UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO DE ESPECIALIZAÇÃO EM TECNOLOGIA JAVA

MARCOS DANIEL BUDTINGER

SISTEMA DE GERENCIAMENTO DE PEDIDOS PARA EMPRESAS QUE ATUAM COM REPRESENTAÇÕES E VENDAS DE PRODUTOS

MONOGRAFIA DE ESPECIALIZAÇÃO

PATO BRANCO
2017

MARCOS DANIEL BUDTINGER

SISTEMA DE GERENCIAMENTO DE PEDIDOS PARA EMPRESAS QUE ATUAM COM REPRESENTAÇÕES E VENDAS DE PRODUTOS

Monografia de especialização apresentada na disciplina de Metodologia da Pesquisa, do Curso de Especialização em Tecnologia Java, do Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, como requisito parcial para obtenção do título de Especialista.

Orientador: Prof. Vinicius Pegorini.

PATO BRANCO
2017

MINISTÉRIO DA EDUCAÇÃO
Universidade Tecnológica Federal do Paraná
Câmpus Pato Branco
Departamento Acadêmico de Informática
Curso de Especialização em Tecnologia Java

TERMO DE APROVAÇÃO

SISTEMA DE GERENCIAMENTO DE PEDIDOS PARA EMPRESAS QUE ATUAM COM REPRESENTAÇÕES E VENDAS DE PRODUTOS

por

MARCOS DANIEL BUDTINGER

Este trabalho de conclusão de curso foi apresentado em 26 de agosto de 2017, como requisito parcial para a obtenção do título de Especialista em Tecnologia Java. Após a apresentação o candidato foi arguido pela banca examinadora composta pelos professores Vinicius Pegorini (orientador), Beatriz Terezinha Borsoi e Robison Cris Brito, membros da banca. Em seguida foi realizada a deliberação pela banca examinadora que considerou o trabalho aprovado.

Vinicius Pegorini
Prof. Orientador (UTFPR)

Beatriz Terezinha Borsoi
Banca (UTFPR)

Robison Cris Brito
Banca (UTFPR)

Robison Cris Brito
Coordenador da IV Especialização
em Tecnologia Java

A Folha de Aprovação assinada encontra-se na Coordenação do Curso.

O oferecimento deste trabalho vai, em primeiro lugar para Deus que sempre esteve ao meu lado e me proporcionou a vida para que chegasse até este ponto de vitória.
Em segundo lugar para toda minha família e professores que tive durante essa jornada acadêmica, os quais nunca deixaram de acreditar em minha pessoa e sempre me motivaram a seguir em frente.

AGRADECIMENTOS

Os agradecimentos vão para todos que sempre acreditam em minha pessoa, em minha perseverança e no ideal de um mundo melhor.

“Se você é um carpinteiro e está fazendo um belo armário de gavetas, você não vai usar um pedaço de compensado na parte de trás porque as pessoas não o enxergarão, pois ele estará virado para a parede. Você sabe que está lá e, então, usará um pedaço de madeira bonito ali. Para você dormir bem à noite, a qualidade deve ser levada até o fim”.

Steve Jobs

RESUMO

BUDTINGER, Marcos Daniel Budtinger. Sistema de gerenciamento de pedidos para empresas que atuam com representações e vendas de produtos. 2017. 46 f. Monografia (Trabalho de especialização) – Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2017.

À medida que as tecnologias avançam muitas aplicações em diferentes áreas visam proporcionar comodidade aos usuários ou clientes. Isso inclui ações como fazer compras ou pagamentos sem sair de casa e ter um sistema para gerenciamento e controle de clientes, fornecedores e outros processos necessários no cotidiano de empresas e negócios. No presente trabalho foi desenvolvido um *software* que tem como objetivo o gerenciamento de pedidos para empresas que atuam com representações e vendas de produtos. O sistema foi desenvolvido utilizando diversas tecnologias, como VRaptor, Bootstrap, HTML5, CSS3, Apache Ivye o ambiente de desenvolvimento Eclipse. O resultado do trabalho é um sistema para gerenciamento e controle de clientes, fornecedores, pedidos e outros cadastros, além de emissão de relatóriose realização de cálculo de comissões sobre os pedidos.

Palavras-chave: VRaptor. Bootstrap. HTML5. CSS3. Apache Ivy.Gerenciamento de pedidos.

ABSTRACT

BUDTINGER, Marcos Daniel. Automation and order management. 2017. 46 f. Monografia (Trabalho de especialização) – Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2017.

Increasingly, people use online platforms, named software, as part of their day-to-day. Since the emergence of Web 2.0, together with the rapid advance of telecommunications and widespread use of Internet, these platforms have been a business strategy for companies. In light of this, this work presents an online web system for ordering management of any kind of products. The platform was developed by employing technologies such as VRaptor, Bootstrap, HTML5, CSS3, and Apache Ivy. Eclipse was adopted as an Integrated Development Environment (IDE). The main features of the software developed can be summarized as follows: order and customer management, control and calculation of commissions, and reports.

Keywords: VRaptor. Bootstrap. HTML5. CSS3. Apache Ivy. Order management.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTAGENS DE CÓDIGOS

LISTA DE SIGLAS

CRUD	*Create, Retrieve, Update and Delete*
CSS	*Cascading Style Sheet*
CX	Caixa
IDE	*Integrated Development Environment*
FTP	*File Transfer Protocol*
HTTP	*Hypertext Transfer Protocol*
JSON	*JavaScript Object Notation*
KG	Quilograma
MVC	*Model-View-Controller*
MVCC	*Multi Version Concurrency Control*
PDF	*Portable Document Format*
PIB	Produto Interno Bruto
RF	Requisitos Funcionais
RNF	Requisitos Não Funcionais
URL	*Uniform Resource Locator*
XHTML	*eXtensible Hypertext Markup Language*
XML	*eXtensible Markup Language*
WAL	*Write Ahead Logs*

SUMÁRIO

1 INTRODUÇÃO

Representantes e revendas devem manter atualizados os dados dos clientes, fornecedores e produtos que compõem a sua empresa. Isso porque é comum clientes e fornecedores mudarem de endereço e alterarem o preço de produtos. Essas informações devem ser atualizadas constantemente, o que é suscetível a erros quando isso é feito em fichas de papéis.

Normalmente há alguém responsável por atualizar essas fichas com as informações corretas. Essas informações geralmente estão desatualizadas porque dependendo da quantidade de clientes, fornecedores e produtos a procura das fichas de cadastro se torna difícil, pois o volume de papel é muito grande. Além da possibilidade de conter informações incompletas. Para melhorar o gerenciamento e a tomada de decisão é útil que as informações pudessem ser atualizadas por meio de um sistema no qual a busca pelo cliente, fornecedor ou produto seja ágil e fácil. Assim, informações atualizadas como de quais produtos foram mais vendidos e qual o percentual de comissão sobre as vendas, podem auxiliar na tomada de medidas que sejam necessárias para resolver problemas, no planejamento e em decisões gerenciais.

Saber qual foi o percentual ganho sobre as vendas em certo período e fazer comparações fica muito mais fácil quando é possível escolher e gerar relatórios de períodos específicos ao invés de os valores das comissões serem obtidos pela soma de venda por venda. Justifica-se nesse contexto, a utilização de tecnologias para o desenvolvimento de um *software* que supra essas necessidades e agilize o processo cotidiano dessas empresas.

Atualmente o setor de Tecnologia da Informação e Comunicação brasileiro representa aproximadamente 8,7% do Produto Interno Bruto (PIB), segundo a Brasscom (2016). Esse percentual tem aumentado substancialmente nos últimos anos devido ao uso intensivo de diversas tecnologias nos mais diversos tipos de empresas e de organizações. Apesar das dificuldades enfrentadas hoje pelo Brasil, o setor continua em expansão, as perspectivas para o futuro são promissoras graças ao aumento de consumo, de infraestrutura e de investimentos em pesquisas e desenvolvimento entre diversos outros fatores.

Com isso diversas empresas sentem a necessidade de atualizar seus procedimentos cotidianos automatizando-os para que dessa forma elas tenham informações mais precisas e que a tomada de decisão seja mais ágil. No ramo de vendas e pedidos deve-se, primeiramente definir qual é o negócio da empresa, o que ela se propõe a fazer para atender o mercado e qual é o seu objetivo que é o que conduz a empresa nos caminhos desejados pela administração.

Por isso nesse trabalho é apresentado o desenvolvido de um sistema que visa auxiliar no gerenciamento de clientes, fornecedores, produtos, pedidos e comissões afim de agilizar os processos internos da empresa e torná-los mais eficientes.

1.1 OBJETIVOS

O objetivo geral se refere ao resultado principal obtido com a realização deste trabalho. É o produto de software gerado. Os objetivos específicos estão relacionados às complementações tanto do sistema desenvolvido como em termos de resultados da realização deste trabalho.

1.1.1 Objetivo Geral

Desenvolver um sistema para gerenciar os pedidos e as comissões de uma empresa de representação e revenda de produtos.

1.1.2 Objetivos Específicos

- Manter atualizados os dados referentes à representação e à revenda;
- Propor um sistema que auxilie no controle e no gerenciamento das atividades cotidianas, disponibilizando os dados quando solicitados.
- Melhorar o processo de venda.
- Possibilitar o cálculo de comissões de forma automatizada com base nas vendas realizadas.
- Manter a segurança e a disponibilidade dos dados armazenados.

1.2 JUSTIFICATIVA

Mesmo com a presença de equipamentos de tecnologia dentro das empresas muitas ainda utilizam planilhas eletrônicas ou editores de texto para controlar os processos da empresa e os dados gerados e manipulados por esses processos. Na empresa que foi utilizada

como modelo para desenvolvimento do sistema objeto da realização deste trabalho grande parte das atividades eram controladas manualmente em planilhas o que demandava muito tempo e era passível de diversos erros, como ocorria frequentemente. Optou-se então pelo desenvolvimento de um sistema customizado às necessidades da empresa.

O sistema desenvolvido visa facilitar a venda e o gerenciamento de clientes, fornecedores, pedidos e as comissões sobre as vendas efetuadas. O usuário em questão tem uma representação de queijos e frios no Estado de São Paulo, na qual o mesmo divulga e vende os produtos de seus fornecedores para os seus clientes e por meio do total dessa venda ele é comissionado.

O sistema consiste em cadastros de clientes, fornecedores, unidades de medida, produtos, categorias e formas de pagamento, geração de pedidos, gráficos e relatórios. Por meio das informações reportadas é possível ter controle e gerenciamento mais rápido e transparente dos processos que a empresa executa em seu cotidiano, garantindo assim a confiabilidade nos dados gerados e manipulados pelos processos do negócio. O sistema disponibiliza relatórios de efetivação de clientes, de pedidos por cliente, de comissões por fornecedor e de período e entre outros que possam auxiliar na tomada de decisão.

1.3 ESTRUTURA DO TRABALHO

O presente trabalho é dividido em capítulos, sendo este o Capítulo 1 que contém as considerações iniciais, os objetivos e a justificativa do trabalho. O Capítulo 2 contém um descritivo das ferramentas, das tecnologias e dos procedimentos utilizados para a criação e o desenvolvimento do projeto. No Capítulo 3 é apresentado o resultado que é o sistema desenvolvido, bem como os diagramas de caso de uso, de classes, de banco de dados, os requisitos do sistema e apresentação das telas. Por fim, está o Capítulo 4 com as considerações finais que são seguidas das referências utilizadas na composição do texto.

2 FERRAMENTAS, TECNOLOGIAS E PROCEDIMENTOS

Neste Capítulo são apresentadas as ferramentas e as tecnologias que foram utilizadas para a modelagem e o desenvolvimento do sistema, bem como os procedimentos técnicos para o uso dessas ferramentas e tecnologias.

2.1 FERRAMENTAS E TECNOLOGIAS

O Quadro 1 apresenta as ferramentas e tecnologias utilizadas na modelagem e no desenvolvimento do sistema.

Ferramenta / Tecnologia	Versão	Disponível em	Aplicação
VRaptor	3	http://vraptor3.vraptor.org/pt/	Framework Model-View-Controller (MVC) para desenvolvimento.
Java	7	https://www.java.com/pt_BR/	Linguagem para desenvolvimento.
PostgreSql	9.5.5	https://www.postgresql.org/	Banco de dados.
Ivy	2.4.0	http://ant.apache.org/ivy/	Gerenciador de dependências.
IDE Eclipse	Neon	https://eclipse.org/	Integrated Development Environment (IDE) de desenvolvimento.
iReport	5.5.0	http://community.jaspersoft.com/project/ireport-designer	IDE desenvolvimento de relatórios.

Quadro 1 - Tecnologias e ferramentas utilizadas na modelagem e na implementação do sistema

2.1.1 PostgreSQL

O PostgreSQL (POSTGRESQL, 2016) é um sistema de banco de dados objeto-relacional de código aberto. Pode ser executado na maioria dos sistemas operacionais incluindo Linux, Mac OS X, Solaris e Windows. Possui amplo suporte a chaves estrangeiras, *joins*, *views*, *triggers* e *stored procedures*. As suas principais características são (SQL MAGAZINE, 2016):

a) Recuperação automática após falha de sistema (*Write Ahead Logs - WAL*);

b) *Multi Version Concurrency Control* (MVCC) ou controle de concorrência de multi versão. Nesse mecanismo, processos de leitura não bloqueiam processos de escrita e vice-

versa, reduzindo drasticamente a contenção entre transações concorrentes e paralisação parcial ou completa (*deadlock*);

c) *Commit, rollback* e *checkpoints* de transações;

d) *Triggers e stored procedures*;

e) *Foreignkeys*;

f) *Backupon-line*;

g) Tamanho ilimitado de registro. Não há limite para o tamanho dos tipos de dados;

h) Índices em *cluster*: cada tabela pode suportar um índice em *cluster*. Esse índice classifica fisicamente os dados na mesma sequência como especificada pelo índice. Um índice de *cluster* permite maior velocidade na recuperação de dados melhorando o desempenho geral do banco de dados.

2.1.2 Linguagem Java

A plataforma ou o ambiente de programação Java permite desenvolver aplicativos utilizando qualquer uma das linguagens criadas para a plataforma Java. Uma grande vantagem dessa plataforma é a de não estar vinculada a um único sistema operacional ou *hardware*, pois seu código gerado executa por meio de uma máquina virtual que pode ser emulada em qualquer sistema operacional que suporte a linguagem C++. Java é uma linguagem orientada a objetos, sendo assim, a maior parte dos elementos de um programa Java são objetos (DEITEL, DEITEL, 2003). Como exceção cita-se os tipos básicos, como o *int* e o *float*.

O código é organizado em classes, que podem estabelecer relacionamentos de herança simples entre si. Chamadas a funções de acesso remoto (*sockets*) e os protocolos *Internet* mais comuns - *Hypertext Transfer Protocol* (HTTP), *File Transfer Protocol* (FTP) e *Telnet* - são suportadas em Java, de forma que a implementação de programas baseados em arquiteturas cliente/servidor é facilitada. Java provê o gerenciamento de memória por meio de *garbage collection* (coleta de lixo). Sua função é a de verificar a memória de tempos em tempos, liberando automaticamente os blocos que não estão sendo utilizados. Esse procedimento pode deixar o sistema com execução mais demorada por manter uma *thread* paralela à execução do programa. Porém, esse procedimento evita problemas como referências perdidas e avisos de falta de memória quando ainda há memória disponível na máquina.

2.1.3 IDE Eclipse

O Eclipse (ECLIPSE, 2016) é um ambiente de desenvolvimento de software multilinguagens e compreende uma IDE e *plugins.* Por meio de *plugins*, a IDE Eclipse pode ser usada para desenvolver *software* em várias linguagens além de Java, como, C, C++, PHP e *Ruby*.

2.1.4 Apache Ivy

O Apache Ivy é um gerenciador de dependência, com foco na flexibilidade e simplicidade (IVY 2016). O gerenciador de dependência facilita o controle de bibliotecas utilizadas no desenvolvimento do projeto que não precisa mais ser realizado de maneira manual.

2.1.5 VRaptor

O VRaptor 3 possibilita produtividade e desenvolvimento rápido e fácil de sistemas Web em Java. É um *framework* MVC brasileiro *opensource* com grande comunidade de desenvolvedores e usuários (VRAPTOR 2016).

O VRaptor 3 foca em simplicidade e, portanto, com diversas funcionalidades ele têm como meta resolver o problema do programador da maneira menos intrusiva possível em seu código. Operações relacionadas à s

Salvar, remover, buscar e atualizar e funcionalidades que costumam ser mais complexas como *upload* e*download* de arquivos, resultados em formatos diferentes (*eXtensible Markup Language - XML*, *JavaScript Object Notation - JSON*, *eXtensible Hypertext Markup Language - XHTML*, etc) são realizadas por meio de funcionalidades simples do VRaptor 3, que sempre procuram encapsular *HttpServletRequest*, *Response*, *Session* e toda a *API* do *javax.servlet*.

2.1.6 IReport

O *iReport* é uma ferramenta que provê suporte para a criação de relatórios nos formatos *Portable Document Format* (PDF) para (arquivo somente leitura, XLS (arquivo do aplicativo *Microsoft Excel*) e HTML.

2.2 PROCEDIMENTOS TÉCNICOS

Para configuração do projeto foi utilizado o *VRaptor Scaffold*[1] que é uma extensão criada por Rodolfo Liviero, inspirado pela funcionalidade de *scaffoldingdo Ruby on Rails*. Com ele, a criação de projetos com o *VRaptor* fica bem mais simples e flexível. A documentação completa está em: http://vraptor3.vraptor.org/pt/docs/vraptor-scaffold-pt/.

Para usá-lo, é necessário ter o *RubyGems*(http://rubygems.org) instalado no sistema e seguir os seguintes passos:

1) Instalar a *gem* vraptor-scaffold:

gem install vraptor-scaffold

2) Criar um novo projeto:

vraptor new nomesoftware --package=br.com. seupacote .nomesoftware

3) Acessar a pasta do projeto e baixar as dependências com:

cd livraria ant resolve

4) Importar o projeto na IDE *Eclipse* ou *Netbeans*.

O projeto criado usa o *ant* (http://ant.apache.org) para gerenciar o *build* e o *Ivy* (http://ant.apache.org/ivy) para gerenciar as dependências. Mas é possível mudá-lo para usar o *Maven* ou o *Gradle* como ferramenta de *build*. O *VRaptorScaffold* gera vários arquivos no projeto, entre eles:

• arquivos de configuração de projeto do eclipse (.project, .classpath e .settings)

• build.xml e build.properties — arquivos de configuração do Ant com algumas *targets* já configuradas, como compile para compilar as classes, war para gerar o war e jetty.run para subir a aplicação num jetty.

[1] VRaptor Scaffold GITHUB - https://github.com/caelum/vraptor-scaffold

• ivy.xml e ivy.jar — arquivos de configuração do Ivy. É necessário editar o arquivo ivy.xml para acrescentar dependências ao projeto e executar o Ant Resolve para baixá-las.

• em src/main/java foram criados três pacotes: controllers, models e repositories, e arquivos padrão caso o desenvolvedor utilize o comando *VRaptor Scaffold* para criar os modelos da sua aplicação.
• em src/main/resources foram criados os arquivos de configuração do hibernate, jpa e log4j.

• em src/main/webapp foram criados arquivos JavaScript e *Cascading Style Sheet* (CSS padronizados para facilitar o início do projeto. Também está configurado o *sitemesh* (http:// sitemesh.org), que facilita a criação de *templates* (como cabeçalhos e rodapés, menus padronizados etc).

3 RESULTADOS

Este capítulo apresenta o resultado da realização do trabalho.

3.1 ESCOPO DO SISTEMA

O sistema consiste em gerenciar pedidos de vendas efetuados pelo usuário que é um representante. Esse usuário possui diversos clientes e fornecedores e trabalha com representação de queijos e frios no Estado de São Paulo. Esse usuário utilizava planilhas para efetuar o controle de pedidos e das comissões e dos seus clientes, fornecedores e respectivos produtos. Esse contexto de interesse e essas necessidades serviam como base para o desenvolvido um sistema.

O sistema possui cadastro de clientes, fornecedores, produtos, pedidos, unidademedida, forma de pagamento, categoria de produto, gráfico de produtos mais vendidos e relatórios.

O usuário efetua o cadastro de categorias de produtos de acordo com sua necessidade e o cadastro de unidades de medida. No cadastro de fornecedores são utilizadas categorias previamente cadastradas, que são categorias cujos produtos ele vai representar.

No cadastro de produtos são informados campos necessários de acordo com requisitos levantados com o cliente, incluindo categoria e unidades de medida previamente cadastradas. É importante ressaltar que se faz necessário ter o fornecedor do produto para que assim seja escolhido posteriormente no cadastro de produto esse fornecedor e a categoria desse produto.

O usuário também faz o cadastro de clientes no sistema também informando todas as informações relevantes para que posteriormente seja possível efetuar o pedido. O cliente no caso é a pessoa ou a empresa que faz um pedido que é repassado para o respectivo fornecedor.

Para efetuar um pedido o usuário deve escolher o fornecedor, o cliente e a forma de pagamento que foram cadastrados anteriormente. Após salvo, o usuário poderá incluir produtos a esse pedido. Os produtos apresentados para incluir são somente produtos do fornecedor escolhido no início do pedido, após a escolha de produtos e da quantidade o sistema apresenta uma tela na qual o usuário pode conferir e alterar o pedido, feito isso o pedido pode ser finalizado e impresso para que possa ser enviado para o cliente por *e-mail*.

A cada venda efetuada e concluída o usuário ganha comissão sobre o valor total da venda, o valor dessa comissão em percentual é informado no cadastrado do fornecedor no sistema.

Na tela inicial do sistema é apresentado um gráfico com os produtos mais vendidos até o momento. Também ficam disponíveis relatórios de comissões com filtro por fornecedor e data, relatório de clientes, de fornecedores, e de produtos por fornecedor.

3.2 MODELAGEM DO SISTEMA

Os requisitos identificados para o sistema agrupam-se em:

a) Controlar os clientes:

- Saber quantos clientes a empresa possui;
- Quem são e sua localização;

- Consultas avançadas, com diversos filtros nos dados armazenados dos clientes;

- Gerar relatórios com os resultados das consultas.

b) Controlar os fornecedores:

- Saber quantos fornecedores a empresa possui;

- Quem são e sua localização;

- Consultas avançadas, com diversos filtros nos dados armazenados dos fornecedores;

- Gerar relatórios com os resultados das consultas.

c) Controlar os produtos:

- Saber quantos produtos a empresa representa;

- Quais são e de onde vem;

- Consultas avançadas, com diversos filtros nos dados armazenados dos produtos;

- Gerar relatórios com os resultados das consultas.

d) Controlar unidade medida:

- Saber com quais unidades de medida a empresa trabalha;

- Poder fazer consultas avançadas, com diversos filtros nos dados armazenados das unidades medidas;

- Gerar relatórios com os resultados das consultas.

e) Controlar Categorias:

- Saber com quais categorias de produtos a empresa trabalha;

- Poder fazer consultas avançadas, com diversos filtros nos dados armazenados das categorias de produtos;

- Gerar relatórios com os resultados das consultas.

f) Controlar forma de pagamento:

- Saber com quais formas de pagamentos a empresa trabalha;

- Poder fazer consultas avançadas, com diversos filtros nos dados armazenados das formas de pagamento;

- Gerar relatórios com os resultados das consultas.

g) Controlar as vendas:

- Saber quantas vendas a empresa efetuou;

- Quais os principais produtos vendidos;

- Consultas avançadas, com diversos filtros nos dados armazenados das vendas;

- Gerar relatórios com os resultados das consultas.

h) Controlar a percentagem de comissão:

- Saber quanto a empresa ganhou sobre a venda;

- Gerar relatórios;

i) Controlar cidades:

- Controlar o cadastro de cidades;

j) Controlar Estados:

- Controlar o cadastro de Estados;

Esses requisitos foram representados como casos de uso, como é mostrado na Figura 1. Os casos de uso são compostos por um ator principal que realizará as atividades de manutenção de cadastros sendo que este precisa estar autenticado para acessar o sistema.

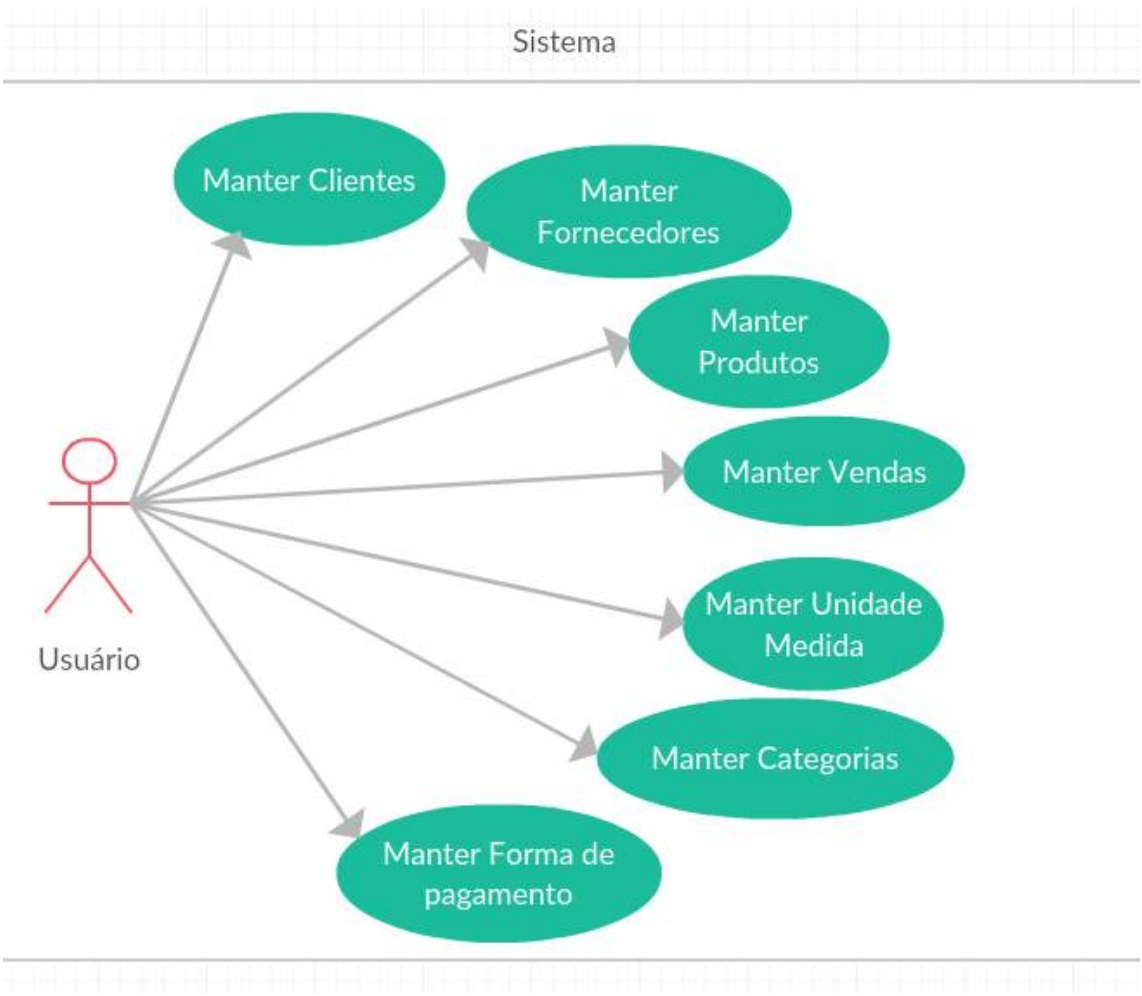

Figura 1 - Diagrama de caso de uso

O Quadro 2 apresenta os requisitos funcionais agrupados por casos de uso apresentados na Figura 1. Nesse quadro RF significa requisito funcional.

	Caso de uso	**Requisito**	**Descrição**
RF01	Manter clientes	Cadastro de clientes	Os clientes da empresa ou negócio.
RF02	Manter fornecedores	Cadastro de fornecedores	Os fornecedores da empresa ou negócio.
RF03	Unidade medida	Cadastro de unidade medida	Unidade medida do produto a ser vendido. Exemplos: kg para quilograma e cx para caixa.
RF04	Categoria	Cadastro de categoria	Categorias dos produtos a serem vendidos exemplo: queijo, frios.
RF05	Manter produtos	Cadastro de produto	Os produtos a serem vendidos.
RF06	Forma de	Cadastro de forma de	Possíveis formas de pagamentos.

	pagamento	pagamento.	
RF07	Manter vendas	Cadastro de nova venda	Inserir nova venda para um cliente
RF08	Manter vendas	Produtos da venda	Inserção de produtos na venda(pedido).
RF09	Manter vendas	Resumo venda	Resumo da venda, produtos inseridos com possibilidade de alteração de quantidade e remoção de produtos.
RF10	Manter vendas	Impressão	Impressão com pré-visualização da venda (pedido).
RF11	Manter vendas	Comissões	Impressão de relatório de comissões sobre as vendas.

Quadro 2 - Requisitos Funcionais

No Quadro 3 estão listados os requisitos não funcionais. Nesse quadro RNF significa requisito não funcional.

	Requisito	**Descrição**
RNF01	Cálculo de comissões	O cálculo de comissões será feito sobre o valor previamente inserido no cadastro do fornecedor no campo “Percentual Comissão (%)”.
RNF02	Acesso ao sistema	O acesso ao sistema deve ser total sem controle de permissões.
RNF03	Relatório de comissões	O relatório de comissões deve ter as seguintes opções de filtros: Data inicial, data final e fornecedor.
RNF04	Relatório de produtos	O relatório de produtos deve ter a seguinte opção de filtro:fornecedor.
RNF05	Quantidade de clientes	Deve ser exibido na tela inicial do sistema a quantidade de clientes que a empresa tem em sua carteira.
RNF06	Quantidade de fornecedores	Deve ser exibido na tela inicial do sistema a quantidade de fornecedores que a empresa tem em sua carteira.
RNF07	Quantidade de vendas(pedidos)	Deve ser exibido na tela inicial do sistema a quantidade de vendas efetuadas até o momento.
RNF08	Quantidade de produtos vendidos.	Deve ser exibido na tela inicial do sistema a quantidade de produtos vendidos até o momento.
RNF09	Manter vendas	Gráfico com produtos mais vendidos

Quadro 3 - Requisitos não funcionais

A Figura 2 apresenta o diagrama de entidades e relacionamentos do sistema.

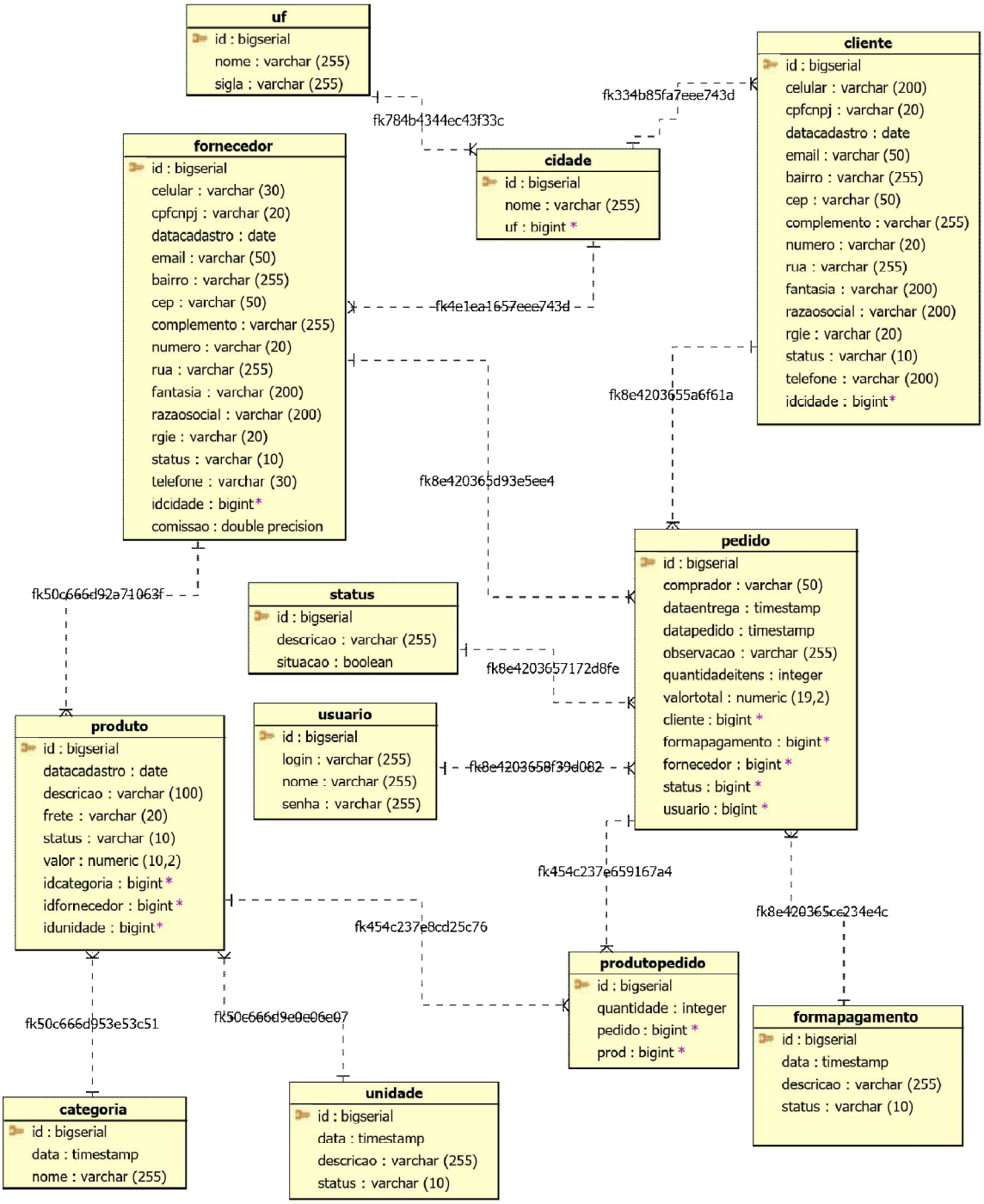

Figura 2 - Diagrama de Entidade Relacionamento

Na Figura 3 está o diagrama de classes elaborado para o sistema

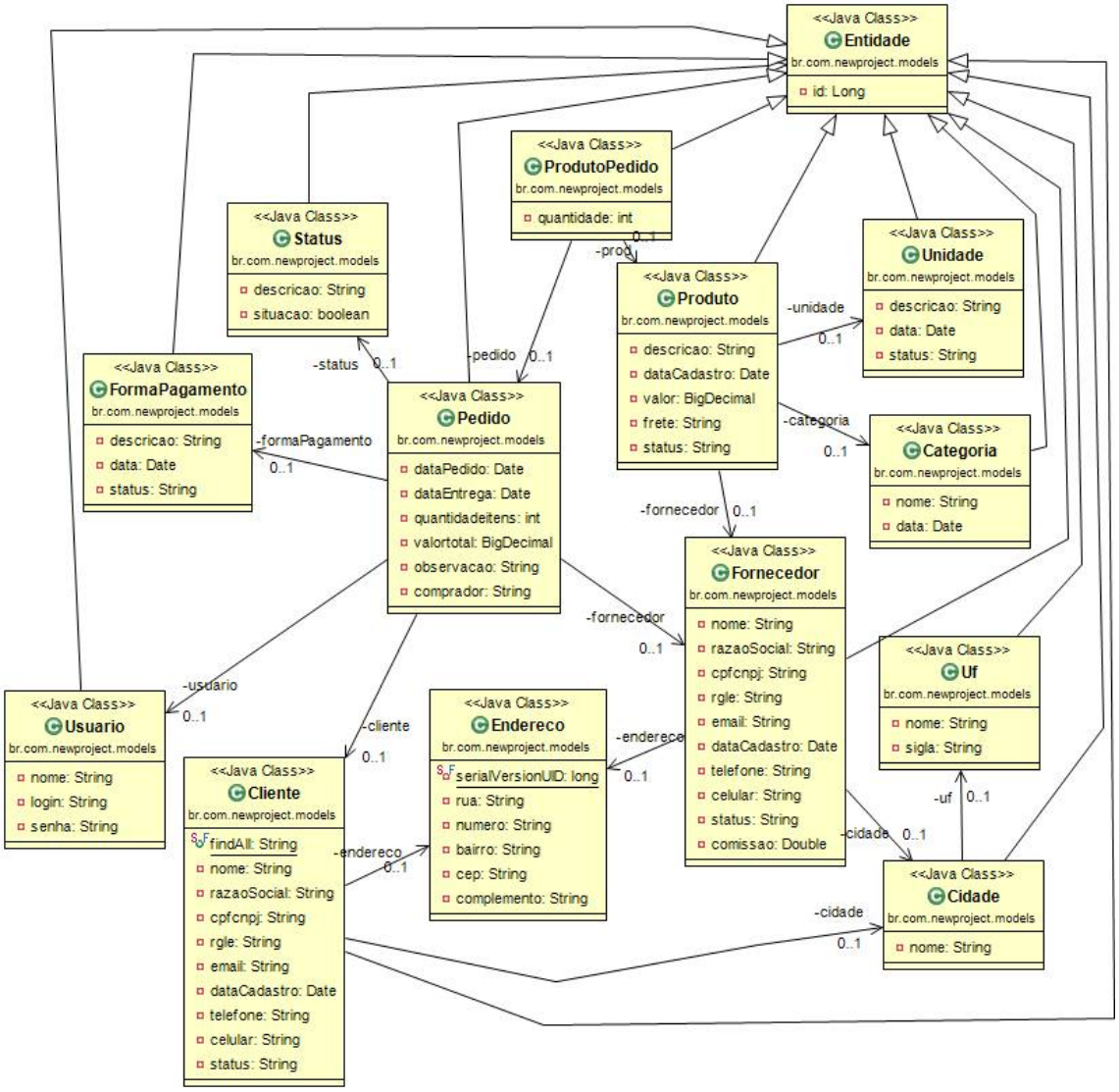

Figura 3 - Diagrama de Classe

3.3 APRESENTAÇÃO DO SISTEMA

Ao acessar o sistema pelo navegador a primeira tela exibida é a tela de *login*. Na Figura 4 está a tela de login do sistema, responsável pela validação de acesso ao sistema por meio de um campo de identificação e da respectiva senha.

Figura 4 - Login do sistema

Como pode ser visto na Figura 4 o usuário deverá informar seu login e sua senha. Em caso de entrada inválida para um desses campos ou os dois é apresentada a mensagem “Usuário e/ou senha inválidos!”. Após realizada a validação de acesso e todos os dados estarem de acordo com o seu cadastro no banco de dados, o sistema será aberto com a tela inicial, a qual pode ser observada na Figura 5.

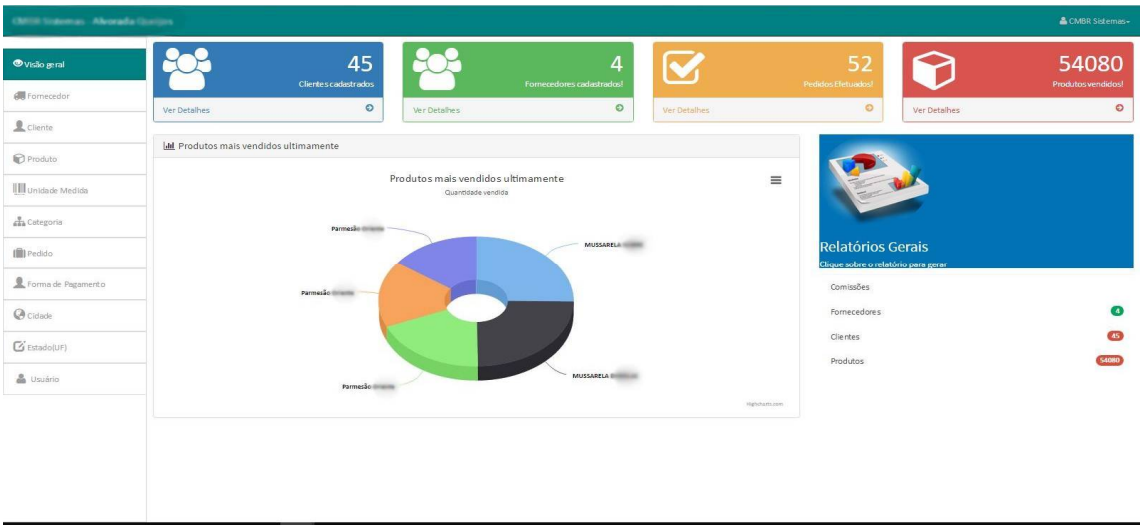

Figura 5 - Tela inicial do sistema

Na parte superior da tela da Figura 5 está o nome do usuário autenticado na aplicação e ao clicar sobre esse campo a opção de “*Logout*” do sistema é exibida, função essa que permite ao usuário desconectar-se do sistema. No lado esquerdo da tela estão os menus de acesso às funções do sistema. Na parte central da tela estão painéis disponibilizando informações de quantidade de cliente, fornecedores, produtos e vendas. No menu “Cadastros” está o cadastro de fornecedor. Ao ser acessada essa opção é aberta uma tela, que é exibida na Figura 6, com uma listagem de todos os fornecedores cadastrados no sistema.

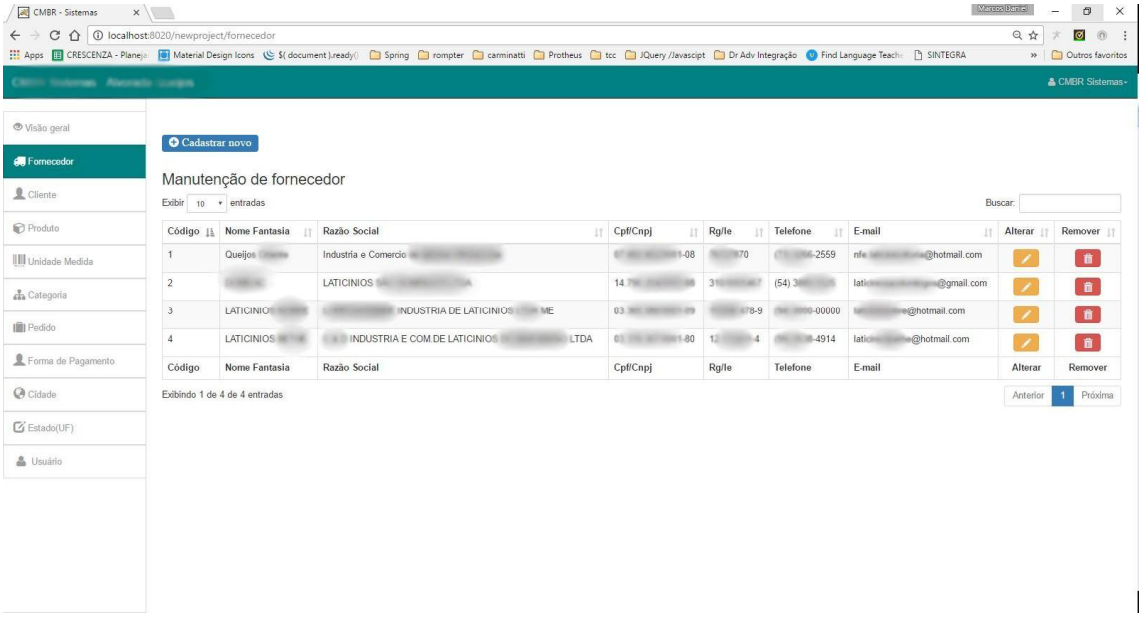

Figura 6 - Listagem de fornecedor

Cada linha da tabela mostrada na Figura 6 representa um fornecedor cadastrado com as suas respectivas informações. Essas informações podem ser ordenadas de acordo com a necessidade do usuário, para tal basta clicar sobre cada coluna da tabela para que ela fique ordenada pela coluna e em ordem crescente ou decrescente. No final de cada linha estão disponíveis dois botões. Um deles tem a função de alterar os dados do fornecedor e o outro tem a função de removê-lo.

Para cadastrar um novo fornecedor foi disponibilizado o botão logo acima da tabela com o texto “Cadastrar novo”, que quando clicado abre a tela para que o usuário entre com as informações do novo fornecedor como mostrado na Figura 7. Nessa tela o usuário entra com os dados do fornecedor e clica no botão cadastrar para incluir o novo fornecedor ou no botão cancelar para cancelar esse cadastro.

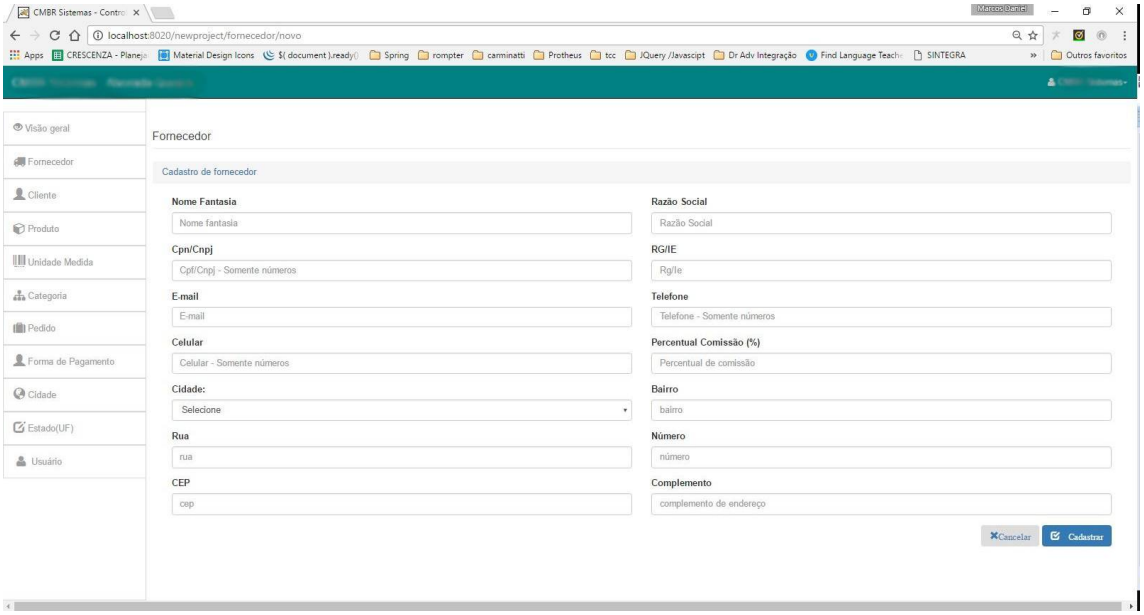

Figura 7 - Novo fornecedor

Para todas as telas do sistema foi adotado o mesmo padrão, como pode ser visto nas Figuras 8, 9, 10 e 11.

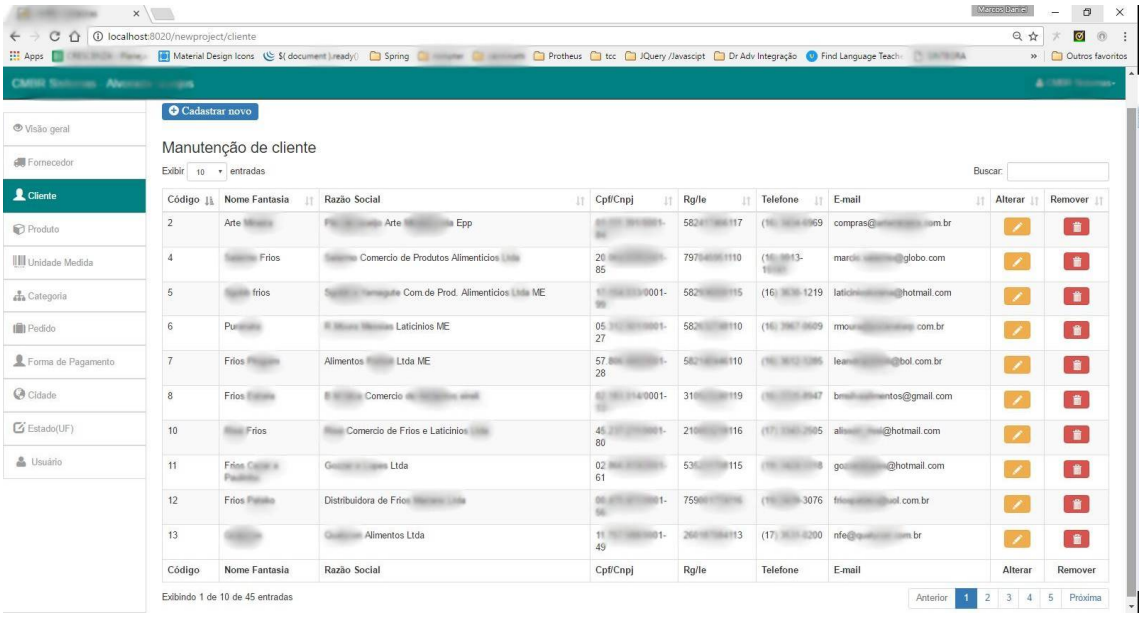

Figura 8 - Listagem de cliente

No cadastro de clientes apresentado na Figura 9 são apresentados os campos conforme elicitados com o usuário do sistema, com as informações necessárias para a realização de um pedido.

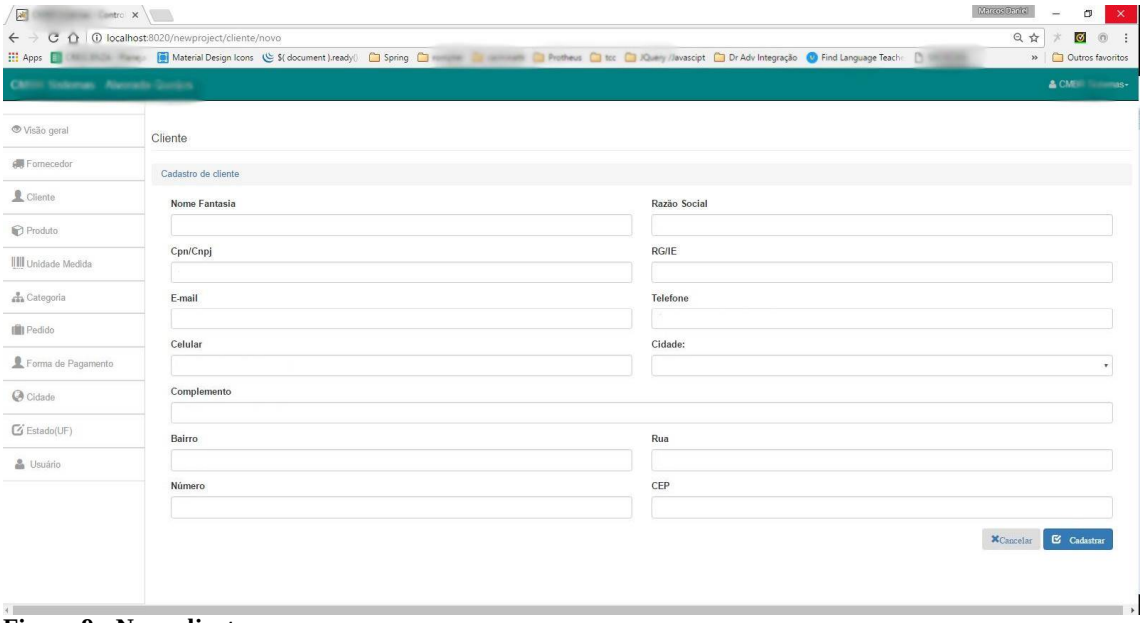

Figura 9 - Novo cliente

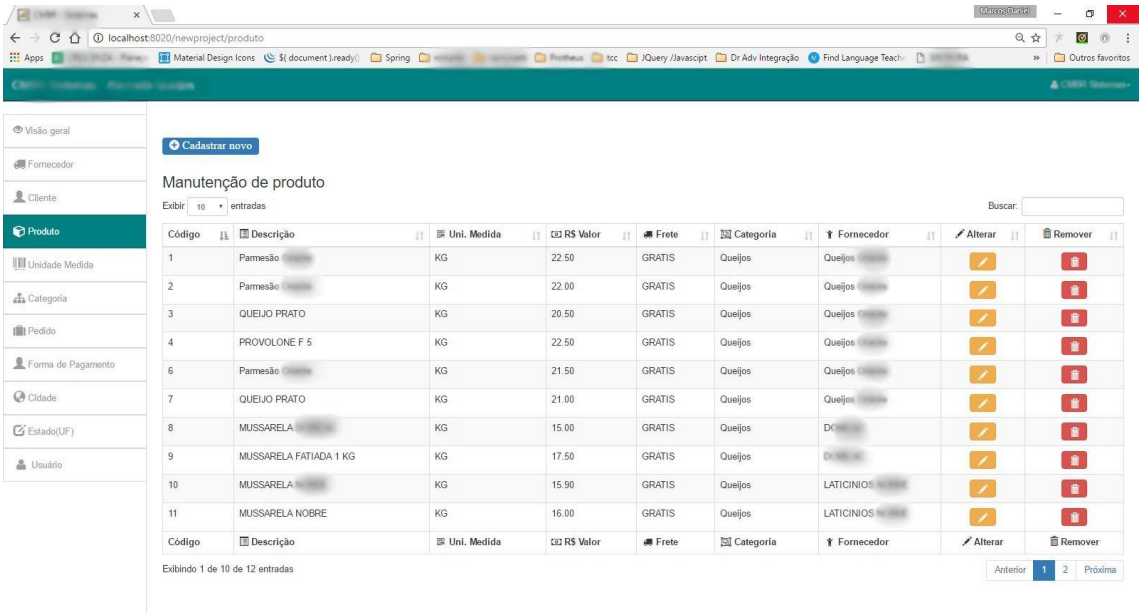

Figura 10 - Listagem de produto

No cadastro de produtos é informada a descrição, o valor de venda, a unidade medida, se o mesmo tem frete e não qual a categoria do produto e o seu respectivo fornecedor, conforme exibido na Figura 11.

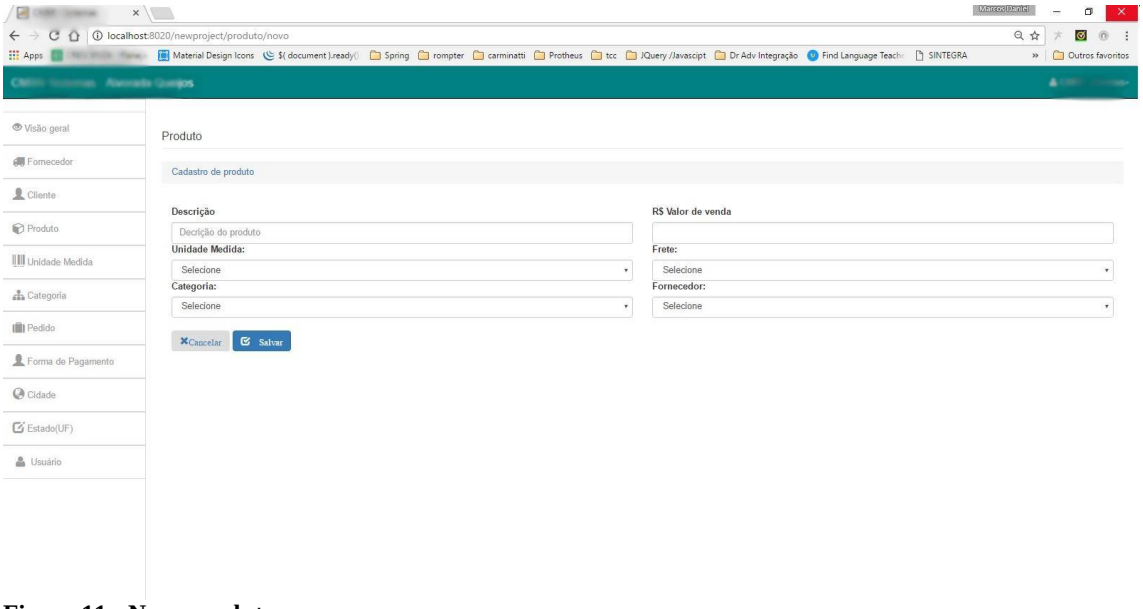

Figura 11 - Novo produto

Quando clicado sobre o menu de pedido é aberta uma tela com a listagem de todos os pedidos efetuados até o momento por ordem decrescente como padrão, como pode ser visto na Figura 12. Nessa tela há mais opções como a de adicionar produtos ao pedido, imprimir o pedido, alterar e remover o respectivo pedido.

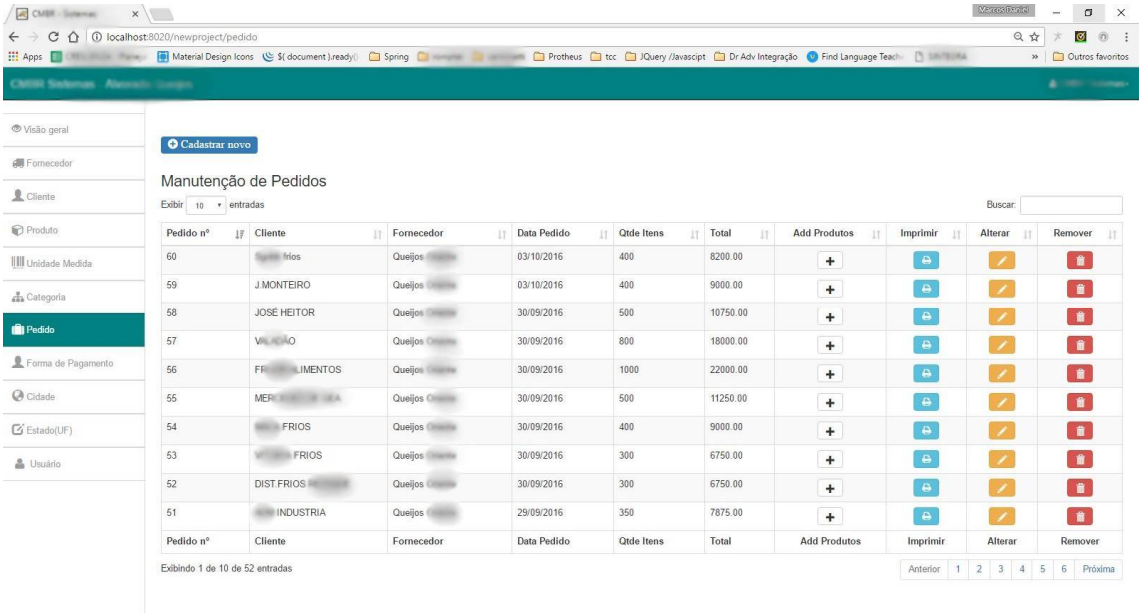

Figura 12 - Listagem de pedido

Ao clicar em cadastrar novo é aberta uma tela para o cadastro de um novo pedido no sistema, como pode ser observado na Figura 13, na qual o usuário entra com os dados necessários para a abertura de um novo pedido.

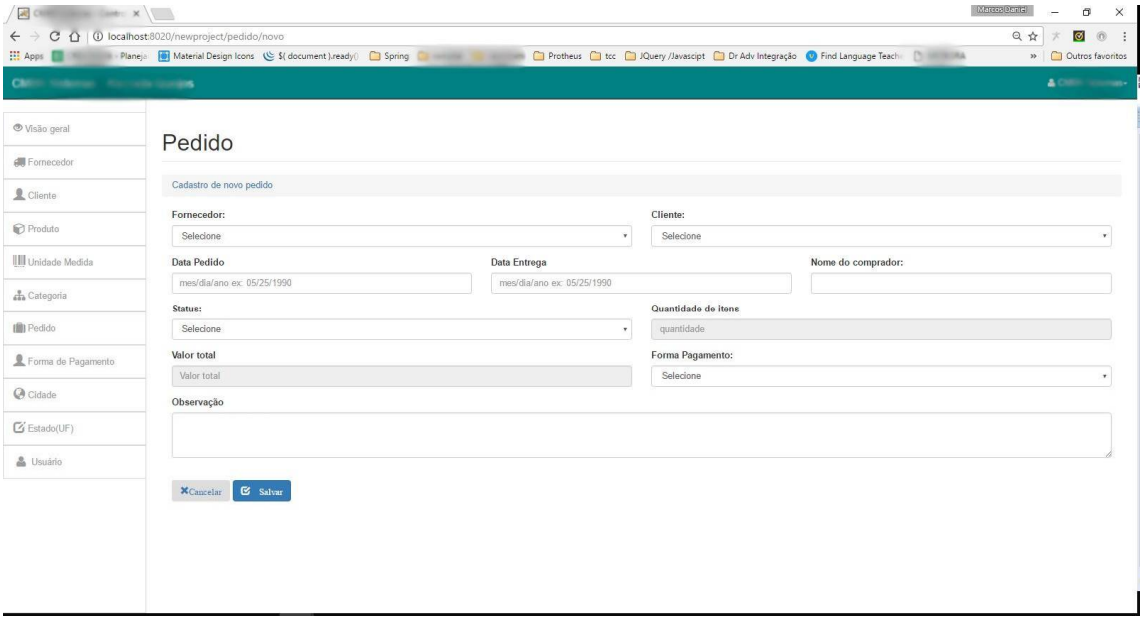

Figura 13 - Novo pedido

Preenchido os campos corretamente ao clicar em salvar o sistema apresenta ao usuário a tela na qual ele adiciona os produtos com a quantidade desejada ao seu pedido, como apresentado na Figura 14, os produtos apresentados são apenas os produtos relacionados ao fornecedor definido na abertura do pedido.

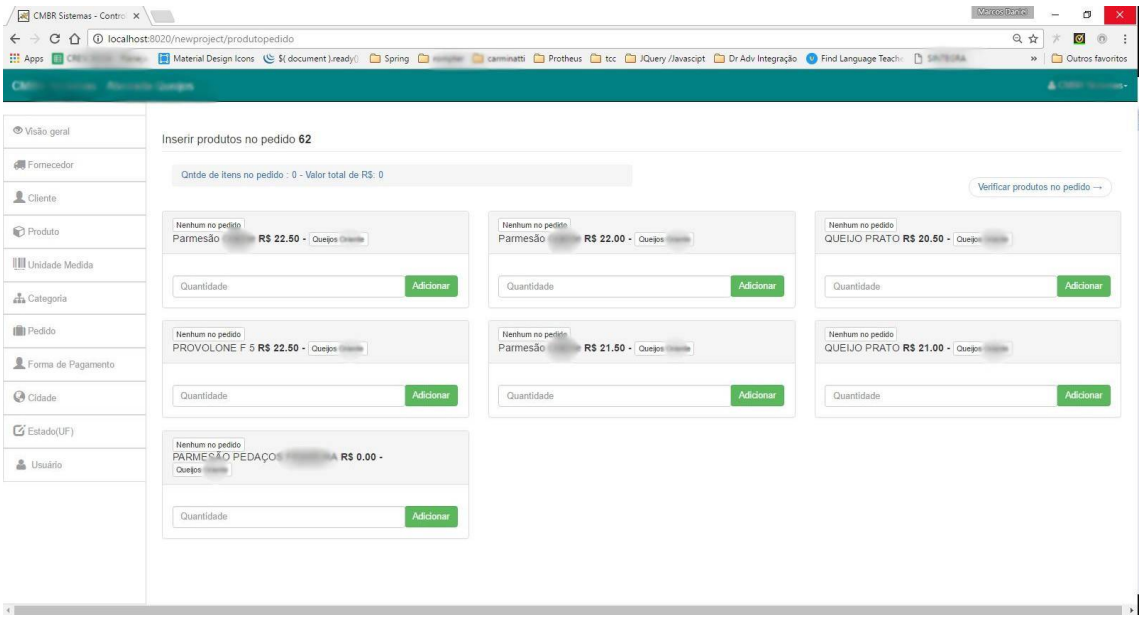

Figura 14 - Adicionando produtos ao pedido

Nessa tela o usuário ao ir adicionando novos produtos ao pedido, consegue verificar a quantidade de itens que tem no pedido e o valor total até o momento, essa informação encontra-se logo acima dos produtos exibidos. Ele também consegue verificar quais produtos já estão no pedido por meio de um indicador na própria exibição do produto. Caso ele não tenha nenhum produto no pedido é informado “Nenhum no pedido” e caso tenha é informada a quantidade de itens no pedido, e também quando adicionado é exibida uma mensagem informando “Produto adicionado com sucesso ao pedido!”, como mostrado na Figura 15.

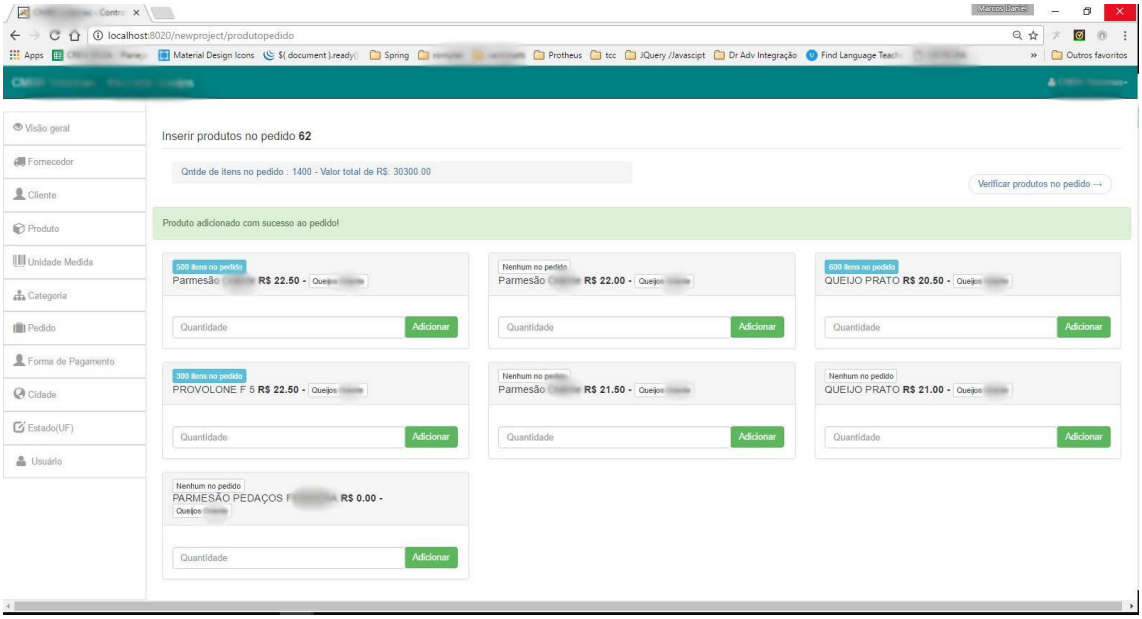

Figura 15 - Pedido com produto adicionados

Após serem adicionado todos os produtos desejados no pedido é necessário clicar em "Verificar produto no pedido" no canto superior direito da tela. É, assim, apresentada uma tela para um aferimento dos produtos no pedido e é possível excluir e alterar a quantidade. Essa tela é apresentada na Figura 16.

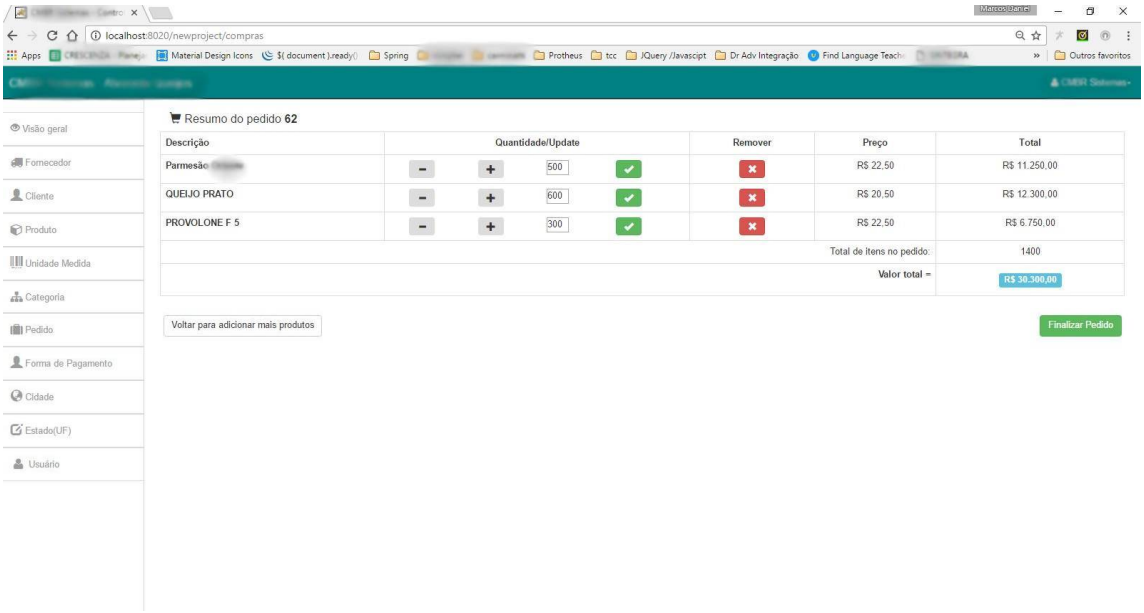

Figura 16 - Conferência e alteração de itens no pedido

Nessa tela (Figura 16) o usuário pode alterar a quantidade dos produtos que adicionou anteriormente no pedido, remover esse produto, voltar para adicionar mais itens no pedido ou finalizar o pedido. Ao clicar em “Finalizar Pedido” é exibido ao usuário uma pré-impressão do pedido na tela para conferência. Dessa tela é possível voltar para realizar ajustes que possam ser necessários ou imprimir o pedido para enviar por e-mail para o fornecedor do produto. Essa tela é apresentada na Figura 17.

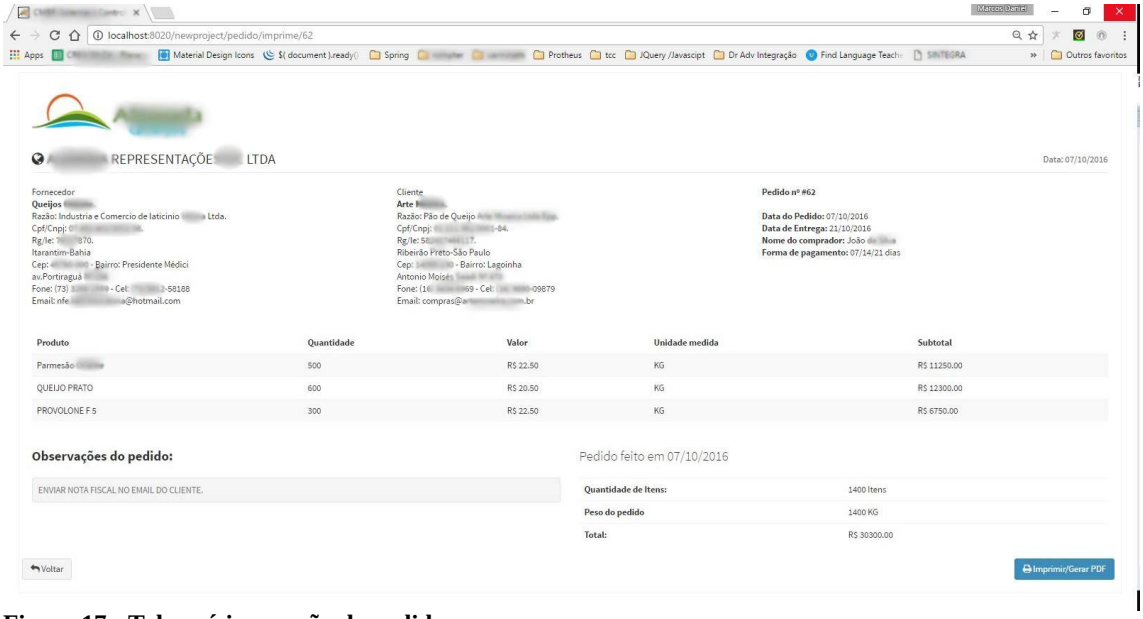

Figura 17 - Tela pré-impressão do pedido.

Nas Figuras 18 e 19 é apresentado um pedido impresso.

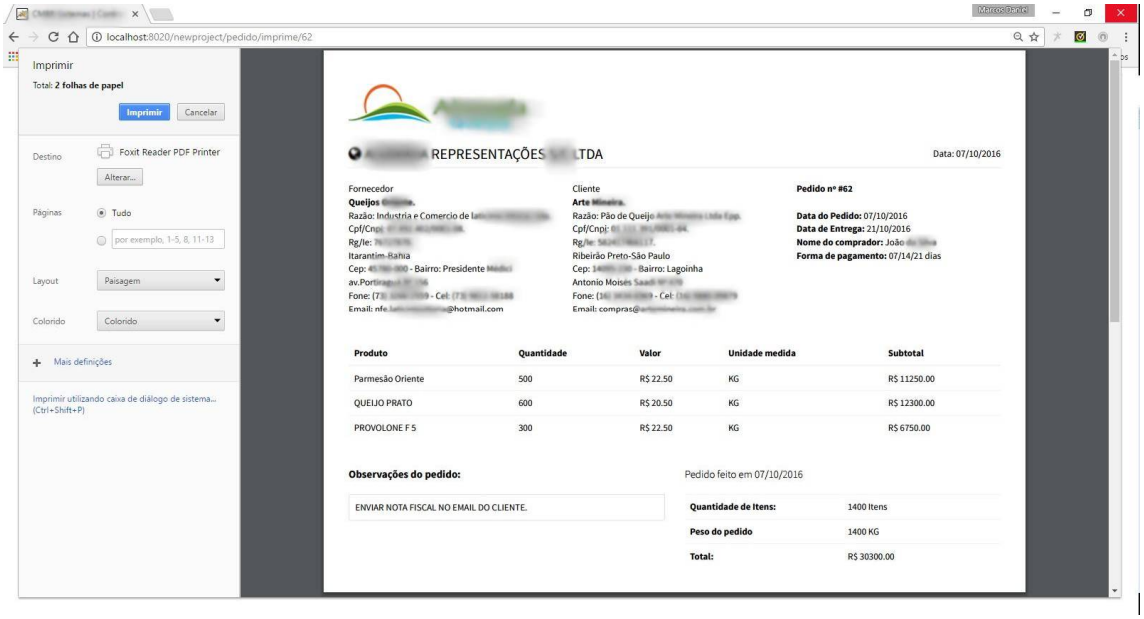

Figura 18 - Tela para imprimir ou salvar pedido.

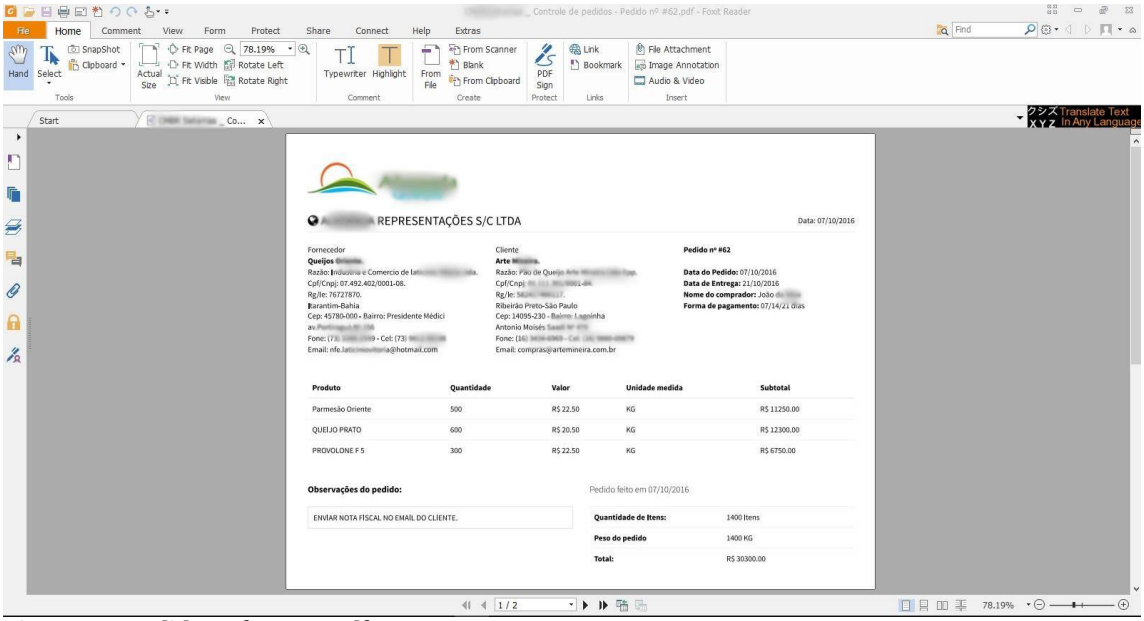

Figura 19 - Pedido no formato pdf

3.4 IMPLEMENTAÇÃO DO SISTEMA

Conforme apresentado no Capítulo 2, para a codificação do sistema foi utilizada a linguagem de programação Java com o framework VRaptor. Nessa Seção serão apresentados trechos de código fonte utilizados no desenvolvimento do sistema.

Na Listagem 1 é possível verificar a forma como é desenvolvido um *interceptor* que é utilizado para que quando o usuário fizer *logout* do sistema e clicar no botão voltar do navegador, ele não consiga retornar para a última tela em que estava no sistema sem efetuar nova autenticação.

```
@Intercepts
publicclass NoCacheInterceptor implements Interceptor{
	privatefinal HttpServletResponse response;

	public NoCacheInterceptor(HttpServletResponse response) {
		this.response = response;
	}

	public HttpServletResponse getResponse() {
		returnresponse;
	}

	@Override
	publicboolean accepts(ResourceMethod method) {
		returnmethod.containsAnnotation(NoCache.class);
	}

	@Override
	publicvoid intercept(InterceptorStack stack, ResourceMethod method,
			Object resourceInstance) throws InterceptionException {
		// set the expires to past
	  getResponse().setHeader("Expires", "Wed, 31 Dec 1969 21:00:00 GMT");

// no-cache headers for HTTP/1.1
	  getResponse().setHeader("Cache-Control", "no-store, no-cache, must-
revalidate");

// no-cache headers for HTTP/1.1 (IE)
	  getResponse().addHeader("Cache-Control", "post-check=0, pre-check=0");

// no-cache headers for HTTP/1.0
	  getResponse().setHeader("Pragma", "no-cache");

stack.next(method, resourceInstance);
	}
}
```

Listagem 1 – Interceptador

A classe *interceptor* tem como função expirar o cache que até então estava no navegador, e faz isso toda vez que houver uma requisição de um método que contenha a

anotação @NoCache. Quando é feita uma requisição para um método que tenha essa anotação ele seta no *header* da resposta HTTP algumas informações que fazem com que o cache volte para a data que não foi utilizado, perdendo assim as informações estavam no cache do navegador.

Para que ele funcione deve ser codificada também uma anotação em tempo de execução a qual leva o nome do NoCache como mostrado na Listagem 2.

```
//a anotaçãovaificardisponívelemtempodeexecucao
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD) // anotaçãoparamétodos
public@interfaceNoCache{

}
```

Listagem 2 – Anotação *NoCache*

Nessa *interface* foi criada uma anotação própria que ficará disponível em tempo de execução e será anotada apenas em métodos. Assim, todo o método que tiver a anotação @NoCache executará todo o código desenvolvido na classe da Listagem 1. Um exemplo de utilização é apresentado na Listagem 3.

```
	@NoCache
	@RestritoUserNormal
	@Path("/home")
	publicvoid home() throws SQLException {
		result.include("qntdprodutos", qntdProdVendidos());
		result.include("qntdpedidos", malDao.buscarTodos().size());
		result.include("qntdfornecedores", forDao.buscarTodos().size());
		result.include("fornecedores", forDao.buscarTodos());
		result.include("qntdclientes", cliDao.buscarTodos().size());
	}

	@NoCache
	@Path("/logout")
	publicvoid logout(Usuario usuario) {
		usuarioWeb.logout();
		carrinho.zera();
		result.redirectTo(IndexController.class).index();
	}
```

Listagem 3 – Exemplo utilização anotação @NoCache

Outro exemplo de interceptor é a classe AutenticacaoInterceptorUserNormal para qual também foi criada uma anotação. Essa classe tem a função de verificar se o usuário efetuou o *login* no sistema, se está autenticado e se o método contém a anotação @RestritoUserNormal, se estiver tudo certo o usuário fará a requisição e terá resposta caso contrário ele é redirecionado para efetuar o login no sistema, o desenvolvimento é descrito na Listagem 4.

```
@Intercepts
publicclass AutenticacaoInterceptorUserNormal implements Interceptor{

	privatefinal UsuarioWeb usuario;
	privatefinal Result result;

	public AutenticacaoInterceptorUserNormal(UsuarioWeb usuario, Result result)
{
		this.usuario = usuario;
		this.result = result;
	}

	@Override
	publicboolean accepts(ResourceMethod method) {
			return !this.usuario.isLogado()
&&method.containsAnnotation(RestritoUserNormal.class);
	}

	@Override
	publicvoid intercept(InterceptorStack stack, ResourceMethod method,
			Object resourceInstance) throws InterceptionException {
			result.redirectTo(IndexController.class).index();
	}

}
```

Listagem 4 – Classe AutenticacaoInterceptorUserNormal

Para a classe AutenticacaoInterceptorUserNormal também foi criada uma anotação que é chamada de RestritoUserNormal, que fica disponível em tempo de execução e é feita antes da declaração dos métodos. Desta maneira qualquer método que contenha essa anotação @RestritoUserNormal executará o código da Listagem 4. Nesse trecho de código é verificado se o usuário está realmente autenticado no sistema, se sim irá proceder normalmente caso contrário voltara para a página *index*, a qual é o login da aplicação. Assim, é possível assegurar que o usuário esteja autenticado no sistema. A implementação desta anotação segue na Listagem 5.

```
//a anotaçãovaificardisponívelemtempodeexecucao
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD) // anotaçãoparamétodos
public@interfaceRestritoUserNormal{

}
```

Listagem 5 – Interface RestritoUserNormal

Na Listagem 6 é possível verificar como foi desenvolvida a parte de *Create, Retrieve, Update and Delete* (CRUD) de clientes, que contém os métodos para adicionar, excluir e editar. Nota-se que todos os métodos têm a anotação @RestritoUserNormal que foi apresentada na Listagem 5, o que faz com que o usuário esteja obrigatoriamente autenticado

no sistema para que possa utilizar esses métodos. No método para adicionar é realizada a verificação da anotação @Post(“/cliente”). Essa anotação faz com que o método espere um *submit* do tipo *POST* do formulário na *view*, o “/cliente” é a *Uniform Resource Locator* (URL) a qual esse formulário será submetido. O método espera um parâmetro do tipo Cliente que vem da *view* com um *Parse* que o próprio *framework* faz. Há, ainda, uma verificação para saber se o atributo cidade da classe Cliente realmente foi preenchido. Caso não tenha sido preenchido é enviada uma mensagem de erro para a *view* informando que o campo é de preenchimento obrigatório. Após isso é definida uma data de cadastro para o cliente, seu *status* para “ATIVO” e, por fim, os dados do cadastro são salvos na base de dados e é retornado para a listagem de clientes.

```
@Resource
@RequestScoped
publicclass ClienteController {

	privatefinal Result result;
	privatefinal ICidadeDao cidDao;
	privatefinal IClienteDao dao;
	privatefinal Validator validator;

	public ClienteController(Result result, ICidadeDao cidDao, Validator
validator, IClienteDao dao) {
		this.result = result;
		this.cidDao = cidDao;
		this.validator = validator;
		this.dao = dao;
	}

	@RestritoUserNormal
	@Get("/cliente/novo")
	publicvoid novo(){
		result.include("cidades", cidDao.buscarTodos());
	}

	@RestritoUserNormal
	@Get("/cliente")
	publicvoid cliente() throws Exception{
		result.include("cidades", cidDao.buscarTodos());
		result.include("clientes", lista());
	}

	@RestritoUserNormal
	@Post("/cliente")
	publicvoid adiciona(Cliente cliente) throws Exception{
	if(cliente.getCidade().getId() == null){
			validator.add(new ValidationMessage("O campo cidade é de
preenchimento obrigatório!", "Cidade"));
			validator.onErrorRedirectTo(this).cliente();
		}
		cliente.setStatus("ATIVO");
		cliente.setDataCadastro(new Date());
		dao.salvar(cliente);
		result.redirectTo(this).cliente();
```

```
	}

	@RestritoUserNormal
	@Delete("/cliente/{id}")
	publicvoid excluir(Long id) throws Exception{
		if(dao.verificaDependencias(id)){
			validator.add(new ValidationMessage("Você não pode excluir
esse cliente! ", "Cliente com Produto Vinculado"));
			validator.onErrorRedirectTo(this).cliente();
		}else{
			Cliente cliente = dao.buscarClientePorId(id);
			cliente.setStatus("INATIVO");
			dao.salvar(cliente);
		}
		result.redirectTo(this).cliente();
	}

	@Get("/cliente/{id}")
	public Cliente edita(Long id) throws Exception {
		result.include("fornecedor", dao.buscarClientePorId(id));
		result.redirectTo(this).novo();
		returndao.buscarClientePorId(id);
	}

	@RestritoUserNormal
	@Get("/clienteS")
	public List<Cliente> lista(){
		returndao.buscarTodos();
	}

}
```

Listagem 6 – Classe ClienteController

Na Listagem 7 está a classe ClienteDao, a qual é responsável pela comunicação com o banco de dados. Essa classe é responsável por salvar, alterar, remover e buscar dados do banco dados. Ela implementa uma interface que obriga o desenvolvedor a sobrescrever os métodos toda vez que ela é implementada. A classe ClienteDao é mostrada na Listagem 7 e a interface que a mesma implementa na Listagem 8, a interface IClienteDao.

```
@Component
publicclass ClienteDao implements IClienteDao{
	privatefinal EntityManager manager;
	public ClienteDao(EntityManager manager) {
		this.manager = manager;
	}

	@Override
	public Cliente salvar(Cliente cliente) {
		returnmanager.merge(cliente);
	}

	@Override
	publicvoid excluir(Cliente cliente) {
		manager.remove(cliente);
	}
```

```
	@Override
	public Cliente buscarClientePorId(Long id) {
		EasyCriteria<Cliente>criteria =
EasyCriteriaFactory.createQueryCriteria(manager, Cliente.class);
		criteria.andEquals("id", id);
		returncriteria.getSingleResult();
	}

	@Override
	public List<Cliente> buscarTodos() {
		EasyCriteria<Cliente>criteria =
EasyCriteriaFactory.createQueryCriteria(manager, Cliente.class);
		criteria.andEquals("status", "ATIVO");
		returncriteria.getResultList();
	}

	@Override
	public List<Cliente> buscarTodosRelatorio() {
		EasyCriteria<Cliente>criteria =
EasyCriteriaFactory.createQueryCriteria(manager, Cliente.class);
		criteria.andEquals("status", "ATIVO");
		criteria.orderByAsc("razaoSocial");
		returncriteria.getResultList();
	}

	@Override
	publicboolean verificaDependencias(Long id) {
		booleanretorno = false;
		EasyCriteria<Pedido>critUser =
EasyCriteriaFactory.createQueryCriteria(manager, Pedido.class);
		critUser.andEquals("cliente", id);
		if(critUser.getResultList().size() > 0){
			retorno = true;
		}
		returnretorno;
	}
}
```

Listagem 7 – Classe ClienteDao

```
publicinterface IClienteDao {
	Cliente salvar(Cliente cliente);
	void excluir(Cliente cliente);
	Cliente buscarClientePorId(Long id);
	List<Cliente> buscarTodos();
	List<Cliente> buscarTodosRelatorio();
	boolean verificaDependencias(Long id);
}
```

Listagem 8 – Interface IClienteDao

4 CONSIDERAÇÕES FINAIS

Um sistema *web* para gerenciamento de pedidos para empresas que atuam com representações e vendas de produtos foi obtido como resultado do desenvolvimento deste trabalho. Esse sistema auxilia no gerenciamento de clientes, fornecedores, produtos, unidade medida, categorias, formas de pagamento e realização de pedidos e fornece uma forma de gerenciamento mais eficaz e visa ser de fácil utilização. Além disso, as comissões sobre os pedidos são geradas automaticamente, tendo apenas a necessidade de emitir um relatório de comissões.

No texto formam apresentadas algumas telas do sistema, exemplificando o padrão de interface adotado, juntamente com o código de algumas operações realizadas. O código foi apresentado com o objetivo de mostrar alguns recursos das tecnologias utilizadas.

É notável que o desenvolvimento de sistemas tornou-se algo muito mais simples devido à grande quantidade de *frameworks* disponíveis, quando o desenvolvedor analisa e utiliza as ferramentas necessárias de acordo com o projeto é certo que há uma agilidade muito grande em termos de tempo de desenvolvimento.

Para este projeto foi utilizado o *framework* VRaptor na versão 3, o qual se mostra bastante produtivo para pequenos sistemas web.

Já para sistemas maiores existe a possibilidade de ele ser bem proveitoso. Contudo, existem outros *frameworks*, como o Spring, que agilizam de uma maneira mais organizada a persistência de dados, a configuração do projeto e a segurança dos dados. Nada impede que as duas tecnologias sejam utilizadas no mesmo projeto. Para isso é necessário o analista ou desenvolvedor analisar o problema, verificar quais as possibilidades de resolução e então decidir qual a melhor tecnologia ou combinação de tecnologias a ser utilizada e para resolver o problema da forma mais flexível, fácil, segura e com a menor demanda de tempo possível.

Durante o desenvolvimento do trabalho surgiram algumas dificuldades, tais como elicitar os requisitos do sistema com o usuário. Ele tinha dificuldade de repassar o que realmente precisava. Outra dificuldade foi em relação ao relatório de comissões, isso porque a comissão é realizada pela soma do percentual sobre fornecedor e não sobre o produto. Por exemplo: Fornecedor 1 paga 2% independente de produto e quantidade de venda. Então optou-se por adicionar um campo no cadastro do fornecedor no qual é informado o valor percentual da comissão sobre a venda. Uma dificuldade na implementação esteve relacionada

à geração do gráfico de produtos mais vendidos pela falta de experiência no uso das tecnologias.

Como trabalhos futuros pretende-se aprimorar mais o sistema de forma que consiga fazer uma versão *standard e* posterior customizar para outros clientes de acordo com sua necessidade, mas sempre tendo como base a versão *standard*. Com base nissoimplementar a nova versão utilizando novas tecnologias separando totalmente o *front-end* do *back-end*, implantação do sistema na nuvem, utilizar micro serviços e ser totalmente RestFull para posteriormente consumir em uma plataforma *mobile* sem ter que alterar a estrutura do projeto.

REFERÊNCIAS

BRASSCOM. Disponível em: <http://www.brasscom.org.br/>. Acesso em: 01 mar. 2016.

DEITEL, Harvey M.; DEITEL, Paul J. **Java como programar**. Porto Alegre: Bookman, 2003. Disponível em: <http://www.microinf.com.br/downloads/Java%20Como%20Programar%20-%20Deitel%20-%204a%20ed_pt-br.pdf>. Acesso em: 27 dez. 2016.

ECLIPSE. **EclipseNeon**. Disponível em: <https://eclipse.org/>. Acesso em: 27 dez. 2016.

IVY. **Apache Ivy**. Disponível em: <http://ant.apache.org/ivy/>. Acesso em: 27 dez. 2016.

POSTGRESQL. **Postgresql**. Disponível em: <https://www.postgresql.org/about/>. Acesso em: 27 dez. 2016.

SQL MAGAZINE. **SQL developer**. Disponível em: <http://www.sqlmagazine.com.br/artigos/postgre/01_Caracteristicas.asp>. Acesso em: 27 dez. 2016.

VRAPTOR. **VRaptor 3**. Disponível em: <http://vraptor3.vraptor.org/pt/>. Acesso em: 27 dez. 2016.