalidades para completar o suporte a tipos. Estas funcionalidades incluíam a habilidade de definir tipos, mas também a habilidade de descrever relações - as quais até este momento eram amplamente utilizadas, mas completamente mantidas pelo usuário. No Postgres, o banco de dados "compreendia" as relações e podia obter informações de tabelas relacionadas utilizando regras.

Iniciando em 1986, a equipe divulgou uma série de documentos descrevendo a base do sistema e em 1988 o projeto possuía um protótipo funcional. A versão 1 foi liberada para um grupo pequeno de usuários em junho de 1989, seguida pela versão 2 com um sistema de regras reescrito em junho de 1990. Para a versão 3, liberada em 1991, o sistema de regras foi reescrito novamente, mas também foram adicionados suporte para múltiplos gerenciadores de armazenamento e um melhorado motor de consultas. Já em 1993, Postgres havia crescido imensamente em popularidade e possuía uma grande demanda por suporte e por novas funcionalidades. Após a liberação da versão 4, a qual era uma simples versão de limpeza, o projeto foi oficialmente abandonado pela Universidade de Berkeley.

Entretanto, devido ao fato do seu código fonte estar sob uma licença BSD, o seu desenvolvimento foi continuado. Em 1994, dois estudantes , Andrew Yu e Jolly Chen, adicionaram um interpretador SQL para substituir a linguagem QUEL (desenvolvida para o Ingres) e o projeto foi renomeado para Postgres95. Com a

divulgação de seu código pela Internet, Postgres95 iniciou uma nova vida como software open source.

Em agosto de 1996, Marc Fournier, Bruce Momjian e Vadim B. Mikheev lançaram a primeira versão externa da Universidade de Berkeley e deram início à tarefa de estabilizar o código herdado. Também em 1996, o projeto foi renomeado para PostgreSQL a fim de refletir a nova linguagem de consulta ao banco de dados: SQL. A primeira versão de PostgreSQL, a 6.0, foi liberada em janeiro de 1997. Desde então, um grupo de desenvolvedores e de voluntários de todo o mundo, coordenados pela Internet, têm mantido o software e desenvolvido novas funcionalidades.

O PostgreSQL é um projeto open source coordenado pelo PostgreSQL Global Development Group. Embora as atividades do grupo sejam patrocinadas por diversas organizações de todo o mundo, seu modelo de desenvolvimento é o modelo Bazar (originalmente apresentado em A Catedral e o Bazar de Eric S. Raymond) (POSTGRESQL, 2014).

3.6. JAVA DATA BASE CONNECTIVITY (JDBC)

JDBC, ou Java Data Base Connectivity (Conectividade Java com Banco de Dados), é um conjunto de interfaces e classes em Java que tem como objetivo padronizar o modo com que um aplicativo qualquer se conecte com banco de dados. É inspirado no padrão Microsoft de acesso a banco de dados, ODBC (Open DataBase Connectivity), porém tem a vantagem de ser multi-plataforma. Além da independência da plataforma, o Java também visa obter independência de banco de dados, conforme será verificado mais adiante. Isto significa que ao se trocar de banco de dados, esperasse pouca alteração na aplicação, aumentando o reuso de um aplicativo.

A API (Application Programming Interface) JDBC permite utilizar caminhos distintos para acessar a base de dados, ou seja, podem-se escolher diferentes drivers de diferentes tipos. O primeiro tipo é a conexão através de uma ponte jdbc-odbc. Levando em consideração que a ponte jdbc-odbc já vem incorporada ao JDK, e juntamente com o driver ODBC, esta opção se torna a mais fácil de utilizar. No segundo tipo, o driver é obtido a partir de uma API nativa. Isto significa que o driver

Java faz chamadas nativas a C ou C++ para subrotinas definidas pelo fornecedor do banco de dados. Esta alternativa exige a instalação de software cliente. O último tipo se dá através de um driver específico JDBC (G. Reese, Database Programming with JDBC & Java, 2000).

O JDBC usa a classe DriverManager e duas interfaces Driver e Connection, para se conectar a um banco de dados. A classe Driver proporciona à JDBC um ponto de partida para a conectividade de banco de dados respondendo às requisições de conexão de DriverManager e fornecendo informações sobre a implementação em questão. A classe DriverManager tem como principal responsabilidade manter uma lista de implementações de drivers. A classe Connection é utilizada para enviar uma série de dados de instruções SQL ao banco de dados e controlar o registro ou o aborto dessas instruções.

A API do JDBC disponibiliza classes para trabalhar com a manipulação de registros de banco de dados. Statements e PreparedStatement são as classes que facilitam a manipulação de dados nessa API. São as instâncias dessas classes que enviam suas declarações SQL para o banco de dados (Brian Jepson, Programando banco de dados em Java, 1997).

4. ANÁLISE E DOCUMENTAÇÃO DO SISTEMA (DIAGRAMAS UML)

4.1. WBS – WorkBreakdownStructure

Na figura 6 é ilustrada a estrutura analítica do trabalho.

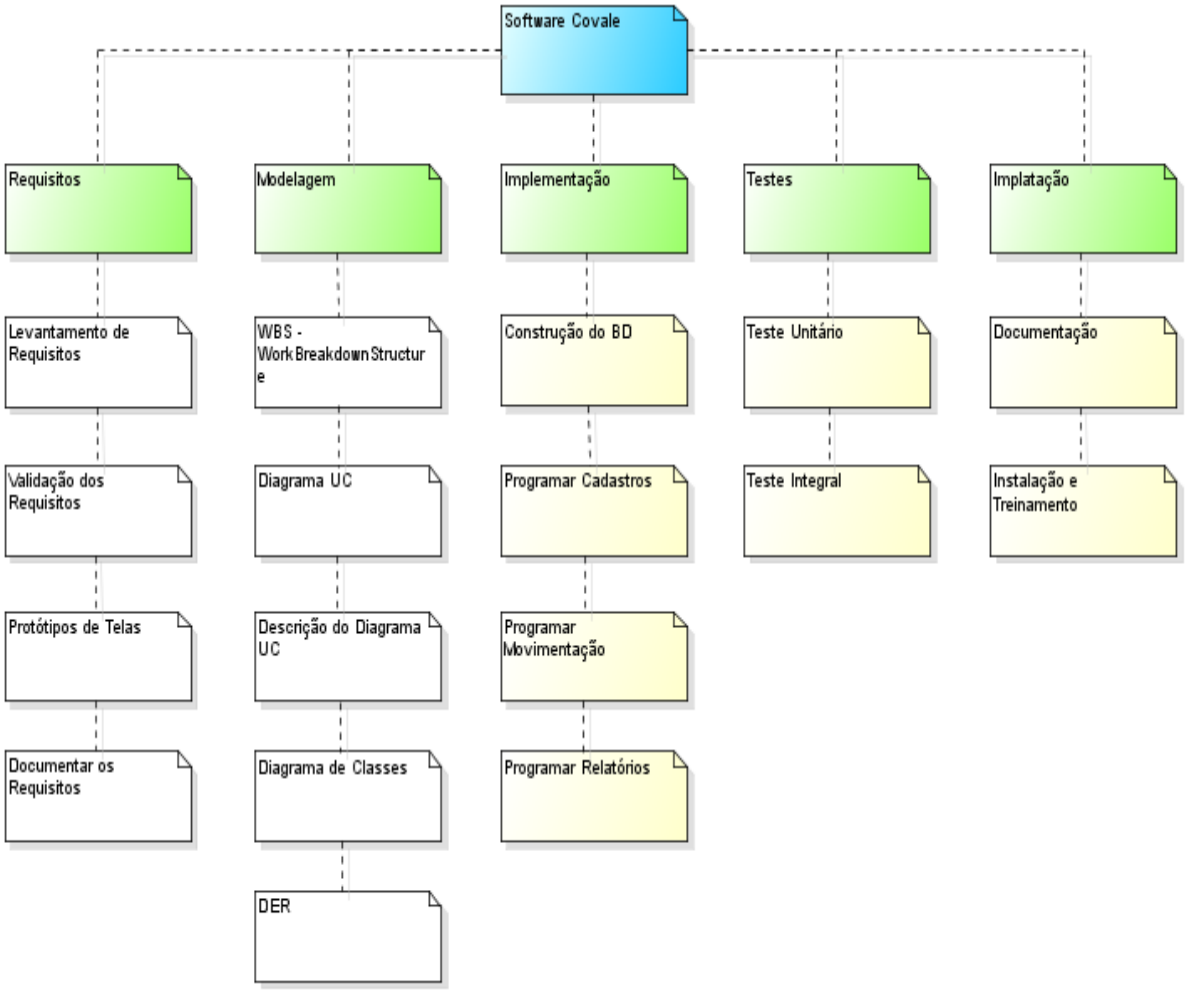

Figura 6. WBS - WorkBreakdownStructure

4.2. Definições das Atividades no Desenvolvimento do Software

O sistema irá fazer cadastro de clientes, que poderá ser alterado, excluído ou visualizado, fará cadastro de fornecedores que também poderá ser alterado, excluído ou visualizado, irá conter um cadastro de produtos, podendo também ser

alterado, excluído ou visualizado, cadastrar a data de validade do produto e o preço e terá um controle de caixa que o qual ficará registrado as movimentações.

4.2.1. Sequenciamentos das Atividades Definidas

Para o desenvolvimento do sistema a análise utilizada será orientada a objetos.

Estrutura:

1. Levantamento dos requisitos
2. Diagrama de Casos de Uso (Use-Case)
3. Diagrama de Classes
4. Diagrama de Entidade Relacionamento (ER)
5. Implementação
6. Testes
7. Instalação
8. Treinamento

4.3. Caso de Uso (Use Case)

É um tipo de classificador representando uma unidade funcional coerente provida pelo sistema, subsistema, ou classe manifestada por sequências de mensagens intercambiáveis entre os sistemas e um ou mais atores.
Casos de uso são narrativas em texto, descrevendo a unidade funcional, e são amplamente utilizados para descobrir e registrar requisitos de sistemas. Os diagramas de Casos de Uso são representações dos Casos de Uso e podem ser representados por uma elipse contendo, internamente, o nome do caso de uso.

4.3.1. Diagrama Caso de Uso – Covale

A figura 7 ilustra o diagrama de Caso de Uso do programa Covale.

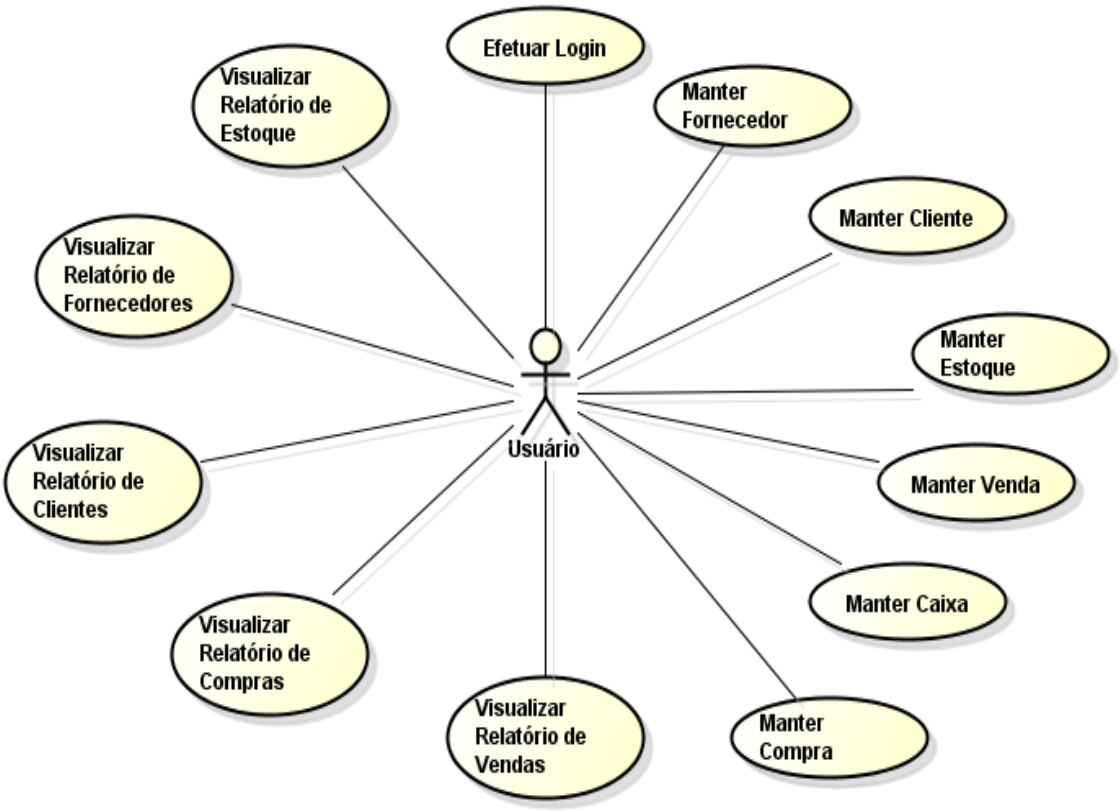

Figura 7. Diagrama de Caso de Uso Geral

4.4. Especificação de Caso de Uso

4.4.1. Efetuar Login

A figura 8 ilustra o diagrama de Caso de Uso Efetuar Login.

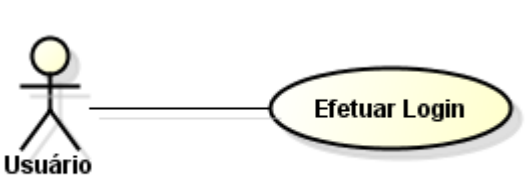

Figura 8 – Diagrama de Caso de Uso – Efetuar Login

Nome do Caso de Uso:	Efetuar Login.

Finalidades:	Permitir que o usuário acessasse o sistema e todas as informações que contém.
Atores:	Usuário.
Evento Inicial:	O usuário entra com o número de login e senha.
Fluxo Principal:	O usuário informa seu número de login e senha; O sistema verifica os dados informados, se estiver correto, o sistema fica acessível para o usuário.
Fluxo Alternativo:	O usuário cancela a operação.
Fluxo de exceção:	Se o usuário digitar número de login e senha incorretamente, o sistema mostrará uma mensagem de erro e o administrador terá de digitar novamente.
Pós-Condições:	Após o usuário ter efetuado o login, terá todo acesso aos dados e informações do sistema.

TABELA 1 – Especificação do Caso de Uso Efetuar Login

4.4.2. Manter Cliente

A figura 9 ilustra o diagrama de Caso de Uso Manter Cliente.

Figura 9 – Diagrama de Caso de Uso – Manter Cliente

Nome do Caso de Uso:	Manter Cliente.
Pré-Condições:	O usuário deverá estar logado no sistema.
Finalidades:	Permitir que o administrador cadastre um cliente no sistema.

Atores:	Usuário.
Evento Inicial:	O usuário seleciona o botão novo;
Fluxo Principal:	O usuário digita o nome do cliente.
Fluxo Alternativo:	O usuário poderá alterar ou excluir o cadastro se necessário.
Fluxo de Exceção:	Se caso o usuário não informar o nome do cliente, o cadastro não será concluído.
Pós-Condição:	O usuário poderá ter acesso aos dados do cliente e fazer qualquer alteração se necessário.

Tabela 2 – Especificação do Caso de Uso Manter Cliente

4.4.3. – Manter Estoque

A figura 10 ilustra o diagrama de Caso de Uso Manter Estoque.

Figura 10 – Diagrama de Caso de Uso – Manter Estoque

Nome do Caso de Uso:	Manter Estoque.
Pré-Condições:	O usuário deverá estar logado no sistema.
Finalidades:	Permitir que o usuário cadastre um produto do estoque no sistema.
Atores:	Usuário.
Evento Inicial:	O usuário seleciona o botão novo.
Fluxo Principal:	O usuário preenche o nome, o custo, o preço e a quantidade de produto no estoque.
Fluxo Alternativo:	O usuário poderá o cadastro se necessário.
Fluxo de Exceção:	Caso o usuário não informar uns dos campos obrigatórios, o cadastro não será concluído.

Pós-Condição:	O usuário poderá ter acesso e fazer qualquer alteração se for necessário.

TABELA 3 – Especificação do Caso de Uso Manter Estoque

4.4.4. Manter Fluxo Caixa

A figura 11 ilustra o diagrama de Caso de Uso Manter Caixa.

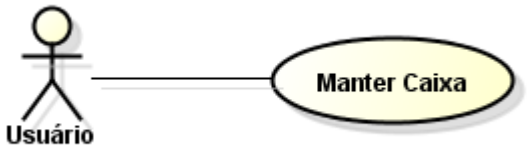

Figura 11 – Diagrama de Caso de Uso – Manter Caixa

Nome do Caso de Uso:	Manter Caixa.
Pré-Condições:	O usuário deverá estar logado no sistema.
Finalidades:	Permitir que o usuário efetue uma venda.
Atores:	Usuário.
Evento Inicial:	O usuário seleciona o cliente.
Fluxo Principal:	O usuário inclui os itens da venda. O usuário finaliza a venda. O usuário escolhe a forma de pagamento e insere os dados das condições de pagamento escolhido.
Fluxo Alternativo:	O usuário poderá alterar a quantidade ou excluir os itens da venda.
Fluxo de Exceção:	Caso o usuário não informar uma data válida para a venda, a venda não será realizada. Caso o usuário não informe o preço e/ou a quantidade este item de venda não será realizado.
Pós-Condição:	A venda é gerada.

TABELA 4 – Especificação do Caso de Uso Manter Fluxo de Caixa

4.4.5. Manter Fornecedor

A figura 12 ilustra o diagrama de Caso de Uso Manter Fornecedor.

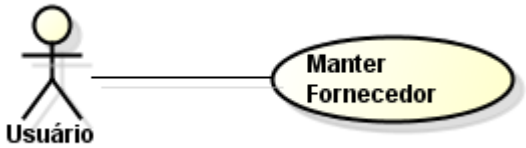

Figura 12 – Diagrama de Caso de Uso – Manter Fornecedor

Nome do Caso de Uso:	Manter Fornecedor.
Pré-Condições:	O usuário deverá estar logado no sistema.
Finalidades:	Permitir que o usuário cadastre um fornecedor no sistema.
Atores:	Usuário.
Evento Inicial:	O usuário seleciona o botão novo.
Fluxo Principal:	O usuário preenche o nome do novo fornecedor.
Fluxo Alternativo:	O usuário poderá alterar ou excluir o cadastro se for necessário.
Fluxo de Exceção:	Se o usuário não informar o nome do fornecedor, o cadastro não será concluído.
Pós-Condição:	O usuário poderá ter acesso aos dados do fornecedor e fazer qualquer alteração se necessário.

TABELA 5 – Especificação do Caso de Uso Manter Fornecedor

4.4.6. Visualizar Relatório de Estoque

A figura 13 ilustra o diagrama de Caso de Uso Visualizar Relatório de Estoque.

Figura 13 – Diagrama de Caso de Uso – Visualizar Relatório de Estoque

Nome do Caso de Uso:	Visualizar Relatório de Estoque.
Pré-Condições:	O usuário deverá estar logado no sistema.
Finalidades:	Permitir que o usuário visualize o estoque.
Atores:	Usuário.
Evento Inicial:	O usuário seleciona a opção estoque no menu.
Fluxo Principal:	O usuário seleciona a opção estoque no menu.
Fluxo Alternativo:	O usuário cancela o relatório de estoque.
Fluxo de Exceção:	Não há fluxo de exceção.
Pós-Condição:	Não há alterações nos registros.

TABELA 6 – Especificação do Caso de Uso Visualizar Relatório de Estoque

4.4.7. Visualizar Relatório de Vendas

A figura 14 ilustra o diagrama de Caso de Uso Visualizar Relatório de Vendas.

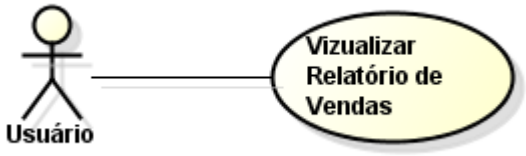

Figura 14 – Diagrama de Caso de Uso – Visualizar Relatório de Vendas

Nome do Caso de Uso:	Visualizar Relatório Vendas.
Pré-Condições:	O usuário deverá estar logado no sistema.
Finalidades:	Permitir que o usuário visualize as vendas por tempo.

Atores:	Usuário.
Evento Inicial:	O usuário informa a data desejada.
Fluxo Principal:	O usuário informa a data desejada. O usuário visualiza o relatório de vendas daquele período.
Fluxo Alternativo:	O usuário cancela a opção relatório de vendas.
Fluxo de Exceção:	Caso o usuário não informar uma data válida, não é visualizado o relatório.
Pós-Condição:	Não há alterações nos registros.

TABELA 7 – Especificação do Caso de Uso Visualizar Relatório de Vendas

4.4.8. Visualizar Relatório de Compras

A figura 15 ilustra o diagrama de Caso de Uso Visualizar Relatório de Compras.

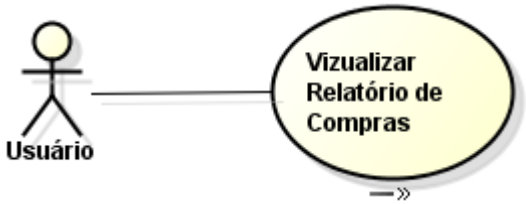

Figura 15 – Diagrama de Caso de Uso – Visualizar Relatório Compra

Nome do Caso de Uso:	Visualizar Relatório de Compras.
Pré-Condições:	O usuário deverá estar logado no sistema.
Finalidades:	Permitir que o usuário visualize o relatório de compras.
Atores:	Usuário.

Evento Inicial:	O usuário seleciona a data desejada.
Fluxo Principal:	O usuário seleciona a data desejada. O usuário visualiza o relatório de compras.
Fluxo Alternativo:	O usuário cancela o relatório de compras.
Fluxo de Exceção:	Caso o usuário não informar uma data válida, o relatório não é visualizado.
Pós-Condição:	Não há alterações nos registros.

TABELA 8 – Especificação do Caso de Uso Visualizar Relatório de Compras

4.4.9. Visualizar Relatório de Clientes

A figura 16 ilustra o diagrama de Caso de Uso Visualizar Relatório de Clientes.

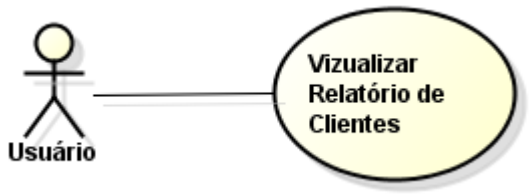

Figura 16 – Diagrama de Caso de Uso – Visualizar Relatório de Clientes

Nome do Caso de Uso:	Visualizar Relatório de Clientes.
Pré-Condições:	O usuário deverá estar logado no sistema.
Finalidades:	Permitir que o usuário visualize o relatório de clientes.
Atores:	Usuário.
Evento Inicial:	O usuário seleciona a letra dos nomes ou a cidades

	dos clientes.
Fluxo Principal:	O usuário seleciona a letra dos nomes ou a cidades dos clientes. O relatório é exibido conforme a opção do usuário.
Fluxo Alternativo:	O usuário cancela o relatório de clientes.
Fluxo de Exceção:	Caso o usuário não informar uma cidade cadastrada, o relatório não é exibido.
Pós-Condição:	.Não há alterações nos registros

TABELA 9 – Especificação do Caso de Uso Visualizar Relatório de Clientes

4.4.10. Visualizar Relatório de Fornecedores

A figura 14 ilustra o diagrama de Caso de Uso Visualizar Relatório de Fornecedores.

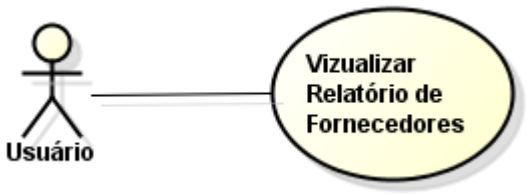

Figura 17 – Diagrama de Caso de Uso – Visualizar Relatório de Fornecedores

Nome do Caso de Uso:	Visualizar Relatório de Fornecedores.
Pré-Condições:	O usuário deverá estar logado no sistema.
Finalidades:	Permitir que o usuário visualize o relatório de fornecedores.
Atores:	Usuário.
Evento Inicial:	O usuário seleciona a letra dos nomes ou a cidades dos fornecedores.

Fluxo Principal:	O usuário seleciona a letra dos nomes ou a cidades dos fornecedores. O relatório é exibido conforme a opção do usuário.
Fluxo Alternativo:	O usuário cancela o relatório de fornecedores.
Fluxo de Exceção:	Caso o usuário não informar uma cidade cadastrada, o relatório não é exibido.
Pós-Condição:	.Não há alterações nos registros

TABELA 10 – Especificação do Caso de Uso Visualizar Relatório de Fornecedores

4.5. Diagrama de Classes

A figura 18 ilustra o Diagrama de Classes do programa Covale.

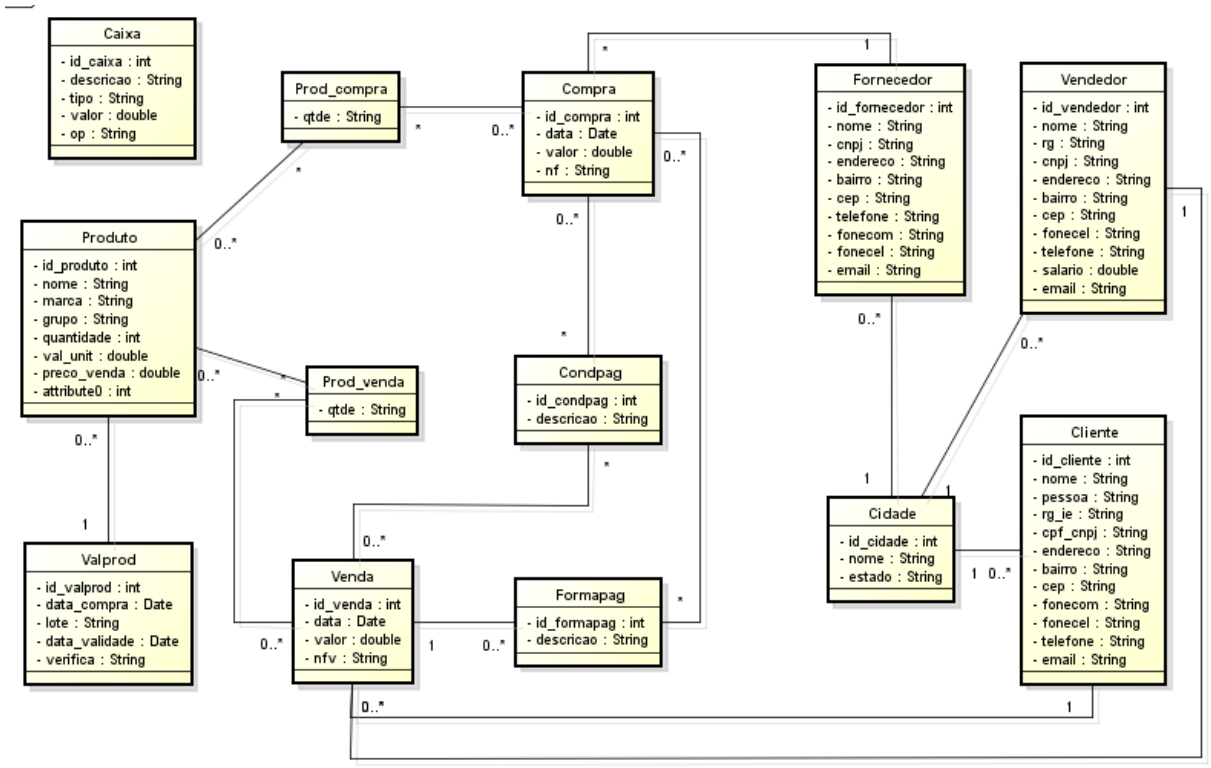

Figura 18 – Diagrama de Classe

4.6. Diagrama ER

A figura 19 ilustra o diagrama de Entidade Relacionamento do programa Covale.

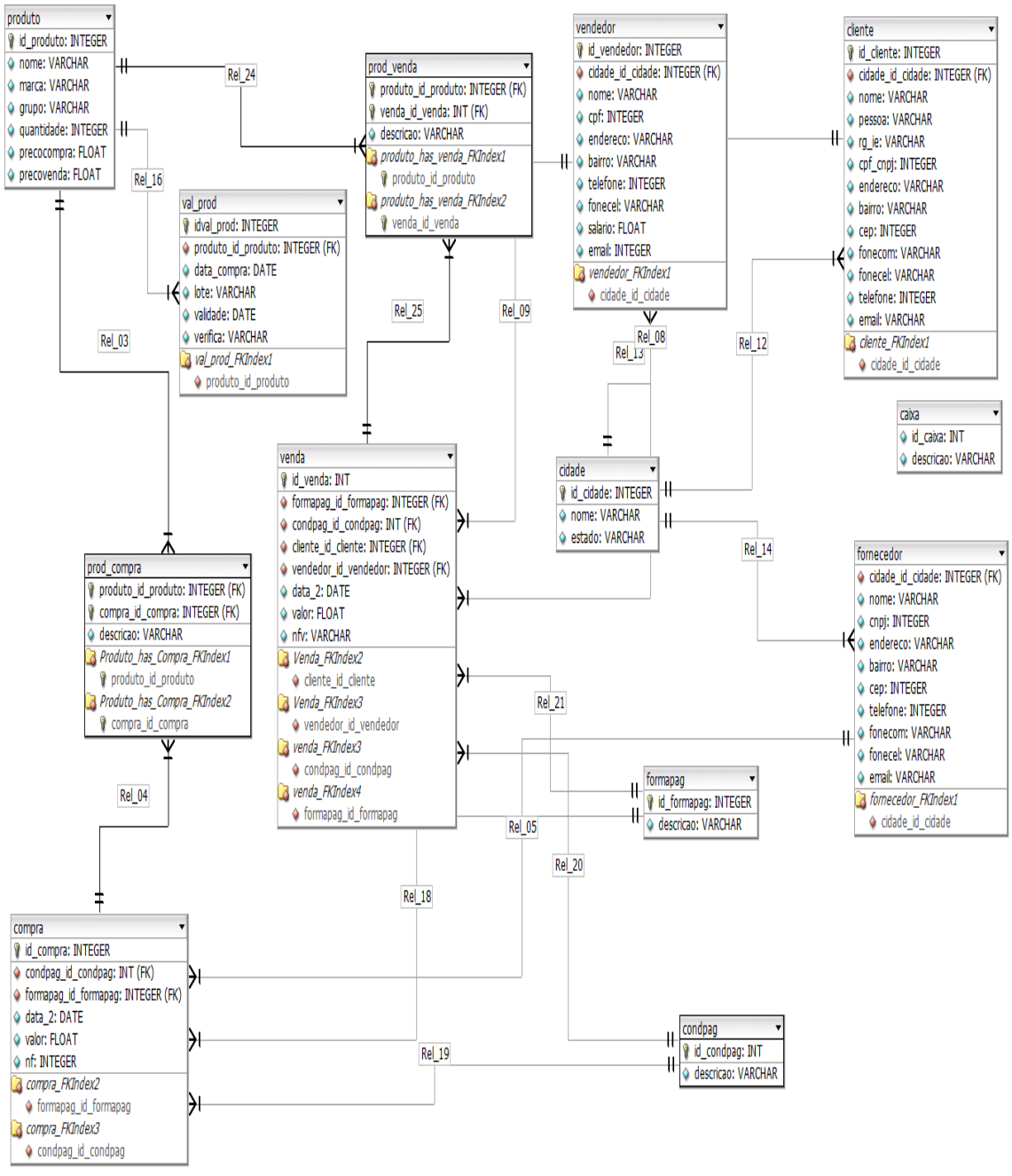

Figura 19 – Diagrama ER

5. IMPLEMENTAÇÃO

Implementação é a fase do ciclo de vida de um software (programa computacional, documentação e dados), no contexto de um sistema de informação, que corresponde à elaboração e preparação dos módulos necessários à sua execução (Laudon, K.; & Laudon, J., Management Information Systems: Managing the Digital Firm, 2010).

Para a implementação do programa foi utilizado o ambiente de desenvolvimento Netbeans com a linguagem de programação Java. O programa está divido em pacotes: br.com.covale.modelo, br.com.covale.dao, br.com.covale.managedBean, br.com.covale.util, e por páginas Web.

5.1. Interfaces

Serão ilustradas a seguir as telas do programa no que se refere a confirmar compra.

A figura 20 ilustra a tela inicial de compras onde o programa apresenta as compras cadastradas.

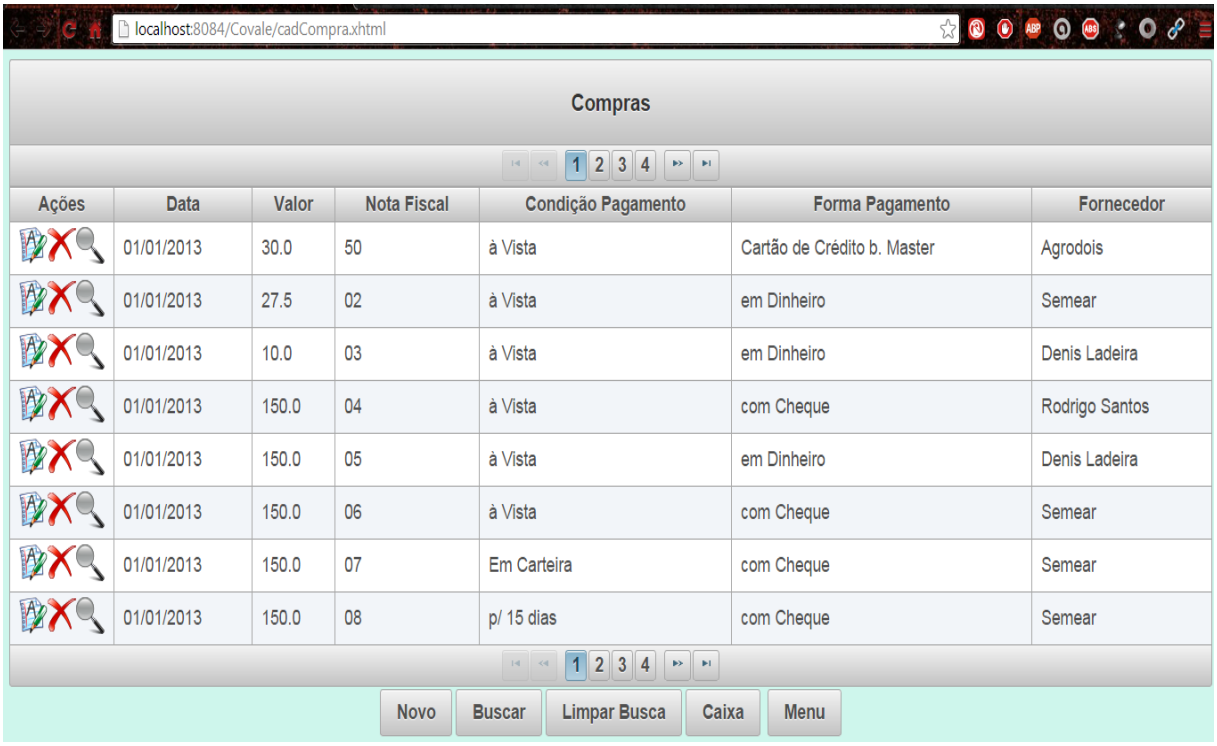

Figura 20 – Tela Inicial de Compras

A seguir, a figura 21 ilustra a tela de Cadastrar Compra quando chamado o botão Novo.

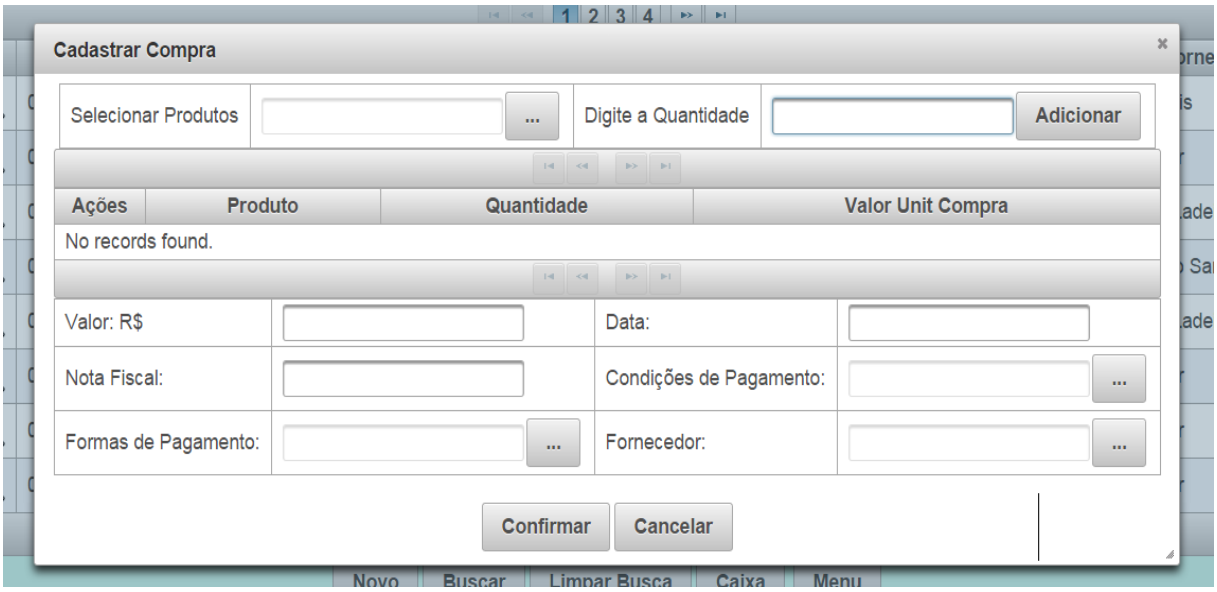

Figura 21 – Tela Cadastrar Compras

A figura 22 ilustra a tela Cadastrar Compra, o campo Selecionar Produtos com um produto selecionado e inserido a quantidade do produto.

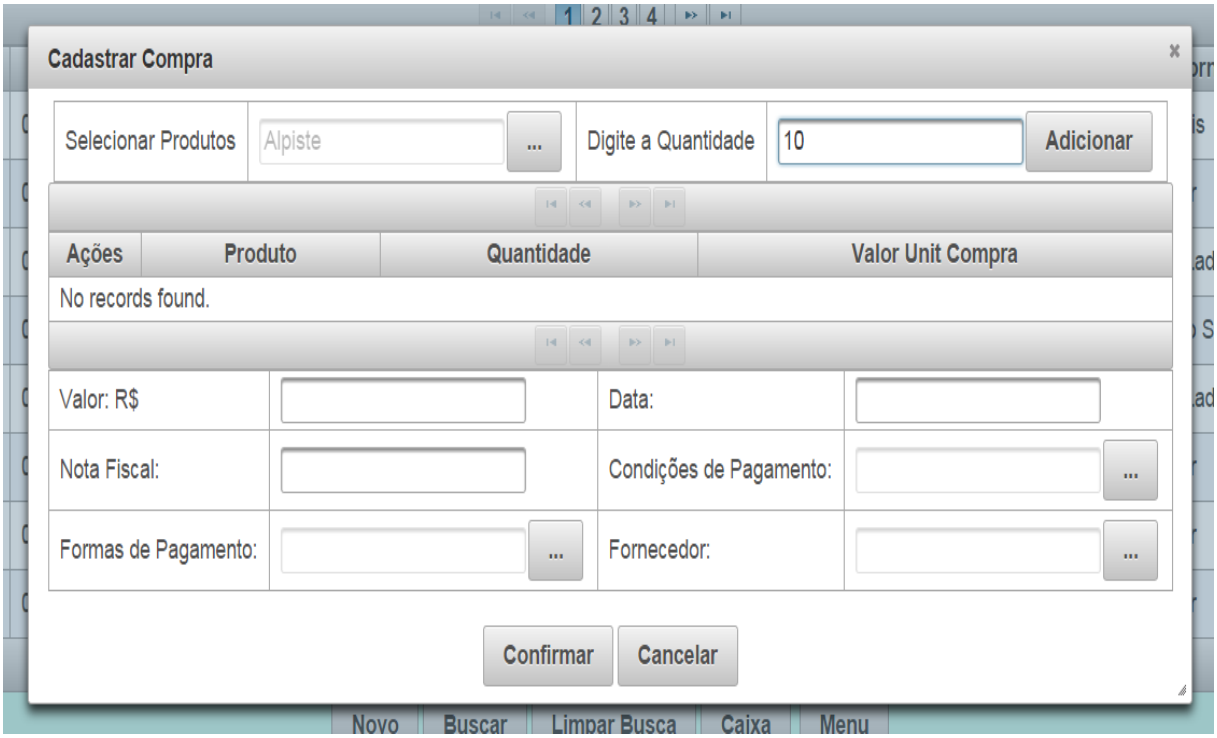

Figura 22 – Tela Cadastrar Compras parte 2

A figura 23 ilustra a tela Cadastrar Compra com o produto selecionado preenchido com a quantidade da compra, o valor unitário de compra e o valor total da compra.

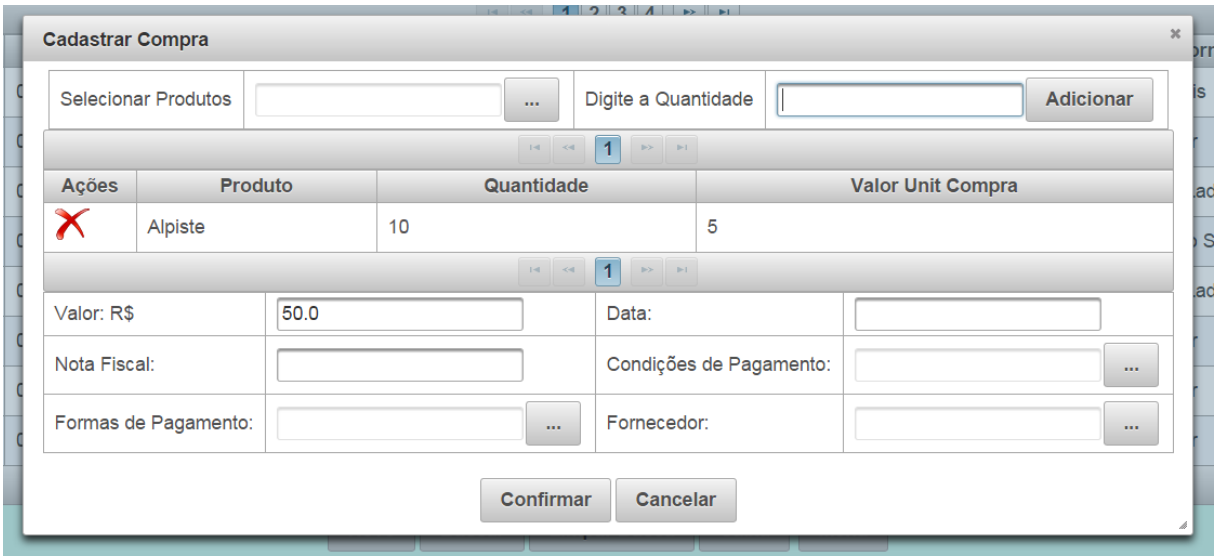

Figura 23 – Tela Cadastrar Compras parte 3

5.2. Código Fonte

Neste subcapítulo serão apresentados alguns trechos do código fonte do software.

A seguir, a figura 24 ilustra a primeira parte do código xhtml da página web com componentes do primefaces cadCompra onde é realizada a compra.

```
<div align="center">
        <h:form id="cadComCadastrar">
            <p:dialog header="Cadastrar Compra" widgetVar="dialogCadastrar" modal="true"
                      id="Cadastrar">
                <p:panelGrid id="panelAlterar" columns="4">

                    <h:outputText value="Selecionar Produtos" />
                    <p:column >
                        <p:inputText value="#{compraMB.produto.nome}" disabled="true" />
                        <p:commandButton value="..." update=":cadComCadastrar:Cadastrar" oncomplete="dialogCadastrar.hide(); dialogProduto.show()"/>
                    </p:column>
                    <h:outputText value="Digite a Quantidade"/>
                    <p:column>
                        <p:inputText value="#{compraMB.produto.quantidade}" />
                        <p:commandButton value="Adicionar" actionListener="#{compraMB.addProduto}"
                        update=":cadComCadastrar:Cadastrar" oncomplete="dialogCadastrar.show()"/>
                    </p:column>

                </p:panelGrid>
                <p:dataTable var="lista" value="#{compraMB.compra.produtos}" id="tabelaProduto" rows="2" paginator="true">

                    <p:column headerText="Ações" width="65">
                        <p:commandLink title="excluir" update=":cadComCadastrar:Cadastrar"
                                       oncomplete="dialogCadastrar.show()"
                                       actionListener="#{compraMB.removeProduto(lista)}"
                                       immediate="true">
                           <p:graphicImage value="./imagens/inativo.png" />

                        </p:commandLink>
                    </p:column>
                    <p:column headerText="Produto" >
                        <h:outputText value="#{lista.nome}" />
```

Figura 24 – Código XHTML parte 1

A figura 25 traz a segunda parte do código fonte XHTML de cadastro de compra.

```
                    </p:column>
                    <p:column headerText="Quantidade" >
                        <h:outputText value="#{lista.quantidade}" />
                    </p:column>
                    <p:column headerText="Valor Unit Compra" >
                        <h:outputText value="#{lista.preco_compra}" />
                    </p:column>

                </p:dataTable>

                <p:panelGrid id="panelAlterar1" columns="4">

                    <h:outputText value="Valor: R$ "/>
                    <p:inputText value="#{compraMB.compra.valor}" maxlength="10" size="20"/>

                    <h:outputText value="Data: "/>
                    <p:inputMask mask="99/99/9999" value="#{compraMB.compra.data}" maxlength="12" size="20"/>
                    <h:outputText value="Nota Fiscal: " />
                    <p:inputText value="#{compraMB.compra.nf}" maxlength="18" size="20" />
                    <h:outputText value="Condições de Pagamento: "/>
                    <p:column >
                        <p:inputText value="#{compraMB.compra.condpag.descricao}" disabled="true"/>
                        <p:commandButton value="..."
                                         oncomplete="dialogCadastrar.hide();dialogIncluirCon.show()"
                                         update=""/>
                    </p:column>
                    <h:outputText value="Formas de Pagamento: "/>
                    <p:column >
                        <p:inputText value="#{compraMB.compra.formapag.descricao}" disabled="true"/>
                        <p:commandButton value="..."
                                         oncomplete="dialogCadastrar.hide();dialogIncluirFop.show()"
                                         update=""/>
```

Figura 25 – Código XHTML parte 2

A figura 26 ilustra a terceira parte do código fonte XHTML de cadastro de compra.

```
                </p:column>
                <h:outputText value="Fornecedor: "/>
                <p:column >
                    <p:inputText value="#{compraMB.compra.fornecedor.nome}" disabled="true"/>
                    <p:commandButton value="..."
                                     oncomplete="dialogCadastrar.hide();dialogIncluirFor.show()"
                                     update=""/>

                </p:column>


            </p:panelGrid>

            <br/>

            <p:commandButton value="Confirmar"
                             actionListener="#{compraMB.cadastrar}"
                             update=":cadComPrincipal:tabelaCompra"
                             oncomplete="dialogCadastrar.hide()" />
            <p:commandButton value="Cancelar"
                             oncomplete="dialogCadastrar.hide()"
                             actionListener="#{compraMB.cancelar}"/>

        </p:dialog>

    </h:form>
</div>
```

Figura 26 – Código XHTML parte 3

A seguir, a figura 27 ilustra as declarações de variáveis no do CompraMB do managedBean.

```
@ManagedBean
@ViewScoped
public class CompraMB {

    private Compra compra = new Compra();
    private List compras = new ArrayList();
    private Produto produto = new Produto();
    private List<Produto> produtos = new ArrayList<Produto>();
    private List<Produto> caux = new ArrayList<Produto>();
    private Venda venda;

    public CompraMB() {

        CompraDao cpDao = new CompraDao();
        compras = cpDao.selecionarTodos();

        ProdutoDao pdDao = new ProdutoDao();
        produtos = pdDao.selecionarTodos();

    }
```

Figura 27 – Declaração de variáveis no pacote managedBean

Na figura 28 é ilustrado o método sem retorno addProduto no pacote managedBean do CompraMB, onde o usuário estando efetivando uma compra, esse é o método que adiciona produtos à compra e já faz o cálculo do valor da compra. O código do método encontra-se comentado para um melhor entendimento do leitor.

```
207     // Método add produto no botão add quantidade produto (compra) já mostrando o valor a ser pago da compra
208     public void addProduto(ActionEvent actionEvent) {
209         // VARIÁVEL COMPRA ADD PRODUTOS À LISTA
210         compra.getProdutos().add(produto);
211 
212         try {
213             /**
214              * valor RECEBE O PREÇO UNITÁRIO DE COMPRA MULTIPLICADO PELA QTDE
215              * O getValor ELE SOMA O VALOR DA COMPRA COM O VALOR DE NOVA INSERÇÃO DE NOVOS PRODUTOS
216              */
217             Double valor = Double.parseDouble(compra.getValor())
218                     + (Double.parseDouble(produto.getPreco_compra())
219                     * Double.parseDouble(produto.getQuantidade()));
220             compra.setValor(valor.toString()); // CONVERTE O VALOR DOUBLE PARA STRING
221         } catch (NumberFormatException e) {
222             Double valor = 0 + (Double.parseDouble(produto.getPreco_compra())
223                     * Double.parseDouble(produto.getQuantidade()));
224             compra.setValor(valor.toString());
225         }
226         produto = new Produto();
227     }
```

Figura 28 – Método addProduto

A figura 29 ilustra o método cadastrar do CompraMB, o qual é o método onde o usuário confirma a compra. O código encontra-se comentado para um melhor entendimento do leitor.

```
82      public void cadastrar(ActionEvent actionEvent) {
83          // CONDIÇÃO PARA QUE NÃO OCORRA CADASTRO SEM INFORMAR A DATA
84          if (!compra.getData().equals("")) {
85 
86              CompraDao cpDao = new CompraDao();
87 
88              if (cpDao.inserir(compra)) { // (compra) É O OBJETO Q MONTA NO LAYOUT(TELA)
89              //SE O RETORNO DO IF FOR VERDADEIRO
90                  compras = cpDao.selecionarTodos(); // VARIÁVEL compras RECEBE UMA LISTA DE COMPRAS
91                  ProdutoDao pdDao = new ProdutoDao(); // INSTANCIAMENTO DO OBJETO pdDao DA CLASSE ProdutoDao
92                  produtos = pdDao.selecionarTodos(); // VARIÁVEL produtos RECEBE UMA LISTA DE PRODUTOS
93                  compra = new Compra(); // LIMPA A COMPRA
94 
95                  FacesContext context = FacesContext.getCurrentInstance();
96                  context.addMessage(null, new FacesMessage(FacesMessage.SEVERITY_INFO,
97                          "Inserido com sucesso", ""));
98              } else {
99                  FacesContext context = FacesContext.getCurrentInstance();
100                 context.addMessage(null, new FacesMessage(FacesMessage.SEVERITY_ERROR,
101                         "Erro ao inserir", "Tente novamente"));
102             }
103         } else {
104             FacesContext context = FacesContext.getCurrentInstance();
105             context.addMessage(null, new FacesMessage(FacesMessage.SEVERITY_ERROR,
106                     "Insira a Data (obrigatório) ", "Tente novamente"));
107         }
108     }
```

Figura 29 – Método cadastrar

A figura 30 ilustra o método inserir do CompraDao.

```
28      public boolean inserir(Compra cp) {
29          String sql = "INSERT INTO compra(data,valor,nf, id_condpag, id_formapag, "
30                  + "id_fornecedor) VALUES (?,?,?,?,?,?) returning id_compra ";
31          boolean retorno = false;
32          try {
33              stmt = conn.getPreparedStatement(sql);
34              stmt.setString(1, cp.getData());
35              stmt.setString(2, cp.getValor());
36              stmt.setString(3, cp.getNf());
37              stmt.setInt(4, cp.getCondpag().getId_condpag());
38              stmt.setInt(5, cp.getFormapag().getId_formapag());
39              stmt.setInt(6, cp.getFornecedor().getId_fornecedor());
40
41              ResultSet rs = stmt.executeQuery();
42              if (rs.next()) {
43                  for (int i = 0; i < cp.getProdutos().size(); i++) {
44                      String sql2 = "INSERT INTO prod_compra(id_compra, id_produto) VALUES(?,?);";
45                      stmt = conn.getPreparedStatement(sql2);
46                      stmt.setInt(1, rs.getInt("id_compra"));
47                      stmt.setInt(2, cp.getProdutos().get(i).getId_produto());
48                      stmt.execute();
49                      // Chama método para atualizar o estoque
50                      adcQtde(cp.getProdutos().get(i));
51                  }
52              }
```

Figura 30 – Método inserir

A figura 31 ilustra o método adcQtde, o qual é o método que atualiza a quantidade de produtos no estoque.

```
273     public boolean adcQtde(Produto p) {
274         boolean returno = false;
275
277         Produto prod = new Produto();
278         ProdutoDao pdDao = new ProdutoDao();
279         // recupera o produto do estoque (recupera o produto já cadastrado)
280         prod = pdDao.selecionarUm(p.getId_produto());
281         Double d1 = Double.parseDouble(prod.getQuantidade()); //prod é do estoque
282         Double d2 = Double.parseDouble(p.getQuantidade()); // p é da compra
283         d1 = d1 + d2;
284         p.setQuantidade(d1.toString());
285         pdDao.update(p);
286         return returno;
287     }
}
```

Figura 31 – Método adcQtde

6. CONCLUSÃO

Esse projeto objetivou o desenvolvimento de um software para a empresa Covale. Foi uma grande experiência trabalhar todo o conhecimento adquirido na faculdade e também incentivou a buscar novas soluções para aumentar o conhecimento posto em prática neste projeto.

Esse sistema traz comodidade ao cliente, pois armazenar as informações num banco de dados, o cliente terá acesso e confiabilidade as informações sobre produtos vendidos, o que os clientes mais consomem e as entradas e saídas da empresa, auxiliando o cliente na tomadas de decisões futuras da empresa.

Como trabalho futuro deseja-se promover algumas modificações no sistema para que ele possa ser reutilizado em empresas do mesmo setor.

REFERÊNCIAS

SOMMERVILLE, Ian; **Engenharia de Software**. 6ª Edição; tradução André Maurício de Andrade Ribeiro; revisão técnica Kechi Hirama. – São Paulo: Addison Wesley, 2003. 580p.

DEITEL, Harvey M.; DEITEL, Paul J. **Java Como Programar**; 8ª Edição; tradução Edson Furmankiewicz; revisão técnica Fabio Luis Picelli Lucchini. – São Paulo: Pearson Prentice Hall, 2010. 1173p.

LEE, Richard C.; TEPFENHART, WILLIAM, M. **Guia Prático de Desenvolvimento Orientado a Objeto**; tradução Celso Roberto Paschoa; revisão técnica José Davi Furlan – São Paulo: Makron Books LTDA, 2001. 550p.

LUCKOW, Décio Heinzelmann; Melo Alexandre Altair de. **Programação Java para a Web**; Revisão gramatical Lia Gabriele Regius - São Paulo: Novatec Editora, 2010. 640p.

MACLAUGHLIN, Brett; POLLICE, Gary & WEST, Davi; **Análise e Projeto Orientado ao Objeto;** tradução Betina Macêdo; revisão técnica Eduardo Velasco – Rio de Janeiro: Alta Books, 2007. 442p.

GUEDES, Gilleanes T. A., **UML 2 uma abordagem prática**; Revisão de texto: Lia Gabriele Regius - São Paulo: Novatec Editora Ltda, 2009. 488p.

REESE G., **Database Programming with JDBC & Java**, Production Editors: Ann Schimer – Printed in the United States of America: Andy Oram, 2000. 350p.

JEPSON, Brian. **Programando banco de dados em Java**. São Paulo : Makron, 1997. 487 p

LAUDON, K., & Laudon, J. **Management Information Systems: Managing the Digital Firm**. New Jersey: Eleventh Edition, 2010. 648p.

BURNETT, Robert Carlisle. **Engenharia de requisitos: conceitos e fundamentos, [S.l.],** [1998].Disponível em: <http://www.bc.furb.br/docs/mo/2004/279170_1_1.pdf>. Acesso em: 12 set. 2014.

FALBO, Ricardo de Almeida. **Engenharia de Software: Notas de Aula**, [2005]. Disponível em: <http://www.inf.ufes.br/~falbo/download/aulas/es-g/2005-1/NotasDeAula.pdf>. Acesso em: 12 set. 2014.

LIMA, Adelcio Marcos de. **Computador e periféricos**, Disponível em: <http://www.adelcio.com.br/arquivos/2012_02_IB/2%20-%20Hardware.pdf>. Acesso em: 12 set. 2014.

WTHREEX, **Conceitos: Gerenciamento de Requisitos**, Disponível em: <http://www.wthreex.com/rup/portugues/process/workflow/requirem/co_rm.htm>. Acesso em: 12 set. 2014.

MACORATTI.NET, **Conceitos : Especificação de requisitos**, Disponível em: <http://www.macoratti.net/07/12/net_fer.htm>. Acesso em: 12 set. 2014.

OFICINADANET, **O que é o NetBeans?**, Disponível em: <http://www.oficinadanet.com.br/artigo/1061/o_que_e_o_netbeans>. Acesso em: 12 set. 2014.

APPNOVATION, **Using Axure for Building Wireframes**, Disponível em: <http://www.appnovation.com/blog/using-axure-building-wireframes>. Acesso em: 12 set. 2014.

CONSULTORIODESOFTWARE, **TUTORIAL - Guia Prático de utilização da ferramenta Astah Community**, Disponível em: <http://www-pet-si.inf.ufsm.br/images/consultoriodesoftware/Astah.pdf>. Acesso em: 12 set. 2014.

DEVMEDIA, **DBDesigner: Modelagem e Implementação de banco de dados**, Disponível em: <http://www.devmedia.com.br/dbdesigner-modelagem-e-implementacao-de-banco-de-dados/30897>. Acesso em: 12 set. 2014.

PGADMIN, **Introduction**, Disponível em: <http://www.pgadmin.org/>. Acesso em: 12 set. 2014.

WILLIAMGAMERS, **Visão Geral sobre PrimeFaces**, Disponível em: <http://williamgamers.wordpress.com/2012/06/04/visao-geral-sobre-primefaces/>. Acesso em: 12 set. 2014.

TOMCAT, **Tomcat**, Disponível em: <http://tomcat.apache.org/>. Acesso em: 12 set. 2014.

NETBEANS.ORG, **Trilha do Aprendizado do Java EE e Java Web**, Disponível em: <https://netbeans.org/kb/trails/java-ee_pt_BR.html>. Acesso em: 12 set. 2014.

SURIAN, Jorge, NICOCHELLI, Luiz, **Apostila de Banco de Dados e SQL**, Disponível em:<http://www.josevalter.com.br/download/sql/Apostila%20de%20Banco%20de%20Dados%20e%20SQL.pdf>. Acesso em: 12 set. 2014.

POSTGRESQL, **History**, Disponível em: <http://www.postgresql.org/about/history/>. Acesso em: 12 set. 2014.