Fundação Educacional do Município de Assis
IMESA - Instituto Municipal de Ensino Superior de Assis

AGUINALDO INACIO

SISTEMA PARA INFORMATIZAÇÃO E GERENCIAMENTO DE OFICINA MECÂNICA

Assis
2014

AGUINALDO INACIO

SISTEMA PARA INFORMATIZAÇÃO E GERENCIAMENTO DE OFICINA MECÂNICA

Trabalho de Conclusão de Curso apresentado ao curso de Análise e Desenvolvimento de Sistemas do Instituto Municipal de Ensino Superior de Assis – IMESA e Fundação Educacional do Município de Assis – FEMA.

Orientador: Dr° Almir Rogério Camolesi
Área de Concentração: Desenvolvimento de Sistemas

Assis
2014

FICHA CATALOGRÁFICA

Inacio, Aguinaldo
Sistemas para informatização e gerenciamento de oficina mecânica/ Aguinaldo Inacio, Fundação Educacional do Município de Assis, 2014.

Orientador: Dr. Almir Rogério Camolesi
Trabalho de Conclusão de Curso
Instituto Municipal de Ensino Superior de Assis – IMESA.

1. Gerenciamento de oficina mecânica, Programação, Linguagem de Programação Java, UML.

SISTEMA PARA INFORMATIZAÇÃO E GERENCIAMENTO DE OFICINA MECÂNICA

AGUINALDO INACIO

Trabalho de Conclusão decurso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Análise e Desenvolvimento de Sistemas, analisado pela seguinte comissão examinadora.

Orientador: Dr° Almir Rogério Camolesi
Analisador: Me. Fabio Eder Cardoso

Assis
2014

DEDICATÓRIA

Dedico este trabalho à minha família,
amigos, professores,
e as pessoas que acreditaram
em meus sonhos.

AGRADECIMENTOS

Agradeço primeiramente a Deus, pois só pela sua graça foi possível chegar até esta etapa da minha vida. Aos meus familiares, pois sempre estiveram ao meu lado, apoiando e incentivando para o meu crescimento.

Ao meu professor e orientador, Dr° Almir Rogério Camolesi, pela orientação, e incentivo, durante todo o período deste trabalho e também durante a caminhada acadêmica.

Aos meus amigos que me apoiaram direta ou indiretamente, torcendo sempre e ajudando nos momentos de dificuldade.

RESUMO

A proposta do trabalho é digitalizar informações administrativas da empresa, facilitando o controle e o armazenamento das mesmas. Neste trabalho, será apresentada a análise de um sistema para o gerenciamento de oficina mecânica. O sistema tem por objetivo atender e suprir necessidades de pequenas empresas de manutenção de veículos.

A análise foi feita utilizando modelagem UML. Para o desenvolvimento foi utilizado tecnologia Java, banco de dados *Postgresql*, e no desenvolvimento de telas foi utilizado tecnologia JSF e *Primefaces*.

ABSTRACT

The proposal of this paper is digitize administrative information of a company making easier the control and storing of them. In this paper the analysis of a managerial system for garages will be presented. The objective of the system is to support the needs of small companies for maintanance of vehicles.

The analysis was done using UML modeling. Was used to develop Java technology, PostgreSQL database, and developing screens was used JSF and Primefaces technology.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

O.S. - Ordem de Serviço.
WBS - Work Breakdown Structure.
SQL - Structure Query Language.
MVC - Model View Controller.
UML - Unified Modeling Language.

Sumário

1-INTRODUÇÃO

A utilização de Computador e programas (*softwares*) está presente em todos os seguimentos da sociedade. Pode-se dizer que não há área que não use computadores, com o constante crescimento da quantidade de informações que temos que é manipulada durante o dia, é praticamente impossível o não uso de computadores. Nos tempos atuais o bem mais valioso é a informação, já que ela pode ser o diferencial entre o fracasso e o sucesso, levando em conta isso as empresas estão valorizando todos os tipos de informações que conseguirem (Pereira e Miranda, 2013).

Softwares específicos, como o proposto neste trabalho, tem a intenção de facilitar a administração de uma empresa de reparação automotiva.

Com a constante venda de veículos, cresce a necessidade de empresas que atuam na área de reparação, de ter um controle de suas informações, informações essas que são controle de serviços, estoque, clientes, veículos, etc.

O controle de todas estas informações, sem a utilização de computadores, consumiria muito tempo dos administradores da empresa, pouca confiabilidades nos dados e também uma demora muito grande no acesso a estas informações.

1.1-OBJETIVO

A proposta do sistema é gerar facilidade na administração das informações da empresa. Permitindo o acesso fácil a cadastros e consultas de informações armazenadas. Gerando informações confiáveis para que os administradores possam tomar as melhores decisões para o futuro do empreendimento.

1.2-JUSTIFICATIVA

No atual mercado de prestadores de serviços há ainda várias empresas que não utilizam sistemas informatizados, a proposta é fornecer um sistema simples e eficiente para auxiliar estas pequenas empresas.

1.3-PUBLICO-ALVO

O *software* é voltado a empresas relacionadas à área de reparação automotiva, pois contem módulos específicos para controle de clientes e veículos.

1.4-ESTRUTURA DE DESENVOLVIMENTO DO TRABALHO

Este trabalho está dividido em capítulos que serão apresentados a seguir.

O capítulo 1 apresenta a justificativa para o desenvolvimento do trabalho.

O capítulo 2 aborda os conceitos de fundamentação teórica das tecnologias utilizadas para o desenvolvimento do *software*.

O capítulo 3 apresenta as etapas e especificações do *software* contemplando o levantamento de requisitos, lista de eventos, caso de uso e suas especificações e os principais diagramas UML (classe, sequência e atividade).

O capítulo 4 descreve a WBS – *Work Breakdown Structure*, o sequenciamento das atividades e o orçamento do *software*.

O capítulo 5 apresenta etapas do desenvolvimento do sistema, exibindo um detalhamento sobre a aplicação desenvolvida assim como a organização e distribuição das camadas do projeto e interfaces criadas para interagir com o usuário final.

2 TECNOLOGIAS DE ANÁLISE E DESENVOLVIMENTO

Neste capítulo foi descrito as tecnologias que foram utilizadas para o desenvolvimento do sistema, as técnicas utilizadas para a etapa de análise de requisitos.

2.1 METODOLOGIAS DE ANÁLISE

“A UML – ***(Unified Modeling Language)*** - é um modelo de linguagem para modelagem de dados orientada a objetos. Com ela, pode-se fazer uma modelagem visual de maneira que os relacionamentos entre os componentes do sistema sejam mais bem visualizados, compreendidos e documentados”. (MARTIN, 1994). Para a fase de análise, foi utilizado à linguagem UML, uma ferramenta que representa de forma visual as funcionalidades do sistema, utilizando diagramas de modelagem UML. Para desenvolvimento dos diagramas será utilizada a ferramenta Astah UML, um editor UML leve e compacto.

2.2 LINGUAGEM DE PROGRAMAÇÃO JAVA

Umas das principais diferenças entre a plataforma Java e as demais linguagens existentes na época é que o *Java é executado sobre uma JVM, ou Java Virtual Machine. Qualquer plataforma de hardware* ou equipamento eletrônico que possa executar uma máquina virtual conseguirá executar Java. Isso justifica o slogan *“Write once, run anywhere*”, ou em português, “escreva uma vez, rode em qualquer lugar”(Luckow, Melo, 2010).

Java é uma linguagem de programação orientada a objeto, desenvolvida por James Gosling, na empresa Sun *Microsystems*. Diferentemente das linguagens convencionais, que são compiladas para código nativo, Java é a única linguagem de programação multi-plataforma, compilada para um *bytecode* que é executado por uma máquina virtual. Depois de compilado, um programa em C ou na maioria das outras linguagens, o compilador transforma seu arquivo-fonte em código de máquina. O programa resultante será executado em outros sistemas baseados na mesma plataforma, mas não funcionará em outras plataformas de outros fabricantes.

Caso seja usado o mesmo programa em outras plataformas, deve transferir o seu código fonte para a nova plataforma e recompilá-lo para produzir o código de máquina específico para esse sistema. Em muitos casos, serão exigidas alterações no código fonte antes que ele seja compilado na nova máquina, devido a diferenças em seus processadores e outros fatores. Os programas Java atingem essa independência através da utilização de uma máquina virtual (JVM – Java *Virtual Machine*), uma espécie de computador dentro de outro. A máquina virtual pega os programas Java compilados e converte suas instruções em comandos que um sistema operacional possa manipular. O mesmo programa compilado, que existe em formato chamado de *bytecode,* pode ser executado em qualquer plataforma e sistema operacional que possua uma JVM. Umas das grades vantagens da linguagem Java é a grade quantidade de fóruns de ajuda na Internet e também é uma linguagens que pode ser usada para qualquer tipo de aplicação, entre elas: web, *desktop,* servidores, *mainframes,* jogos, aplicações móveis, chips de identificação, etc.

2.3 JSF (Java Server Faces)

JSF (Java *server faces*): 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. A proposta é que os sistemas sejam desenvolvidos com a mesma facilidade e produtividade que se desenvolve sistemas desktop (até mesmo com ferramentas que suportam clicar-e-arrastar componentes) (Faria, 2013).

A tecnologia JSF, é um framework que permite o desenvolvimento de aplicações web colocando componentes em formulários e ligando-os a objetos Java permitindo a separação entre conexões com servidores e webservices, regras de negócio e lógica. Fornecendo grande quantidade de componentes visuais pré-prontos permitindo o desenvolvimento ágil. Outra característica é a capacidade de trabalhar em conjunto com outras bibliotecas de componentes visuais.

2.4 PRIMEFACES

PrimeFaces é uma bibliotecas de componentes ricos em Java *Server Faces*. A suíte de componentes inclui diversos campos de entrada, botões, tabelas de dados, árvores, gráficos, diálogos, etc (Faria, 2013).

2.5 HIBERNATE

O mapeamento objeto/relacional (ORM) refere-se a técnica de mapear os registros do banco de dados em objetos e persistir as informações contidas nos objetos em forma de linhas e colunas (Maulo, 2008).

Hibernate é um *framework* de mapeamento objeto/relacional muito utilizado na linguagem Java. Sua função e gerar códigos SQL automaticamente, facilitando a vida do desenvolvedor, permitindo ao desenvolvedor se preocupar mais com a aplicação e menos com banco de dados.

2.6 BANCO DE DADOS POSTGRESQL

O PostgreSQl é um sistema de gerenciamento de bando de dados objeto-relacional (SGBDOR), ele foi o pioneiro em muitos conceitos objeto_relacionais que agora estão se tornando disponíveis em alguns bancos de dados comerciais.(Solgate, 2005).

Um sistema muito poderoso, sofisticado, estável, com alto desempenho. Possui licença *Open Source,* gratuito acessível e de fácil utilização e instalação. Este sistema suporta bases de dados bastante extensas, adequadas ao uso de uma grande empresa. Podemos observar as capacidades do PostgreSQL, percebendo assim que não existe qualquer tipo de limitação para grandes bases de dados:

- Tamanho máximo da Base de Dados Ilimitado *
- Tamanho máximo de uma tabela 64 TB
- Tamanho máximo de uma linha de uma tabela 1.6 TB
- Número máximo de índices por tabela ilimitado *

2.7 JASPER REPORTS

O *JasperReports* é uma biblioteca escrita em Java, de código fonte *open source*, projetada para ajudar o desenvolvedor com a tarefa de criar relatórios para aplicações, tanto Desktop como Web, fornecendo uma API que facilita sua geração(Gonçalves,2009).

O *JasperReports* é um *framework* para a geração de relatórios. É uma ferramenta totalmente *open source* e gratuita, e a mais utilizada com esse propósito atualmente. Entre as funcionalidades do JasperReports destaca-se:

- É capaz de exportar relatórios para diversos formatos diferentes, tais como PDF, HTML, XML, XLS, etc.
- Aceita diversas formas de entrada de dados, tais como um arquivo XML ou CSV, conexão com o banco de dados, uma sessão do Hibernate, uma coleção de objetos em memória, etc.
- Permite o uso de diagramas, gráficos, e até códigos de barras.

A ferramenta *JasperReports* permite a construção de relatórios, desde o mais simples ao mais complexo para aplicações Java ou diretamente em aplicações web, e é um dos mais utilizados no mundo. Permite a inserção de fórmulas nos relatórios e também o recebimento de dados de um sistema, através de uma conexão direta a um banco de dados relacional.

2.8 PADRÃO DE PROJETOS UTILIZANDO CAMADAS

Padrões de projeto podem ser vistos como uma solução que já foi testada para um problema. Desta forma , um padrão de projeto geralmente descreve uma solução ou uma instância da solução que foi utilizada para resolver um problema específico. Padrões de projetos são soluções para problemas que alguém um dia teve e resolveu aplicando um modelo que foi documentado e que você pode adaptar integralmente ou de acordo com necessidade de sua solução (Macoratti).

O padrão MVC, divide o software em três camadas: modelo, visão e controle, permitindo assim o desenvolvedor isolar as funções de cada classe em suas respectivas camadas, facilitando a manutenção do sistema.

A camada modelo contém as classes que descrevem os objetos da aplicação, por exemplo, alunos, disciplinas, turmas, professores, etc. À primeira vista, os objetos do tipo Modelo podem ser vistos como a primeira camada de interação com qualquer banco de dados que você possa estar usando na sua aplicação. Mas em geral eles representam os principais conceitos em torno do qual você programa suas aplicações.

A camada Visão contém as classes que fazem a interação com o usuário. Uma aplicação pode utilizar mais de uma interface diferente, e pode trocar de interface, sem que seja necessário interferir com o restante da aplicação. Para conseguir isso, a interface captura as solicitações do usuário e as converte em mensagens para o controlador. As exceções lançadas pelo controlador são capturadas pelos métodos da visão que geram as ações correspondentes para repassá-las ao usuário. Para poder fazer isso, a camada deve ter acesso a objetos controladores.

A camada controle lida com as requisições dos usuários. Responsável por retornar uma resposta com a ajuda das camadas Modelo e Visão. Os Controles tem a função de receber as solicitações da camada visão, buscar os dados através da camada modelo, e enviar para a camada visão convertendo de forma adequada para a mesma.

2.9 ANÁLISE E OBJETIVOS FUNCIONAIS DO SISTEMA

O objetivo é desenvolver um aplicativo para gerenciamento de oficinas, seguindo os requisitos abaixo:

Manter

- Cliente
- Fornecedor
- Produto
- Usuários
- Veículos
- Serviços

Pesquisar

- Venda
- Itens de Venda
- Serviços efetuados

Visualizar Relatório

- Estoque
- Serviços

2.9.1 DIAGRAMA DE CASO DE USO

Um diagrama de Caso de Uso descreve uma funcionalidade proposta para um novo sistema que será projetado. Pode-se dizer que um Caso de Uso é um documento narrativo que descreve uma sequência de passos que um ou mais atores utilizam para realizar com sucesso um determinado processo. (BOOCH; JACOBSON; RUMBAUGH, 2005)

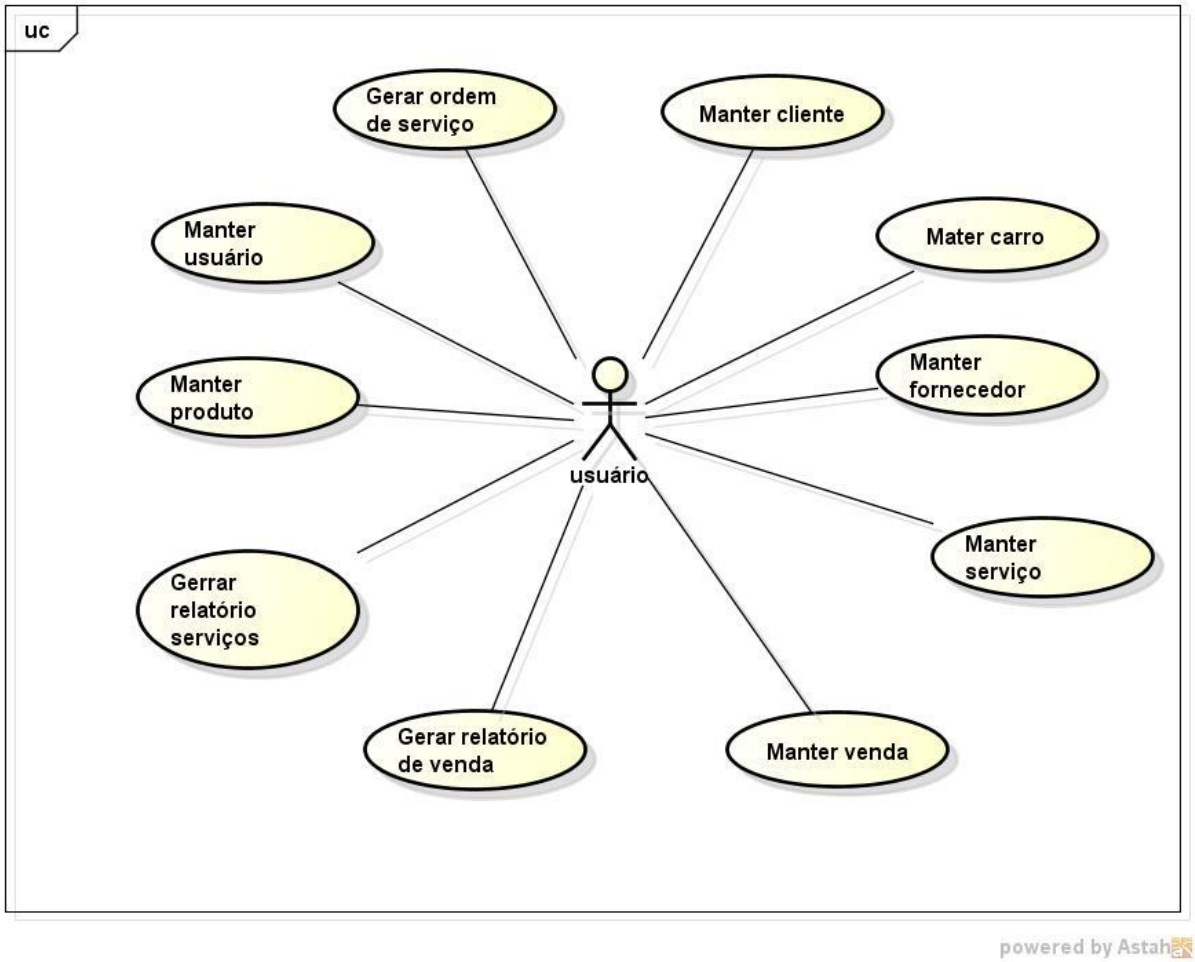

Figura 1 DIAGRAMA DE CASO DE USO

2.9.2 Narrativa UC

Caso de Uso: Manter Cliente.

Ator: Usuário

Fluxo Principal

1. O ator inicia o caso de uso selecionando cadastro de clientes.

2. O sistema oferece opções de manutenção.

3. O ator informa que deseja incluir um novo cliente. **[A1, A2]**

4. O sistema oferece a interface para inclusão.

5. O ator entra com as informações e seleciona salvar. **[E1]**

6. O sistema informa que as informações foram salvas e encerra o caso de uso

Fluxo Alternativo A1: Alteração

3. O ator consulta um cliente para alteração.

4. O sistema oferece o cliente para alteração. **[E2]**

5. O ator entra com as informações e seleciona salvar.

6. O sistema informa que as informações foram salvas e encerra o caso de uso

Fluxo Alternativo A2: Exclusão

3. O ator consulta um cliente para exclusão. **[E2]**

4. O sistema oferece o cliente para a exclusão.

5. O sistema solicita a confirmação da exclusão.

6. O ator confirma a exclusão.

7. O sistema confirma que a exclusão foi efetuada e encerra o caso de uso.

Fluxo de Exceção E1: Cliente Já Cadastrado.

6. O sistema informa que o cliente já possui cadastro e não salvas as informações.

7. O sistema retorna ao passo 2 do Fluxo Principal.

Fluxo de Exceção E2: Cliente Não Cadastrado.

4. O sistema informa que o cliente não esta cadastrado.

5. O sistema retorna para o passo 2 do Fluxo Principal.

Caso de Uso: Manter Veículo.
Ator: Usuário.

Fluxo Principal

1. O ator inicia o caso de uso selecionando cadastro de veículos.

2. O sistema oferece opções de manutenção.

3. O ator informa que deseja incluir um novo veículo. **[A1, A2]**

4. O sistema oferece a interface para inclusão.

5. O ator entra com as informações e seleciona salvar. **[E1]**

6. O sistema informa que as informações foram salvas e encerra o caso de uso

Fluxo Alternativo A1: Alteração

3. O ator consulta um veículo para alteração.

4. O sistema oferece o veículo para alteração. **[E2]**

5. O ator entra com as informações e seleciona salvar.

6. O sistema informa que as informações foram salvas e encerra o caso de uso

Fluxo Alternativo A2: Exclusão

3. O ator seleciona um veículo para exclusão.

4. O sistema oferece o veículo para exclusão.

5. O sistema solicita a confirmação da exclusão.

6. O ator confirma a exclusão.

7. O sistema confirma que a exclusão foi efetuada e encerra o caso de uso.

Fluxo de Exceção E1: Veículo Já Cadastrado.

6. O sistema informa que o veículo já possui cadastro e não salva as informações.

7. O sistema retorna ao passo 2 do Fluxo Principal.

Fluxo de Exceção E2: Veículo Não Cadastrado.

4. O sistema informa que o veículo não esta cadastrado.

5. O sistema retorna para o passo 2 do Fluxo Principal.

Caso de Uso: Manter Usuário.

Ator: Usuário

Fluxo Principal

1. O ator inicia o caso de uso selecionando cadastro de usuário.

2. O sistema oferece opções de manutenção.

3. O ator informa que deseja incluir um novo usuário. **[A1, A2]**

4. O sistema oferece a interface para inclusão.

5. O ator entra com as informações e seleciona salvar. **[E1]**

6. O sistema informa que as informações foram salvas e encerra o caso de uso

Fluxo Alternativo A1: Alteração

3. O ator consulta um usuário para alteração.

4. O sistema oferece o usuário para alteração. **[E2]**

5. O ator entra com as informações e seleciona salvar.

6. O sistema informa que as informações foram salvas e encerra o caso de uso

Fluxo Alternativo A2: Exclusão

3. O ator seleciona um usuário para exclusão.

4. O sistema oferece o usuário para a exclusão.

5. O sistema solicita a confirmação da exclusão.

6. O ator confirma a exclusão.

7. O sistema confirma que a exclusão foi efetuada e encerra o caso de uso.

Fluxo de Exceção E1: Usuário Já Cadastrado.

6. O sistema informa que o usuário já possui cadastro e não salvas as informações.

7. O sistema retorna ao passo 2 do Fluxo Principal.

Fluxo de Exceção E2: Usuário Não Cadastrado.

4. O sistema informa que o usuário não esta cadastrado.

5. O sistema retorna para o passo 2 do Fluxo Principal.

Caso de Uso: Manter Fornecedor.

Ator: Usuário

Fluxo Principal

1. O ator inicia o caso de uso selecionando cadastro de fornecedores.

2. O sistema oferece opções de manutenção.

3. O ator informa que deseja incluir um novo fornecedor. **[A1, A2]**

4. O sistema oferece a interface para inclusão.

5. O ator entra com as informações e seleciona salvar. **[E1]**

6. O sistema informa que as informações foram salvas e encerra o caso de uso

Fluxo Alternativo A1: Alteração

3. O ator consulta um fornecedor para alteração.

4. O sistema oferece o fornecedor para alteração. **[E2]**

5. O ator entra com as informações e seleciona salvar.

6. O sistema informa que as informações foram salvas e encerra o caso de uso

Fluxo Alternativo A2: Exclusão

3. O ator consulta um fornecedor para exclusão.

4. O sistema oferece o fornecedor para exclusão.

5. O sistema solicita a confirmação da exclusão.

6. O ator confirma a exclusão.

7. O sistema confirma que a exclusão foi efetuada e encerra o caso de uso.

Fluxo de Exceção E1: Fornecedor Já Cadastrado.

6. O sistema informa que o fornecedor já possui cadastro e não salvas as informações.

7. O sistema retorna ao passo 2 do Fluxo Principal.

Fluxo de Exceção E2: Fornecedor Não Cadastrado.

4. O sistema informa que o fornecedor não esta cadastrado.

5. O sistema retorna para o passo 2 do Fluxo Principal.

Caso de Uso: Manter Produto.

Ator: Usuário

Fluxo Principal

1. O ator inicia o caso de uso selecionando cadastro de produtos.

2. O sistema oferece opções de manutenção.

3. O ator informa que deseja incluir um novo produto. **[A1, A2]**

4. O sistema oferece a interface para inclusão.

5. O ator entra com as informações e seleciona salvar. **[E1]**

6. O sistema informa que as informações foram salvas e encerra o caso de uso

Fluxo Alternativo A1: Alteração

3. O ator consulta um produto para alteração.

4. O sistema oferece o produto para alteração. **[E2]**

5. O ator entra com as informações e seleciona salvar.

6. O sistema informa que as informações foram salvas e encerra o caso de uso

Fluxo Alternativo A2: Exclusão

3. O ator seleciona um produto para exclusão.

4. O sistema oferece o produto para exclusão.

5. O sistema solicita a confirmação da exclusão.

6. O ator confirma a exclusão.

7. O sistema confirma que a exclusão foi efetuada e encerra o caso de uso.

Fluxo de Exceção E1: Produto Já Cadastrado.

6. O sistema informa que o produto já possui cadastro e não salvas as informações.

7. O sistema retorna ao passo 2 do Fluxo Principal.

Fluxo de Exceção E2: Produto Não Cadastrado.

4. O sistema informa que o produto não esta cadastrado.

5. O sistema retorna para o passo 2 do Fluxo Principal.

Caso de Uso: Manter Serviço.

Ator: Usuário.

Fluxo principal.

1. O ator inicia o caso de uso selecionando cadastro de serviços.

2. O sistema oferece opções de manutenção.

3. O ator informa que deseja incluir um novo serviço. **[A1, A2]**

4. O sistema oferece a interface para inclusão.

5. O ator entra com as informações e seleciona salvar. **[E1]**

6. O sistema informa que as informações foram salvas e encerra o caso de uso

Fluxo Alternativo A1: Alteração

3. O ator consulta um serviço para alteração.

4. O sistema oferece o serviço para alteração. **[E2]**

5. O ator entra com as informações e seleciona salvar.

6. O sistema informa que as informações foram salvas e encerra o caso de uso

Fluxo Alternativo A2: Exclusão

3. O ator seleciona um serviço para exclusão.

4. O sistema oferece o serviço para exclusão.

5. O sistema solicita a confirmação da exclusão.

6. O ator confirma a exclusão.

7. O sistema confirma que a exclusão foi efetuada e encerra o caso de uso.

Fluxo de Exceção E1: Serviço Já Cadastrado.

6. O sistema informa que o serviço já possui cadastro e não salvas as informações.

7. O sistema retorna ao passo 2 do Fluxo Principal.

Fluxo de Exceção E2: Serviço Não Cadastrado.

4. O sistema informa que o serviço não esta cadastrado.

5. O sistema retorna para o passo 2 do Fluxo Principal.

Caso de Uso: Gerar Ordem de Serviço.

Ator: Usuário

Fluxo Principal.

1. O ator inicia o caso de uso selecionando gerar ordem de serviço.

2. O sistema oferece a interface de ordem de serviço.

3. O ator seleciona um cliente e veículo para a ordem de serviço.

4. O sistema retorna as informações do cliente e do veículo.

5. O ator seleciona os serviços desejados para o veículo.

6. O ator seleciona o grupo de produtos.

7. O sistema apresenta os produtos do grupo selecionado.

8. O ator seleciona os produtos desejados. **[E1]**

9. O sistema calcula os preços e impostos dos produtos e serviços.

10. O ator informa que deseja finalizar a ordem de serviço.

11. O sistema salva as informações, imprime a ordem de serviço e encerra o caso de uso.

Fluxo de exceção [E1]: Quantidade de produto insuficiente.

10. O sistema informa que a quantidade não esta disponível no estoque.

11. O sistema informa o ator que é necessário realizar solicitação do produto.

12. O sistema retorna para o passo 7 do Fluxo Principal.

Caso de Uso: Gerar Relatório de Serviço.

Ator: Usuário

Fluxo principal.

1. O ator inicia o caso de uso selecionando gerar relatório de serviço.

2. O sistema oferece a interface para gerar relatório de serviço.

3. O ator insere as informações e confirma.

4. O sistema gera o relatório, imprime e encerra o caso de uso.

Caso de Uso: Gerar Relatório de Estoque.

Ator: Usuário

Fluxo principal.

1. O ator inicia o caso de uso selecionando gerar relatório de estoque.

2. O sistema oferece a interface para gerar relatório de estoque.

3. O ator insere as informações e confirma.

4. O sistema gera o relatório, imprime e encerra o caso de uso.

Caso de Uso: Gerar Relatório de Venda.
Ator: Usuário

Fluxo principal.

1. O ator inicia o caso de uso selecionando gerar relatório de venda.

2. O sistema oferece a interface para gerar relatório de venda.

3. O ator insere as informações e confirma.

4. O sistema gera o relatório, imprime e encerra o caso de uso.

2.9.3 DIAGRAMA DE CLASSE

Um diagrama de classes representa a estrutura e relações entre classes que servem de modelo para objetos.

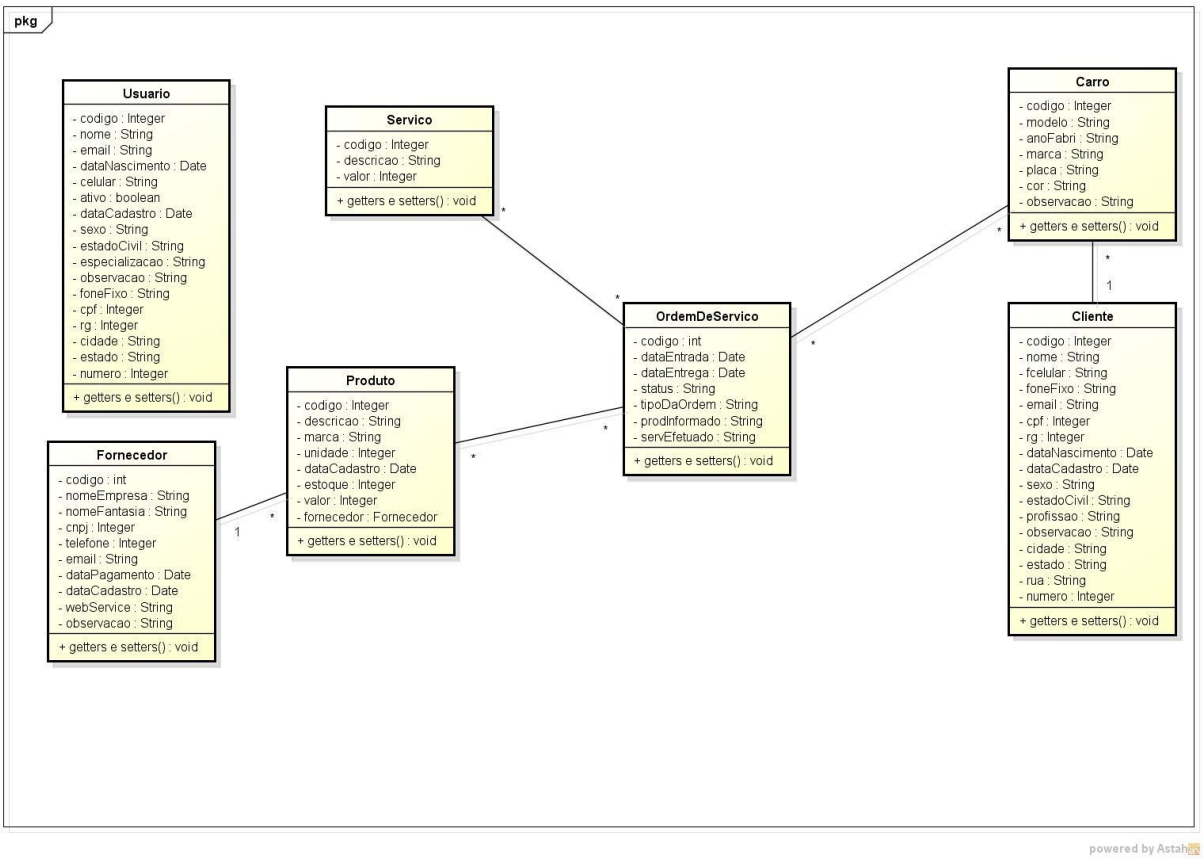

Figura 2 DIAGRAMA DE CLASSE

2.9.4 DIAGRAMA DE ATIVIDADES

O diagrama de atividades representa os fluxos conduzidos por processamentos. É essencialmente um gráfico de fluxo, mostrando o fluxo de controle de uma atividade para outra (BOOCH; JACOBSON; RUMBAUGH, 2000).

2.9.5 GERAR ORÇAMENTO

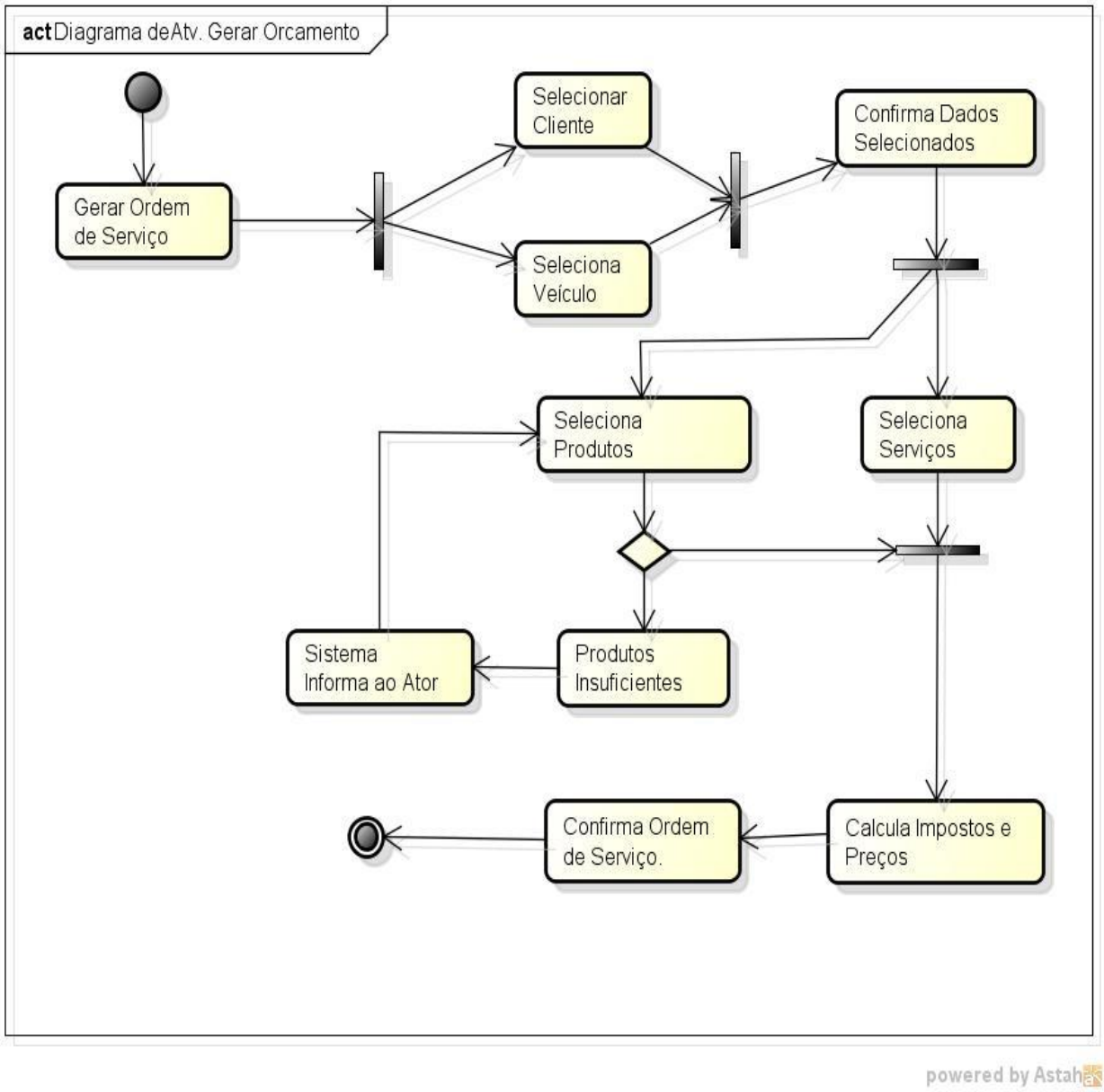

Figura 3 GERAR ORÇAMENTO

2.9.6 MANTER VEICULO

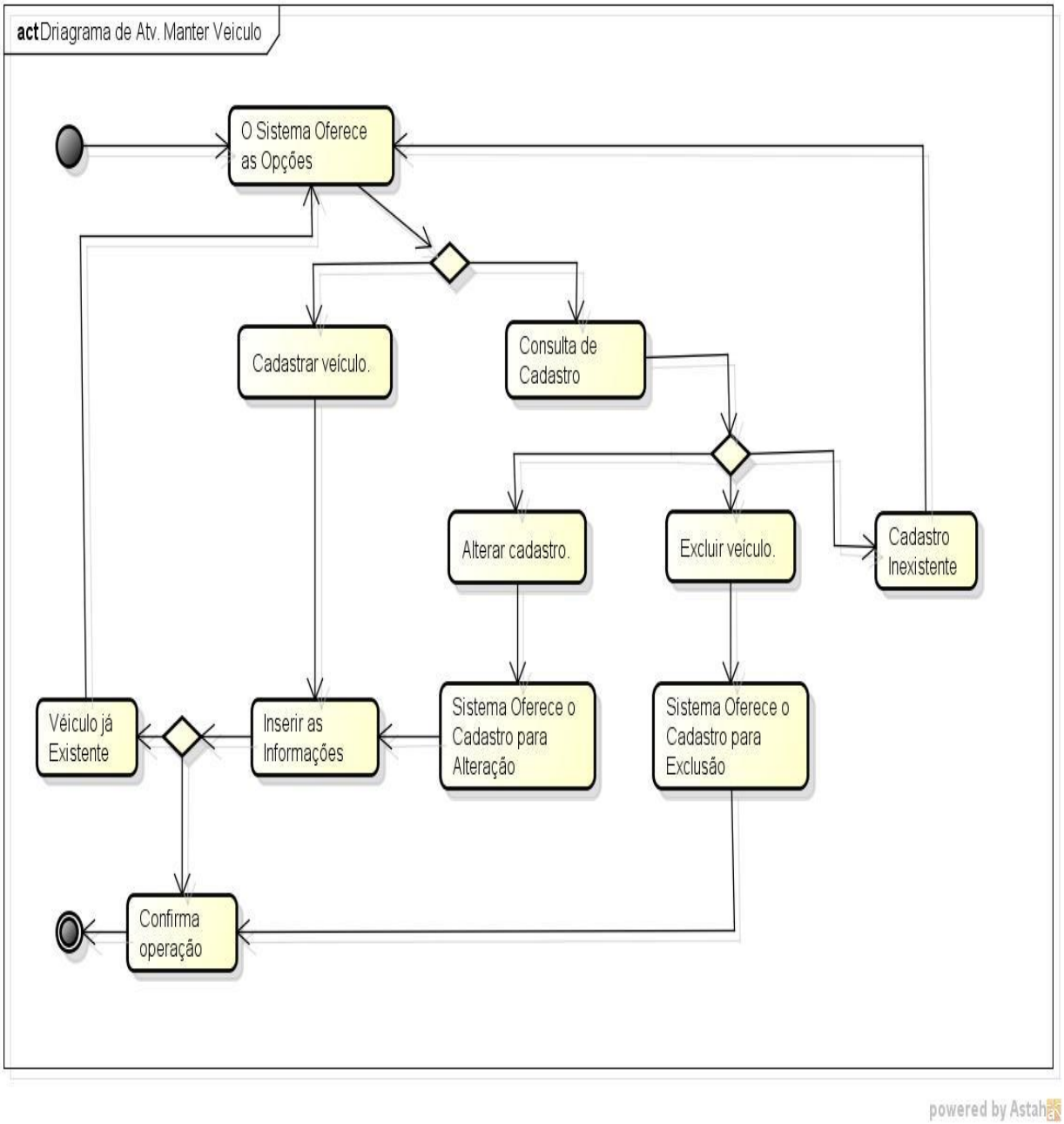

Figura 4 MANTER VEICULO

2.9.7 DIAGRAMA ENTIDADE RELACIONAMENTO

O Diagrama Entidade-Relacionamento tem o objetivo de representar as estruturas de dados da forma visual.

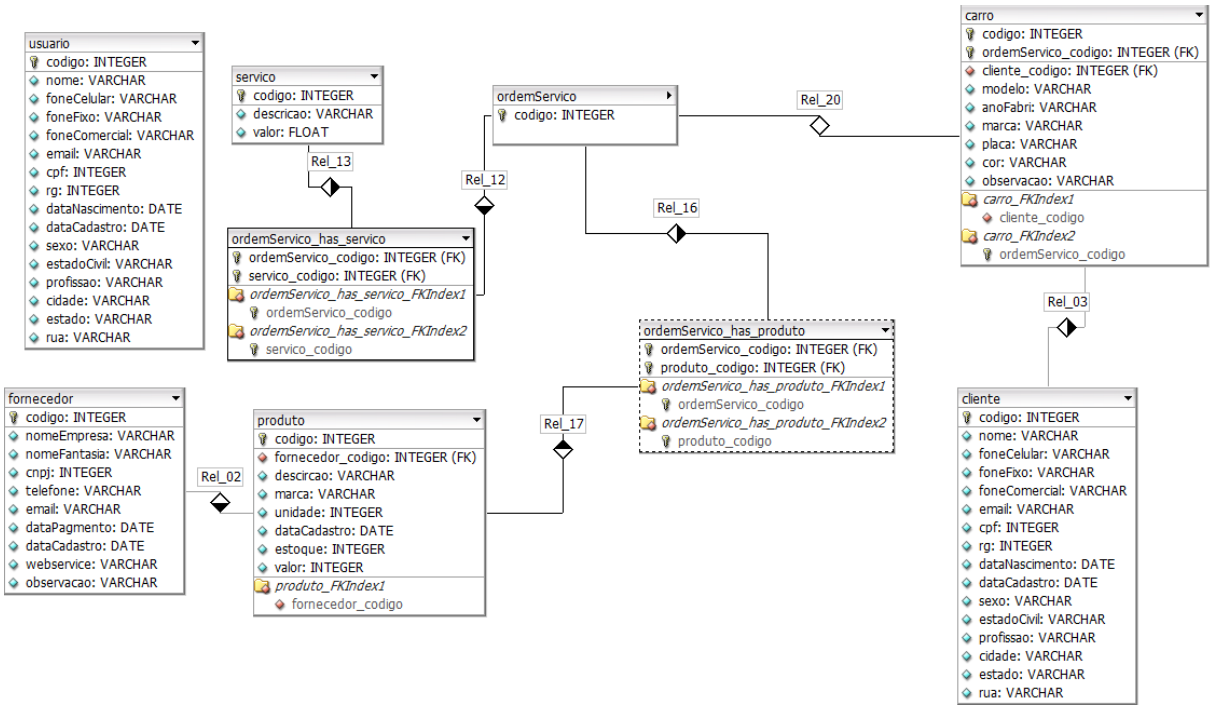

Figura 5 DIAGRAMA ENTIDADE RELACIONAMENTO

2.9.8 ESTRUTURA DE DESENVOLVIMENTO DO SISTEMA

O sistema foi desenvolvido com base na Estrutura Analítica de Trabalho (WBS Work Breakdown Structure), estrutura que subdivide os trabalhos de desenvolvimento em componentes menores para facilitar o gerenciamento das etapas.

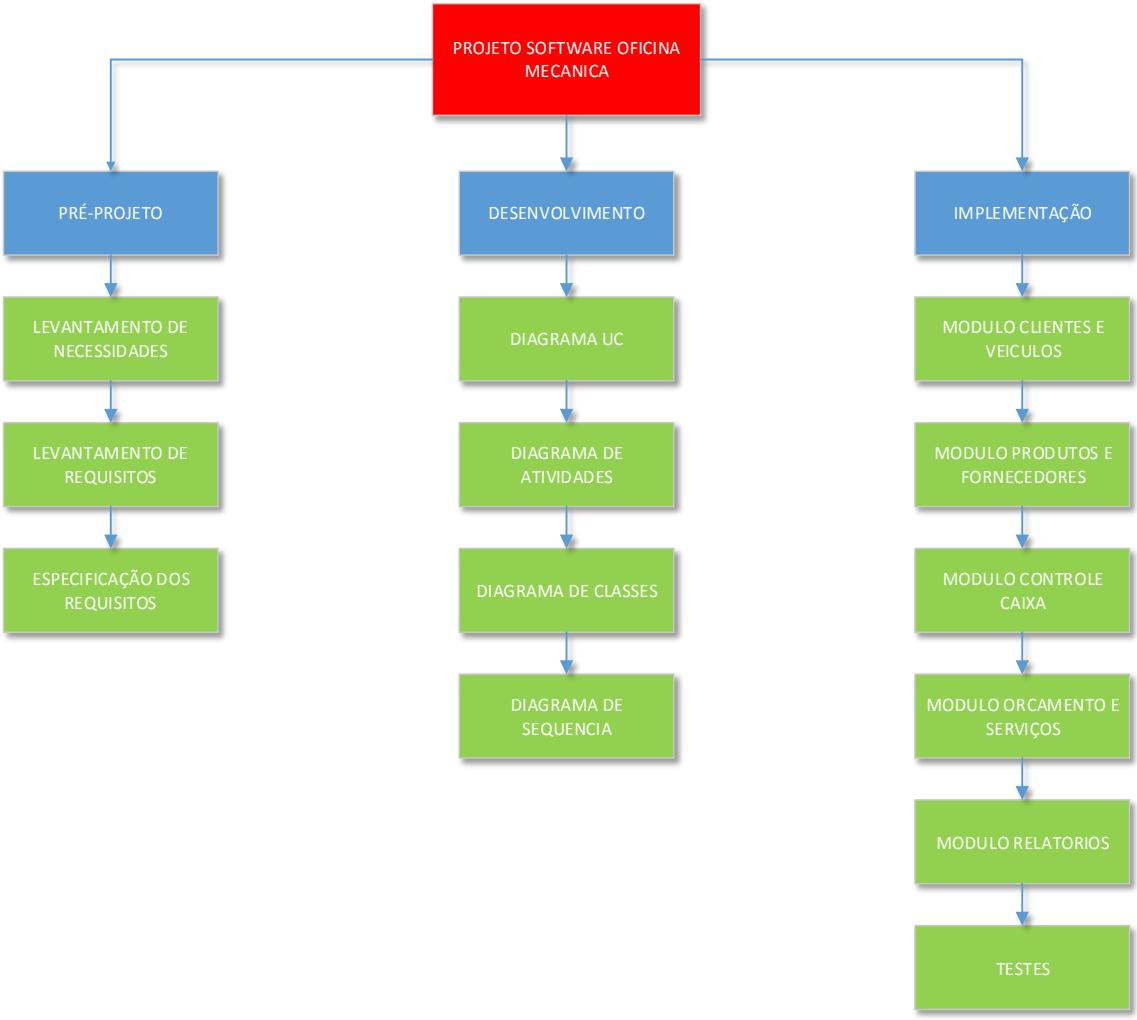

Figura 6 DIAGRAMA WBS

2.9.9 SEQUENCIAMENTO DAS ATIVIDADES

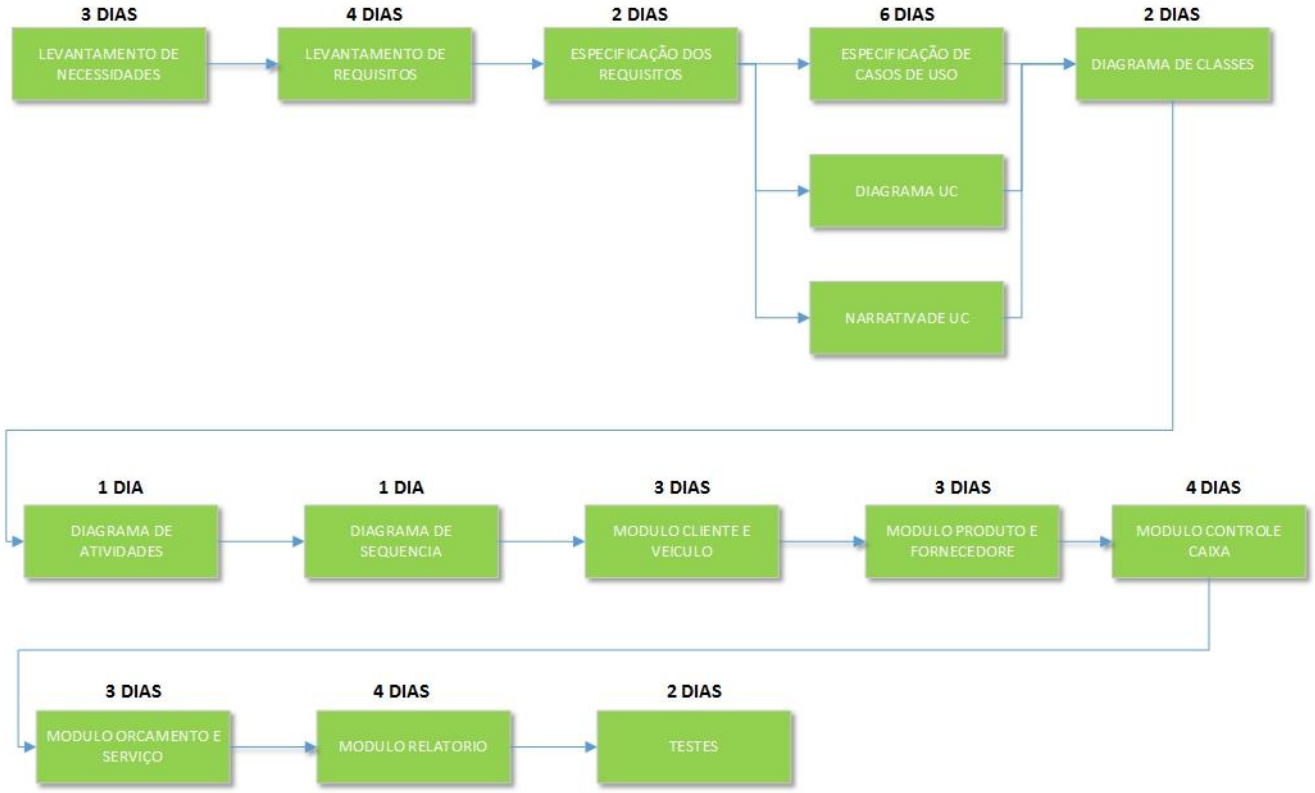

Figura 7 SEQUENCIAMENTO DAS ATIVIDADES

4 ORÇAMENTO DO PROJETO E ESTIMATIVA DE CUSTOS

Custo Analista – Programador

Custo Diário: R$ 80,00 (Oitenta Reais);

Total de Dias: 30;

Custo Total: (30 * 80,00) = R$ 2.400,00 (Dois mil quatrocentos reais).

01 computadores

o Valor unitário = R$2.000,00

o Dias (de uso) = 26 dias

o Depreciação = R$2.000,00 / 24 meses (02 anos. Tempo de depreciação) = R$83,34/mês

30 dias _ 83,34

26 dias _ x

X = R$72,23

o Custo nos 26 dias = R$72,23 * 2 computadores = R$ 144,46

01 impressora

o Valor = R$400,00

o Dias (de uso) = 26 dias

o Depreciação = R$400,00 / 24 = R$16,67

30 dias _ 16,67

26 dias _ x

X = R$14,45

o **Custo da Impressora = R$14,45**

Sistema Operacional Microsoft Windows 7 64 bits: R$ 300,00 (trezentos reais).

Depreciação 2 anos: R$ 300,00 / 24 (meses) = R$ 12,50 (doze reais e cinquenta centavos) por mês.

Custo de um dia: R$ 12,50 / 30 (dias) = R$ 0,41 (quarenta e um centavos) por dia.

Custo de 26 dias * R$ 0,41=10,66

IDE NETBEANS

Gratuito.

SGBD POSTGRESQL

Gratuito.

Java

Gratuito.

Custo Total do Projeto = R2.400,00 + R$144,46 + R$14,45 + R$10,66 =**2570,57**

5. IMPLEMENTAÇÃO DA APLICAÇÃO

Para a implementação do sistema HELPMEC, foi utilizado o ambiente de Desenvolvimento *Netbeans,* juntamente com a linguagem de programação Java, *framework Hibernate* e banco de dados Postgresql. A figura 8 mostra os pacotes na IDE Netbeans

5.1 – ORGANIZAÇÃO DOS PACOTES E CLASSES DO SISTEMA

Para uma melhor organização, o sistema foi organizado em pacotes. A figura 8 apresenta os pacotes.

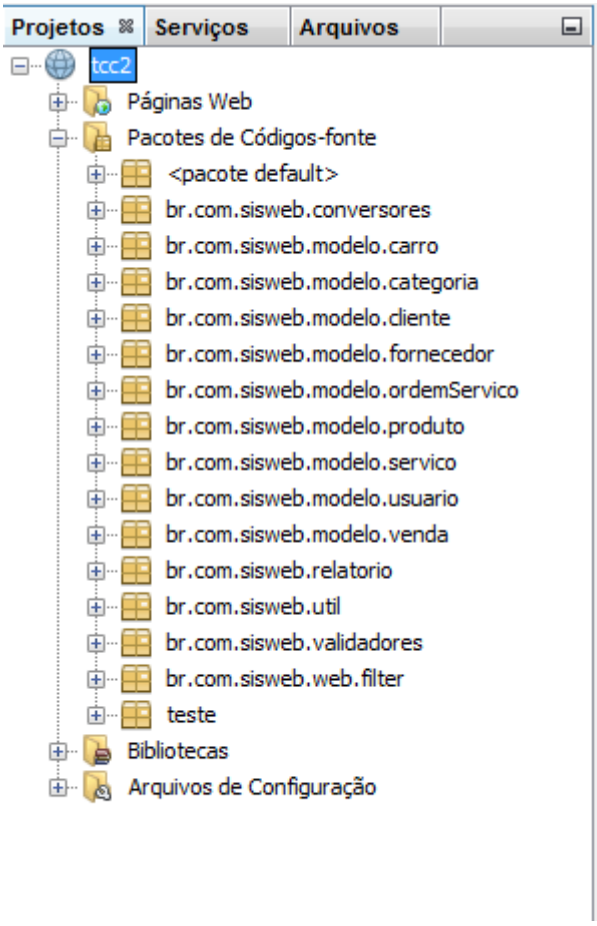

Figura 8 ORGANIZAÇÃO DE PACOTES

Pacote conversores: Onde contém as classes que convertem valores para armazenamento.

Pacote: br.com.sisweb.util: É nele que ficam as classes, que criam a conexão realizam o acesso ao banco de dados através do *Hibernate*.

Pacote: br.com.sisweb.validadores: É nele que ficam as classes, que tem funções de validar dados.

Pacote: br.com.sisweb.filter: É nele que ficam as classes, que a função de filtrar e tratar possíveis erros.

A figura 9 representa o modelo do pacote produto.

Figura 9 CLASSE PRODUTO

```
public class Produto implements Serializable {

    private static final long serialVersionUID = -3445365595630310499L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "cod_produto")
    private Integer codigo;

    private String descricao;

    private String marca;

    private String unidade;

    @Column(name = "data_cadastro")
    @Temporal(javax.persistence.TemporalType.DATE)
    private Date dataCadastro;

    private Integer estoque;

    private Integer valor;

    private Integer valorCusto;

    @ManyToMany(mappedBy = "produtosList")
    private List<OrdemServico> ordens;
```

A figura 10 representa o controle do pacote produto

Figura 10 CLASSE PRODUTODAOHIBERNATE

```
public class ProdutoDAOHibernate implements ProdutoDAO {

    private Session session;
    Categoria categoria = new Categoria();

    public void setSession(Session session) {
        this.session = session;
    }

    @Override
    public void salvar(Produto produto) {
        this.session.save(produto);
    }

    @Override
    public void atualizar(Produto produto) {
        this.session.update(produto);
    }

    @Override
    public void excluir(Produto produto) {
        this.session.delete(produto);
    }

    @Override
    public Produto retornaPorCodigo(Integer codigo) {
        Criteria criteria = this.session.createCriteria(Produto.class).add(Restrictions.eq("codigo", codigo));
        Produto produto = (Produto) criteria.uniqueResult();
        return produto;
    }
```

A classe ProdutoRN, figura 11, representa a camada visão, responsável por fazer a filtragem das regras de negócio do sistema, trabalha em conjunto com a classe ProdutoMB, figura 12, do sistema recebendo as requisições do usuário.

Figura 11 CLASSE PRODUTORN

```
public class ProdutoRN {

    private final ProdutoDAO produtoDAO;

    public ProdutoRN() {
        produtoDAO = DAOFactory.criarProdutoDAO();
    }

    public void salvar(Produto produto) {
        this.produtoDAO.salvar(produto);
    }

    public void atualizar(Produto produto) {
        this.produtoDAO.atualizar(produto);
    }

    public void excluir(Produto produto) {
        this.produtoDAO.excluir(produto);
    }

    public List<Produto> listar() {
        return this.produtoDAO.listar();
    }

    public Produto pesquisarPorDescricao(String descricao) {
        return this.produtoDAO.pesquisarPorDescricao(descricao);
    }

    public Produto retornaPorCodigo (Integer codigo){
      return this.produtoDAO.retornaPorCodigo(codigo);
    }
```

A classe ProdutoMB, figura 12, faz a junção da tela, com a aplicação, permitindo assim que as requisições do usuário cheguem as classes Java.

Figura 12 CLASSE PRODUTOMB

```
@ManagedBean
@ViewScoped
public class ProdutoMB implements Serializable{

    private Produto produto;
    private String pagina;
    private List<Produto> produtos;
    private List<SelectItem> produtoSelect;

    public ProdutoMB(){
        this.produtos = new ArrayList<Produto>();
        this.pagina="produto";
        this.produto = new Produto();
    }

     public String inserir() {
        ProdutoRN pRN = new ProdutoRN();

        if (this.produto.getCodigo() != null && this.produto.getCodigo() != 0) {
            pRN.atualizar(getProduto());
            FacesMessage msg = new FacesMessage("ATUALIZADO COM SUCESSO!!!");
            FacesContext.getCurrentInstance().addMessage(null, msg);
        } else {
            pRN.salvar(getProduto());
            FacesMessage msg = new FacesMessage("CADASTRADO COM SUCESSO!!!");
            FacesContext.getCurrentInstance().addMessage(null, msg);
        }
        novo();
        return pagina;
    }
```

5.2 – ORGANIZAÇÃO DAS PÁGINAS DO SISTEMA

As páginas estão armazenadas no diretório “Paginas web”, contém as páginas xhtml, para cadastro e consulta, como visto na figura 9 tem-se também um diretório resources.

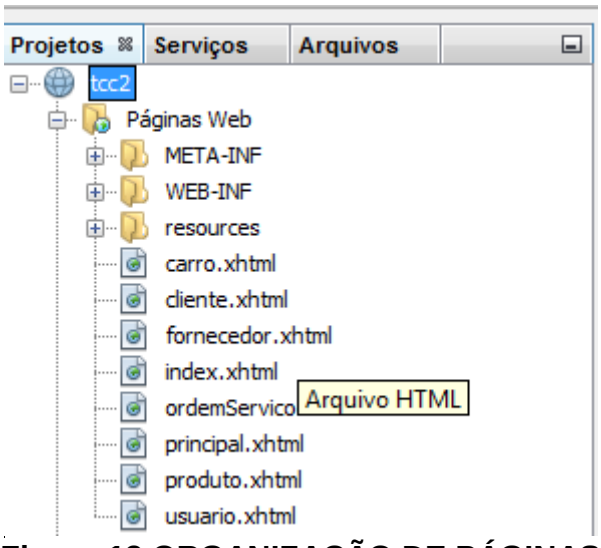

Figura 13 ORGANIZAÇÃO DE PÁGINAS

5.4 – INTERFACE DO SISTEMA

Ao acessar o sistema, abrirá as opções contendo um menu no lado esquerdo: Cadastros, Movimentações, Consultas, Relatórios. Conforme a Figura 10:

Figura 14 INTERFACE

6- CONCLUSÃO

O software *Helpmec* foi desenvolvido para facilitar o gerenciamento de oficinas, agilizando processos e reduzindo gastos. Tudo realizado de forma flexível e simples, tendo em vista uma melhor facilidade no controle das informações satisfazendo assim as necessidades dos clientes.

Na fase de levantamento de requisitos, foram definidas todas as funcionalidades para o sistema, e depois de realizada esta fase, foi feita as modelagens do sistema, como: caso de uso, diagrama de classe, diagrama entidade relacionamento, e diagrama de sequência. Diagramas estes que foram de grande importância no desenvolvimento do sistema, pois ajudaram a ter uma visão completa do sistema.

Uma das dificuldades no desenvolvimento do sistema foi o inicio do aprendizado da linguagem junto com o inicio do desenvolvimento do sistema, tal dificuldade foi sanada com ajuda dos professores e leitura de livros.

As tecnologias empregadas mostraram-se eficientes no desenvolvimento do sistema, pois oferece várias ferramentas para auxiliar o desenvolvimento e grandes fóruns de ajuda na Internet.

Para desenvolvimentos futuros, pretende-se iniciar o desenvolvimento de mais funcionalidade para o sistema permitindo assim atender todas as necessidades dos futuros clientes.

7-REFERÊNCIAS

Astah. Disponível em:

<http://astah.net/ / > Acesso em: 15 de Fevereiro de 2014

BOOCH, Grady; JACOBSON, Ivar; RUMBAUGH, James. **UML Essencial** –Um breve guia para a linguagem-padrão de modelagem de objetos. 2°Edição. Tradução de Vera Pezerico e Christian Thomas. Porto Alegre: Bokkman, 2000.

BOOCH, Grady; JACOBSON, Ivar; RUMBAUGH, James. **UML Guia do Usuário**. 2° Edição. Tradução Fábio Freitas da Silva e Cristiana de Amorim Machado. Rio de Janeiro: Elsevier, 2005

Deitel, Harvei M. **Java Como Programar**, 8º Edição/ Harvei M. Deitel & Paul J. Deitel; tradução Edson FurmanKiewicz; revisão técnica Fabio Luis Picelli Lucchini.- São Paulo: Pearson Prentice Hall, 2010.

Edson Golçalves. Desenvolvendo Relatórios Profissionais com iReport para Netbeans IDE**. 1° edição. Editora Ciência Moderna.** 2009.

Hibernate Disponível em:

<http://hibernate.org/> Acesso em 03 de Abril de 2014

Java Reporting com JasperReports e iReport Open Source
Apostila. Sandro Miguel
JasperReport Disponível em:
<http://community.jaspersoft.com/> Acesso em 10 de Março de 2014

Java. Disponível em :

<https://www.java.com/pt_BR/> Acesso em 02 de Abril de 2014

Luckow,Décio Heinzelmann **Programação Java Para Web**/ Décio Heinzelmann Luckow e Alexandre Altair de Melo. São Paulo : Novatec Editora, 2010.

MACORATTI; **Diagrama de Classes.** Disponível em:
< http://www.macoratti.net> Acesso em: 08 de Maio de 2014

MAULO, Fabio; **NHibernate.** Disponível em:< https://community.jboss.org> Acesso em: 10 de Maio de 2014.

Netbeans. Disponível em:

<https://www.java.com/pt_BR/> Acesso em 02 de Fevereiro de 2014

Postgresql Disponível em:

<http://www.postgresql.org/> Acesso em 06 Abril de 2014

Thiago Faria**. Java EE7 com JSF,PrimeFaces e CDI**. 2013

Vanessa Rocha Solgate. **Apostila sobre o Banco de Dados Postgres**. 2005.

ANEXO – CRONOGRAMA

O cronograma abaixo refere-se ao fluxo de tarefas cumpridas .

Tarefa	Inicio	termino	dias	Jan/14	Fev/14	Mar/14	Abr/14	Mai/14	Jun/14	Jul/14	Ago/8
Levantamento de necessidades	02/01	12/01	10d								
Levantamento de requisitos	13/01	23/01	10d								
Especificação de requisitos	23/01	31/01	7d								
Especificação de caso de uso	01/02	07/02	7d								
Diagrama de caso de uso	08/02	14/02	7d								
Narrativa de caso de uso	15/02	28/02	13d								
Diagrama de classe	01/03	05/03	5d								
Diagrama de atividade	06/03	12/03	6d								
Diagrama de sequencia	12/03	17/03	6d								
Programação	18/03	30/06	103d								
Testes	01/07	31/08									
Entrega	01/09										