INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SUL-RIO-GRANDENSE - IFSUL, *CAMPUS* PASSO FUNDO
CURSO DE TECNOLOGIA EM SISTEMAS PARA INTERNET

BRIANDO MANUEL ALMADA BETTENCOURT

PLANEJAMENTO E DESENVOLVIMENTO DE UM SISTEMA DE GERENCIAMENTO DE OFICINAS MECANICAS

Prof. Wilian Bouviér

PASSO FUNDO, 2015

BRIANDO MANUEL ALMADA BETTENCOURT

PLANEJAMENTO E DESENVOLVIMENTO DE UM SISTEMA DE GERENCIAMENTO DE OFICINAS MECÂNICAS

Monografia apresentada ao Curso de Tecnologia em Sistemas para Internet do Instituto Federal Sul-Rio-Grandense, *Campus* Passo Fundo, como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet.

Orientador: **Prof. Wilian Bouviér**

PASSO FUNDO, 2015

BRIANDO MANUEL ALMADA BETTENCOURT

PLANEJAMENTO E DESENVOLVIMENTO DE UM SISTEMA DE GERENCIAMENTO DE OFICINAS MECÂNICAS

Trabalho de Conclusão de Curso aprovado em ____/____/____ como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet

Banca Examinadora:

_____

Prof. Wilian Bouviér

_____

Prof. Me. André Fernando Rollwagen

_____

Prof.ª. Carmen Vera Scorsatto

_____

Coordenação do Curso

PASSO FUNDO, 2015

DEDICATÓRIA

A mim, pela persistência, a minha noiva, pela paciência e aos meus amigos pela compreensão, apoio e o estímulo em todos os momentos.

AGRADECIMENTOS

Agradecimentos também a toda equipe docente e administrativa do IFSUL, em especial a meu orientador, professor Wilian Bouviér, de quem recebi ótimas orientações e sugestões, e a todos aqueles a qual recebi a honra de ter seus ensinamentos. Agradeço também a todos os amigos e companheiros de percurso ou não, os quais vencemos cada desafio e, mesmo não conseguido concluir o curso juntos, a amizade concretizada permanecerá. E por fim, agradeço a todos que de alguma forma contribuíram para a realização deste trabalho

EPÍGRAFE

“Primeiro aprenda ciência da computação e toda a teoria.
Depois desenvolva um estilo de programação.
E aí esqueça tudo e apenas ‘hackeie’."
George Carrette

RESUMO

Este trabalho apresenta planejamento e desenvolvimento de um sistema de gerenciamento de oficinas mecânicas, com o uso da linguagem de programação Java, banco de dados Postgresql, frameworks JavaServer Faces, Primefaces, Hibernate e demais ferramentas necessárias para o desenvolvimento do sistema. O sistema desenvolvido com base nas tecnologias estudadas consiste em uma ferramenta que auxilia o funcionamento de uma oficina mecânica e seus funcionários no atendimento a clientes, realização de serviços e controle de seu estoque de peças.

Palavras-chave: Oficina mecânica; Java; Postgresql; JavaServer Faces; Primefaces.

ABSTRACT

This paper presents design and development of a management garages system using the Java programming language, PostgreSQL database, frameworks JavaServer Faces, Primefaces, Hibernate and other tools needed to develop the system. The system developed based on the technologies studied consists of a tool that helps the operation of a machine shop and its employees in customer service, fulfillment services and control of your parts inventory.

Keywords: mechanical workshop; Java; Postgresql; JavaServer Faces; Primefaces.

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

ANFAVEA - Associação Nacional dos Fabricantes de Veículos Automotores, p.13

CRUD - *Create, Read, Update* e *Drop*, p.49

DAO - *Data Access Object*, p.50

ER – Entidade-Relacionamento, p.24

HD - *Hard Drive*, p.12

IBGE - Instituto Brasileiro de Geografia e Estatística, p.13

IDE - *Integrated Development Environment*, p.43

IEEE - *Institute of Electrical and Electronics Engineers*, p.16

IFSUL – Instituto Federal Sul-rio-grandense, p. 15

IU - Interface de Usuário, p.29

JCP - Java *Community Process*, p.29

JSF - JavaServer Faces, p.29

MPEs - Micros e Pequenas Empresas, p.12

OS - Ordem de Serviço, p.32

PC - *Personal Computer*, p.12

SGBDR - Sistemas de Gerenciamento de Banco de Dados Relacionais, p.25

TI – Tecnologia da Informação, p.12

TSPI – Tecnologia em Sistemas Para a Internet, p.15

UML - *Unified Modeling Language*, p.22

SUMÁRIO

1 INTRODUÇÃO

Já sabemos que o uso de computadores é indispensável em todas as áreas de atuação do ser humano, sendo elas comerciais, industriais, governamentais ou residenciais. O que está se tornando indispensável é o uso da internet, nenhum ramo acima descrito funciona bem sem seu uso.

A necessidade de consultar informações, o uso de e-mail para comunicação entre clientes e fornecedores e até mesmo o uso da nota fiscal eletrônica juntamente com a estabilidade e velocidade da internet atual, torna praticamente obrigatório o seu uso.

Este pesquisador vem trabalhando a muitos anos no setor de manutenção na área de Tecnologia da Informação (TI), sempre encontrando problemas com mau funcionamento de computadores em diversos ramos de atividade do setor comercial. Os problemas com os computadores vão desde uma queda de energia até contaminação por vírus, incluindo também o mau funcionamento da máquina ou do Hard Drive (HD). Devido a necessidade de uma limpeza ou até mesmo da reinstalação do sistema operacional, o cliente fica um ou mais dias sem o seu computador pessoal (*Personal Computer* – PC), observando que em alguns casos existe a perda de dados.

Partindo então do que foi descrito acima, decidiu-se de que se no lugar de um sistema instalado no PC fosse feito um sistema na web, não haveria a necessidade de ficar sem o controle de sua empresa por um ou mais dias e também não haveria a perda de dados, pois tudo estaria salvo na nuvem, ou seja, na internet e poderia ser acessado por qualquer computador, notebook e, dependendo do sistema desenvolvido, por smartfones e tablets.

Neste trabalho iremos desenvolver um sistema de gerenciamento para a oficina mecânica Auto Mecânica Pedrinho[1], utilizando a linguagem Java e o banco de dados postgresql, juntamente com os frameworks javaserver faces e primefaces.

1.1 MOTIVAÇÃO

De acordo com o Portal Brasil (2013), do governo federal, as Micro e Pequenas Empresas (MPEs) representam 20% do Produto Interno Bruto brasileiro, são responsáveis por 60% dos 94 milhões de empregos no país e constituem 99% dos 6 milhões de estabelecimentos formais existentes no país.

[1] Auto mecânica Pedrinho está situada na cidade de Passo Fundo RS e está em atividade a mais de 40 anos

Esse mercado está carente de um sistema de Gerenciamento Operacional de baixo custo e funcional. Um exemplo de MPEs são as Oficinas Mecânicas.

O mercado automotivo vem crescendo consideravelmente, dados da Associação Nacional dos Fabricantes de Veículos Automotores (ANFAVEA) relatam que em abril de 2013 a produção total de veículos foi de 340,9 mil unidades.

Apenas na cidade de Passo Fundo – RS, de acordo com dados de 2010 do Instituto Brasileiro de Geografia e Estatística (IBGE), existem 66.916 automóveis, um carro para 2,7 habitantes. (IBGE 2013). Segundo o site Audamec, existem no Rio Grande do Sul 10.356 oficinas e no Brasil um total de 100.192 oficinas (AUDAMEC, 2013).

Através desses dados, constata-se a viabilidade do desenvolvimento de softwares específicos para a área de reparação e manutenção desta frota que aumenta dia a dia.

O número de automóveis nas ruas está cada vez maior, fazendo com que surjam, cada vez mais, novas oficinas mecânicas. Para acompanhar este crescimento, faz-se necessário um sistema para gerenciamento, controle de serviços prestados e de vendas realizadas, que atenda demandas específicas de uma oficina mecânica e tenha um preço acessível.

Hoje tona-se uma necessidade, no gerenciamento de estabelecimentos de qualquer natureza, a utilização de algum sistema informatizado.

1.2 OBJETIVOS

Este trabalho tem por objetivo transcrever todo o procedimento necessário para a criação de um software e pôr em pratica o conhecimento adquirido no curso de Tecnologia em Sistemas Para a Internet (TSPI) do Instituto Federal Sul Rio-grandense (IFSul).

1.2.1 Objetivo Geral

O trabalho tem por objetivo de planejar e desenvolver um sistema gerenciador para empresas do ramo de reparação de veículos.

1.2.2 Objetivos específicos

- Pesquisar o funcionamento de uma oficina mecânica, observando as necessidades operacionais básicas.

- Conhecer o funcionamento de um sistema de controle de estoque.

- Fazer o levantamento de requisitos completo com diagramas de classes e esquemas conceituais de banco de dados.

- Implementar um protótipo do sistema que irá gerenciar os dados importantes para a empresa, possibilitando a manipulação destes dados através de inserções, alterações e exclusões.

2 REFERENCIAL TEÓRICO

Neste capítulo é apresentado o estudo sobre a análise e levantamento de requisitos, as linguagens de programação, em especial a linguagem Java, frameworks JavaServer Faces e Primefaces e o conceito de Banco de Dados Relacional.

Estes conceitos são relevantes para o desenvolvimento do sistema e auxiliam no bom andamento de cada etapa.

2.1 LEVANTAMENTO DE REQUISITOS

Levantamento de requisitos é parte mais importante na construção de um sistema de software. Nenhuma outra parte do trabalho inutiliza o sistema se for feita de forma errada. Todo o sistema é baseado nos requisitos levantados, que correspondem à listagem de todas as coisas que o sistema deve fazer, ou seja, as funcionalidades que o sistema deve possuir. Os requisitos levantados são divididos em requisitos funcionais e requisitos não funcionais.

Os requisitos funcionais abordam o quê o sistema deve fazer, como deve se comportar a certas entradas e às mais variadas situações.

Requisitos não-funcionais estão relacionados ao uso da aplicação em termos de desempenho, usabilidade, confiabilidade, segurança, disponibilidade e tecnologias envolvidas. Não é preciso o cliente dizer sobre eles, pois são características mínimas de um software de qualidade, às quais o desenvolvedor deve estar atento.

Tabela 1 - Levantamento de Requisitos

Procedimento	Descrição
Entrevistas não estruturadas	Informal ou sem agenda pré-definida
Entrevistas estruturadas	Com uma agenda pré-definida
Observação do comportamento	Observar os usuários em seu ambiente de trabalho
Aprendizagem com o usuário	Analisa e discute com o usuário a maneira como é feito o trabalho
Prototipagem	Desenvolvimento de um modelo que simulará o sistema real
Brainstorming	Reunião com várias pessoas onde todos discutem um tema central

Análise de textos	O usuário descreve as necessidades textualmente
Reutilização de requisitos	Reaproveitamento de padrões ou requisitos de outros sistemas

Fonte: Macedo (2015)

2.1.1 Classificação dos requisitos

Os requisitos podem ser classificados de várias formas, a finalidade desta classificação é melhor compreender a relação entre objetos, tarefas e as próprias funções do sistema. Uma forma bastante aceitável entre analista é que a classificação seja entre requisitos funcionais e não-funcionais.

2.1.2 Requisitos Funcionais

Os requisitos funcionais são aqueles que fazem parte do sistema, como um relatório específico, um campo a mais em um cadastro, etc. Eles normalmente têm a finalidade de agregar valor ao usuário ou facilitar o trabalho que ele desenvolve. Requisitos funcionais serão implementados no próprio sistema e da junção desses requisitos o corpo do sistema será montado. Correspondem à listagem de todas as coisas que o sistema deve fazer, ou seja, as funcionalidades que o sistema deve possuir.

Requisitos funcionais evidentes são efetuados com conhecimento do usuário.

Requisitos funcionais ocultos são efetuados pelo sistema sem o conhecimento explícito do usuário.

2.1.3 Requisitos não funcionais

São atributos de qualidade ou restrições que se coloca sobre como o sistema deve realizar seus requisitos funcionais.

Definem os atributos do sistema enquanto ele executa seu trabalho.

São aqueles relacionados ao ambiente onde o sistema está inserido

2.1.4 Classificação dos Requisitos Não Funcionais

Esses requisitos declaram características de qualidade que o sistema deve possuir e que estão relacionadas às suas funcionalidades. Temos algumas divisões dentro desse tipo de requisitos.

2.1.4.1 Confiabilidade

São medidas quantitativas da confiabilidade do sistema, como por exemplo, o tempo médio entre falhas e recuperação de falhas.

2.1.4.2 Portabilidade

Trata-se da facilidade de migrar o sistema para outras plataformas. Que devemos dar uma atenção, para que o sistema rode em qualquer lugar.

2.1.4.3 Segurança

São descritas as particularidades sobre acessos ao sistema, segurança com o uso de login, restringir acesso de algumas pessoas, entre outros.

2.1.4.4 Usabilidade

São descritos os requisitos que se relacionam ou afetam a usabilidade do sistema. Coisas relacionadas à facilidade de uso, sobre a necessidade de treinamentos para os usuários.
A análise e refinamento dos requisitos é uma Etapa importantíssima do processo de documentação e projeto.

Requisitos obrigatórios e desejados

Indicam uma prioridade no desenvolvimento.

- Obrigatórios devem ser feitos;
- Desejáveis, se sobrar tempo e recurso;

Por exemplo:

- A interface Web deve ser obrigatória;
- Acesso através de celular deve ser desejável;

Requisitos permanentes e transitórios

Pode ou não mudar? Decisão do analista.

Decisão relativa a tempo e custo de desenvolvimento.

Permanente é mais fácil e rápido implementar, porém mais caro de manter se o sistema mudar.

Transitório é mais caro e complexo desenvolver, porém mais barato e rápido de manter.

Os requisitos transitórios devem ser suscetíveis a acomodar mudanças.

Requisitos mais importantes devem ser transitórios, espera-se que possam mudar no futuro e essa mudança causa maior impacto no sistema. (Gurgel, 2013)

2.2 UML

A Unified Modeling Language (UML) é um modelo de linguagem para modelagem de dados orientado a objetos, usada para especificar, construir, visualizar e documentar um sistema de software.

A UML permite que desenvolvedores visualizem os produtos de seus trabalhos em diagramas padronizados.

É importante distinguir entre um modelo UML e um diagrama (ou conjunto de diagramas) de UML. O último é uma representação gráfica da informação do primeiro, mas o primeiro pode existir independentemente. A UML é um modo de padronizar as formas de modelagem.

2.2.1 Diagramas

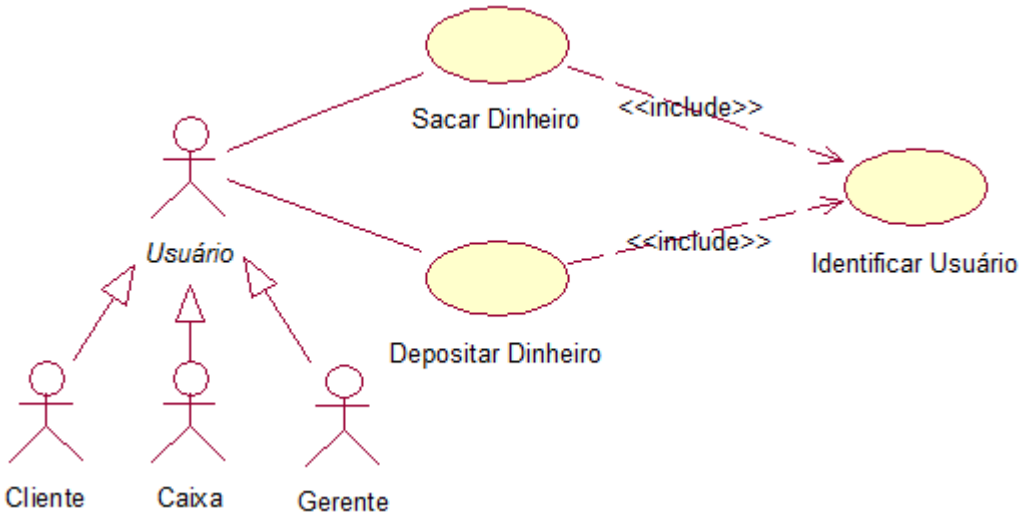
Casos de Uso:

Diagrama mais geral da UML;

Usado geralmente nas fases de Levantamento e Análise de Requisito do Sistema;

Mostra como o sistema irá se comportar.

Figura 1 - Exemplo de caso de uso



Fonte: Devmedia (2015)

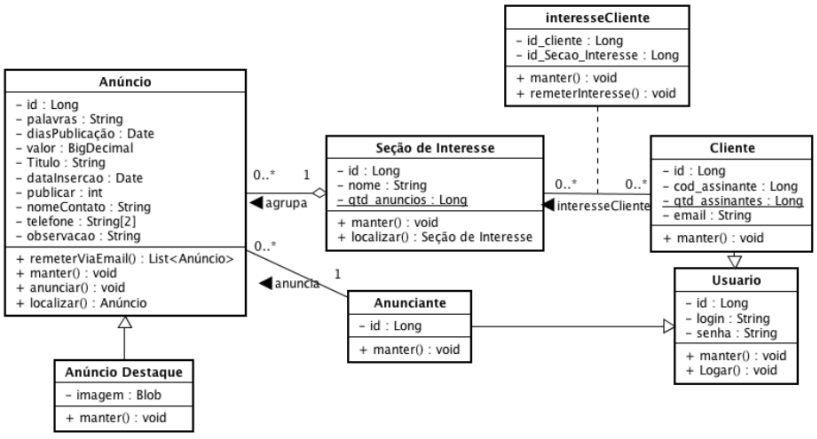
Na figura 1, foi criado um ator chamado Usuário que representa os usuários (atores Cliente, Caixa e Gerente) de um sistema bancário. O ator Usuário irá executar os casos de uso Sacar Dinheiro e Depositar Dinheiro. (Devmedia, 2015).

Diagrama de Classes

É o diagrama mais utilizado da UML, ele serve de apoio para a maioria dos outros diagramas. O diagrama de classes define a estrutura de classes do sistema e estabelece como as classes se relacionam;

É o mais importante e o mais utilizado diagrama da UML.

Figura 2 - Exemplo de diagrama de classes



Fonte: Rocha (2015)

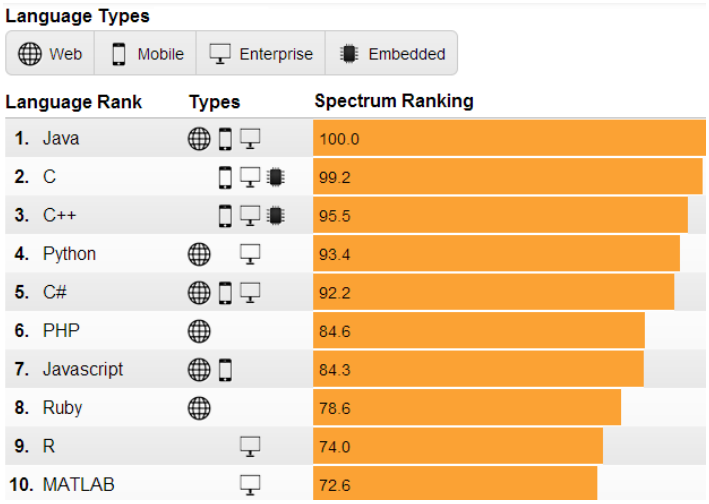
A figura 2 representa como o diagrama de classes permite a visualização das classes que comporão o sistema com seus respectivos atributos e métodos, bem como os relacionamentos entre as classes.

2.3 JAVA

Existem várias linguagens de programação para a criação de sistemas e dentre elas se destaca a linguagem Java, de acordo com o ranking do Instituto de Engenheiros Elétricos e Eletrônicos (Institute of Electrical and Electronics Engineers - IEEE) ela lidera o ranking das linguagens de programação sendo a que vem sendo mais utilizada no mundo. (Canaltech, 2015)

O Java aparece em primeiro lugar no ranking, atingindo 100 pontos, mas é seguido de perto pela linguagem C com 99,2 pontos e em terceiro lugar está o C++ com 95,5. Outras linguagens de programação web também muito utilizadas, PHP e Ruby, só aparecem na sexta e oitava posições, respectivamente, como mostra a figura 3.

Figura 3 - Ranking das Linguagens



Fonte: Canaltech 2015

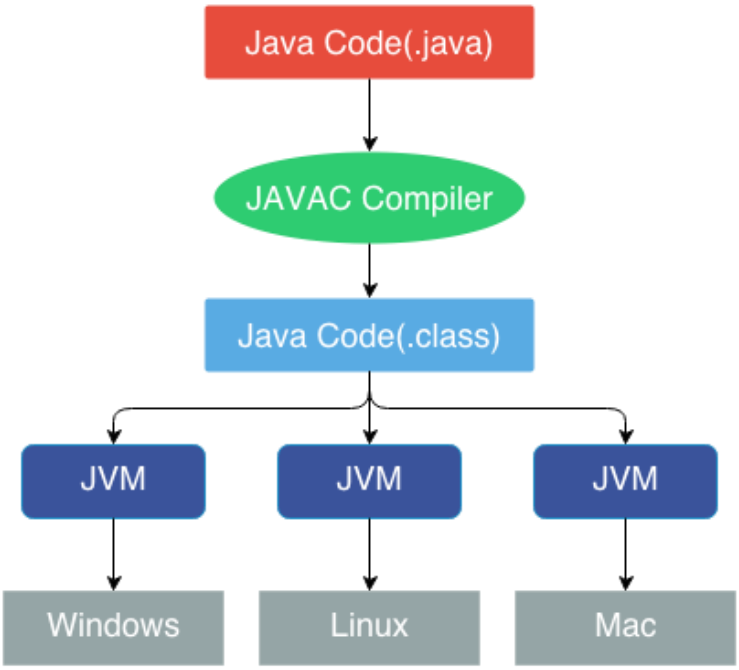
Lançada em 1995 pela Sun Microsystems e atualmente mantida pela Oracle, é uma linguagem de programação multiplataforma que, de acordo com a Oracle, é executada em mais de 850 milhões de computadores pessoais e em bilhões de dispositivos em todo o mundo, inclusive telefones celulares e demais dispositivos móveis.

De acordo com Santos Júnior (2006, p. 3): “A linguagem JAVA é multiplataforma, esta afirmação reporta ao fato de que um programa escrito na linguagem JAVA pode ser executado em qualquer plataforma(sistema operacional) sem necessidade de alterações no código-fonte. ”

Como diz o slogan da Sun MicroSystems para exemplificar os benefícios multiplataforma da Linguagem Java:(Javafree, 2009) “Escreva uma vez, execute em qualquer lugar”.

A figura 4 apresenta um modelo de Java Virtual Machine.

Figura 4 - Java Virtual Machine



Fonte: Codeuse (2015)

2.4 JavaServer Faces

O JavaServer Faces (JSF) foi criada através do Java Community Process (JCP), uma entidade formada pelas mais importantes empresas de tecnologia do mundo.

O JSF é um framework de interface de usuário (IU) para aplicações Java Web.

Facilita a construção de uma IU usando um conjunto de componentes de IU reutilizáveis.

Oferece um modelo simples para conectar os eventos gerados pelo cliente ao código da aplicação do servidor.

Permite personalizar os componentes de UI para que sejam facilmente construídos e reutilizados.

Segundo Faria:

> ...é um framework web baseado em Java que tem como objetivo simplificar o desenvolvimento de interfaces (telas) de sistemas para a web, através de um modelo de componentes reutilizáveis. Java EE 7 com JSF, PrimeFaces e CDI (2015, pág. 14).

2.5 Primefaces

PrimeFaces é uma biblioteca de componentes ricos em JavaServer Faces. A suíte de componentes inclui diversos campos de entrada, botões, tabelas de dados, árvores, gráficos, diálogos, etc. Os componentes do PrimeFaces possuem funcionalidade de Ajax integrado, baseado na API de Ajax do JSF (Faria, 2015).

2.6 BANCO DE DADOS RELACIONAL

Segundo Ricarte (2002) um banco de dados relacional organiza seus dados em relações. Cada relação pode ser entendida como uma tabela, na qual, cada coluna corresponde aos atributos da relação e cada linha corresponde aos elementos da relação.

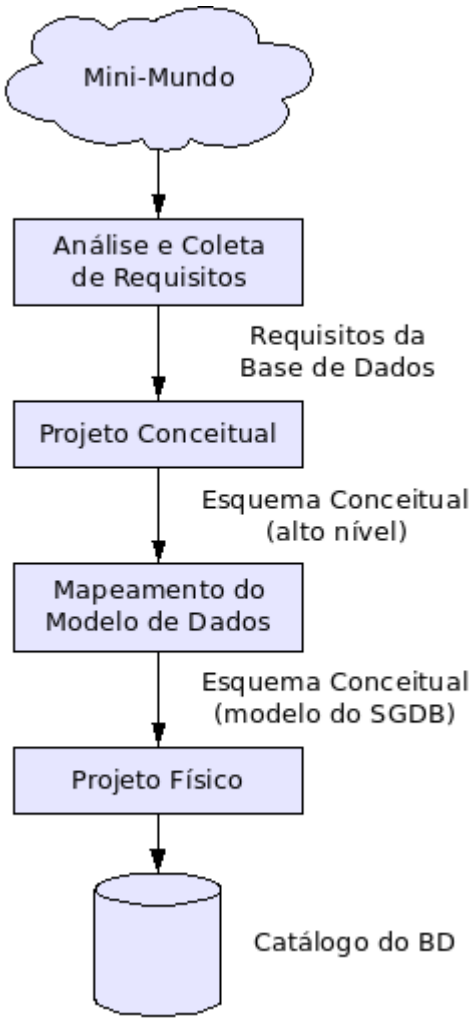
Projetar o banco de dados adequadamente aumenta a confiabilidade da aplicação. De acordo com Oliveira (2002):

> O projeto do banco de dados está para o sistema da mesma forma que a estrutura do prédio está para o edifício. Se não for dada a devida atenção ao desenho do banco de dados, pode-se comprometer todo o desenvolvimento do sistema.

Um banco de dados relacional é um mecanismo de armazenamento que permite a persistência de dados. Para isso, existem os Sistemas de Gerenciamento de Banco de Dados Relacionais (SGBDR). Um SGBDR é um software que controla armazenamento, recuperação, exclusão, segurança e integridade dos dados. O uso mais comum de SGBDRs é para implementar funcionalidades simples do tipo Criação, Leitura, Atualização e Remoção (CRUD).

Essa visão de dados organizados em tabelas oferece um conceito simples e familiar para a estruturação dos dados, sendo um dos motivos do sucesso de sistemas relacionais de dados.

Figura 5 - Fases projeto de um banco de dados



Fonte: Meira (1997)

A figura 5 faz uma descrição simplificada do processo de projeto de um banco de dados.

2.6.1 Modelo entidade-relacionamento

Modelagem de dados conceitual ou modelo Entidade-Relacionamento (ER) é o início para o desenvolver e projetar o banco de dados do sistema. É a descrição da estrutura de um banco de dados, independente de banco será usado.

A análise e os levantamentos de requisitos e a modelagem de dados devem ser muito bem feitos, muito bem detalhados, por que depois de pronto é muito mais difícil de corrigir alguma falha desta etapa.

É definido:

- Quais as tabelas (entidades) teremos;

- Quantas tabelas teremos;
- Quais os atributos;
- Quais as chaves primarias;
- Quais os relacionamentos;
- Quais as chaves estrangeiras.

Componentes do modelo ER

- Entidades – tabelas onde serão salvos os dados do sistema
- Atributos – dados das tabelas
- Relacionamentos – associações entre os atributos, ligações entre as tabelas

As chaves estrangeiras são definidas pelos relacionamentos

Modelo Conceitual

- Relacionamento entre as classes.

Modelo lógico

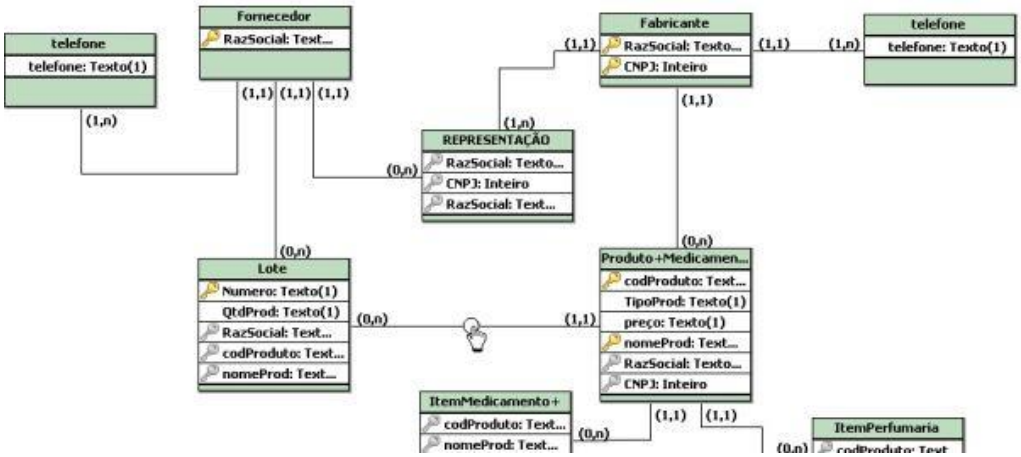
- Representação das classes, atributos e campos.

Modelo físico

- Corresponde às restrições da tecnologia escolhidas no modelo lógico e sua implementação em um ambiente operacional com resultado factível com hardware e software.

A figura 6 é um exemplo de modelo entidade-relacionamento de um sistema farmacêutico.

Figura 6 - Modelo ER logico



Fonte: Baixaki (2015)

Algumas convenções para a padronização de nomes de tabelas e colunas de banco de dados (Tarifa, 2015).

Tabela 2 - Normatização

Número de palavras	Norma	Exemplo
1 Palavra	Utilizar as 3 primeiras letras da palavra	Nome: Pessoas Abreviatura: PES
2 Palavras	Utilizar as 2 primeiras letras da primeira palavra e a 1ª letra da segunda palavra	Nome: Pessoas Físicas Abreviatura: PEF
3 ou mais Palavras	Utilizar a 1ª letra das duas primeiras palavras e a 1ª letra da última palavra	Nome: Tipos Contrato Prestadores Abreviatura: TCP

Fonte: Elaborado pelo autor (2015)

Nomes de Tabelas:

{abreviatura do módulo}_{abreviatura da tabela}_{nome da tabela}

Tabelas Auxiliares n/n:

{abreviatura tabela principal}_{nome tabela principal}_{nome tabela auxiliar}

Nomes de Colunas:

{abreviatura da tabela}_{nome da coluna}

Chaves Primárias:

{abreviatura da tabela}_{nome da tabela}_pk

Chaves Estrangeiras:

{abreviatura da tabela}_{nome da chave primária referenciada}_fk

Relações de Pertinência:

{abreviatura da tabela}_{nome do atributo}_{nome da entidade}

O nome da tabela deve ser sempre no plural.

Para escrever uma tabela durante o levantamento de requisitos, pode-se usar o modelo abaixo:

Tabela 3 - Modelo de tabela banco

Nome do campo	Tipo de dados	Restrição de integridade e comentários

Fonte: Elaborado pelo autor (2015)

Tabela 4 - Exemplo Banco de Dados

Usuarios		
usu_id_pk	Serial	Not null
usu_nom	Varchar, 50	Not null
usu_user	Varchar, 20	Not null
usu_senha	Varchar, 30	Not null
usu_dat_cad	Date	Not null
usu_ult_ace	Date	Not null

Fonte: Elaborado pelo autor (2015)

3 METODOLOGIA

O desenvolvimento do sistema está dividido em:

-Planejamento

-Construção

-Implantação

-Avaliação e Manutenção

Neste trabalho será explicado as 4 etapas, mas, será mostrado apenas o planejamento e a construção do sistema de gerenciamento de oficinas mecânicas

3.1 PLANEJAMENTO

É necessário realizar um levantamento dos requisitos do sistema junto com o cliente, avaliar os problemas e as necessidades do cliente e documentar essas informações que serão fundamentais o desenvolvimento do projeto.

A fase de planejamento inclui a identificação do cliente, o levantamento dos requisitos, a modelagem do sistema, o projeto do banco de dados o levantamento dos recursos e custos, avaliar a viabilidade e a elaboração do cronograma inicial.

3.2 CONSTRUÇÃO

Na construção do sistema é necessário definir níveis de segurança da aplicação, versionamento do código fonte, construção da aplicação em si, criação de teste, promover carga de testes extremos e avaliação do produto final. Fazer um levantamento de erros e correções.

3.2.1 Testes

Diversas atividades de testes são executadas a fim de se validar o produto de software, testando cada funcionalidade de cada módulo, levando em consideração a especificação feita na fase de projeto. Ao final dessa atividade, os diversos módulos do sistema são integrados, resultando no produto de software.

3.2.2 Documentação

Uma importante tarefa é a documentação do projeto interno do software para propósitos de futuras manutenções e aprimoramentos. As documentações mais importantes são das interfaces externas.

3.3 IMPLANTAÇÃO

Realizar a implantação do sistema junto ao cliente, prover treinamentos para os operadores do sistema. Verificar possíveis problemas e realizar correções da aplicação.

- Desenvolver um plano de implantação da aplicação;
- Pacote de Entrega ao Cliente;
- Diretrizes de entrega do produto;
- Treinamento.

3.4 AVALIAÇÃO E MANUTENÇÃO

Realizar uma avaliação do sistema junto com o cliente para garantir a qualidade do produto, realizar manutenções sempre que necessário para manter o bom funcionamento da aplicação.

É importante sempre seguirmos as etapas de desenvolvimento de sistemas, essas etapas vão garantir a entrega de um produto de qualidade condizente com a necessidade do cliente. Cada etapa tem seus próprios requisitos e suas metas que serão elaboradas pela equipe de desenvolvimento. Deve-se estar atento ao cronograma para cumprir as metas estabelecidas dentro do prazo. Sempre avaliar junto com o cliente o produto final.

4 PLANEJAMENTO E DESENVOLVIMENTO DO SISTEMA

4.1 LEVANTAMENTO DE REQUISITOS

4.1.1 Descrição do proposito do sistema

A oficina mecânica Auto Mecânica Pedrinho necessita de um sistema de informação para apoiar a realização de suas atividades principais, a saber: serviços de manutenção e reparação automotiva assim como a venda de autopeças. Para que essas atividades sejam apoiadas, é necessário controlar as informações acerca de serviços e estoque.

Devem ser fornecidas facilidades de consulta ao serviço de reparação de um determinado veículo, permitindo consultas a respeito do serviço e autopeças utilizadas.

O sistema devera controlar os serviços e o estoque de peças da oficina, gerando relatórios informando a quantidade de peças em estoque, além disso o sistema deve manter um histórico de serviços em um veículo e o histórico do cliente, como também, gerar relatórios de clientes em atraso

O sistema deverá relatar todos os serviços de todos os veículos que estão em serviço na oficina, listando os que foram feitos, os que estão em andamento e os que estão por fazer.

4.1.2 Descrição do minimundo

O cliente ao entrar na oficina é atendido pelo gerente, é feito um cadastro, caso não tenha, e logo após é aberta uma ordem de serviço com os defeitos que o veículo apresenta e demais serviços que devem ser realizados, como por exemplo, uma troca de óleo ou uma revisão periódica só então o veículo é encaminhado ou para o atendimento do mecânico ou para uma fila de espera com a ordem de serviço fixada no para-brisa. Em alguns casos o proprietário deve mostrar para o mecânico o que está acontecendo com o veículo, caso não haja mecânico disponível o gerente, que tem conhecimento de mecânica, sai com o veículo, juntamente com o proprietário, e anota numa prancheta o que o proprietário lhe mostra.

O atendimento pelo telefone requer uma atenção especial, quando há necessidade de buscar o veículo do cliente, este deve ser atendido o quanto antes possível, depois de recolhido o veículo ele vai para a fila de espera como os demais veículos.

Quando o mecânico seleciona o próximo veículo a ser reparado, lê a ordem de serviço e começa a reparação.

Durante o processo de reparação cada serviço é informado aos responsáveis pelo uso do sistema, que neste caso são dois atendentes que ficam no balcão e que fornecem as peças necessárias para o referido serviço. Durante a reparação podem surgir outros defeitos que são informados ao cliente e então são adicionados a ordem de serviço. Somente depois de realizado o conserto e feito um teste, o serviço é dado por encerrado e o mecânico passa para o próximo carro. O gerente da oficina comunica o proprietário ou responsável pelo veículo que o serviço está pronto. Somente depois de acertado o pagamento do serviço é que o veículo é entregue e o processo é encerrado finalizando assim a ordem de serviço.

A maioria das peças utilizadas nos serviços são da própria oficina, que mantem uma pequena seção de peças, caso não possua as peças então elas são compradas em outras lojas de autopeças da cidade. Em alguns casos as peças são compradas por encomenda ou o proprietário do veículo traz as peças.

Tendo uma descrição do funcionamento nota-se que a oficina mecânica Auto Mecânica Pedrinho necessita de um sistema de informação para gerenciar o atendimento aos seus clientes. O negócio principal da oficina é a manutenção e a reparação de veículos.

Clientes levam veículos à oficina mecânica para serem consertados, para passarem por revisões periódicas que incluem por exemplo as trocas de correias ou troca de óleo. O cliente pode agendar um concerto ou uma revisão. Sobre um cliente, deseja-se saber: o nome completo, endereço completo, telefone celular e residencial e cpf ou cnpj, caso o cliente seja pessoa jurídica.

Cada veículo é recebido pelo gerente da oficina que identifica os serviços a serem executados, preenche uma ordem de serviço (OS) e prevê uma data aproximada de entrega. Os veículos possuem código, placa, descrição (Fabricante, Modelo e ano de fabricação), nome do responsável e o atual proprietário. Um veículo pode trocar de proprietário.

A partir da OS, calcula-se o valor de cada serviço e o valor de cada peça necessária à execução do serviço.

O cliente necessita autorizar a execução dos serviços.

Cada OS possui um número, uma data de emissão, um valor parcial e uma data aproximada para conclusão dos trabalhos, o(s) mecânico(s), assim como serviços e peças. Uma OS pode envolver várias peças e vários serviços.

Sobre um serviço, deseja-se saber: a descrição e o valor.

A descrição é apenas o que foi realizado, por exemplo: troca de óleo do motor.

Os valores dos serviços são cadastrados por tipo de serviço ou por tempo de realização do serviço, que é por tempo estimado[2], ficando a cargo do gerente da oficina. A hora de serviço terá 3 classificações, “a” valor maior, “b” valor intermediário e “c” de valor mais baixo, que será utilizado de acordo com a dificuldade, nível de conhecimento ou equipamento utilizado para a reparação do veículo.

Sobre as peças o gerente da oficina necessita manter um cadastro de todos os itens que são comprados, catalogando eles por código, código do fabricante, descrição do item (Grupo e subgrupo), tipo do item, fabricante, aplicação (montadora, modelo, motor, combustível e ano do veículo), observações e quantidade, como demonstra a tabela 5.

Tabela 5 - Cadastro de peças

Cód.	Cod_fab	Descrição (grupo e sub-grupo)	Tipo	Fab_Item	Aplicação (montadora, modelo, combustível e ano	Obs.	Quant.
00001	000001	Amortecedor dianteiro	Turbo gás	Cofap	VW gol/saveiro/parati 1.6 98-02	S/AC	5
00002	000002	Óleo motor	Sintético	Mobil	Todos / álcool	DH	6
00003	000003	Pistão freio traseiro	Comum	Valeo	VW Santana 1.8 2.0 95-97		0

Fonte: Elaborado pelo autor (2015)

As peças e serviços poderão ser cadastradas previamente ou durante o preenchimento da ordem de serviço

Os pedidos de peças são feitos na visita de um vendedor ou quando termina o estoque, no caso da visita do vendedor deverá ser gerado um relatório do estoque para saber o que está em falta ou quase no fim, no caso de terminar o estoque o pedido é feito por e-mail ou telefone,

[2] O mecânico já sabe quanto tempo demora o serviço

diretamente da distribuidora. O sistema manterá um cadastro de fornecedores com seus contatos (vendedores)

Dos fornecedores deve-se saber o nome, razão social, CNPJ, vendedor, contato, telefone e e-mail.

Cada serviço será feito por um ou mais mecânicos e cada mecânico pode fazer um ou mais serviços.

Os funcionários são divididos em gerente (administrador), atendente e mecânico. Cada funcionário possui código, nome, endereço, telefone, tipo (admin, atendente ou mec.) e observações. Os funcionários poderão ser usuários do sistema e terão um nome de usuário, uma senha e controle de acesso com data dos acessos ao sistema.

O gerente é o usuário administrador do sistema, ele pode cadastrar cliente, cadastrar funcionário, cadastrar peças, cadastrar serviços, abrir ordem de serviço e cadastrar outros usuários do sistema.

O atendente só pode cadastrar usuário, cadastrar peças e abrir ordem de serviço. Não pode alterar valores de serviços e peças já registradas e não pode cadastrar funcionários e usuários.

Tomando por base o contexto do sistema, foram identificados os seguintes requisitos de usuário:

- Funcionalidade (finalidade do produto);
- Usabilidade (esforço para utilizar, aprender o produto);
- Confiabilidade (frequência de falhas, recuperabilidade);
- Eficiência (característica relacionada ao desempenho);
- Manutenibilidade (esforço necessário para modificar);
- Portabilidade (capacidade de transferir o produto para outros ambientes).

4.1.3 Requisitos Funcionais

RF1 – Clientes

O sistema deve permitir o cadastro, a consulta, a exclusão e a alteração de clientes.

O sistema deve cadastrar o cliente, cadastrando o nome, CPF, endereço, telefone e veiculo.

RF2 – Serviços, Produtos e Peças

O sistema deve permitir o cadastro, a consulta, a exclusão e a alteração de serviços, produtos ou peças.

RF3 – Funcionários

O sistema deve permitir o cadastro, a consulta, a exclusão e a alteração de funcionários.

O sistema deve cadastrar os funcionários, cadastrando nome, CPF, endereço, telefone e função.

As funções dos funcionários serão:

Administrador, gerente e atendente.

RF4 – Ordem de Serviço

O sistema deve permitir aos usuários a abertura, alteração e finalização de uma ordem de serviço.

O sistema deve permitir a inclusão, exclusão e alteração de peças, produtos e serviços em uma determinada OS.

RF5 – Venda Avulsa

O sistema deverá permitir a venda de produtos e peças sem a necessidade de abertura de uma OS.

RF6 – Relatórios

O sistema deverá gerar relatórios de OS em aberto, histórico de um veículo, clientes.

Os relatórios devem ter a opção de visualização em PDF e XLS.

RF7 – Veículos

O sistema deve permitir o cadastro, a consulta, a exclusão e alteração de veículos.

O sistema deve registrar a entrada e saída de veículos da oficina.

RF8 – Vendas

O sistema deve permitir a baixa automática do estoque quando da venda de um produto.

RF9 - Controle de acesso

A função só pode ser acessada por usuário com perfil de gerente ou administrador.

RF10 - Cadastros

Todas as funções relacionadas ao cadastro devem ser efetuadas em uma única tela.

RF11 - Funcionalidades

As funcionalidades do sistema são acessíveis aos usuários de acordo com seu nível de privilégio.

RF12 - Relatórios

Os Relatórios devem ter níveis de acesso.

RF13 - Funcionamento do Clientes

O cliente deve ser cadastrado sempre antes de se gerar uma ordem de serviço. Deverá constar Nome, Telefone, Endereço.

4.1.4 Requisitos Não-Funcionais

RNF1 - Sistema

Tempo de resposta desejável menor que 10 segundos.

RNF2 - Linguagem

Utilização da linguagem Java para Web.

RNF3 – Telas do sistema

A interface do sistema será agradável e objetiva ao usuário, as funcionalidades e informações deveram estar bem visíveis.

RNF4 - Alertas

As mensagens de alerta serão simples e explicativas sobre os erros gerados.

RNF5 – Banco de Dados

O Sistema usará um banco de dados relacional, garantindo a segurança dos dados. O banco de dados será o PostgreSQL.

RNF6 – Acesso

Todas as funcionalidades do sistema são acessíveis aos usuários de acordo com seu nível de privilégio no sistema. Isto é realizado através de um sistema de CPF/Senha.

O sistema só poderá ser acessado por usuários cadastrados no sistema.

O acesso ao sistema se dará pela informação de CPF e senha.

O administrador do sistema terá acesso à todas as áreas do sistema.

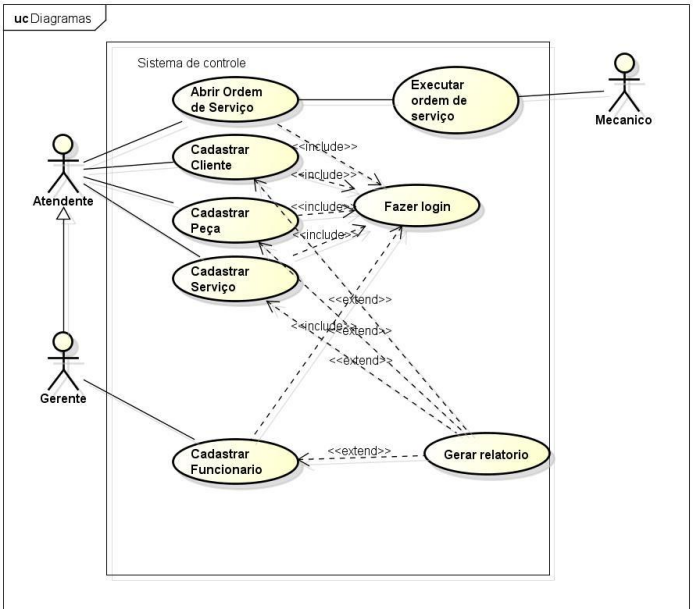
Os gerentes do sistema terão acesso à todas as áreas do sistema, exceto as áreas restritas ao desenvolvedor.

Os atendentes não terão acesso as áreas de cadastro de funcionários, produtos e serviços.

4.1.5 Diagrama de Caso de Uso

A figura 7 mostra o caso de uso geral do sistema.

Figura 7 - Diagrama de caso de uso



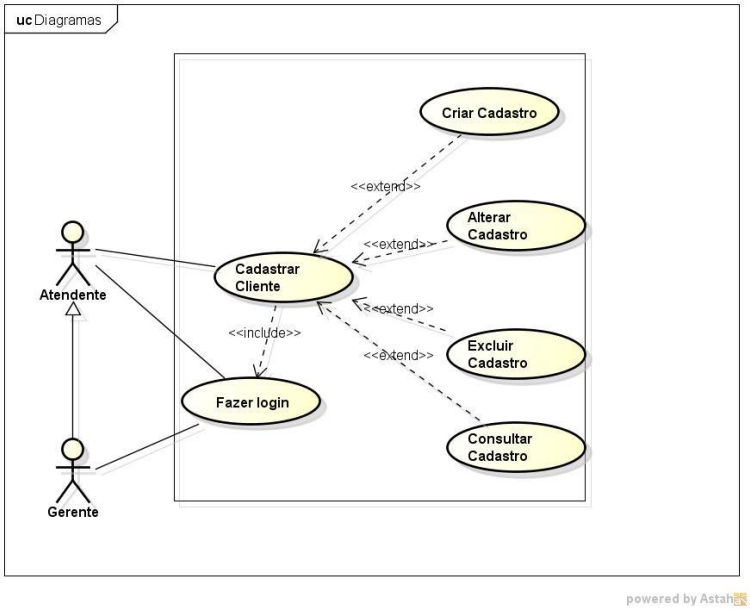
Fonte: Elaborado pelo autor (2015)

O ator gerente herda do ator atendente as funções de abrir ordem de serviço, cadastrar cliente cadastrar peça e cadastrar serviço. Somente o gerente pode cadastrar funcionário.

Ambos os atores necessitam fazer login.

A figura 8 mostra o caso de uso do cadastro de cliente

Figura 8 - Caso de uso Cadastro cliente

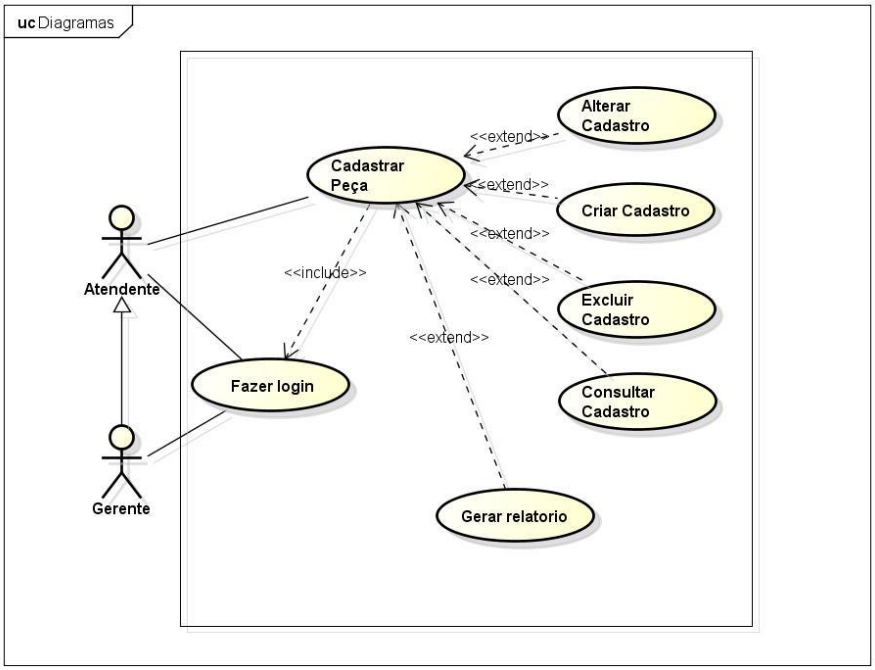


Fonte: Elaborado pelo autor (2015)

O gerente herda do atendente a função de cadastrar cliente, que inclui criar cadastro, alterar cadastro, excluir cadastro e consultar cadastro.

A figura 9 exemplifica o caso de uso do cadastro de peças

Figura 9 - Caso de uso Cadastro de peças



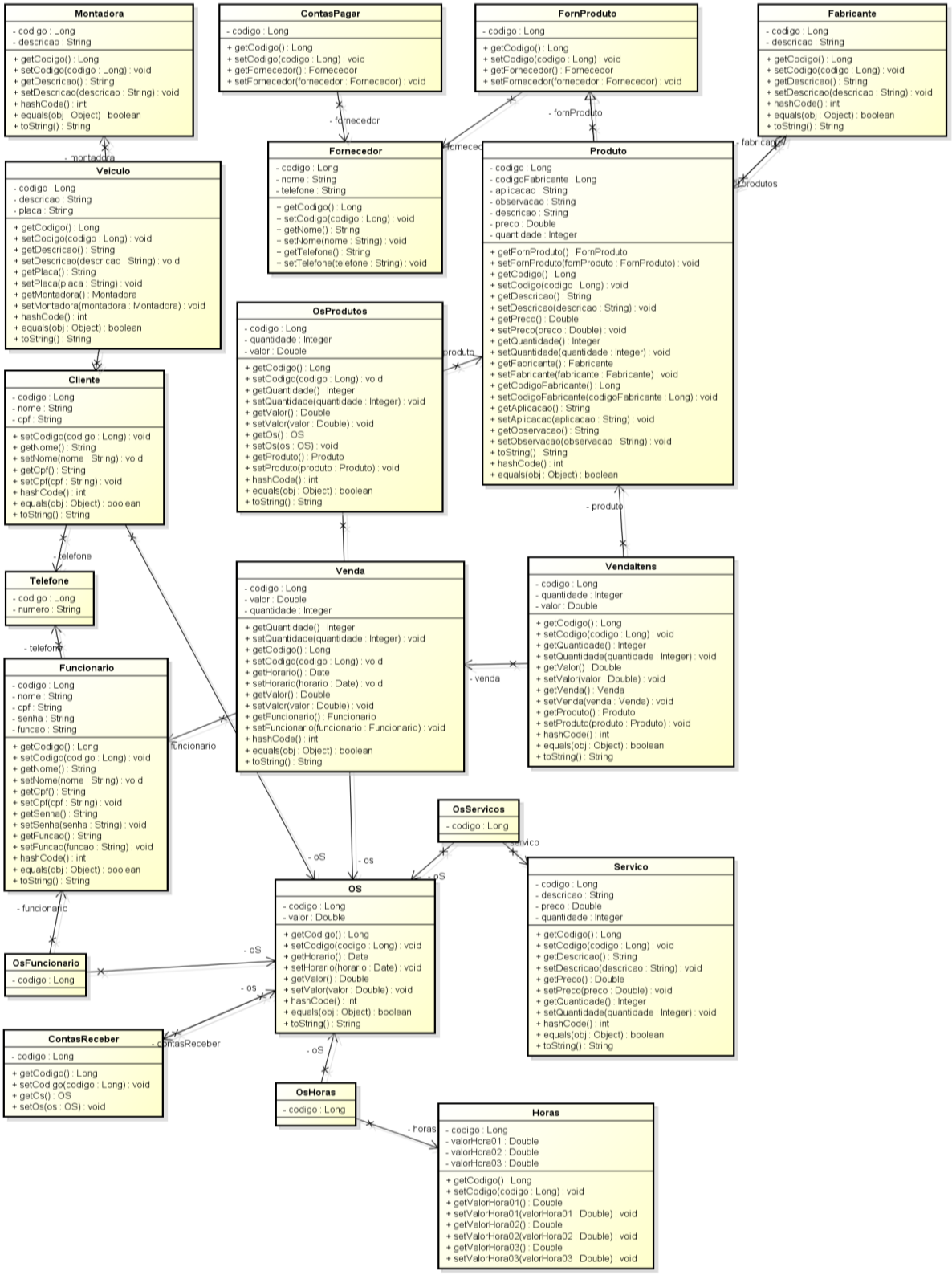
Fonte: Elaborado pelo autor (2015)

O gerente herda do atendente a função de cadastrar peça, que inclui criar cadastro, alterar cadastro, excluir cadastro, consultar cadastro e gerar relatório.

4.1.6 Diagrama de classes

A figura 10 mostra o diagrama de classes completo do sistema.

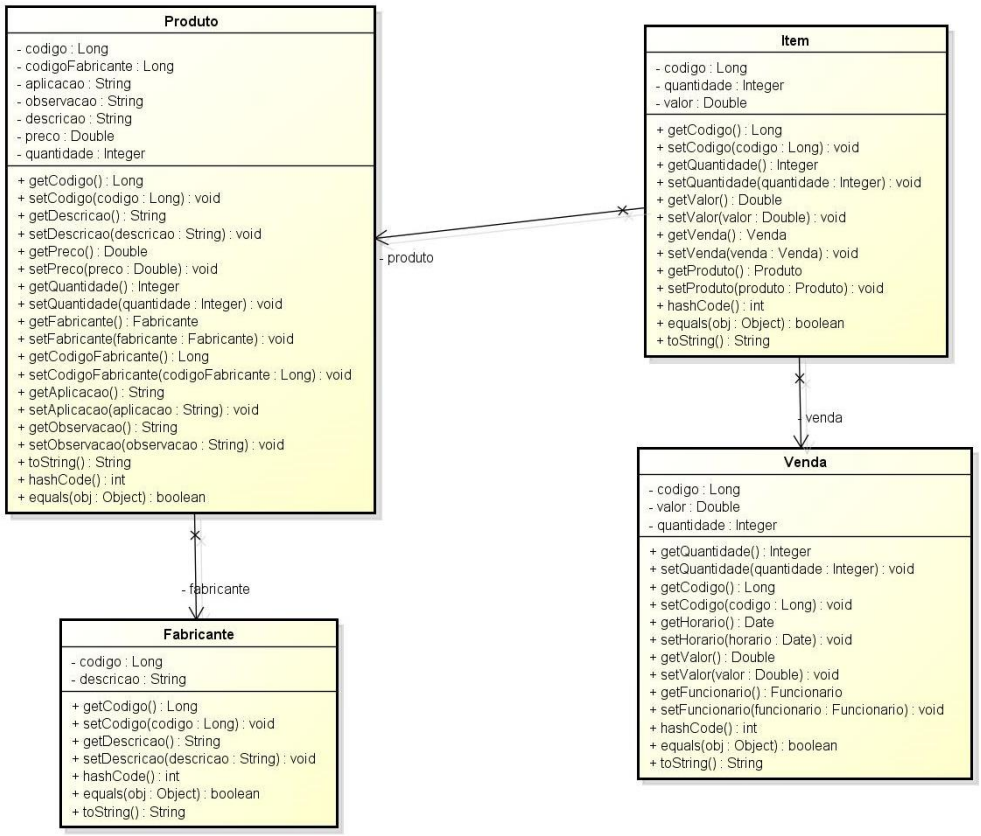
Figura 10 - Diagrama de classes



Fonte: Elaborado pelo autor (2015)

Na figura 11 temos destacado o diagrama de classes das vendas com as entidades produto, item, fabricante e venda e seus respectivos relacionamentos. Todas as classes com seus atributos e metodos

Figura 11 - Diagrama de classes Vendas

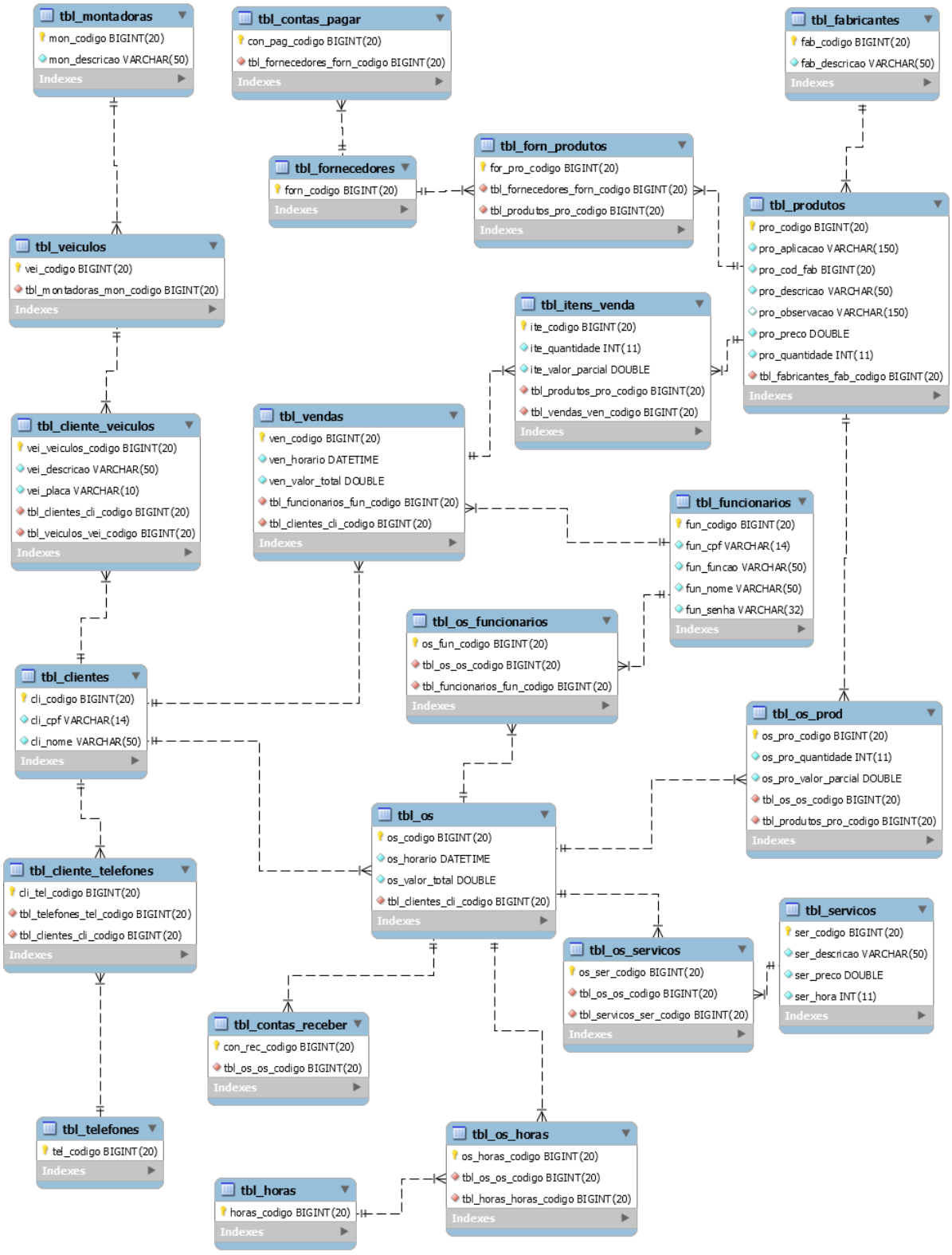


Fonte: Elaborado pelo autor (2015)

4.1.7 Modelo ER

O modelo ER completo é visto na figura 12.

Figura 12 - Modelo ER



Fonte: Elaborado pelo autor (2015)

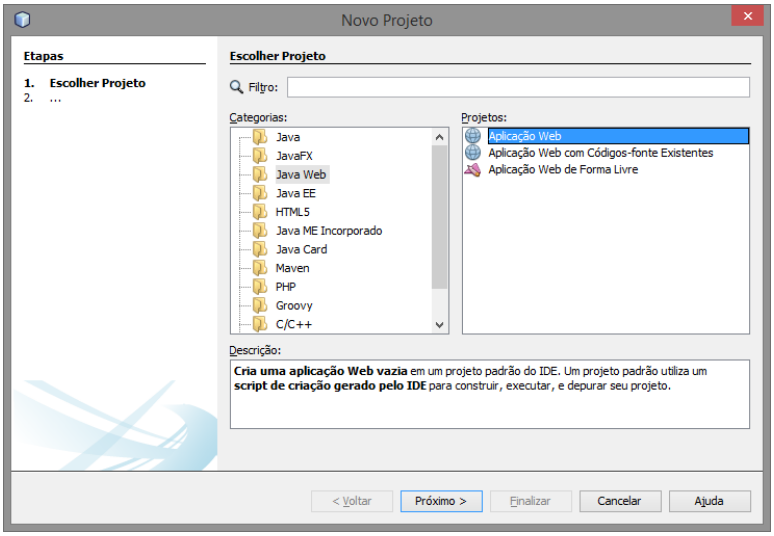
5 DESENVOLVIMENTO E CONSTRUÇÃO

Para realizar o desenvolvimento do sistema foi utilizado o ambiente de desenvolvimento integrado (*integrated development environment* – IDE) netbeans.

5.1 CONFIGURAÇÃO INICIAL DO PROJETO

O passo demostrado na figura 13 é o passo inicial para o desenvolvimento de uma aplicação Java web no netbeans.

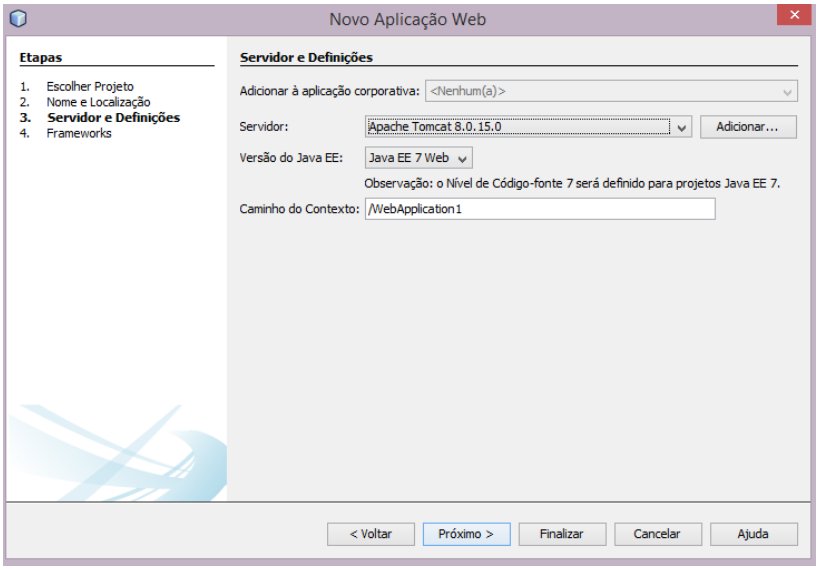
Figura 13 - Aplicação Web



Fonte: Elaborado pelo autor (2015)

Na figura 14 é o passo da escolha do servidor, que para este projeto foi escolhido o Apache Tomcat que é vinculado a instalação do proprio netbeans.

Figura 14 - Apache Tomcat

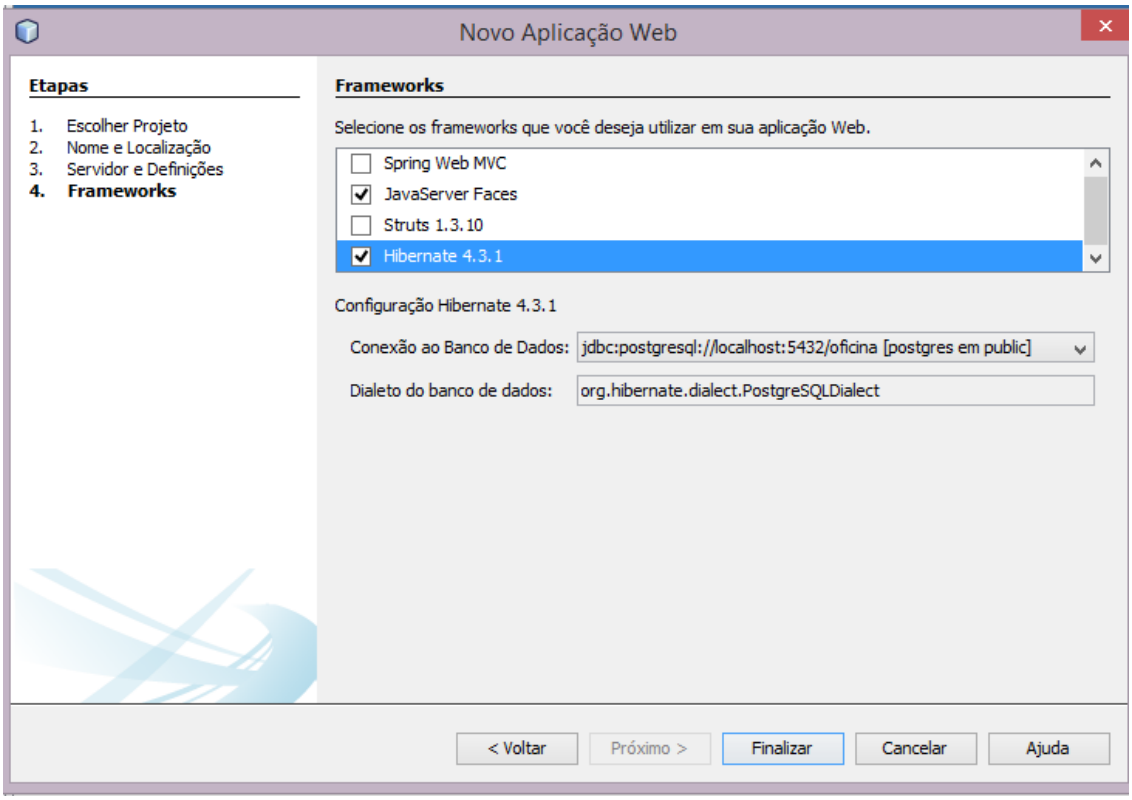


Fonte: Elaborado pelo autor (2015)

É importante que durante a construção do projeto sejam tomadas algumas precauçoes , como as escolhas do *framework javaserver faces* e o *hibernate*.

No caso do *hibernate* , há a necessidade que o banco de dados já tenha sido criado para que seja feita a conexão de maneira correta. Como mostra a figura 15.

Figura 15 - Hibernate

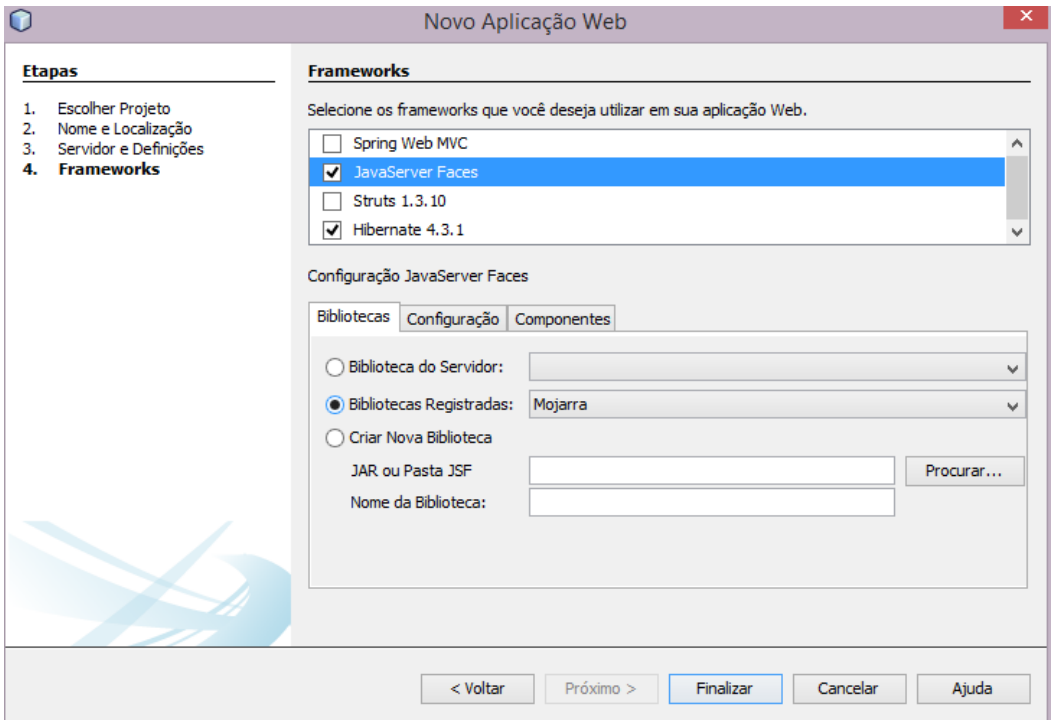


Fonte: Elaborado pelo autor (2015)

A configuração do *JavaServer* faces também é fundamental que seja feita durante a construção do projeto.

A figura 16 apenas mostra a seleção da biblioteca Mojarra, que é a biblioteca oficial do javaserver faces.

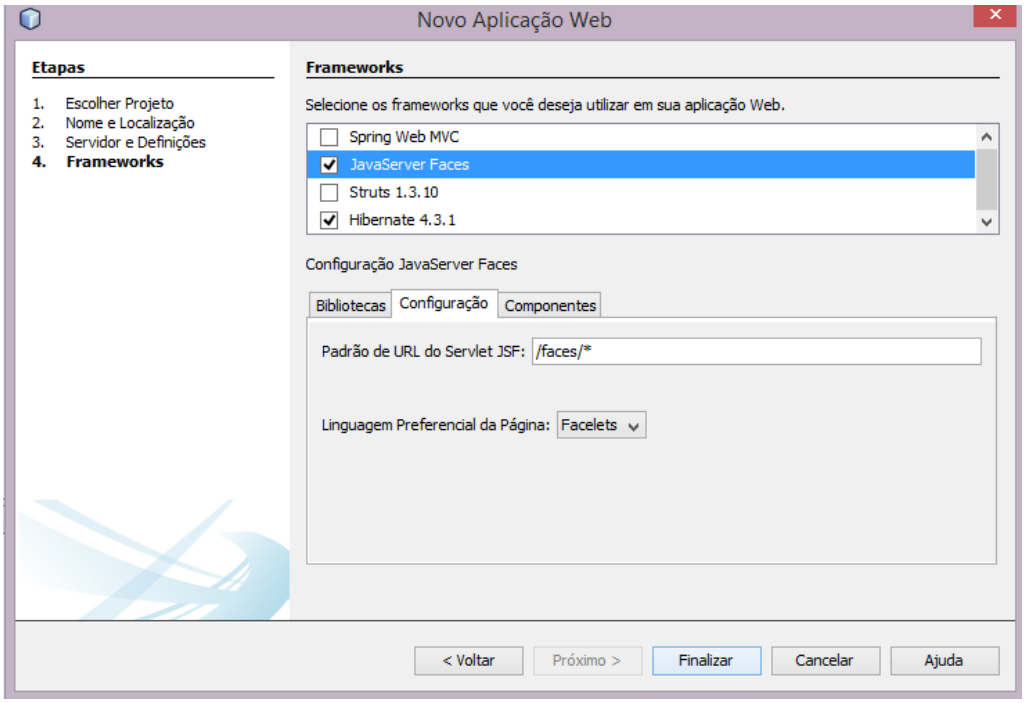
Figura 16 - JavaServer Faces



Fonte: Elaborado pelo autor (2015)

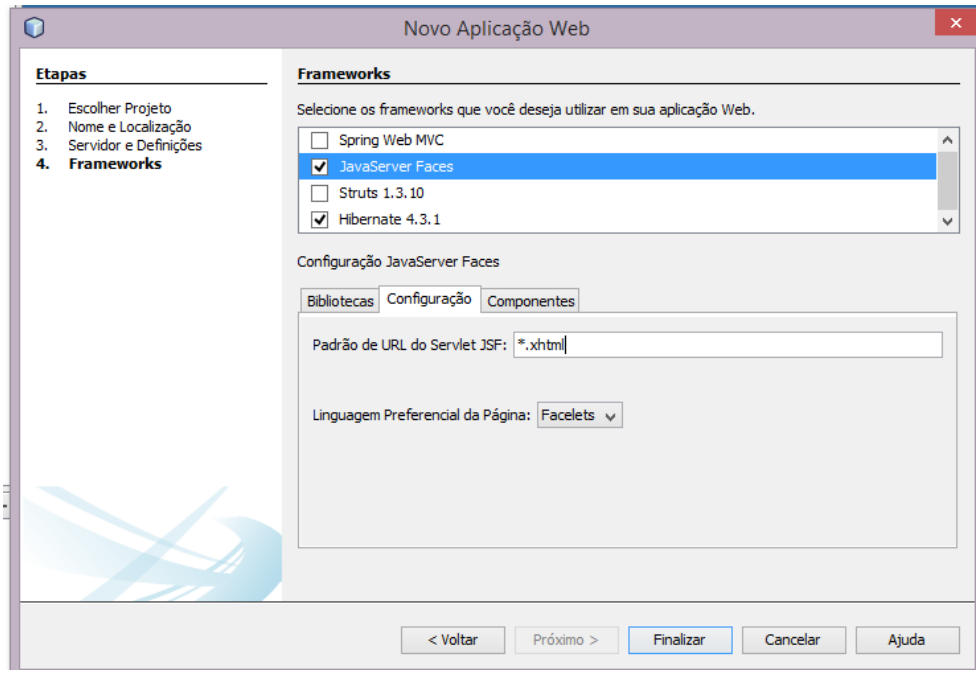
As figuras 17 e 18 demonstram um passo muito importante para a contrução do projeto. O campo "Padrao de url do servlet jsf" vem por padrão como: "/faces/*", deve ser mudado para "*.xhtml".

Figura 17 - Faces



Fonte: Elaborado pelo autor (2015)

Figura 18 - Xhtml



Fonte: Elaborado pelo autor (2015)

5.2 CONFIGURAÇÃO DO *HIBERNATE*

A escolha do *hibernate* durante a construção do projeto, resulta na criação automática do arquivo "hibernate.cfg.xml" que contém as configurações necessárias para a criação do banco de dados e a integração do sistema com o banco, ou seja, o CRUD (*create*, *read*, *update* e *drop*).

Mas ele não está completo e na documentação do *hibernate*, no item 1.1.4 encontra-se o modelo de configuração completo, como mostra a figura 19.

Figura 19 - hibernate.cfg.xml

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

    <session-factory>

        <!-- Database connection settings -->
        <property name="connection.driver_class">org.hsqldb.jdbcDriver</property>
        <property name="connection.url">jdbc:hsqldb:hsql://localhost</property>
        <property name="connection.username">sa</property>
        <property name="connection.password"></property>

        <!-- JDBC connection pool (use the built-in) -->
        <property name="connection.pool_size">1</property>

        <!-- SQL dialect -->
        <property name="dialect">org.hibernate.dialect.HSQLDialect</property>

        <!-- Enable Hibernate's automatic session context management -->
        <property name="current_session_context_class">thread</property>

        <!-- Disable the second-level cache  -->
        <property name="cache.provider_class">org.hibernate.cache.internal.NoCacheProvider</property>

        <!-- Echo all executed SQL to stdout -->
        <property name="show_sql">true</property>

        <!-- Drop and re-create the database schema on startup -->
        <property name="hbm2ddl.auto">update</property>

        <mapping resource="org/hibernate/tutorial/domain/Event.hbm.xml"/>

    </session-factory>

</hibernate-configuration>
```

1.1.4. Hibernate configuration

Fonte: jboss (2015)

Deve-se ficar atento para as configurações da conexão com o banco de dados correto, como demonstra a figura 20 e o mapeamento das tabelas na figura 21

Figura 20 - Conexão com Postgresql

```
<property name="hibernate.connection.driver_class">org.postgresql.Driver</property>
<property name="hibernate.connection.url">jdbc:postgresql://localhost:5432/oficina</property>
<property name="hibernate.connection.username">postgres</property>
<property name="hibernate.connection.password">postgres</property>
```

Fonte: Elaborado pelo autor (2015)

Na primeira execução do programa a diretiva “<property name= “hbm2ddl.auto”>” deve estar como *create*, para que sejam criadas as tabelas e logo após deve-se mudar para “update” como mostra a figura 21.

Figura 21 - Mapeamento das tabelas

```
<!-- Drop and re-create the database schema on startup -->
<property name="hbm2ddl.auto">update</property>
<!--create só para a criação do projeto -->
<!-- validate: validate the schema, makes no changes to the database.-->
<!-- update: update the schema.-->
<!-- create: creates the schema, destroying previous data.-->
<!-- create-drop: drop the schema at the end of the session.-->
<mapping class="br.com.oficina.domain.Cliente"/>
<mapping class="br.com.oficina.domain.Fabricante"/>
<mapping class="br.com.oficina.domain.Funcionario"/>
<mapping class="br.com.oficina.domain.Item"/>
<mapping class="br.com.oficina.domain.Montadora"/>
<mapping class="br.com.oficina.domain.OS"/>
<mapping class="br.com.oficina.domain.Produto"/>
<mapping class="br.com.oficina.domain.Servico"/>
<mapping class="br.com.oficina.domain.Veiculo"/>
<mapping class="br.com.oficina.domain.Venda"/>
```

Fonte: Elaborada pelo autor (2015)

5.3 CRIAÇÃO DOS PACOTES JAVA:

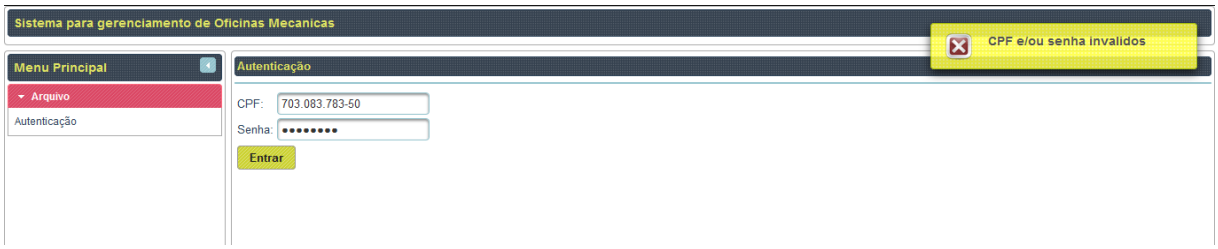
Seguindo as normas de criação de Pacotes Java, foram criados pacotes com a seguinte nomenclatura:

- br.com.oficina.bean - Managed Bean, manipulação das telas web.
- br.com.oficina.dao - manipulações do banco de dados (DAO – Data Access Object).
- br.com.oficina.domain - domínio, onde ficam as entidades.
- br.com.oficina.util - utilitários do sistema.

5.4 TELAS DO SISTEMA

A página inicial ou tela inicial do sistema é simples e limpa e a autenticação é realizada pelo cpf e senha do usuário. Caso seja invalido uma mensagem clara aparece na lateral da página. Essa mensagem assim como todas as outras não desaparecerão sozinhas da tela, ou o usuário fecha ela ou efetua alguma ação na página, como mostra a figura 22

Figura 22 - Tela inicial e autenticação



Fonte: Elaborado pelo autor (2015)

O sistema funciona da seguinte maneira:

As páginas web somente trabalham com as classes "*bean*", os métodos definidos dentro da classe requisitada é que fazem chamadas as outras classes.

Como mostra a figura 24 a tela de autenticação faz um elo com a classe autenticaBean que por sua vez com o metodo funcionarioLogado(figura 25) e por fim a consulta ao banco de dados para buscar o cpf(figura 26).

Figura 23 – Autenticação - xhtml

```
<ui:define name="conteudo" >
    <h:form>

        <p:toolbar>
            <f:facet name="left">
                <h:outputText value="Autenticação" title="Autenticação"/>
            </f:facet>
        </p:toolbar>

        <p:separator/>

        <h:panelGrid columns="2">
            <p:outputLabel value="CPF:" />
            <p:inputMask mask="999.999.999-99" id="cpf" size="20" value="#{autenticacaoBean.funcionarioLogado.cpf}">
                <f:validateBean/>
                 <p:focus for="cpf"/>
            </p:inputMask>
            <p:outputLabel value="Senha:"/>
            <p:password maxlength="32" size="20" value="#{autenticacaoBean.funcionarioLogado.senha}">
                <f:validateBean/>
            </p:password>

        </h:panelGrid>

        <h:panelGrid columns="1">
            <p:commandButton value="Entrar" update=":msgGlobal"
                             action="#{autenticacaoBean.autenticar()}"/>
        </h:panelGrid>

    </h:form>
```

Fonte: Elaborado pelo autor (2015)

Figura 24 – Autenticação - bean

```
public Funcionario getFuncionarioLogado() {
    if (funcionarioLogado == null) {
        funcionarioLogado = new Funcionario();
    }
    return funcionarioLogado;
}

public void setFuncionarioLogado(Funcionario funcionarioLogado) {
    this.funcionarioLogado = funcionarioLogado;
}

public String autenticar() {
    try {
        FuncionarioDAO funcionarioDAO = new FuncionarioDAO();
        funcionarioLogado = funcionarioDAO.autenticar(funcionarioLogado.getCpf(), DigestUtils.md5Hex(funcionarioLogado.getSenha()));
        if (funcionarioLogado == null) {
            FacesUtil.adicionarMsgErro("CPF e/ou senha invalidos");
            return null;
        } else {
            FacesUtil.adicionarMsgInfo("Funcionario autenticado com sucesso");
            return "/pages/principal.xhtml?faces-redirect=true";
        }

    } catch (RuntimeException ex) {
        FacesUtil.adicionarMsgErro("Erro ao tentar autenticar no sistema" + ex.getMessage());
        return null;
    }
}
```

Fonte: Elaborado pelo autor (2015)

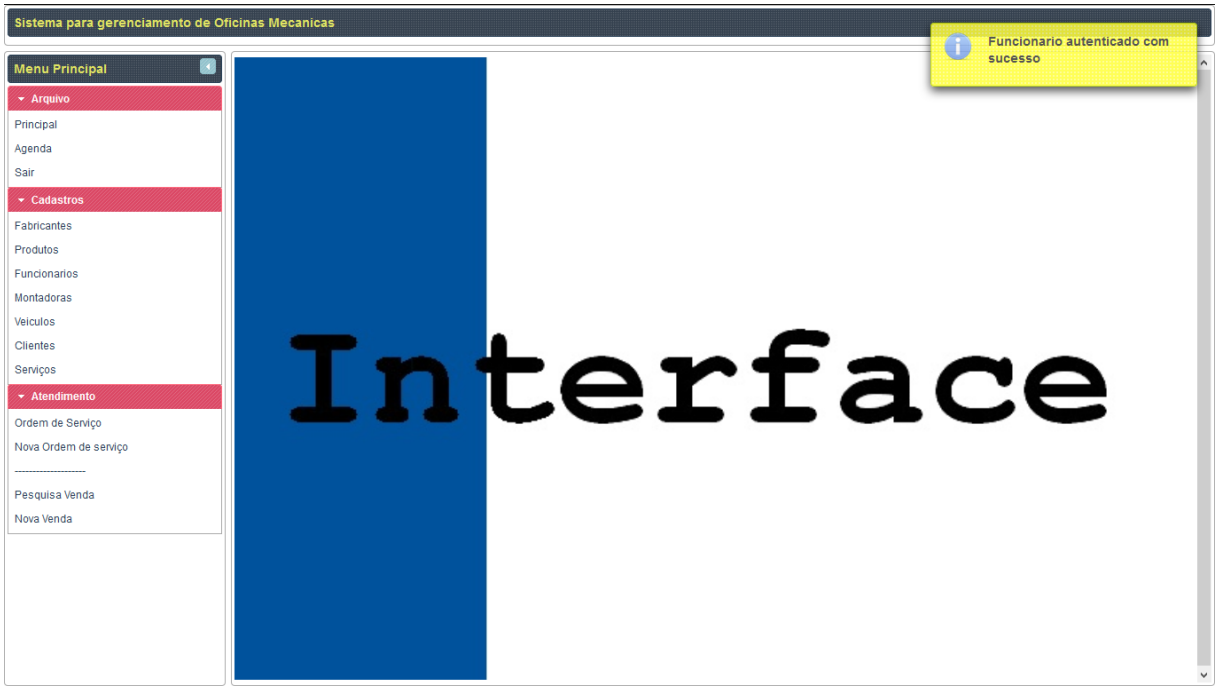
Figura 25 – Autenticação - DAO

```
'

public Funcionario autenticar(String cpf, String senha) {
    Session sessao = HibernateUtil.getSessionFactory().openSession();
    Funcionario funcionario = null;

    try {
        Query consulta = sessao.getNamedQuery("Funcionario.autenticar");
        consulta.setString("cpf", cpf);
        consulta.setString("senha", senha);

        funcionario = (Funcionario) consulta.uniqueResult();
    } catch (RuntimeException ex) {
        throw ex;
    } finally {
        sessao.close();
    }
    return funcionario;

}
```

Fonte: Elaborado pelo autor (2015)

Quando o usuário estiver logado ira aparecer um menu completo na arte esquerda da tela e um logotipo no centro e também a mensagem de sucesso como mostra a figura 26.

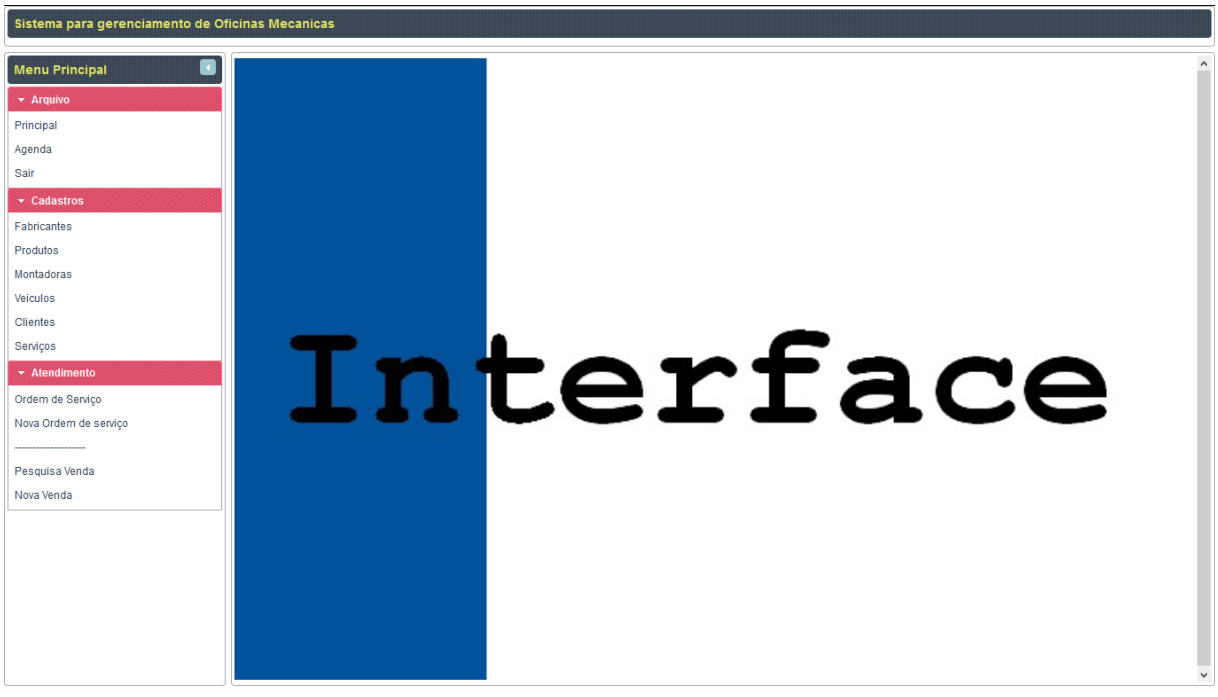
Figura 26 - Tela de administrador autenticado



Fonte: Elaborado pelo autor (2015)

O usuário da figura 26 é o administrador, portanto o menu está completo. Na figura 27 o usuário é o atendente e não tem permissão de cadastrar novos usuários. Não aparece no menu a opção Funcionários.

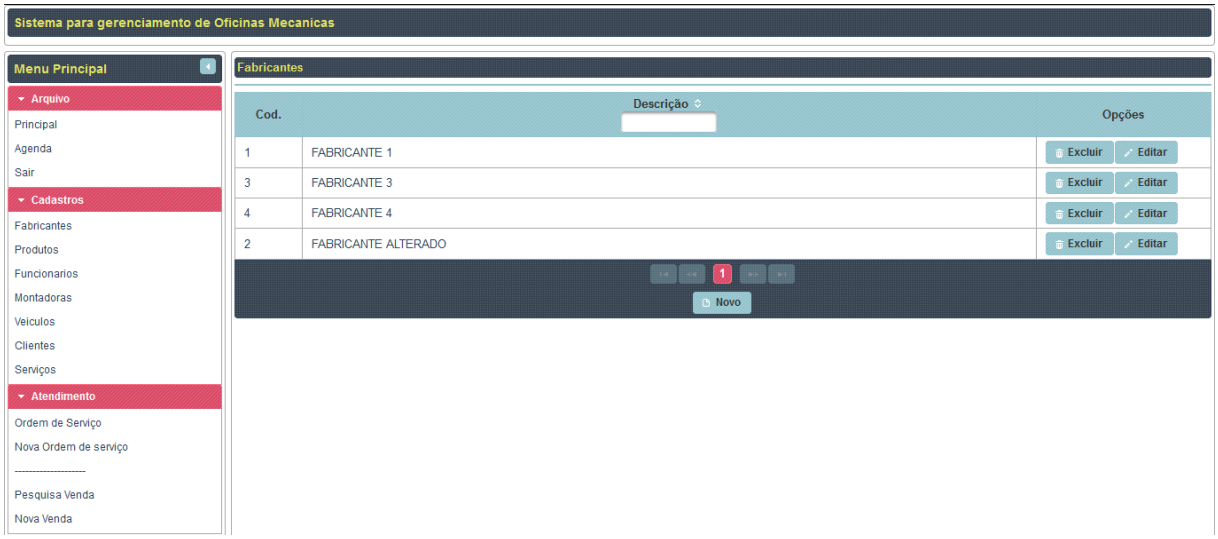
Figura 27 - Tela de atendente autenticado



Fonte: Elaborado pelo autor (2015)

A figura 28 mostra a tela de fabricantes, que tem as opções de excluir, editar ou cadastrar um novo fabricante. O fabricante poderá ser listado por ordem alfabética ou o usuário poderá digitar o nome.

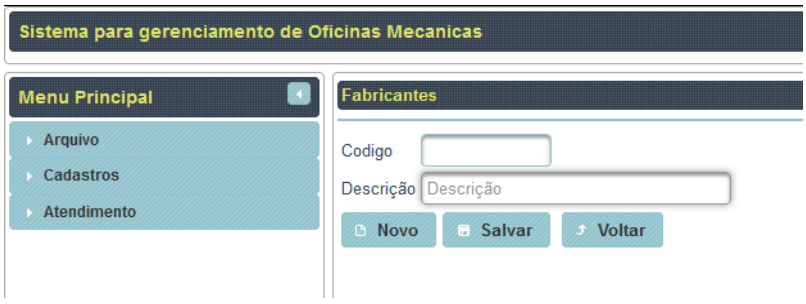
Figura 28 - Lista de fabricantes



Fonte: Elaborado pelo autor (2015)

O cadastro de fabricantes terá apenas o nome do fabricante (figura 30) e já deverá estar cadastrado a maioria dos fabricantes conhecidos, a lista de fabricantes é apenas para facilitar o cadastro de novos produtos e caso surja no mercado um novo fabricante o usuário do sistema poderá cadastrar um novo.

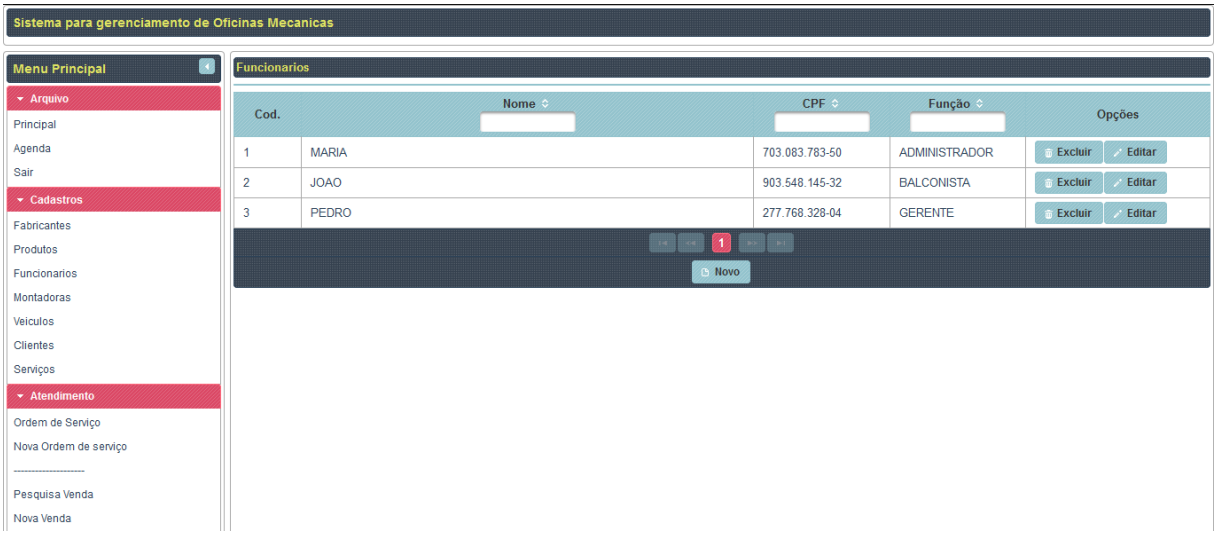
Figura 29 - Cadastro de fabricantes



Fonte: Elaborado pelo autor (2015)

A listagem de funcionários (figura 30), como somente será acessada pelo administrador do sistema, mostrará o cpf e terá as opções de excluir, editar e novo. Poderá ser feita uma pesquisa pelo nome, cpf ou função.

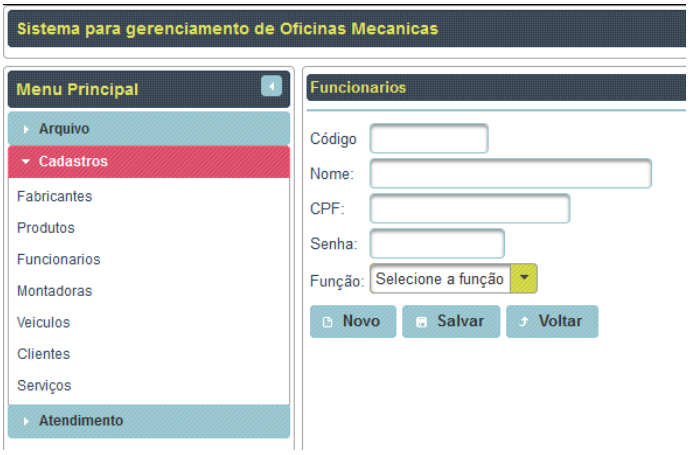
Figura 30 - Lista de funcionários



Fonte: Elaborado pelo autor (2015)

O cadastro de funcionários (figura 31) será simples e deverá ser preenchido na frente do usuário para que ele digite uma senha que será criptografada no banco de dados, por questão de segurança apenas o usuário saberá a sua senha.

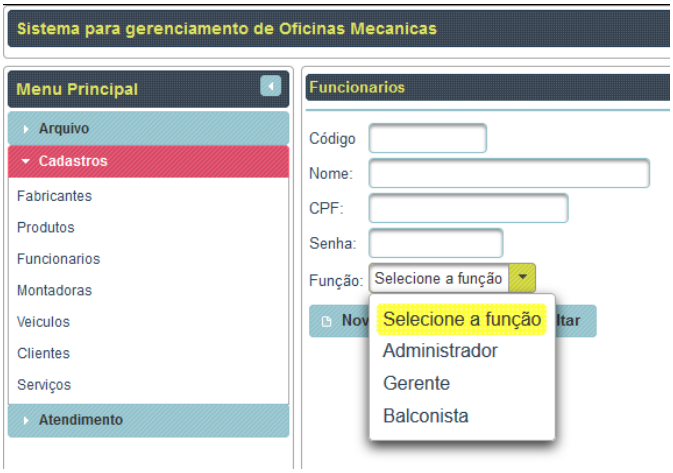
Figura 31 - Cadastro de funcionários



Fonte: Elaborado pelo autor (2015)

A função do funcionário (figura 32) está incorporada ao sistema, não poderá ser modificada.

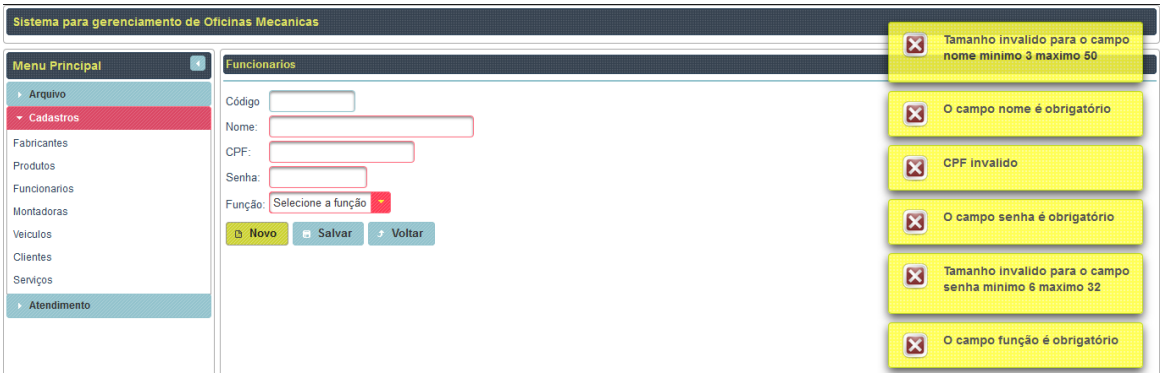
Figura 32 - Tipos de funcionários



Fonte: Elaborado pelo autor (2015)

Todos os campos da tela (figura 33) terão validações.

Figura 33 - Campos com validações



Fonte: Elaborado pelo autor (2015)

Desde o começo do curso existiu a preocupação em entender a validação de CPF e ao realizar este projeto descobriu que o próprio hibernate faz isso (figura 34) sem a necessidade de criação de novos códigos, apenas com a notação “@CPF (message=”CPF invalido”) ” na variável que representa o CPF.

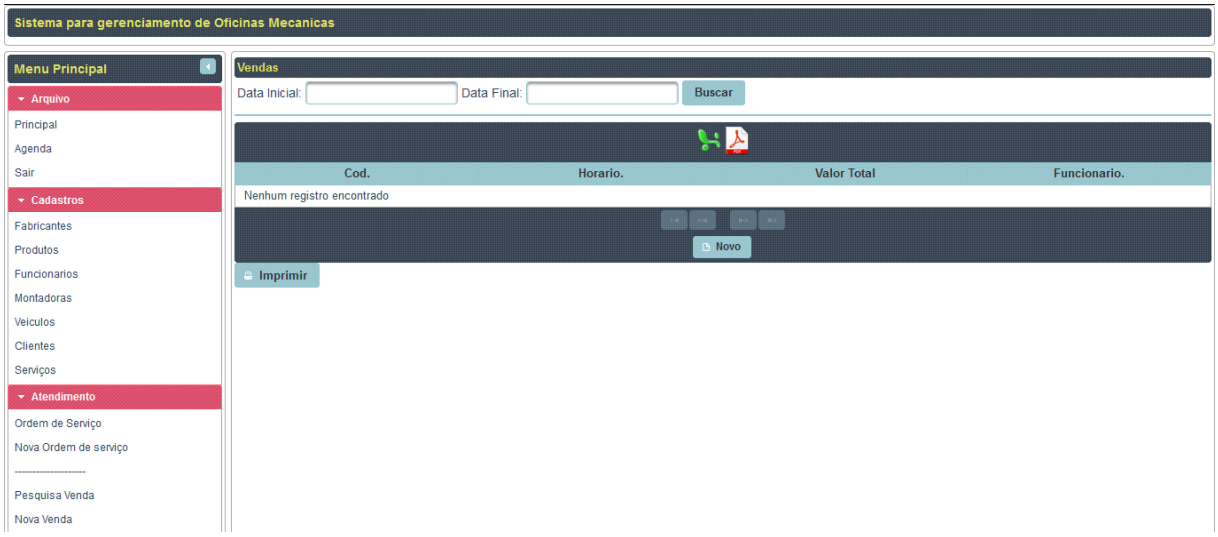
Figura 34 - Classe Funcionário, CPF

```
@Entity
@Table(name = "tbl_funcionarios")
@NamedQueries({
    @NamedQuery(name = "Funcionario.listar", query = "SELECT fu FROM Funcionario fu"),
    @NamedQuery(name = "Funcionario.buscarPorCodigo", query = "SELECT fu FROM Funcionario fu WHERE fu.codigo = :codigo"),
    @NamedQuery(name = "Funcionario.autenticar", query = "SELECT fu FROM Funcionario fu WHERE fu.cpf = :cpf AND fu.senha = :senha")})
public class Funcionario implements Serializable {
    @Id
    @Column(name = "fun_codigo")
    //  @GeneratedValue(strategy = GenerationType.AUTO)
    @SequenceGenerator(name = "seq_fun", sequenceName = "seq_fun_id", allocationSize = 1)
    @GeneratedValue(generator = "seq_fun", strategy = GenerationType.SEQUENCE)
    private Long codigo;

    @NotEmpty(message = "O campo nome é obrigatório")
    @Size(min = 3, max = 50, message = "Tamanho invalido para o campo nome minimo 3 maximo 50")
    @Column(name = "fun_nome", length = 50, nullable = false)
    private String nome;

    @CPF(message = "CPF invalido")
    @Column(name = "fun_cpf", length = 14, nullable = false, unique = true)
    private String cpf;

    @NotEmpty(message = "O campo senha é obrigatório")
    @Size(min = 6, max = 32, message = "Tamanho invalido para o campo senha minimo 6 maximo 32")
    @Column(name = "fun_senha", length = 150, nullable = false)
    private String senha;

    @NotEmpty(message = "O campo função é obrigatório")
    @Column(name = "fun_funcao", length = 50, nullable = false)
    private String funcao;
```

Fonte: Elaborado pelo autor (2015)

A tela de vendas, além das opções já conhecidas, tem a opção de busca por um determinado período, e também a opção de geração de relatórios em Excel ou pdf.

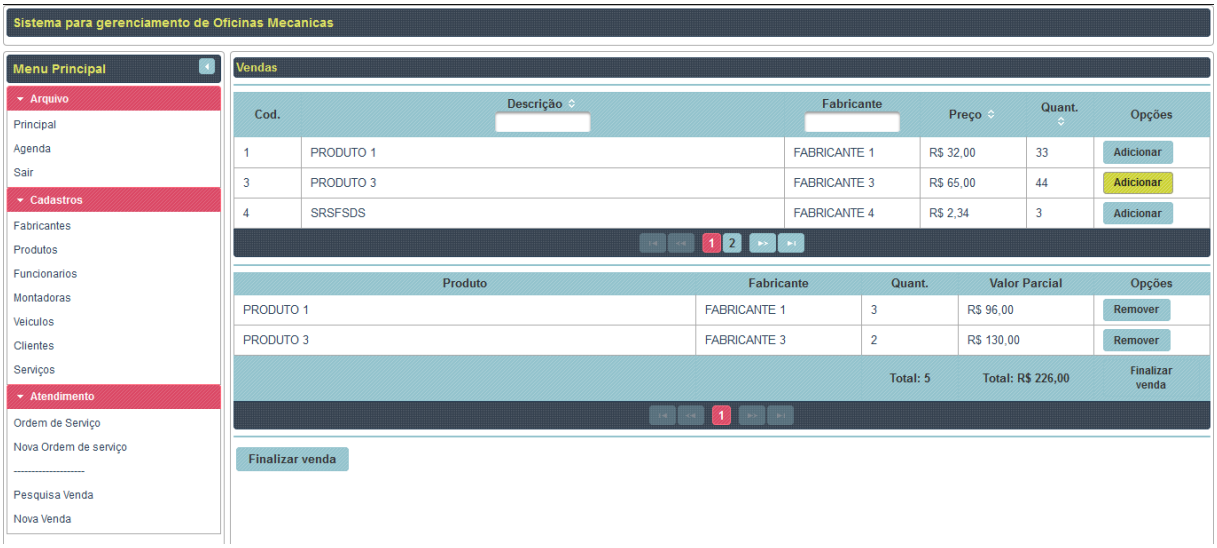
Figura 35 - Tela de vendas



Fonte: Elaborado pelo autor (2015)

A venda é feita de maneira simples, numa mesma tela (figura 36) está a listagem de produtos e o carrinho de compras. O usuário vai adicionando ao carrinho e depois finaliza a venda

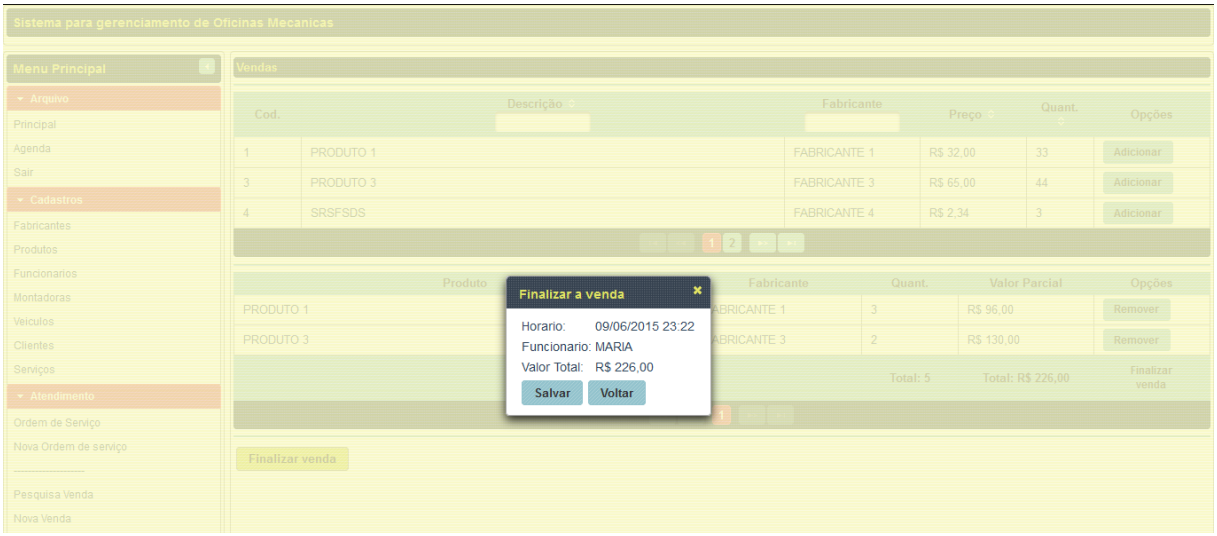
Figura 36 - Itens da venda



Fonte: Elaborado pelo autor (2015)

Na finalização da venda surge uma janela (figura 37), com o horário da venda o nome do usuário e o valor total, para a confirmação da venda.

Figura 37 - Finalização da venda



Fonte: Elaborado pelo autor (2015)

Uma dificuldade surgiu durante a construção da tela de confirmação de venda (figura 37), a janela que aparecia não era clicavel, ficava como aparece a tela de fundo na figura 37. Um problema interessante, pois na documentação do *primefaces* não tinha nada a respeito.

A solução foi uma pesquisa na internet para descobrir que no componente “<p: dialog>” estava faltando o parâmetro “*modal=”true*””, que corrigiu o problema (figura 38).

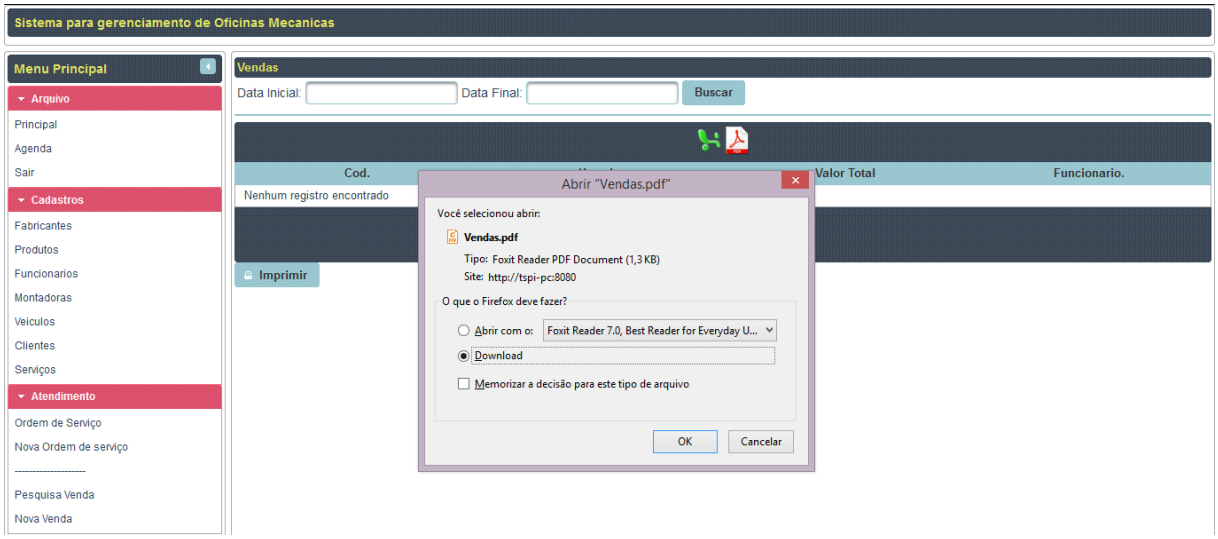
Figura 38 - Dialog do primefaces, finalização da venda

```
<p:dialog closable="true" draggable="true" modal="true" resizable="false"
          header="Finalizar a venda"
          widgetVar="wvDlgFinVenda"
          appendTo="@(body)">
```

Fonte: Elaborado pelo autor (2015)

Para gerar relatórios em pdf ou Excel (figura 39) basta adicionar ou netbeans as bibliotecas “poi” e “iText” respectivamente.

Figura 39 - Relatórios



Fonte: Elaborado pelo autor (2015)

O parâmetro “dataexporter” (figura 40), é responsável pela exportação da tabela desejada, num determinado formato

Figura 40 - dataexporter

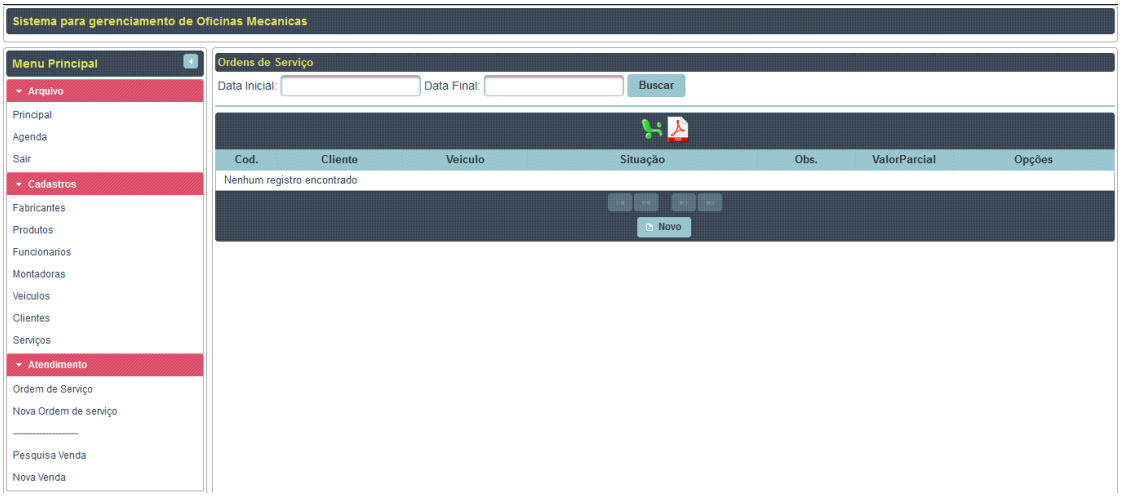
```
<f:facet name="header">
    <h:commandLink>
        <p:graphicImage library="images" name="excel.png"/>
        <p:dataExporter type="xls" target="tblVendasPesquisa" fileName="Vendas"  />
    </h:commandLink>

    <h:commandLink>
        <p:graphicImage library="images" name="pdf.png"/>
        <p:dataExporter type="pdf" target="tblVendasPesquisa" fileName="Vendas" encoding="ISO-8859-1"/>
    </h:commandLink>

</f:facet>
```

Fonte: Elaborado pelo autor (2015)

As figuras 41, 42, 43 e 44 ilustram este trabalho, pois o desenvolvimento e funcionamento das mesmas é semelhante as anteriores.

A listagem de ordem de serviço (figura 42), é semelhante a listagem de vendas, possui uma pesquisa por data, opção de relatórios em pdf e Excel e o botão “Novo”, que abre uma nova ordem de serviço.

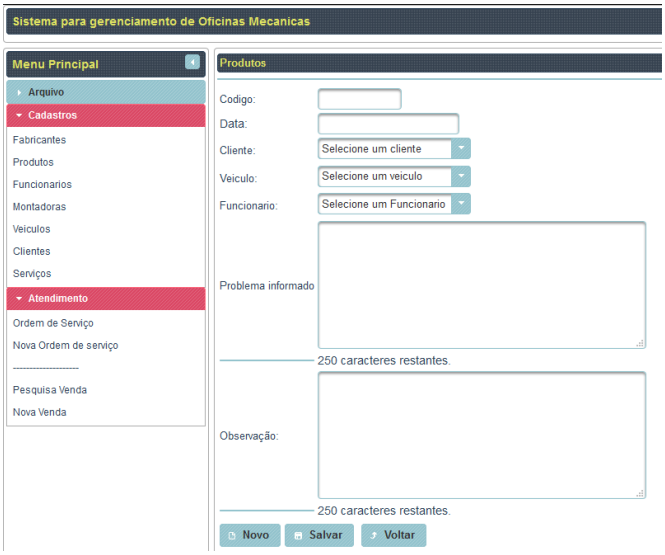
Figura 41 - Listagem de ordens de serviço



Fonte: Elaborado pelo autor (2015)

Uma nova ordem de serviço (figura 42), segue o modelo do cadastro de funcionários. A ordem de serviço possui data, uma listagem de clientes, veículos e funcionários, assim como os campos “Problema informado” e “Observações”, onde o atendente registra os problemas do veículo, informados pelo proprietário e algumas observações pertinentes ao serviço.

Figura 42 - Nova ordem de serviço

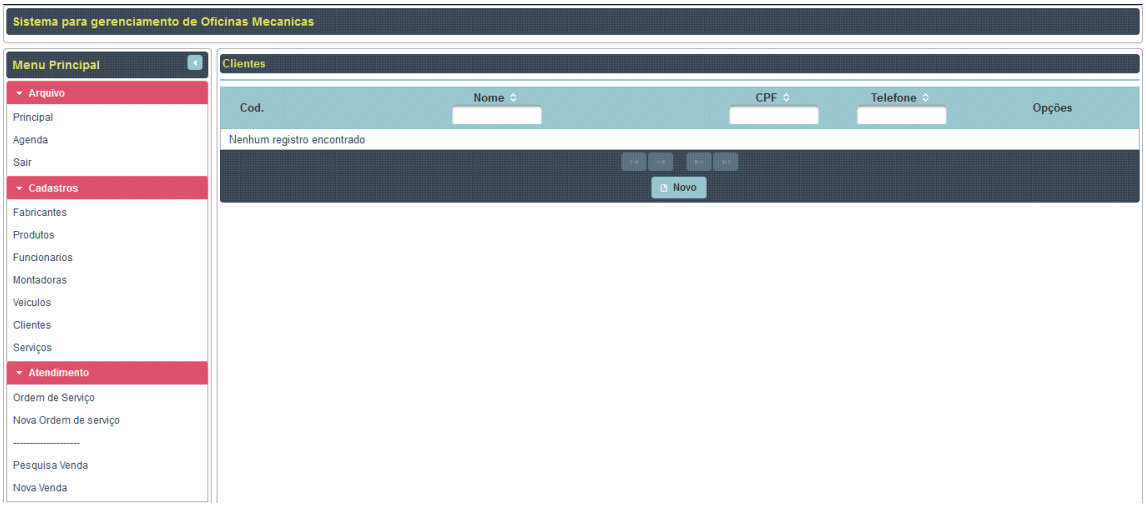


Fonte: Elaborado pelo autor (2015)

A listagem de clientes (figura 43), funciona semelhante a listagem de funcionários.

De modo simples e intuitivo, a tela de listagem de clientes tem a pesquisa por nome, cpf e telefone para uma procura de um determinado cliente e também as opções de editar, excluir e cadastrar um novo cliente

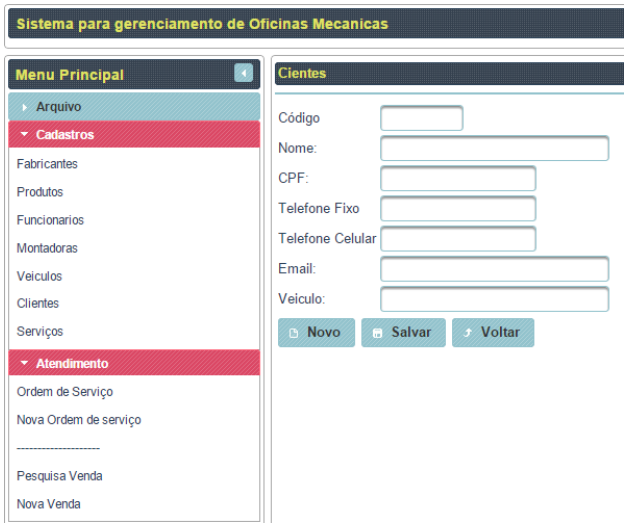
Figura 43 - Listagem clientes



Fonte: Elaborado pelo autor (2015)

O cadastro de clientes (figura 44) e o cadastro de funcionários possuem o mesmo funcionamento.

Figura 44 - Cadastro de clientes



Fonte: Elaborado pelo autor (2015)

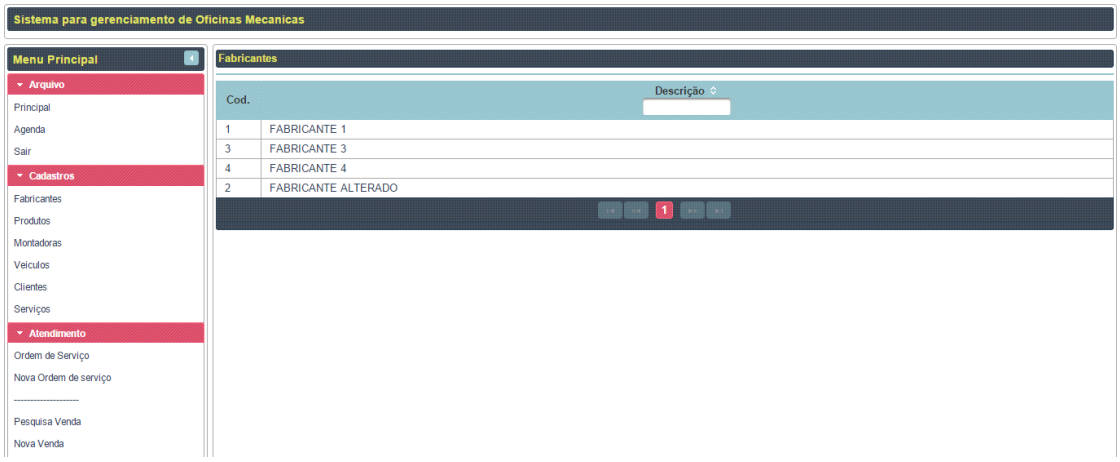
O sistema não permite que um usuário não autenticado acesse qualquer outra página, a não ser a página de autenticação, como mostra a figura 45.

Figura 45 - Digitando o endereço da página sem estar logado

Fonte: Elaborado pelo autor (2015)

A figura 46 mostra uma das restrições aplicada ao funcionário gerente, ele tem acesso a listagem de fabricantes, mas não pode cadastrar, excluir ou editar.

Figura 46 - Tela de fabricantes do funcionário gerente



Fonte: Elaborado pelo autor (2015)

No processo de desenvolvimento do sistema, quando os dados são do mesmo tipo, os processos de leitura, edição, exclusão e cadastro, são similares, tornando o reaproveitamento de código um procedimento comum.

6 CONSIDERAÇÕES FINAIS

Para desenvolver o sistema proposto neste trabalho foi necessário, além de rever todo o aprendizado obtido durante os anos do curso, estudar novas tecnologias que estão surgindo no mercado. Isso reforça o fato de que a tecnologia da informação está sempre em evolução, fazendo com que quem trabalhe nesta área tenha a necessidade de estar sempre atualizado, sempre atento para o que está acontecendo no mundo da programação.

Foi aprendido que um sistema nunca estará pronto, havendo sempre a necessidade de atualizações.

Notou-se que muita coisa muda e evolui, mas a base é a mesma, a etapa de levantamento de requisitos e de construção do banco de dados mantem-se estável.

O projeto terá continuidade e provavelmente num prazo de um ano, a seguir, estará disponível para comercialização.

7 REFERÊNCIAS

AUDAMEC. *Marketing e Pesquisa automotiva*. Disponível em: <http://www.audamec.com.br/Numero-de-oficinas-mecanicas-no-Brasil/2/n/>. Acesso em: 11 out. 2013.

BAIXAKI. *brModelo*. Disponível em: <http://www.baixaki.com.br/download/brmodelo.htm>. Acesso em: 26 fev. 2015

BATEBYTE. *Processo de Desenvolvimento de Sistemas com Qualidade*. Disponível em <http://www.batebyte.pr.gov.br/modules/conteudo/conteudo.php?conteudo=1492&gt>. Acesso em: 19 mai. 2015

CANALTECH. *Java lidera ranking das linguagens de programação mais utilizadas no mundo.* Disponível em: <http://corporate.canaltech.com.br/noticia/programacao/Java-lidera-ranking-das-linguagens-de-programacao-mais-utilizadas-no-mundo/>. Acesso em: 11 mar. 2015

CODEUSE. *Let's Learn Java – Part 1 – Introduction to Java.* Disponível em: <http://codeuse.com/introduction-to-java/>. Acesso em: 11 mar. 2015

DEVMEDIA. *Desenvolvimento de Software Dirigido por Caso de Uso*. Disponível em: <http://www.devmedia.com.br/desenvolvimento-de-software-dirigido-por-caso-de-uso/9148>. Acesso em: 03 mar. 2015

DIAS NETO, Arilo Cláudio. *Bancos de Dados Relacionais - Artigo Revista SQL Magazine 86*. Disponível em: <http://www.devmedia.com.br/bancos-de-dados-relacionais-artigo-revista-sql-magazine-86/20401>. Acesso em: 01 dez. 2013

FARIA, Thiago. *Java EE 7 com JSF, PrimeFaces e CD.* 2. Ed. S.I. s.n. 2015

FUNPAR. *Atividade: Design da Classe.* Disponível em: <http://www.funpar.ufpr.br:8080/rup/process/activity/ac_cldes.htm>. Acesso em: 19 mar. 2015

GEARY, David. *Core JavaServerTN Faces.* 3. Ed. Rio de Janeiro: Alta Books, 2012

GONÇALVES, Edson. *Desenvolvendo Aplicações Web com JSP Servlets, JavaServer Faces, Hibernate, EJB 3 Pesistence e Ajax.*1. ed. Rio de Janeiro: Editora Ciência Moderna Ltda., 2007

GUEDES, Gilleanes T. A. *UML 2:* uma abordagem pratica. 2. Ed. São Paulo: Novatec, 2011.

GURGEL, Bruno. *Análise e Projeto Orientados a Objetos - Requisitos*. Disponível em: <http://docente.ifrn.edu.br/brunogurgel/disciplinas/2013/apoo/aulas/4-Requisitos.pdf>. Acesso em 19 mai. 2015.

HEUSER, Carlos Alberto. *Projeto de Banco de Dados.* 6. ed. Porto Alegre: Bookman, 2009.

IBGE Cidades, 2010. Disponível em: <http://www.ibge.gov.br/cidadesat/painel/painel.php?codmun=431410#>. Acesso em: 10 out. 2013.

JAVA. O que é a tecnologia Java e por que é necessária? Disponível em: <http://www.java.com/pt_BR/download/faq/whatis_java.xml>. Acesso em: 01 dez. 2013.

JAVAFREE. A capital do café agora bebe JAVA. Disponível em: <http://javafree.uol.com.br/topic-4326-A-capital-do-cafe-agora-bebe-JAVA.html>. Acesso em 17 mai. 2015.

JBOSS. *Hibernate Community Documentation.* Disponível em: <http://docs.jboss.org/hibernate/orm/5.0/manual/en-US/html/ch01.html#tutorial-firstapp-configuration>. Acesso em: 02 fev. 2015

MAGALHÃES, Guilherme. *Como Funciona o Processo de Desenvolvimento de Software.* Disponível em: <http://protocoloti.blogspot.com.br/2012/03/como-funciona-o-processo-de.html>. Acesso em: 10 mar. 2015

MACEDO, Diego. Analista de T.I *Um pouco de tudo sobre T.I.* Disponível em: <http://www.diegomacedo.com.br/levantamento-e-analise-de-requisitos/?print=print>. Acesso em: 12 mar. 2015

MEDEIROS, Ernani Sales de. *Desenvolvendo Software com UML 2.0*: definitivo.1. ed. São Paulo, 2004

MEIRA, Fabio. *Sistemas de Bancos de Dados*. Disponível em: <https://chasqueweb.ufrgs.br/~paul.fisher/apostilas/basdad/unimar/index.htm>. Acesso em: 03 mar. 2015

NETBEANS. *Introdução ao JavaServer Faces 2.x.* Disponível em: <https://netbeans.org/kb/docs/web/jsf20-intro_pt_BR.html>. Acesso em: 02 jan. 2015

NOGUEIRA, Admilson. *UML - Unified Modeling Language - Introdução e Histórico.* Disponível em: <http://www.linhadecodigo.com.br/artigo/763/uml-unified-modeling-language-introducao-e-historico.aspx#ixzz3XxSK6NRa>. Acesso em: 12 dez. 2014

OLIVEIRA, A.F. Estudo Adaptativo Para Incorporação de Transações no Sistema de Comunicação Aberto – DiretoGNU. Maio de 2002. Disponível em: <http://www.inf.lasalle.tche.br/direto/docs/Anexo2Transacoes.pdf>. Acesso em: 05 jan. 2013

PINTO, Hudson Lamounier. *Atividades básicas ao processo de desenvolvimento de Software*. Disponível em: <http://www.devmedia.com.br/atividades-basicas-ao-processo-de-desenvolvimento-de-software/5413>. Acesso em: 10 mai. 2015

PORTAL BRASIL. Mapa das micro e pequenas empresas. Disponível em: <http://www.brasil.gov.br/economia-e-emprego/2012/02/o-mapa-das-micro-e-pequenas-empresas>. Acesso em: 09 out. 2013.

RICARTE, Ivan Luiz Marques. Bancos de dados relacionais. Disponível em: <http://www.dca.fee.unicamp.br/cursos/PooJava/javadb/bdrel.html >. Acesso em: 01 dez.2014

ROCHA, Givanaldo. *Engenharia de Software, Diagrama de classe.* Disponível em: <http://docente.ifrn.edu.br/givanaldorocha/disciplinas/engenharia-de-software-licenciatura-em-informatica/diagrama-de-classes>. Acesso em: 12 jun. 2015

ROMANATO, Allan. Entenda como funciona a java virtual machine jvm. Disponível em: <http://www.devmedia.com.br/entenda-como-funciona-a-java-virtual-machine-jvm/27624>. Acesso em: 01 dez. 2014

SANTOS JÚNIOR, João B. dos. Linguagem de Programação Java. 2006. Disponível em: <http://www.tvdi.inf.br/upload/artigos/apostilalinguagemjava.pdf>. Acesso em: 01 dez. 2014.

SOMMERVILLE, Ian. *Engenharia de Software.* 8. Ed. São Paulo: Pearson Addison Wesley, 2007

TARIFA, Alexandre. Padrões de nomenclaturas – Guia de consulta rápida. Disponível em: < http://www.linhadecodigo.com.br/artigo/253/padroes-de-nomenclaturas-guia-de-consulta-rapida.aspx>. Acesso em: 26 fev. 2015