FACULDADE INTERAMERICANA DE PORTO VELHO

LUIZ COSTA VELOSO

TRABALHO DE CONCLUSÃO DE CURSO

SISTEMA DE INFORMAÇÃO PARA CONTROLE DE SERVIÇOS E ESTOQUE DE PEÇAS - SICSEP

Porto Velho

2014

LUIZ COSTA VELOSO

TRABALHO DE CONCLUSÃO DE CURSO

SISTEMA DE INFORMAÇÃO PARA CONTROLE DE SERVIÇOS E ESTOQUE DE PEÇAS

Trabalho de conclusão de curso apresentado como requisito obrigatório para obtenção do grau de Bacharel na Faculdade Interamericana de Porto Velho - UNIRON no Curso de Sistemas de Informação.

Orientadores: José Avani das Chagas Junior, Marcelo de Araújo Rech e Nayme Petrus Abi Abib.

Porto Velho

2014

LUIZ COSTA VELOSO

SISTEMA DE INFORMAÇÃO DE CONTROLE DE SERVIÇOS E ESTOQUE DE PEÇAS

Trabalho de conclusão de curso apresentado a Faculdade Interamericana de Porto Velho – UNIRON como parte dos requisitos para obtenção do grau de bacharelado em Sistemas de Informação no curso de Sistemas de Informação.

Aprovado no dia 14 de junho de 2014.

_____
Prof. Ms Autran Dias de Almeida
Coordenador do Curso

BANCA EXAMINADORA

_____
Prof. Esp. Marcelo Araújo Rech
Presidente da Banca

_____
Prof. Esp. José Avani das Chagas Junior

_____
Prof. Ms. Nayme Petrus Abi Abib
Coordenador de TCC

DEDICATÓRIA

A minha mãe, esposa, filhas e ao meu pai Francisco Reinaldo pelo exemplo de homem que é, também pela a paciência, amor, carinho, por serem minha inspiração e acreditado que eu iria conseguir concluir este curso, dedico-lhes com muito amor essa conquista como gratidão.

AGRADECIMENTO

Agradeço primeiramente a Deus pela oportunidade de enriquecer os meus conhecimentos, a minha esposa, Clarice Frank, pelo apoio, paciência e estimulo, ao professor Nayme Petrus, Ancelmo Luiz e em especial ao meu amigo Wesly Lopes que juntos contribuímos um com o outro para alcançar este objetivo com Vitória.

RESUMO

O objetivo do trabalho foi elaborar um sistema para Empresa Mix Motos que possibilite a efetivação de sistematização, tratamento, análise e integração das informações denominado Sistema de Informação para Controle de Serviços e Estoque de Peças – SICSEP tendo como objetivo principal controlar o Estoque e Serviços além de gerenciar informações dos clientes e funcionários por meio de um registro de dados previamente cadastradas. O trabalho foi executado em três fases distintas, análise, projeto e Implementação. Na fase de análise foram levantados todos os requisitos necessários para a elaboração do sistema, sendo realizadas reuniões com o cliente para que pudéssemos alcançar o resultado esperado. No projeto foram foi definidas todas as funcionalidades, as ferramentas de programação, modelagem, armazenamento de dados e criação de interface (Desenvolvimento de Telas). Por fim ficou a fase de implementação, onde se executou a codificação do sistema com base em todos os dados que foram obtidos no decorrer das etapas de análise e projeto. Para o desenvolvimento deste sistema foram utilizadas tecnologias atuais e bem sucedidas renomadas na orientação a objeto, utilizado o modelo de projeto MVC, linguagem C#, Framework Bootstrap, estes foram criados na ferramenta Visual Studio 2010 com banco de dados SQL Server 2008 juntamente com a ferramenta Enterprise Architect (EA).

Palavras-chave: Projeto, Análise, Implementação, Controlar, Gerenciar.

ABSTRACT

The objective was to develop a system for Motorcycles Mix Company that enables the execution of systematization, processing, analysis and integration of information called Information System for Service Control and Parts Inventory - SICSEP having as main objective to control the stock and Services beyond to manage customer information and employees through a record previously registered data. The work was carried out in three distinct phases, analysis, design and implementation. In the analysis phase all the necessary preparation for the system requirements, client meetings being held so we could reach the expected result were raised. In all design features, the programming tools, modeling, data storage and creation of interface (Development of Screens) have been defined. Finally got the implementation phase, where we performed the coding system based on all the data that were obtained during the stages of analysis and design. To develop this system renowned current and successful technologies in object orientation were used, used the model of MVC, C # language design, Framework Bootstrap, these were created in Visual Studio 2010 with SQL Server 2008 database tool with the tool Enterprise Architect (EA).

Keywords: Design, Analysis, Implementation, Control, Manage.

LISTA DE FIGURAS

LISTA DE QUADROS

ÍNDICE

1 INTRODUÇÃO

O mercado atual está altamente competitivo, forçando as empresas a buscarem, em sistemas de informação automatizados, soluções para aperfeiçoamento da qualidade de seus serviços e controle de seus produtos.

Quando se fala em qualidade de serviços de lojas de autopeças deve-se enfatizar a satisfação do cliente com o serviço oferecido e o período gasto com a execução do mesmo. Afinal, a grande maioria das pessoas utiliza seu transporte para fins de trabalho, com isso, o tempo demasiado gera prejuízo, diminui a possibilidade de mais faturamento e cria insatisfação para o cliente.

Em relação ao controle de estoque, não serve somente para saber a quantidade que se tem de determinado produto ou o que saiu e entrou, mas vai muito além, sendo de suma importância para a empresa, pois controlam os desperdícios, os possíveis desvios, apuram-se valores para fins de análise, bem como, verifica-se onde há um excessivo investimento, no qual prejudica o capital de giro.

Um sistema de informação possibilita avaliar os seus serviços, planejar suas compras de maneira organizada, dando uma visão sobre quais precisam de mais investimentos, eliminando assim produtos em estoque sem demanda e evitando a utilização de capital em compras desnecessárias baseado em dados precisos de toda sua logística.

Como não é diferente, a empresa Mix Motos também se encontra forçada a investir em melhorias para competir com a concorrência, buscando, em um sistema de informação, soluções para eliminar investimentos desnecessários e melhoria da qualidade dos serviços prestados.

Diante do exposto, foi desenvolvido um Sistema de Informação para Controle de Serviços e Estoque de Peças – SICSEP, que tem como objetivo gerenciar os dados dos clientes e funcionários através de um registro de informações previamente cadastradas, o estoque será mantido pelo sistema controlando a entrada e saída dos produtos, evitando assim, possíveis extravios, perdas, erros de conferência e efetivando uma melhor distribuição dos investimentos entre seus produtos. Quanto aos serviços, serão controlados pelo sistema identificando qual funcionário o realizou.

2 ANÁLISE

A análise do sistema SICSEP foi realizada após o levantamento de requisitos através de entrevistas com o cliente para abstrair suas necessidades e assim desenvolver um sistema eficaz e eficiente.

Complementa Silva (2007) afirmando que, o objetivo da etapa de análise é a modelagem do domínio do problema, em contraste com a modelagem da solução computacional.

2.1 Entrevistas

Através das entrevistas foi realizada a análise levantando-se todos os requisitos Funcionais, não funcionais e de informação necessário para o desenvolvimento do sistema que para tanto foram realizado as entrevistas qualitativa e quantitativa.

2.1.1 Entrevista Qualitativa

A entrevista qualitativa é realizada de uma forma aberta, onde o entrevistado é estimulado a expor livremente o que deseja do sistema, conforme em APÊNDICE C.

Para o sistema SICSEP esta entrevista teve como objetivo abstrair de uma forma macro na visão do cliente as funcionalidades a qual o sistema teria que realizar, servindo como base para a criação do sistema.

2.1.2 Entrevista Quantitativa

A entrevista quantitativa foi realizada através da elaboração de perguntas, de forma objetiva esclarecendo as dúvidas que fiquem da entrevista qualitativa, conforme APÊNDICE D, uma vez que a entrevista é conceituada por Lakatos (2009, p. 197) como “encontro de duas pessoas, a fim de que uma delas obtenha informações a respeito de determinado assunto, mediante uma conversação de natureza profissional”.

2.2 Descrição do Sistema Anterior

À Micro Empresa Mix Motos não possui um sistema de informação. Atualmente o processo de controle dos serviços realizados na loja pelos mecânicos e do estoque é feito todo manualmente pelo proprietário em blocos de pedido, onde são preenchidas as informações do mecânico, atendente, serviço, produto, moto, valores e cliente. No final do expediente é feito a conferência destes. Este sistema impossibilita o controle em tempo real e eficaz, forçando a empresa conferir o produto sempre que vai efetuar uma venda, realizar pedido também quando necessita saber de informações sobre os serviços que estão em andamento.

2.3 Objetivo do Sistema

O objetivo do sistema desenvolvido é gerenciar os serviços e estoque da empresa Mix Motos.

2.4 Resumo Executivo

O Sistema desenvolvido para empresa Mix Motos tem como nome Sistema de Informação para Controle de Serviços e Estoque de Peças – SICSEP que procederá da seguinte forma.

O cliente, ao chegar à loja, dirige-se ao balcão de atendimento onde o funcionário irá atender a sua solicitação. Se este for somente comprar peça, o atendente verificará a disponibilidade da mesma no sistema e efetuará a venda. Se o cliente desejar trocar a peça ou realizar algum serviço em sua moto, este deverá informar ao atendente os dados pessoais e do veículo para abertura de sua ordem de serviço.

Após esses procedimentos, o veículo será encaminhado à oficina onde o mecânico irá efetuar a troca ou identificar os problemas, informando os serviços e as peças que serão usados na manutenção, deixando em aberto o seu pedido até a finalização dos seguintes procedimentos:

- ✓ Concluído os serviços, o cliente vai ao caixa para efetuar o pagamento;
- ✓ O atendente de caixa finalizará o pedido;
- ✓ O sistema efetuará a baixa no estoque das peças que foram utilizadas;

- ✓ Clientes e funcionários serão controlados através de dados comerciais previamente registrados em um banco de dados;
- ✓ O estoque será mantido pelo sistema controlando a entrada e saída dos produtos, emitirá alerta de quantidade mínima em estoque e informará aos usuários os serviços que foram finalizados e os que estão em aberto;
- ✓ A segurança dos dados se dará através de backup automatizado;
- ✓ Informações adicionais poderão ser consultadas no manual do Sistema de Informação para Controle de Serviços e Estoque de Peças.

2.5 Requisitos

A fase de requisitos é definida pelas necessidades do cliente, quanto aos meios necessários para o funcionamento adequado do sistema.

Os requisitos são classificados em: Requisitos Funcionais, Não Funcionais e de Informação.

Descreve o conceituado Professor Martins (2009) esclarecendo que o requisito é considerado uma característica do sistema ou até mesmo de alguma coisa que o próprio sistema é apropriado para ser realizado e alcançar seus objetivos, como fundamento para essa ocasião, nos quais são identificados como se segue.

2.5.1 Requisitos Funcionais

Booch et al (2005) conceitua os requisitos funcionais como tudo que o sistema deve realizar com fundamento em alterações de dados entre o meio externo e interno a partir da interface do sistema.

Requisito funcional define-se na necessidade do cliente, baseado no processamento do sistema. Desta forma detectamos os seguintes requisitos funcionais para o sistema SICSEP.

- ✓ Restrição de permissões para Atendente e Gerente.
- ✓ Manter as informações:
- ✓ Funcionário
- ✓ Cliente

- ✓ Moto
- ✓ Produto
- ✓ Serviços
- ✓ Controlar o estoque e serviço.

Por final, os requisitos funcionais são na verdade funções ou atividades que o sistema oferece e que são expressamente definidos. Apresentam as funcionalidades que se almeja do sistema para ser realizado na empresa, de maneira completa e consistente. Uma vez que o usuário anseia pelo serviço a ser oferecido, recebendo às intenções para qual o sistema foi arquitetado.

2.5.2 Requisitos não Funcionais

Requisitos Não Funcionais são definidos pelos meios externos ao sistema, isto é, complementos e materiais no qual supre as necessidades do sistema para o seu funcionamento adequado, entre os quais Hardware e Software.

No transcorrer da idealização desse projeto foram detectados os seguintes requisitos não funcionais, que são:

Hardware:

- ✓ Placa mãe com Micro processador Dual Core 3.0Ghz ou superior contendo Memória (RAM) DDR3 de 4GB ou superior;
- ✓ Hard Disk de 50GB de espaço livre, de 7200rpm;
- ✓ Impressora.

Software

- ✓ Sistema Operacional Windows 7 (SP1) ou superior;
- ✓ Sistema Gerenciador de Banco de Dados Microsoft SQL Server 2012

2.5.3 Requisitos de Informação

O requisito de informação deste sistema define-se no resultado do processamento do mesmo, ou seja, as informações as quais o cliente espera que este realize. Desta forma na fase de levantamento de requisitos foram definidos os seguintes requisitos de informação ao qual o sistema teria que realizar:

- ✓ Informar quais os produtos está abaixo da quantidade necessária em estoque;

- ✓ Emitir relatório de Produtos abaixo da quantidade necessário em estoque;
- ✓ Consultar Informações de:
- ✓ Produtos;
- ✓ Serviços;
- ✓ Cliente;
- ✓ Moto;
- ✓ Ordem de Serviço;
- ✓ Funcionário

3 PROJETO

Projetar algo antes de realizar é de suma importância para alcançar resultado satisfatório.

No decorrer da criação do projeto para este sistema foi definida todas as funcionalidades, para tal foram usadas as ferramentas de programação, modelagem, armazenamento de dados e criação de interface (Desenvolvimento de Telas).

Segundo definição do site “sparxsystems”1, a Sparx Sistemas Enterprise Architect, é considerada um tipo de ferramenta usada na modelagem UML com fundamento na ideia, bem como no desenvolvimento de sistemas de software. Esta, utilizada para o Desenvolvimento dos Diagramas de Caso de Uso, Classe, Conceitual, Sequência, Tela e DDL.

O Sistema Gerenciador de Banco de Dados Microsoft SQL Server 2012 para armazenamento dos Dados, onde depois de elaborada as tabelas gerou-se o Diagrama de Entidade e Relacionamento como representação das tabelas do Banco.

Para a Implementação do sistema foi usada a ferramenta de codificação Visual Studio Profissional 2012 da Microsoft, utilizando a linguagem C# na Camada de Negocio e o Framework CSS Bootstrap 3.0 do Twitter para o designer da Camada de Apresentação.

3.1 CASOS DE USO

Silva (2007) explica que, um caso de uso é uma funcionalidade atômica de um sistema, subsistema ou classe. Para um melhor entendimento, é um exemplo de conjunto de caso de uso que pode ser observado em um caixa eletrônico de banco. Cada uma das funcionalidades acessíveis nos menus e sub menus é um caso de uso do software do caixa eletrônico, como emissão de extrato de conta corrente ou aplicação em caderneta de poupança e outras funções.

Com essa ideia foram identificados todos os Casos de Usos do sistema conforme figura 1.

---

[1] Introdução ao ***Enterprise Architect*** . Disponível no site: ttp://www.sparxsystems.com/enterprise_architect_user_guide/10/standard_uml_models/whatisuml.htm Acesso em 30.05.2014, 10:44;00.

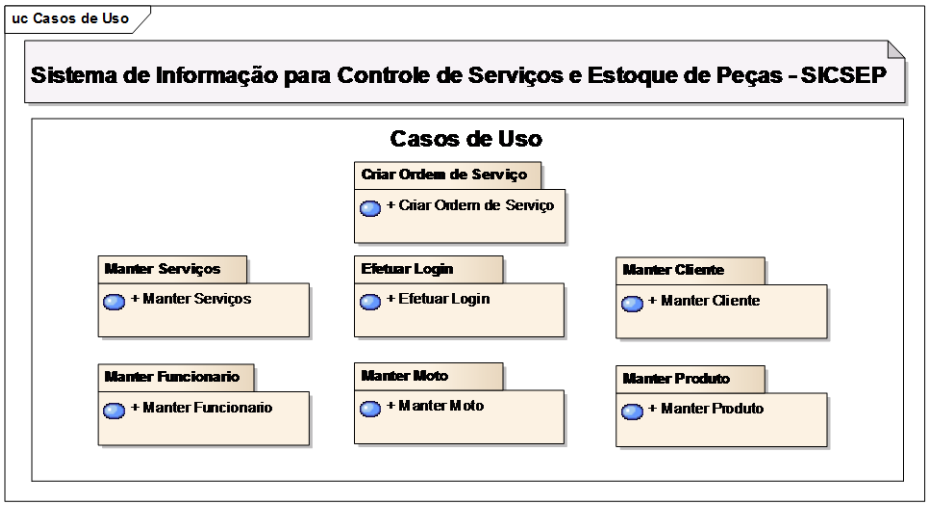

Figura 1. Caso de Uso identificados no Sistema.

Silva (2007) declara que o caso de uso principal comprova a funcionalidade eficaz, narrando o desenvolvimento para aqueles que estejam preocupados em alcançar sua finalidade através do sistema, que neste caso é o de Controle de Serviços e Estoque de Peças em uma empresa no ramo de vendas de Motos. A figura abaixo exibe o caso de uso Principal do sistema.

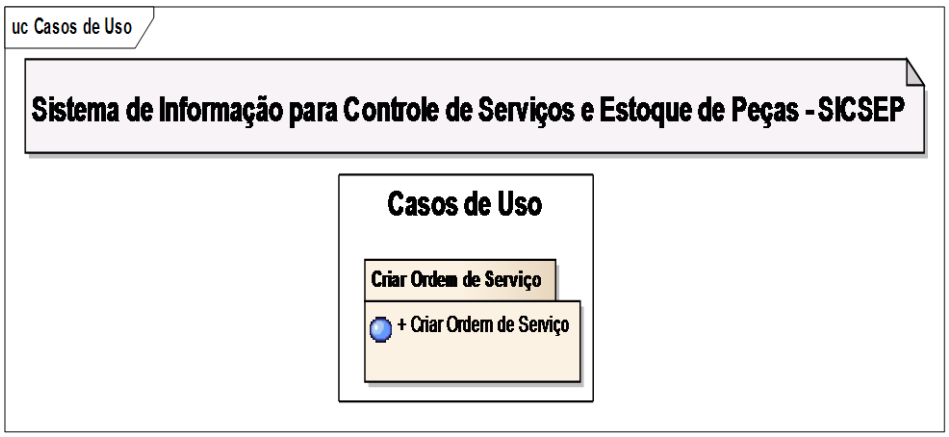

Figura 2. Casos de Uso Principal

Neste item, o Caso de Uso Secundário se enquadra como complemento a funcionalidade do Caso de Uso principal. A figura abaixo exibe os Casos de Uso Secundário do sistema.

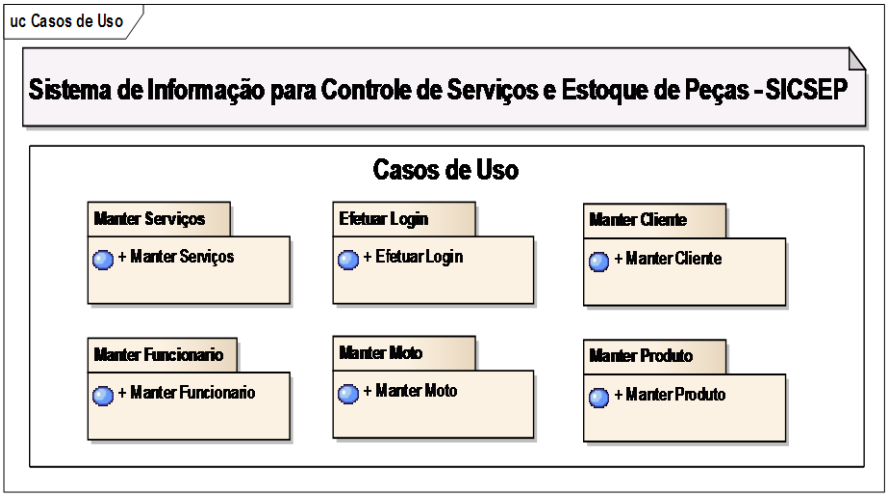

Figura 3. Caso de Uso Secundário

O Diagrama de Caso de Uso tem por objetivo apresentar todas as funcionalidades do sistema, bem como suas ligações, provando todos os requisitos adaptados com os Stakeholders (interessados) por meio de técnicas para a sua forma de colher dados. Neste entendimento, define Guedes (2008) que:

> O Diagrama de Caso de Uso procura, por meio de uma linguagem simples, possibilita a compreensão do comportamento externo do sistema por qualquer pessoa, tentando apresentar o sistema por intermédio de uma perspectiva do usuário.

Percebe-se então que, um Caso de Uso expõe procedimentos do sistema de em conformidade com os pedidos dos usuários. Pois, tais procedimentos são escritos em forma de narrações relatados todos os passos que apresenta a forma de sua realização, bem como a integração dos participantes do sistema trabalhado.

3.1.1 Diagrama de Caso de Uso Principal.

O Diagrama de Caso de Uso Principal descreve o que é necessário para que os interessados consiga criar uma ordem de serviço.

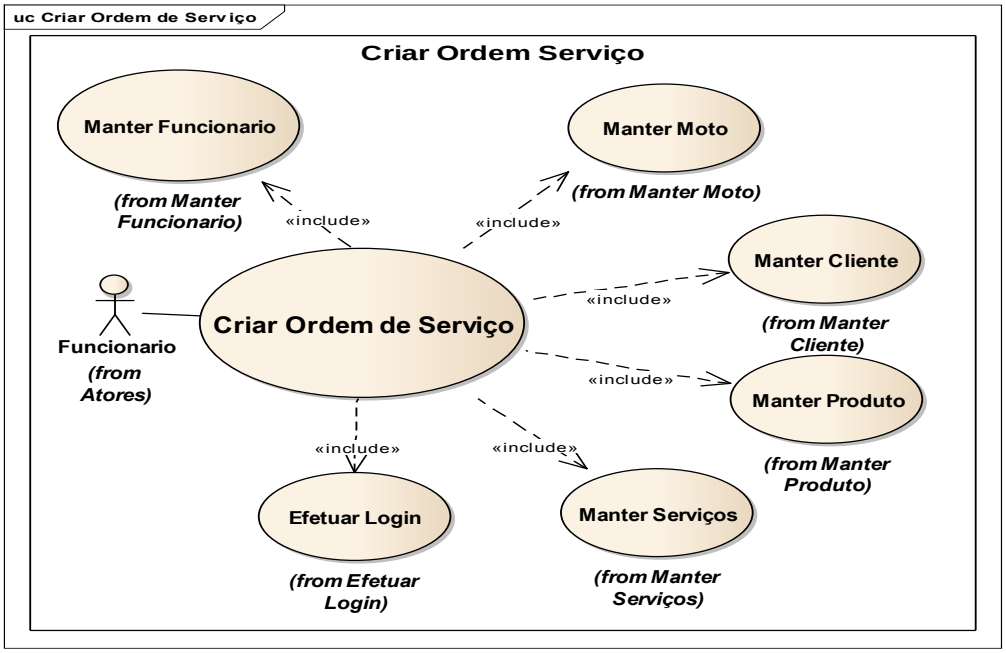

Figura 4. Diagrama de Caso de Uso Principal

Caso de uso estendido da Figura acima.

Criar Ordem de Serviço - Fluxo Principal**:**

1 - **Sistema**: exibe a tela de venda

2 - **Funcionário**: consulta o Cliente cadastrado.

4 - **Sistema**: retorna os dados do cliente

5 - **Funcionário**: consulta moto.

4 - **Sistema**: retorna os dados da moto

5 - **Funcionário**: consulta o mecânico que realizará os serviços

6 - **Sistema**: identifica

7 - **Funcionário**: consulta o produto

8 - **Sistema**: retorna o produto consultado e armazena

9 - **Funcionário**: consulta serviços

10 - **Sistema**: retorna o serviço e armazena

11 - **Funcionário**: finaliza o pedido

12 - **Sistema**: armazena e baixa os produtos.

Criar Ordem de Serviço - Fluxo Alternativo

2.1 - Clientes não possui cadastro

2.1.1 - **Sistema:** exibe tela de Cadastro

2.1.2 - **Funcionário**: realiza o cadastro do cliente.

2.1.3 - **Sistema**: retorna ao fluxo principal 2.

5.1 - Moto não possui cadastro

5.1.1 - **Sistema** exibe tela de Cadastro

5.1.2 - **Funcionário**: realiza o cadastro da moto.

5.1.3 - **Sistema**: retorna ao fluxo principal 3.

7.1 - Produtos em falta

7.1.1 - **Sistema**: informa "Produto inexistente"

3.1.2 Diagrama de Caso de Uso Secundário

Os Diagramas de Caso de Uso Secundário auxiliam o Diagrama de Caso de Uso Principal dando uma maior compreensão do funcionamento deste.

3.1.2.1 Diagrama de Atores

A figura abaixo tem o diagrama de atores representando a parte interessada que fará a interação com o sistema.

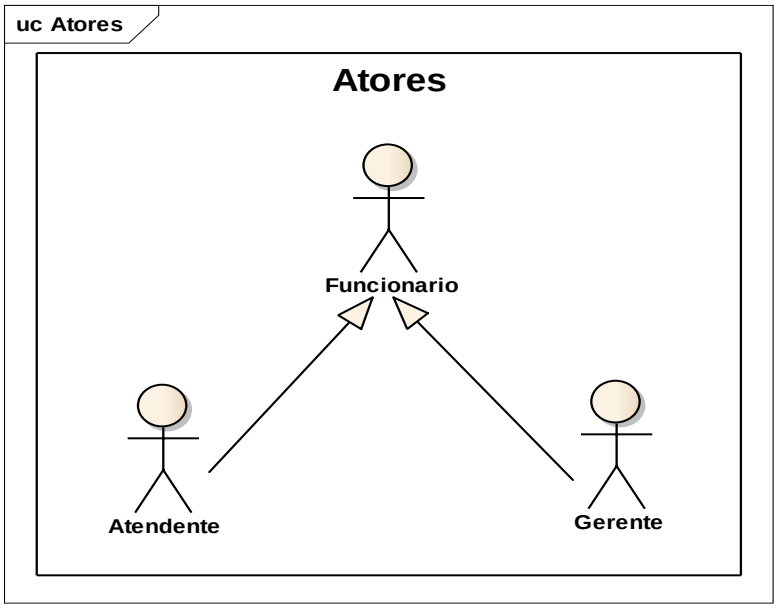

Figura 5. Diagrama de Atores

Este caso de uso representa os funcionários que acessarão o sistema independente de suas permissões.

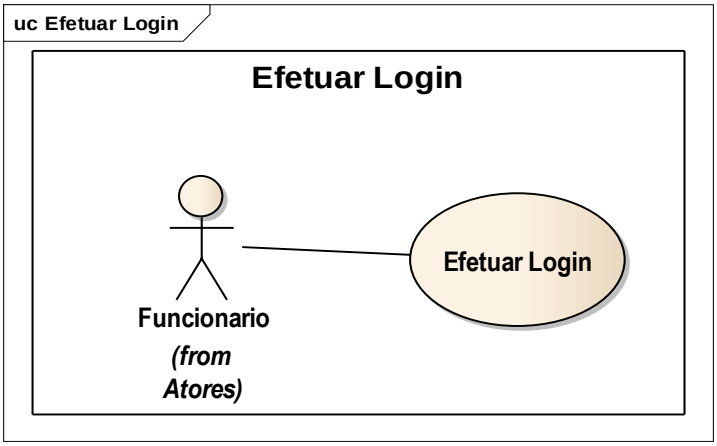

Figura 6. Diagrama de Caso de Uso Efetuar Login.

Caso de uso estendido da Figura acima.

Efetuar Login - Fluxo Principal

1 - **Sistema**: exibe tela de login.

2 - **Funcionário**: informa seu login e senha.

3 - **Sistema**: valida sua permissão.

Efetuar Login - Fluxo Alternativo

2.1 - Funcionários não autenticados

2.2 - Retorna ao fluxo principal

3.1.2.2 Diagrama Manter Cliente

Este Caso de Uso mostra de que forma se dará o cadastro do Cliente e quem poderá cadastrá-lo.

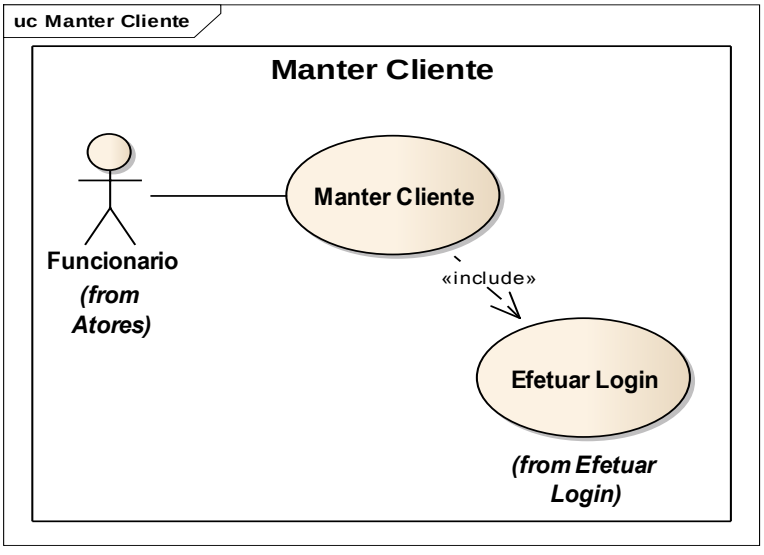

Figura 7. Diagrama de Caso de Uso Manter o cliente.

Caso de uso estendido da Figura acima.

Manter Cliente – Fluxo Principal

1 - **Sistema**: exibe tela de cadastro

2 - **Funcionário**: informa os dados do cliente e finaliza

3 - **Sistema**: informa "cadastrado com sucesso"

Manter Cliente – Fluxo Alternativo

2.1 - Cliente cadastrado.

2.2 - Retorna ao fluxo principal

2.3 - Cancela a Inserção do Cliente

3.1.2.3 Diagrama Manter Funcionário

Este caso de uso mostra de que forma se dará o cadastro da moto e quem poderá cadastra-lo.

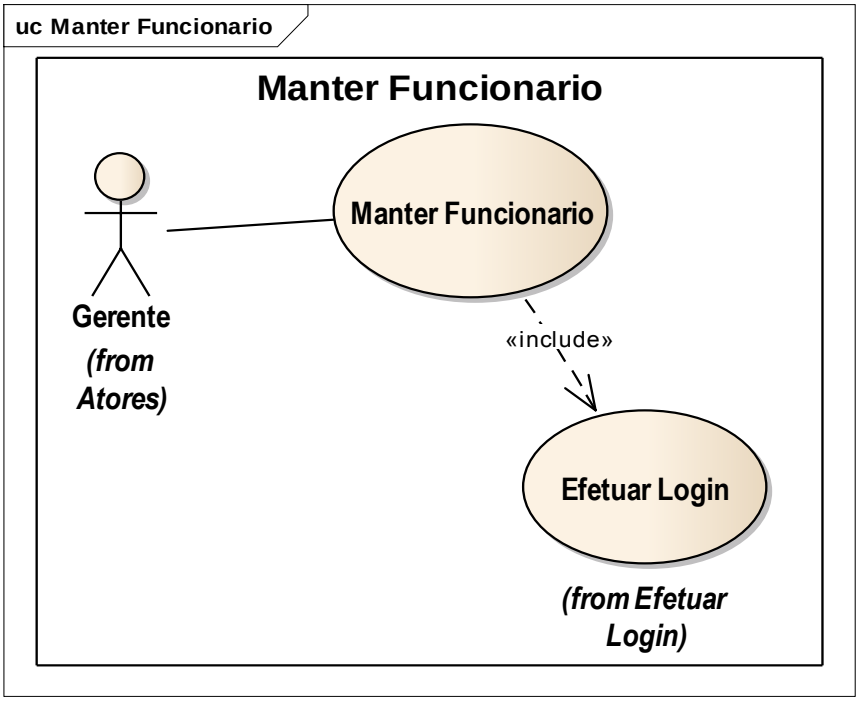

Figura 8. Diagrama de Caso de Uso Manter o Funcionário.

Caso de Uso estendido da figura acima.

Manter Funcionário – Fluxo Principal

1 - **Sistema**: exibe tela de cadastro.

2 - **Gerente**: Informa os dados do novo funcionário e finaliza.

3 - **sistema**: informa "cadastrado com sucesso”.

3.1.2.4 Diagrama Manter Moto

Este caso de uso mostra de que forma se dará o cadastro da moto e quem poderá cadastra-lo.

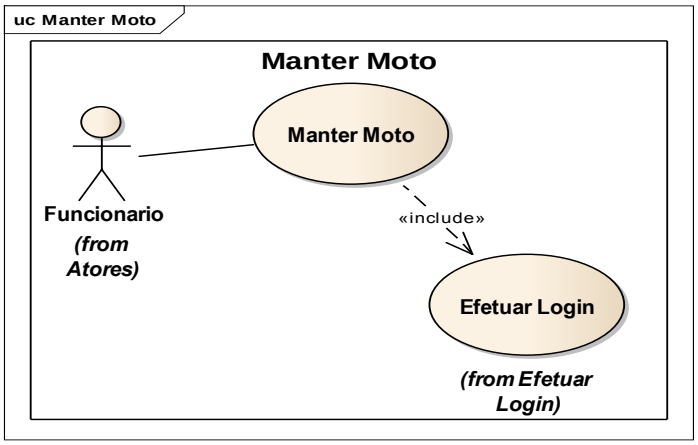

Figura 9. Diagrama de Caso de Uso Manter Moto.

Caso de uso estendido da Figura acima.

Manter Moto – Fluxo Principal

1 - **Sistema**: exibe tela de cadastro

2 - **Funcionário**: Informa os dados da moto

3 - **Sistema**: informa "cadastrado com sucesso"

Manter Moto – Fluxo Alternativo

2.1 - **Sistema**: Moto Inserida no Sistema.

2.2 - **Sistema**: Retorna ao fluxo principal.

3.1 - **Funcionário**: Cancela a inserção.

3.1.2.5 Diagrama Manter Produto

Este caso de uso mostra de que forma se dará o cadastro dos produtos e quem poderá cadastra-lo.

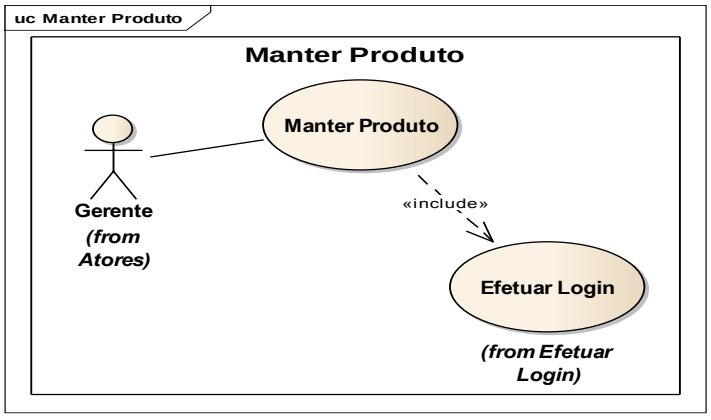

Figura 10. Diagrama de Caso de Uso Manter Produto.

Caso de uso estendido da Figura acima.

Manter Produto - Fluxo Principal

1 - **Sistema**: exibe tela de cadastro

2 - **Gerente**: Consulta o Produto

3 - **Sistema**: exibe os dados do produto

4 - **Gerente**: insere a nova quantidade

5 - **Sistema**: atualiza.

Manter Produto - Fluxo alternativo

2.1 - Produtos não estão cadastrados

2.1.1 - **Gerente**: informa os dados do produto

2.1.2 - **Sistema**: retorna ao fluxo principal 2

3.1.2.6 Diagrama Manter Serviços

Este Caso de Uso mostra de que forma se dará o cadastro dos tipos de Serviços que serão realizados na loja e quem poderá cadastra-lo.

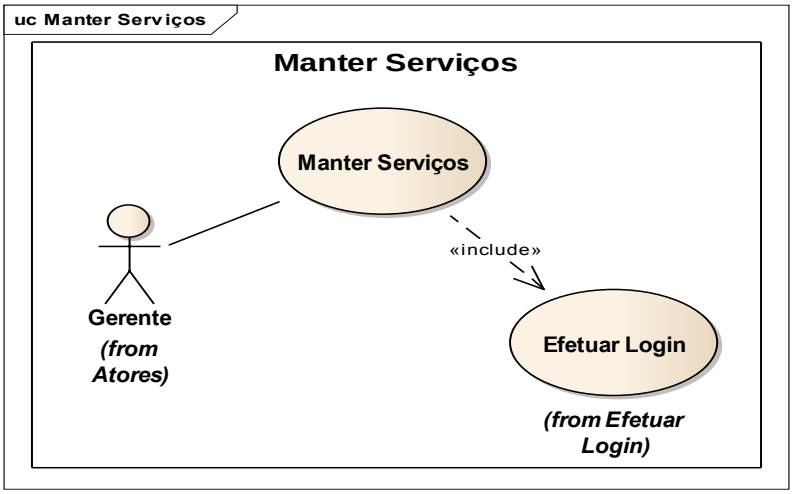

Figura 11. Diagrama de Caso de Uso Manter serviços.

Caso de uso estendido da Figura acima.

Manter Serviço – Fluxo Principal

1 - **Sistema**: Exibe Tela cadastro

2 - **Gerente**: Insere o serviço

3 - **Sistema**: atualiza

3.2 Diagrama de Sequência

Diagrama de Sequência é utilizado para descrever as sequências das entradas de dados, desde a percepção de quem vai manter inseridos os dados até a saída de confirmação de conclusão da ação.

Wazlawick (2004) entende que “o diagrama de sequência é uma ferramenta que deve ser utilizada sempre em função dos casos de uso. Ele captura o comportamento de um único caso de uso, ou seja, mostra a interação entre os objetos ao longo do tempo, apresentando os objetos que participam da interação e da sequência das mensagens trocadas”.

Diagrama de Sequência identificado no Sistema.

A figura abaixo exibe de uma maneira macro todos os diagramas de sequência existente no sistema.

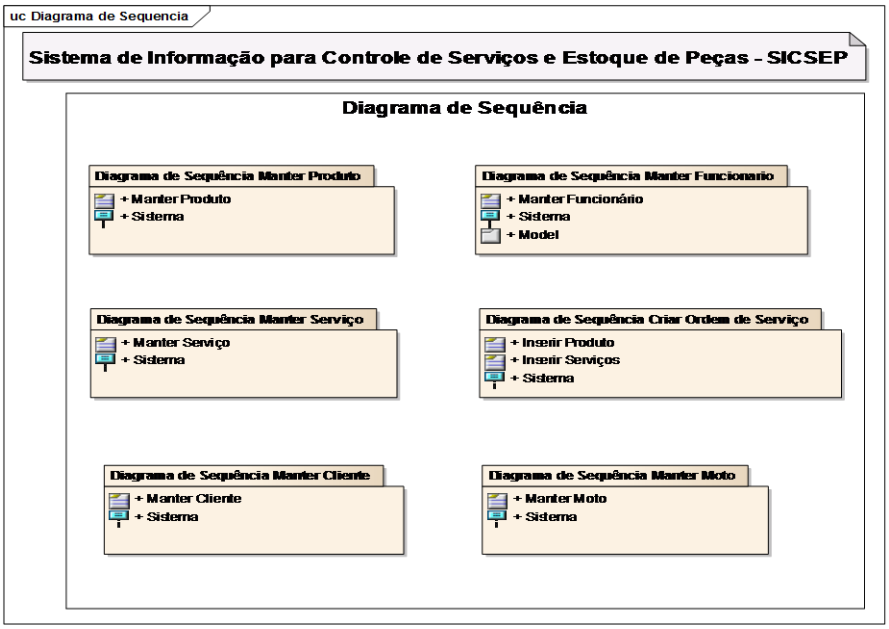

Figura 12. Diagrama de Sequência identificado no Sistema

3.2.1 Diagrama de Sequência Criar Ordem de Serviço

Este diagrama descreve a sequência de ações necessárias para a criação de uma Ordem de Serviço de maneira visual.

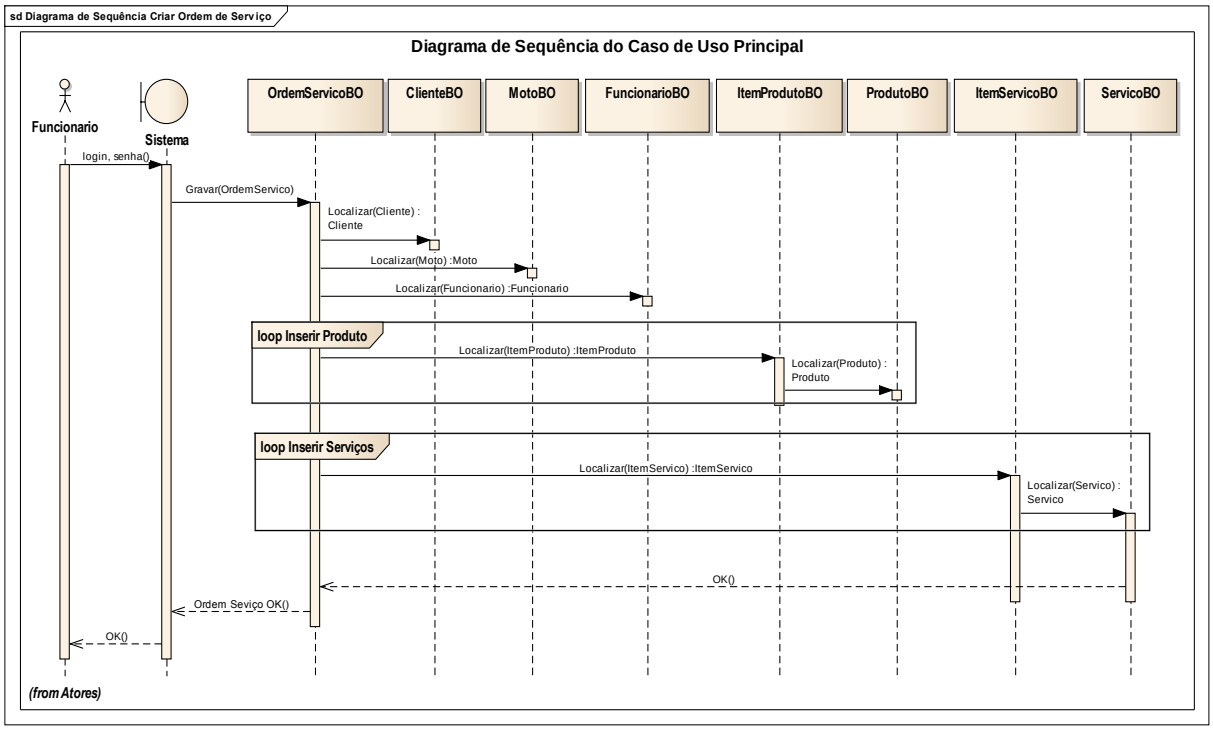

Figura 13. Diagrama de Sequência Criar Ordem de Serviço

3.2.2 Diagrama de Sequência Manter cliente

Este diagrama descreve a sequência de ações necessárias para Manter o Cliente de maneira visual.

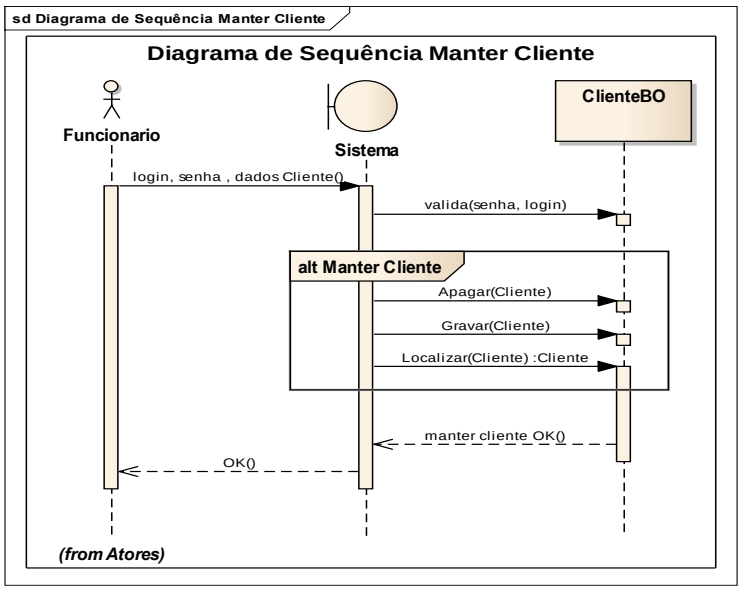

Figura 14. Diagrama de Sequência Manter cliente

3.2.3 Diagrama de Sequência Manter Funcionário.

Este diagrama descreve a sequência de ações necessárias para manter o funcionário e quem poderá realizar de forma visual.

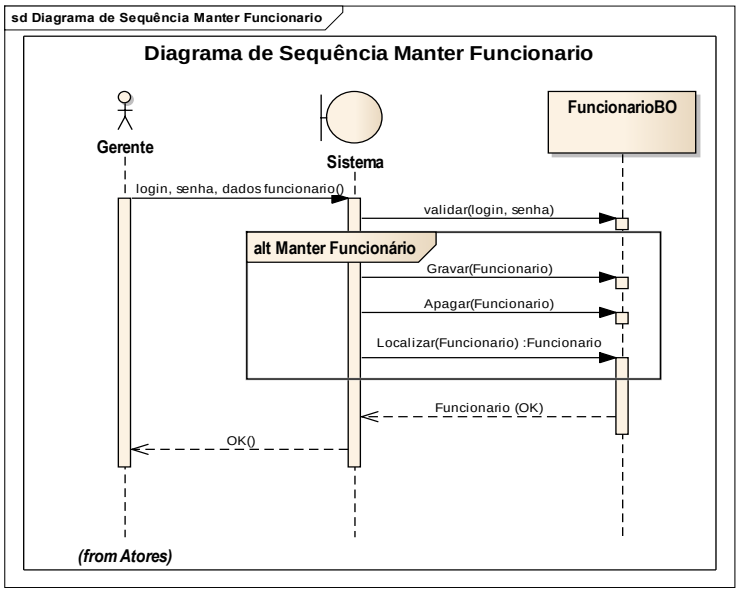

Figura 15. Diagrama de Sequência Manter Funcionário.

3.2.4 Diagrama de Sequência Manter Moto

Este diagrama descreve a sequência de ações necessárias para manter a moto de maneira visual.

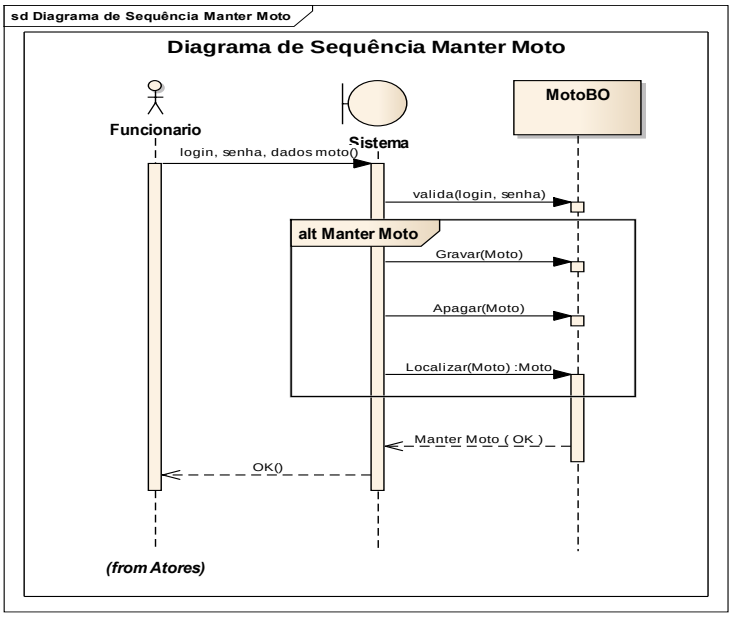

Figura 16. Diagrama de Sequência Manter Moto

3.2.5 Diagrama de Sequência Manter Produto.

Este diagrama descreve a sequência de ações necessárias para manter o produto e quem poderá realizar de forma visual.

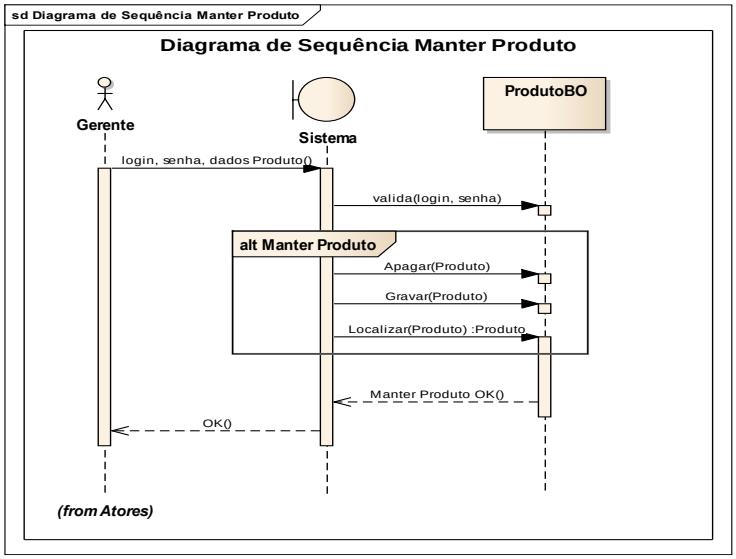

Figura 17. Diagrama de Sequência Manter Produto.

3.2.6 Diagrama de Sequência manter serviço.

Este diagrama descreve a sequência de ações necessárias para manter o serviço e quem poderá realizar de forma visual.

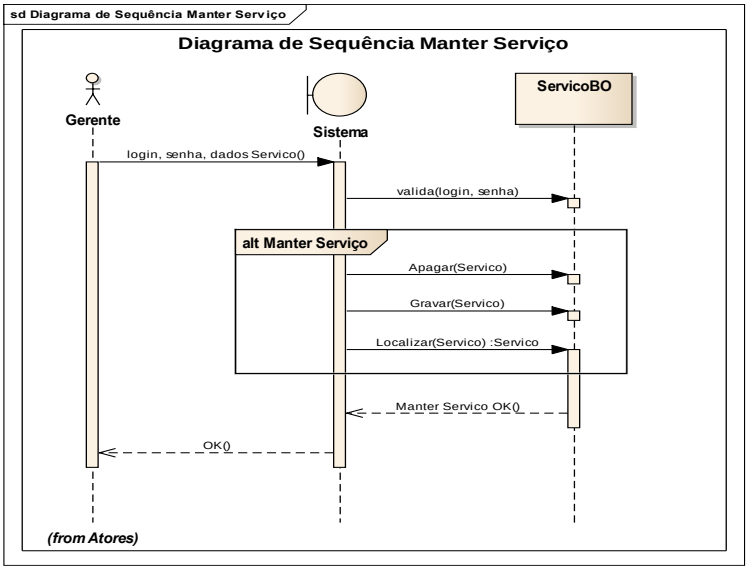

Figura 18. Diagrama de Sequência manter serviço.

3.3 Diagrama Conceitual

De acordo com Lima (2008), o Diagrama Conceitual é uma armação estática que mostra definição do mercado atual, bem como suas analogias assim como o sistema a ser desenvolvido. Auxiliando no entendimento do sistema desenvolvido conforme mostra a figura abaixo:

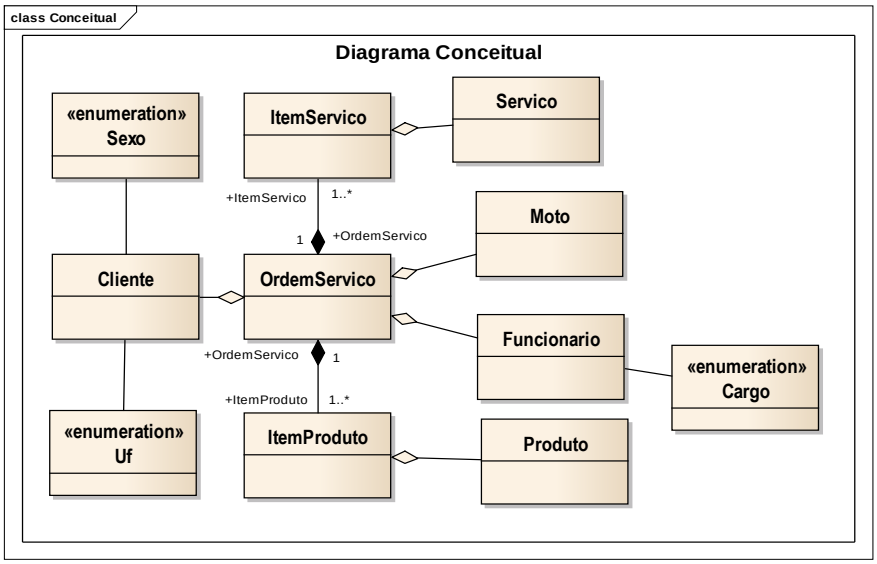

Figura 19. Diagrama Conceitual

3.4 Diagrama de Classe

No pensamento de Guedes (2009) este Diagrama mostra a armação das divisões usadas pelo sistema, comandando as características e técnicas que cada divisão possui, bem como de ordenar de que forma se integram e transmitem dados entre si. O diagrama de classe demonstra todas as classes que serão codificadas com os atributos necessários para a codificação.

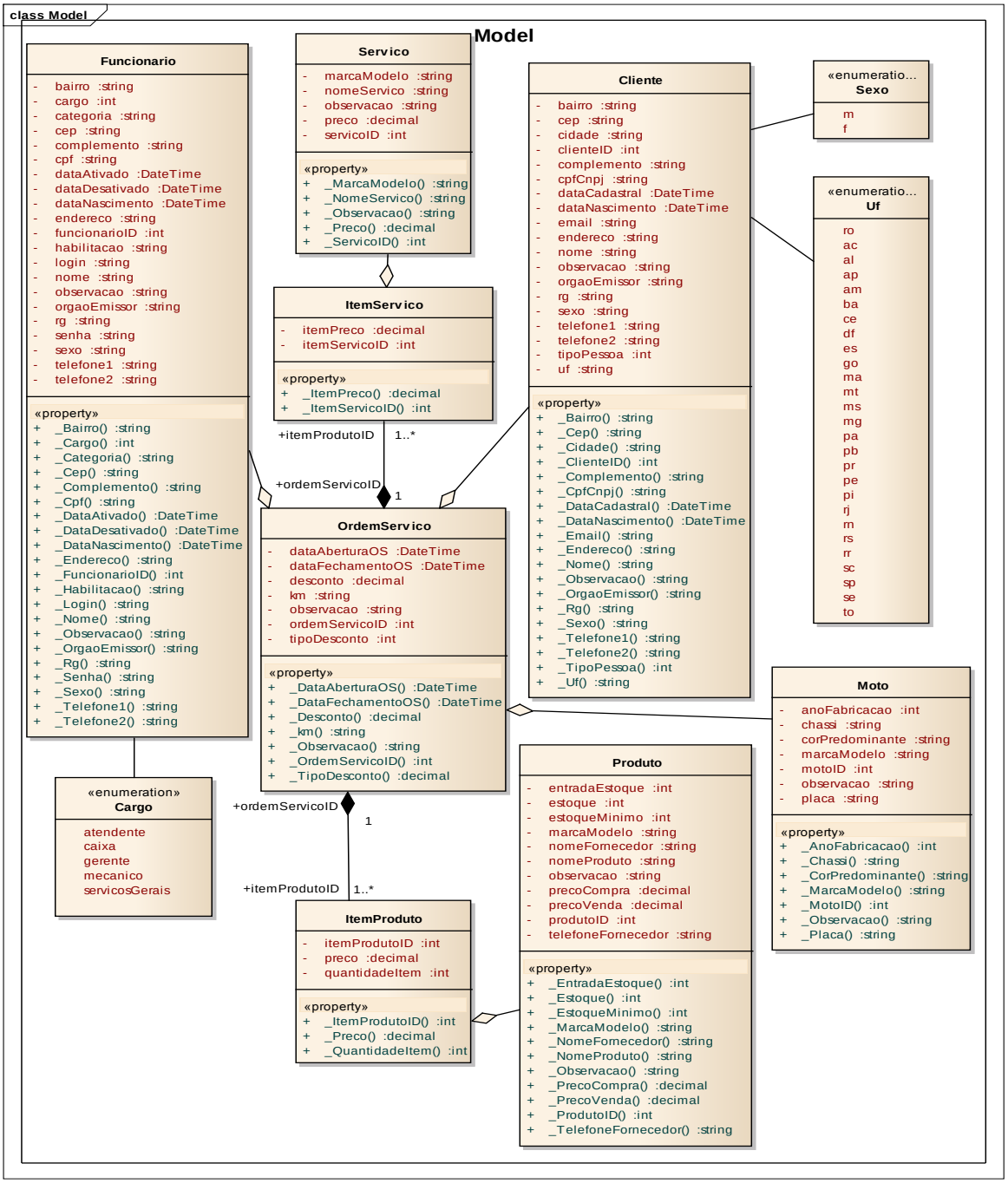

Figura 20. Diagrama de Classe da Camada Model.

3.4.1 Classe Cliente

A classe Cliente é responsável por criar espaços no disco para manipulação das informações inerentes a este que vão ser adicionadas conforme os nomes dos seus atributos.

No quadro abaixo temos todos os atributos pertencentes a esta classe.

Atributo	Notas	Tipo
bairro	Bairro = aqui será inserido pelo usuário o nome do bairro onde o cliente mora.	string
cep	CEP = aqui será inserido o cep da rua em que o cliente mora.	string
cidade	Cidade = aqui será inserida a cidade em que o cliente mora.	string
clienteID	Cliente ID = este ficará responsável por identificar unicamente os clientes desta classe.	int
complemento	Complemento = será inserido qualquer complemento do endereço do cliente.	string
cpf/cnpj	CPF ou CNPJ= aqui será inserido o número de cadastro de pessoa física do cliente ou o CNPJ da empresa se quando jurídica.	string
dataCadastral	Data de abertura de cadastro = aqui será inserida a data em que foi criado a cadastro do cliente.	DateTime
dataNascimento	Data de Nascimento = campo em que será inserida a data em que o cliente nasceu.	DateTime
email	E-mail = será inserido aqui o e-mail do cliente.	string
endereco	Endereço = aqui será inserido o nome da rua e o número da casa do cliente.	string
nome	Nome = aqui será inserido pelo usuário o nome do novo funcionário da empresa.	string
observacao	Observação = aqui será inserida qualquer observação necessária sobre o cliente.	string
orgaoEmissor	Órgão emissor = aqui o usuário irá inserir o órgão que expediu o registro único.	string
RG	RG = aqui será inserido o registro único do cliente.	string
sexo	Sexo = será identificado aqui pelo usuário o sexo do	string

Atributo	Notas	Tipo
	cliente.	
telefone1	Telefone = será inserido o número do telefone do cliente.	string
telefone2	Telefone celular 1 = será inserido aqui o número do telefone celular cliente.	string
tipoPessoa	Tipo pessoa = define se pessoa física ou jurídica.	int
uf	Uf = aqui será inserido o estado em que o cliente mora.	string

Quadro 1. Atributo da Classe Cliente.

3.4.2 Classe Funcionário

A Classe Funcionário é responsável por criar espaços no disco para manipulação das informações inerentes á este que vão ser adicionadas conforme os nomes dos seus atributos.

No quadro abaixo temos todos os atributos pertencentes a esta classe.

Atributo	Notas	Tipo
bairro	Bairro = aqui será inserido pelo usuário o nome do bairro onde o funcionário mora.	string
cargo	Cargo = aqui será definido pelo usuário o cargo que o funcionário exercerá na empresa.	int
categoria	Categoria = aqui será inserida a categoria na qual o funcionário pertence definido em sua habilitação.	string
cep	CEP= aqui será inserido o cep da rua em que funcionário mora.	string
complemento	Complemento = será inserido qualquer complemento do endereço do funcionário.	string
cpf	CPF = aqui será inserido o número de cadastro de pessoa física do funcionário.	string
dataAtivado	Data de Ativado = aqui será inserida a data em que o funcionário passou a fazer parte do sistema.	DateTime
dataDesativado	Data de desativado = aqui será inserida a data em que o funcionário foi desativado.	DateTime
dataNascimento	Data de Nascimento = campo em que será inserido a data em que o funcionário nasceu.	DateTime

Atributo	Notas	Tipo
endereco	Endereço = aqui será inserido o nome da rua e o número da casa do funcionário.	string
FuncionarioID	Funcionário ID = este ficará responsável por identificar unicamente os funcionários desta classe.	int
habilitacao	Habilitação = aqui será inserido o número da habilitação do funcionário.	string
login	Login = aqui será criado o login de acesso que o funcionário irá utilizar para acessar o sistema.	string
nome	Nome = aqui será inserido pelo usuário o nome do novo funcionário da empresa.	string
observação	Observação = aqui será inserida qualquer observação necessária sobre o funcionário.	string
orgaoEmissor	Órgão emissor = aqui o usuário irá inserir o órgão que expediu o registro único.	string
rg	RG = aqui será inserido o registro único do funcionário.	string
senha	Senha = aqui será definida uma senha para os funcionários que tiver que interagir com o sistema.	string
sexo	Sexo = será identificado aqui pelo usuário o sexo funcionário.	string
telefone1	Telefone = será aqui o número do telefone do funcionário.	string
telefone2	Telefone celular 1 = será inserido aqui o número do telefone celular funcionário.	string

Quadro 2. Atributo da Classe Funcionário

3.4.3 Classe Item de Produto

A classe Item Produto é responsável por criar espaços no disco para manipulação das informações inerentes a este, que vão ser adicionadas conforme os nomes dos seus atributos. Também ficará responsável em processar os serviços criando uma lista conforme a inserção de quantidades.

No quadro abaixo temos todos os atributos pertencentes a esta classe.

Atributo	Notas	Tipo
itemProdutoID	Item de Produto ID =Este ficará responsável pelo identificador único do item de produto.	int

Atributo	Notas	Tipo
preco	Preço = Campo que será responsável por receber automaticamente o preço do produto que estará vindo do banco de dados.	decimal
quantidadeItem	Quantidade de Item = Campo que será inserido a quantidade do produto a ser vendida.	int

Quadro 3. Atributo da Classe Item de Produto.

3.4.4 Classe Item de Serviço

A classe Item Serviço é responsável por criar espaços no disco para manipulação das informações inerentes a este, que vão ser adicionadas conforme os nomes dos seus atributos. Também ficará responsável em processar os serviços criando uma lista conforme a inserção de quantidades.

No quadro abaixo temos todos os atributos pertencentes a esta classe.

Atributo	Notas	Tipo
itemPreco	Item Preço = Este é responsável por armazenar o preço do produto que está vindo da classe Produto.	decimal
itemServiçoID	Item de Serviço ID = Este ficará responsável pelo identificador único do item de Serviço.	int

Quadro 4. Atributo da Classe Item serviço.

3.4.5 Classe Moto

A classe Moto é responsável por criar espaços no disco para manipulação das informações inerentes a este que vão ser adicionadas conforme os nomes dos seus atributos.

No quadro abaixo temos todos os atributos pertencentes a esta classe.

Atributo	Notas	Tipo
anoFabricacao	Ano de fabricação = será inserido aqui o ano em que a moto foi fabricada.	int
chassi	Chassi = será inserido aqui o número do chassi da moto.	string
corPredominante	Cor predominante = a cor da moto que mais predomina definida no documento da moto.	string

Atributo	Notas	Tipo
marcaModelo	Marca e modelo = será inserida aqui a marca e o modelo da moto.	string
motoID	Moto ID = este ficará responsável por identificar unicamente todas as motos desta classe.	int
observacao	Observação = aqui será inserida qualquer observação necessária sobre a moto.	string
placa	Placa = será inserida aqui o número da placa da moto.	string

Quadro 5. Atributo da Classe Moto.

3.4.6 Classe Ordem Serviço

A classe Ordem Serviço é responsável por criar espaços no disco para manipulação das informações inerentes a estes que vão ser adicionadas conforme os nomes dos seus atributos. Também é responsável em buscar nas Classes Agregada, Cliente, Moto, Funcionário, Produto e Serviço, as informações destas e fazer o processamento gerenciando o produto e serviço.

No quadro abaixo temos todos os atributos pertencentes a esta classe.

Atributo	Notas	Tipo
dataAberturaOS	Data de abertura da OS = será inserida aqui a data de criação da ordem de serviço.	DateTime
dataFechamentoOS	Data de fechamento da OS = será inserida aqui a data em que a ordem de serviço foi finalizada.	DateTime
desconto	Desconto = o valor de desconto do produto quando necessário.	decimal
km	Km = aqui será inserido a numeração atual da quilometragem em que a moto deu entrada para efetuar algum tipo de serviço na loja.	string
observacao	Observação = aqui será inserida qualquer observação necessária que diz respeito aquela ordem de serviço.	string
ordemServicoID	Ordem de serviço ID = este ficará responsável por identificar unicamente todas as ordens de serviço desta classe.	int
tipoDesconto	Tipo de desconto = este ficará responsável por definir se o desconto é em espécie ou porcentagem.	int

Quadro 6. Atributo da Classe Ordem de Serviço.

3.4.7 Classe Produto

A classe Produto é responsável por criar espaços no disco para manipulação das informações inerentes á este que vão serem adicionadas conforme os nomes dos seus atributos.

No quadro abaixo temos todos os atributos pertencentes a esta classe.

Atributo	Notas	Tipo
entradaEstoque	Entrada de estoque = será inserido a nova quantidade de produtos que foi comprada para loja.	int
estoque	Estoque = fornecerá automaticamente para o usuário a quantidade do produto que estiver em estoque no momento da inserção.	int
estoqueMinimo	Estoque mínimo = o usuário definirá aqui um número mínimo do produto em estoque.	int
marcaModelo	Marca e Modelo = será definida a marca e o modelo do produto.	string
nomeFornecedor	Nome do fornecedor = será inserido aqui o nome do fornecedor que forneceu o produto.	string
nomeProduto	Nome do produto = será inserido aqui o nome do produto.	string
observacao	Observação = aqui será inserida qualquer observação inerente sobre o produto.	string
precoCompra	Preço de compra = será inserido aqui o valor que o produto custou para empresa.	decimal
precoVenda	Preço de venda = aqui será inserido automaticamente o valor que custará para o cliente o produto.	decimal
produtoID	Produto ID = este ficará responsável por identificar unicamente todos os produtos desta classe.	int
telefoneFornecedor	Telefone do fornecedor = será inserido aqui o número do telefone do fornecedor para eventual contato com o mesmo.	string

Quadro 7. Atributos da Classe Produto.

3.5 Classe Serviço

A classe Serviço é responsável por criar espaços no disco para manipulação das informações inerentes a este, que vão ser adicionadas conforme os nomes dos seus atributos.

No quadro abaixo temos todos os atributos pertencentes a esta classe.

Atributo	**Notas**	**Tipo**
marcaModelo	Marca e Modelo = será definido aqui para qual marca e modelo de moto é aquele serviço.	string
nomeServiço	Nome do Serviço = Campo que será inserido o nome do serviço que a loja irá disponibilizar aos clientes.	string
observacao	Observação = aqui será inserida qualquer observação necessária que diz respeito a serviço.	string
preco	Preço = Campo que será inserido o valor que custará o serviço.	decimal
servicoID	Serviço ID = Este ficará responsável pelo identificador único dos serviços	int

Quadro 8. Atributo da Classe Serviço.

3.6 Diagrama DDL

O diagrama DDL foi gerado através do diagrama de classe para assim de posse dessas informações, gerar o script do banco de dados para armazenamento das informações.

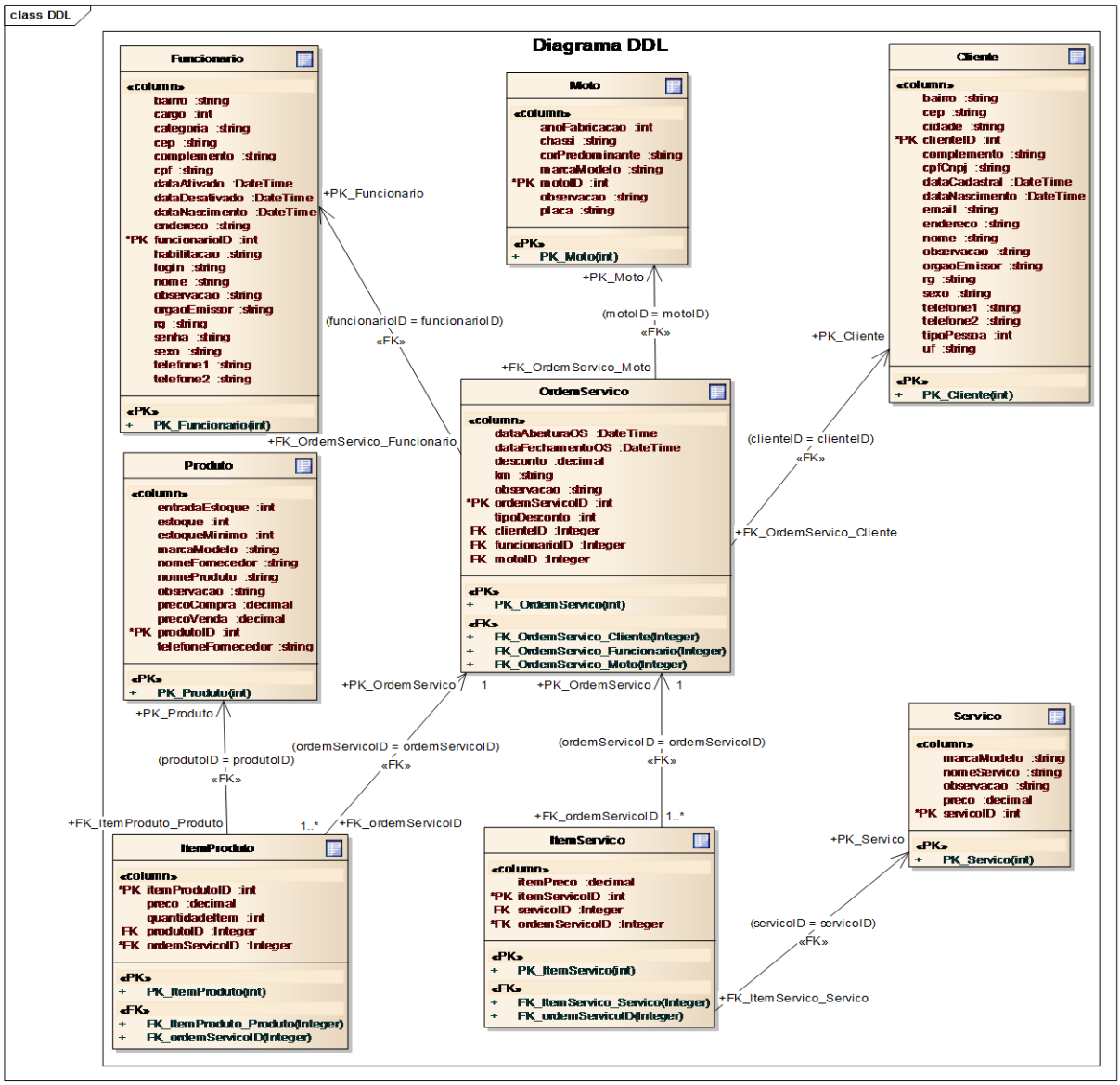

Figura 21. Diagrama DDL.

3.7 Diagrama de Tela (Interface)

Diagrama de tela demonstra como o usuário irá interagir com o sistema através das páginas Web que o sistema terá.

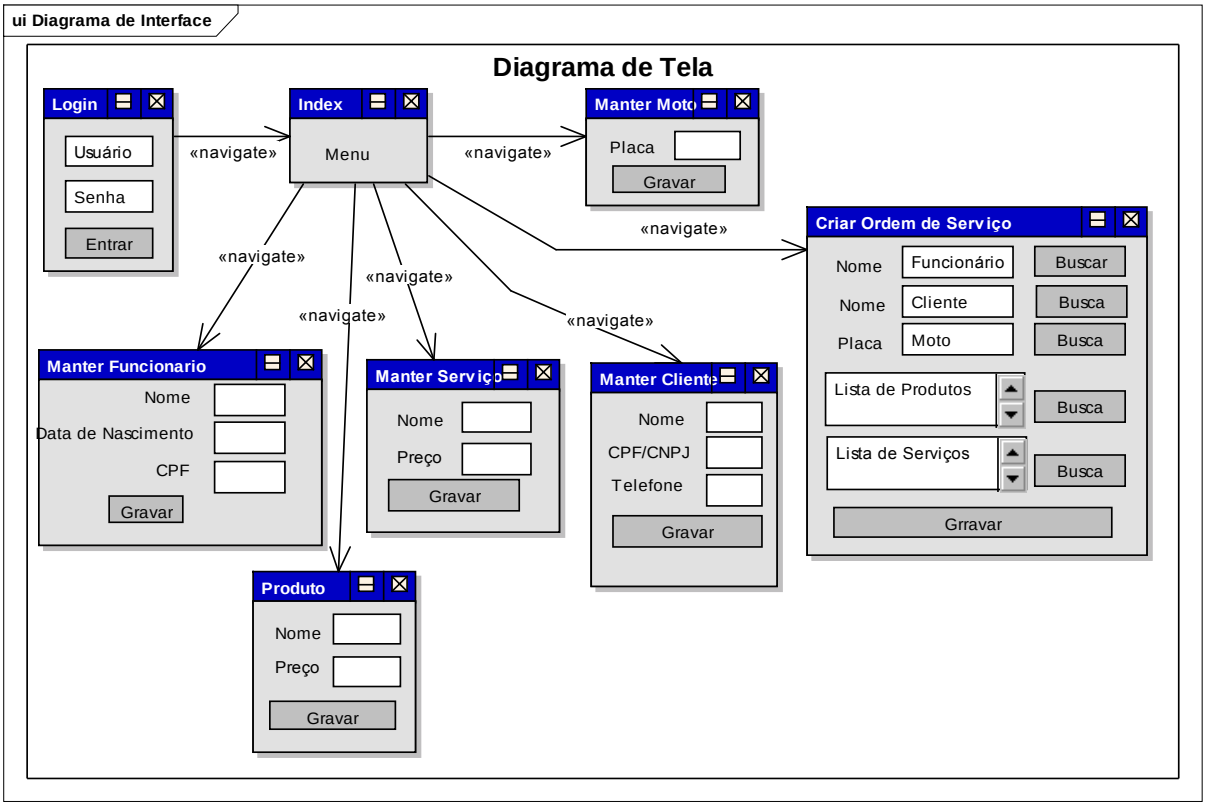

Figura 22. Diagrama de Tela.

3.8 Diagramas de Entidade e Relacionamento

O diagrama de entidade e relacionamento foi gerado pela ferramenta SQL Server 2012, após a criação de todas as tabelas do banco de dados, representando a iteração entre as tabelas através de suas chaves primárias e secundárias.

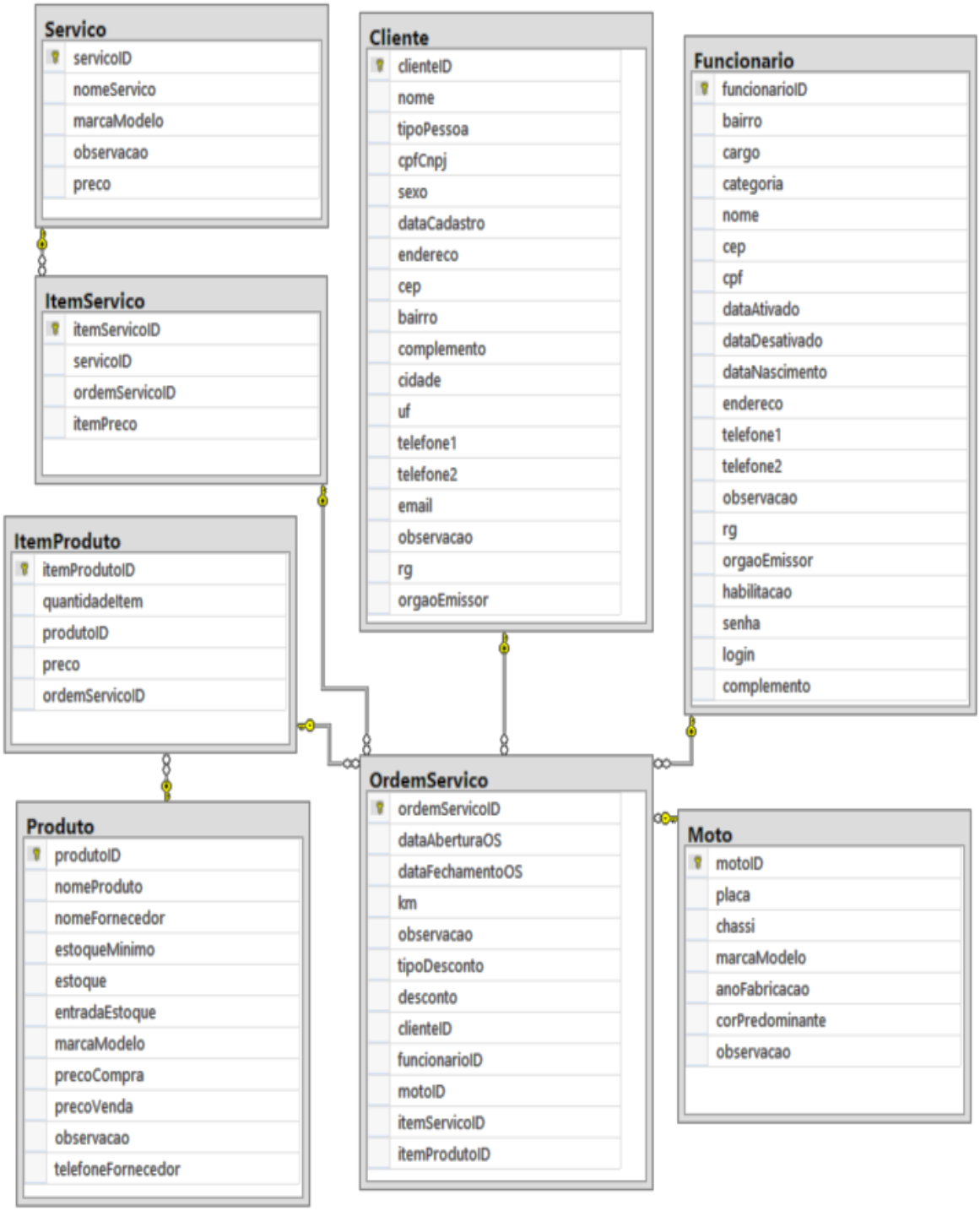

Figura 23. Diagrama de Entidade e Relacionamento

4 IMPLEMENTAÇÃO

Neste item é que são executadas todas as fases das codificações de acordo com as fases já citadas. De igual modo é desenvolvido um banco de dados, bem como as classes de exposição responsável pela integração de quem vai utilizar o programa na empresa.

Houve bastante preocupação na criação do programa para que tudo ocorresse de acordo com a necessidade da empresa para tanto foi criado uma camada de apresentação de fácil interação para o usuário e uma codificação eficaz na camada de codificação de classe para atender o cliente tudo o que este esperava do sistema.

4.1 Camada de Apresentação

Silva (2007) esclarece que, a categoria que apresenta as informações falando estatisticamente sobre a interface, bem como o responsável que assumem a recepção das informações, assim como os comandos usados pelos os usuários e, demonstrando a resposta almejada advindo do sistema criado.

Em síntese, segue todas as informações para a execução do programa a ser utilizado pelo sistema proposto.

4.1.1 Tela de Login

Tela de validação do usuário de maneira a utilizar o sistema apenas aqueles que estão pré-cadastrados no sistema com suas credenciais.

Figura 24. Tela de Login

4.1.2 Tela de Início

Tela de apresentação onde todos os funcionários que acessarem o sistema irá ser redirecionado para ela e assim apresentar o menu conforme sua credencial.

Figura 25. Tela de Início

4.1.3 Tela de Informações da Ordem de Serviço

Tela onde se dará o processo de criação da ordem de serviço apresentando todas as informações pertinentes a ela.

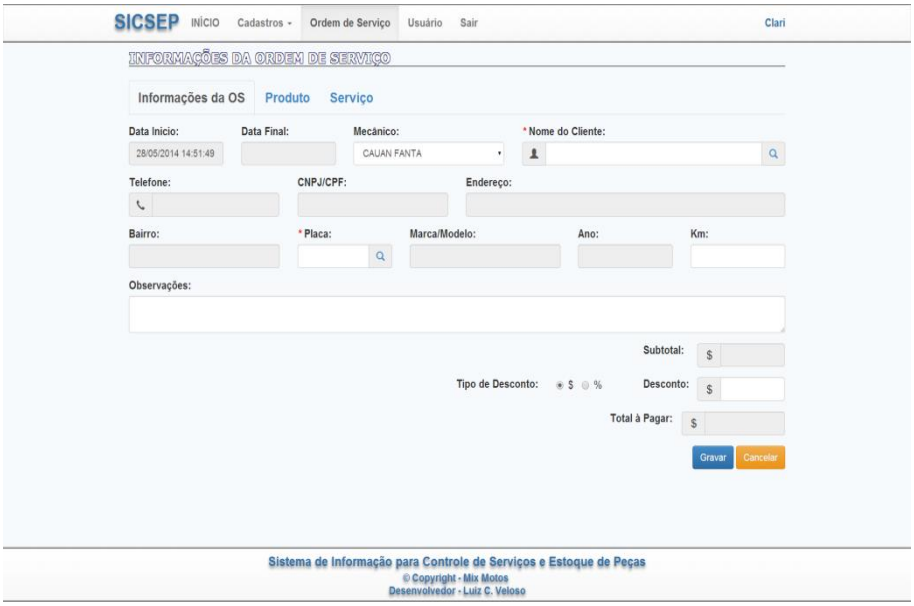

Figura 26. Tela de Informações de Ordem de Serviço

4.1.4 Tela de Informações do Cliente

Tela de inserção de todas as informações do cliente como também de consulta do mesmo.

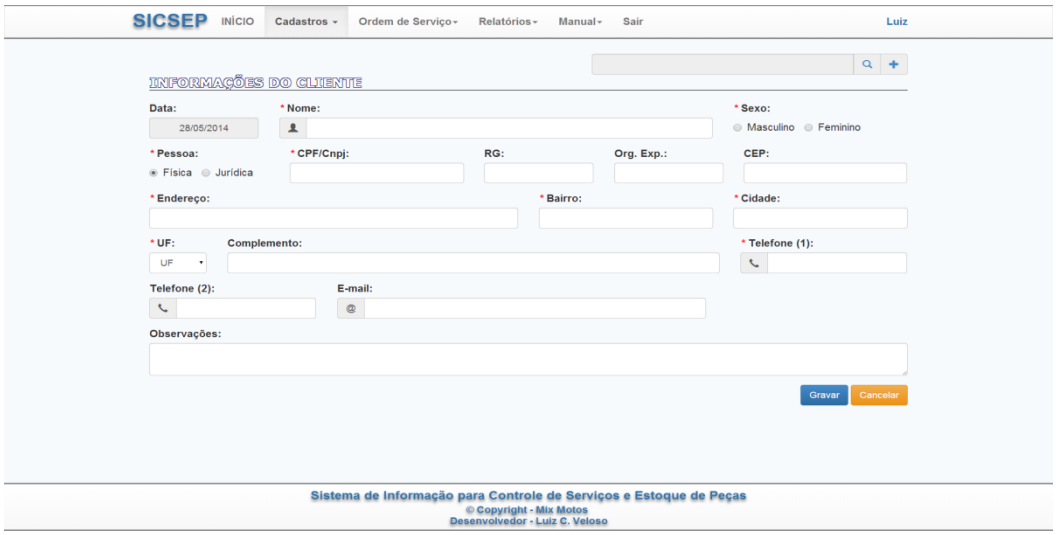

Figura 27. Tela de Informações de Cliente

4.1.5 Tela de Informações do Funcionário

Tela onde será feita todo o processo de alteração, inserção das informações do funcionário.

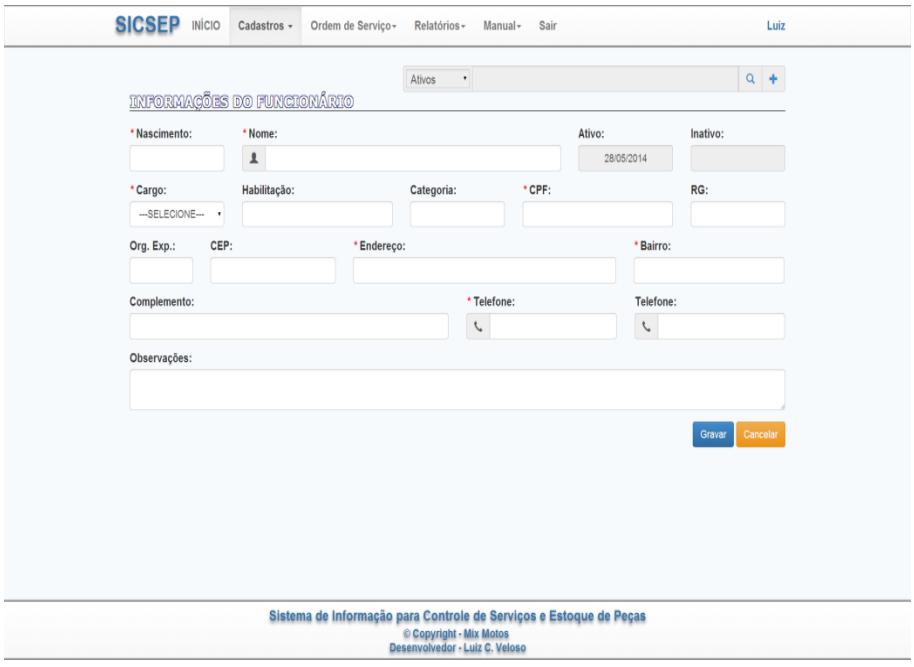

Figura 28. Tela de Informações de Funcionário

4.1.6 Tela de Informações da Moto

Tela onde será feita todo o processo de alteração, inserção das informações da Moto.

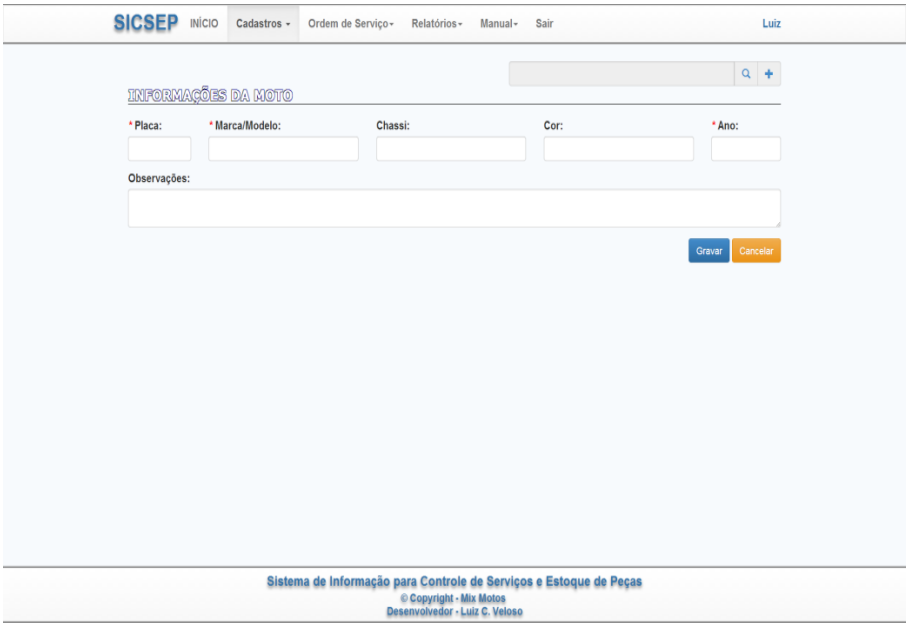

Figura 29. Tela de Informações de Moto

4.1.7 Tela de Informações do Produto

Tela onde serão cadastrados todos os produtos da loja.

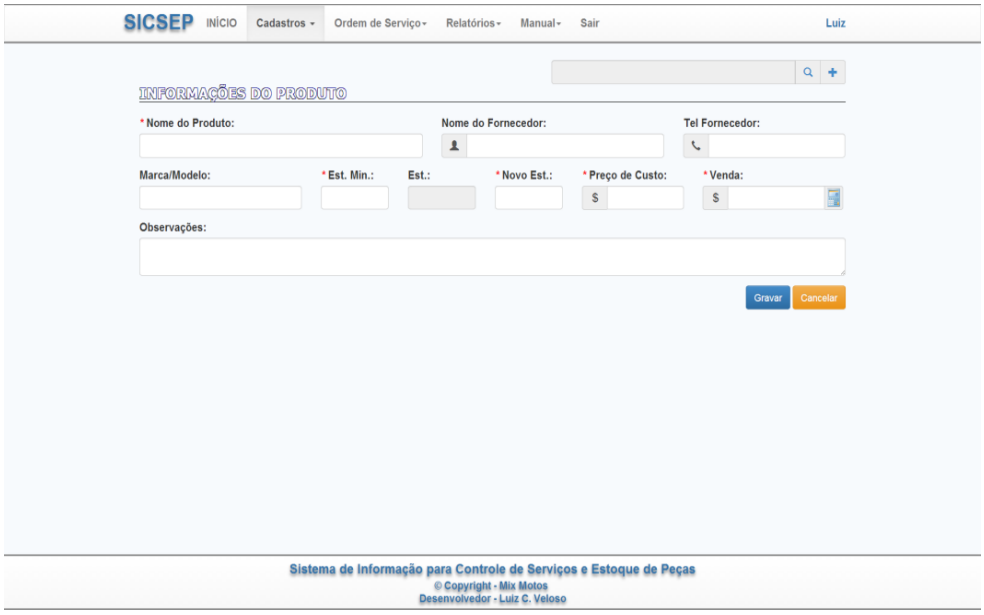

Figura 30. Tela de Informações de Produto

4.1.8 Tela de Informações de Serviço

Tela de Inserção dos serviços da loja.

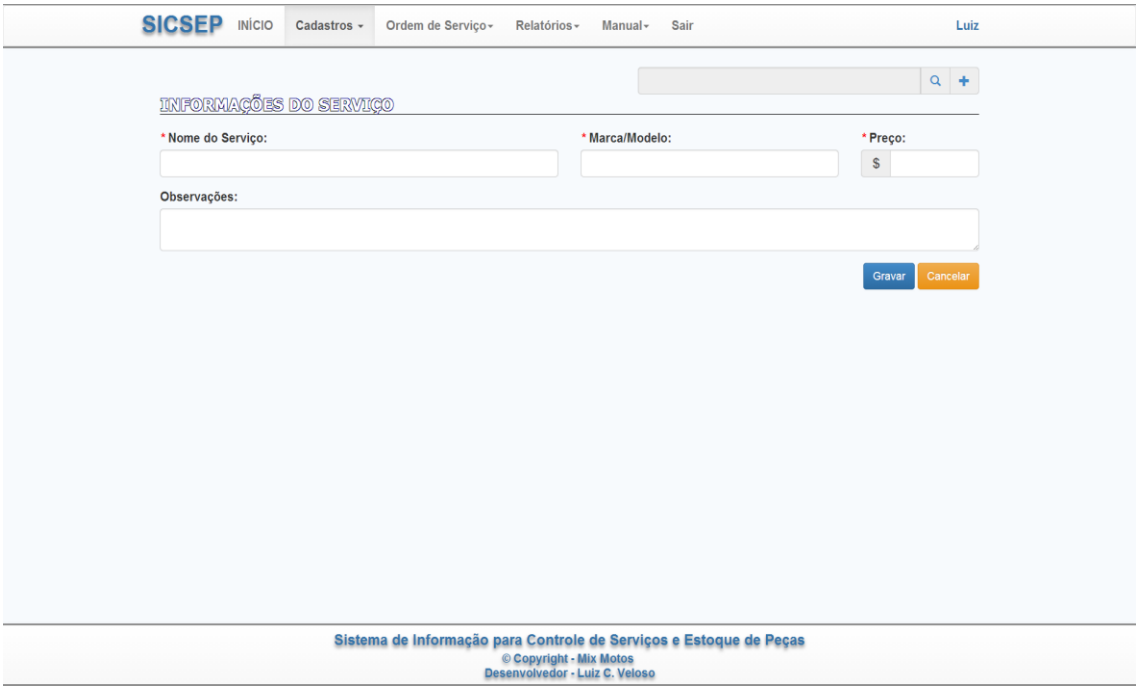

Figura 31. Tela de Informações de Serviço

4.2 Camada de Codificação de Classes

Para a codificação do projeto foi usado à ferramenta de programação Visual Studio 2012 da Microsoft utilizando a linguagem C# seguindo o modelo MVC.

Para dar uma dimensão do sistema e a linguagem usada bem como a estrutura da codificação foram extraídos do projeto os códigos pertinentes a Serviço representando a camada de negócio composta pelas classes Model, DAO e BO.

4.2.1 Camada Model - Classe Serviço

Na Model são codificados os atributos e os métodos Get e Set que são as propriedades que encapsula os atributos.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Negocio.Model
{
    /// <summary>
    /// A classe Serviço é responsável por criar espaços no disco quando instanciada
para manipulação das
    /// informações inerentes a este, que vão ser adicionadas conforme os nomes dos
    /// seus atributos.
    /// </summary>
```

```
    public class Servico
    {
        //---------------------------------///-------------------------///------------
        #region ATRIBUTOS

        /// <summary>
        /// Nome do Serviço = Campo que será inserido o nome do serviço que a loja irá
disponibilizar aos
        /// clientes.
        /// </summary>
        private string nomeServico;
        /// <summary>
        /// Serviço ID = Este ficara responsável pelo identificador único dos serviços
        /// </summary>
        private int servicoID;
        /// <summary>
        ///Preço = Campo que será inserido o preço que custará o serviço.
        /// </summary>
        private decimal preco;
        /// <summary>
        /// Observação = aqui será inserida qualquer observação necessária que diz
respeito a serviço.
        /// </summary>
        private string observacao;
        /// <summary>
        /// Marca e Modelo = será definido aqui para qual a marca e modelo de moto é
aquele serviço.
        /// </summary>
        private string marcaModelo;
        #endregion
        //---------------------------------///-------------------------///------------
        #region PROPRIEDADES
        /// <summary>
        /// Marca e Modelo = será definido aqui para qual a marca e modelo de moto é
aquele serviço.
        /// </summary>
        public string _MarcaModelo
        {
            get { return marcaModelo; }
            set { marcaModelo = value; }
        }
        /// <summary>
        /// Nome do Serviço = Campo que será inserido o nome do serviço que a loja irá
disponibilizar aos
        /// clientes.
        /// </summary>
        public string _NomeServico
        {
            get
            {
                return nomeServico;
            }
            set
            {
                nomeServico = value;
            }
        }
        /// <summary>
        /// Serviço ID = Este ficara responsável pelo identificador único dos serviços
        /// </summary>
        public int _ServicoID
        {
```

```
            get
            {
                return servicoID;
            }
            set
            {
                servicoID = value;
            }
        }
        /// <summary>
        ///Preço = Campo que será inserido o preço que custará o serviço.
        /// </summary>
        public decimal _Preco
        {
            get
            {
                return preco;
            }
            set
            {
                preco = value;
            }
        }
        /// <summary>
        /// Observação = aqui será inserida qualquer observação necessária que diz
respeito a serviço.
        /// </summary>
        public string _Observacao
        {
            get { return observacao; }
            set { observacao = value; }
        }
        #endregion
    }//end Servico
}//end namespace Model
```

4.2.2 Camada BO – Classe Serviço BO

Camada BO é codificada toda a regra de negócio do sistema, como validação, formatação e outros para repassar para camada DAO de maneira pronta para que esta receba sem ter que aplicar nenhuma regra de negócio.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data;
using Negocio.Model;
using Negocio.DAO;

namespace Negocio.BO
{
    /// <summary>
    /// Classe da camada BO responsável pela Regra de negócio.
    /// </summary>
    public class ServicoBO
```

```
    {
        #region ASSINATURAS DE METODOS E DECRARAÇÕES
        /// <summary>
        /// Cria uma lista do Tipo da Classe.
        ///</summary>
        IList<Servico> listaServico;
        /// <summary>
        /// Cria um objeto da camada DAO para dar acesso aos metodos inerente da
Classe.
        /// </summary>
        ServicoDAO objServicoDAO;
        /// <summary>
        /// Cria o objeto para reservar um espaço no disco após instanciado.
        /// </summary>
        Servico objServico;
        /// <summary>
        /// Cria as variaveis.
        /// </summary>
        string valor, id;
        decimal numDecimal;
        int numInt;
        #endregion
        //---------------------------------///------------------------///-----------
---------------------///
        #region CRUD
        /// <summary>
        /// Método responsável em preparar os dados para gravar ou alterar enviando
para a camada DAO.
        /// </summary>
        /// <param name="objServico">Parâmetro que recebe as informações do
Serviço.</param>
        public void Gravar(Servico objServico)
        {
            try
            {
                valor = objServico._Preco.ToString();
                id = objServico._ServicoID.ToString();
                objServicoDAO = new ServicoDAO();
                if (String.IsNullOrEmpty(objServico._NomeServico))
                    throw new Exception("O campo Nome é obrigatório!!!");
                if (String.IsNullOrEmpty(objServico._MarcaModelo))
                    throw new Exception("O campo Marca/Modelo é obrigatório!!!");
                //todo REFERENCIA (RELATIVO AO TryParse http://msdn.microsoft.com/en-
us/library/9zbda557.aspx)
                if (String.IsNullOrEmpty(objServico._Preco.ToString()))
                    throw new Exception("O campo Preço é obriatório!!!");
                if (!Decimal.TryParse(valor, out numDecimal))
                    throw new Exception("Digite um Preço Válido ex: 10,34");
                else if (int.TryParse(id, out numInt) && objServico._ServicoID != 0)
                {
                    try
                    {
                        objServicoDAO.Alterar(objServico);
                    }
                    catch (Exception erro)
                    {

                        throw new Exception("Não foi possível Alterar. erro:" +
erro.Message);
                    }
                }
                else
```

```
                {
                    try
                    {
                        objServicoDAO.Gravar(objServico);
                    }
                    catch (Exception erro)
                    {

                        throw new Exception("Não foi possível Salvar. erro:" +
erro.Message);
                    }
                }
            }
            catch (Exception erro)
            {
                throw new Exception(erro.Message);
            }
        }
        /// <summary>
        /// Método que recebe um id e chama o método excluir da camada DAO.
        /// </summary>
        /// <param name="objServico">Parâmetro que recebe o id do Serviço.</param>
        public void Excluir(Servico objServico)
        {
            try
            {
                objServicoDAO = new ServicoDAO();
                objServicoDAO.Excluir(objServico);
            }
            catch (Exception erro)
            {

                throw new Exception("Não foi possível EXCLUIR. erro: " +
erro.Message);
            }
        }
        #endregion
        //---------------------------------///-------------------------///------------
        #region BUSCAS
        /// <summary>
        /// Método responsável em buscar uma lista de Serviços.
        /// </summary>
        /// <param name="servico">Parâmetro que recebe o nome do serviço.</param>
        /// <returns>Retorna uma lista com os serviços encontradas.</returns>
        public DataTable BuscarListaServico(Servico servico)
        {
            objServicoDAO = new ServicoDAO();
            try
            {
                DataRow dr;
                DataTable dt = new DataTable();
                listaServico = new List<Servico>();

                dt.Columns.Add("ServicoID");
                dt.Columns.Add("Nome do Serviço");
                dt.Columns.Add("Marca/Modelo");
                dt.Columns.Add("Preço");

                listaServico = objServicoDAO.BuscarListaServico(servico);

                // Percorre a lista com o resultado da consulta
```

```
                if (listaServico != null)
                {
                    foreach (Servico objServico in listaServico)
                    {
                        dr = dt.NewRow();

                        dr["ServicoID"] = objServico._ServicoID;
                        dr["Nome do Serviço"] = objServico._NomeServico;
                        dr["Preço"] = objServico._Preco;
                        dr["Marca/Modelo"] = objServico._MarcaModelo;
                        dt.Rows.Add(dr);
                    }
                }
                return dt;
            }
            catch (Exception erro)
            {
                throw new Exception("Não foi possível Fazer a BUSCA. erro: " +
erro.Message);
            }
        }
        /// <summary>
        /// Método responsável em buscar um tipo de Serviços.
        /// </summary>
        /// <param name="servico">Parâmetro que recebe o id do Serviço.</param>
        /// <returns>Retorna um objeto com preenchido com as informações de um serviço
especifico.</returns>
        public Servico BuscarServico(Servico servico)
        {
            try
            {
                objServicoDAO = new ServicoDAO();
                objServico = new Servico();
                objServico = objServicoDAO.BuscarServico(servico);
                return objServico;
            }
            catch (Exception erro)
            {
                throw new Exception("Não foi possível Fazer a BUSCA. erro: " +
erro.Message);
            }
        }
        #endregion
    }
}
```

4.2.3 Camada DAO – Classe Serviço DAO

Camada DAO é responsável por se conectar com o Banco de Dados, receber os dados que vem da camada BO e transformar estes em relacional para enviar para o Banco de Dados.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.SqlClient;
using Negocio.Model;
```

```
namespace Negocio.DAO
{
    /// <summary>
    ///Classe DAO, responsável por fazer as persistências na tabela de serviço no
banco.
    /// </summary>
    public class ServicoDAO
    {
        #region ASSINATURA DOS METODOS
        /// <summary>
        /// Instancia de um novo objeto do tipo StringBuilder.
        /// </summary>
        StringBuilder sql = new StringBuilder();
        /// <summary>
        ///   Instancia de um novo objeto do tipo SqlCommand.
        /// </summary>
        SqlCommand comando = new SqlCommand();
        /// <summary>
        /// Instancia de uma lista para receber as informações dos produtos buscados
no banco.
        /// </summary>
        IList<Servico> listaServico;

        #endregion
        //---------------------------------///------------------------///------------
        #region CRUD
        /// <summary>
        /// Método que converte o objeto em liguagem sql e grava no banco as
informações.
        /// </summary>
        /// <param name="objServico">Informações do serviço recebida como
parâmetro.</param>
        public void Gravar(Servico objServico)
        {
            try
            {
                sql.Remove(0, sql.Length);
                comando.Parameters.Clear();

                sql.Append("Insert Into Servico(nomeServico,preco, observacao,
marcaModelo) values(@nomeServico, @preco, @observacao, @marcaModelo)");

                comando.CommandText = sql.ToString();

                comando.Parameters.AddWithValue("@nomeServico",
objServico._NomeServico);
                comando.Parameters.AddWithValue("@preco", objServico._Preco);
                comando.Parameters.AddWithValue("@observacao",
objServico._Observacao);
                comando.Parameters.AddWithValue("@marcaModelo",
objServico._MarcaModelo);

                ConexaoBanco.Crud(comando);
            }
            catch (Exception erro)
            {
                throw new Exception(erro.Message);
            }
        }
        /// <summary>
```

```
        /// Método que converte o objeto em liguagem sql persistindo no banco as
informações para alteração do serviço.
        /// </summary>
        /// <param name="objServico">Informações do serviço recebida como
parâmetro.</param>
        public void Alterar(Servico objServico)
        {
            try
            {
                sql.Remove(0, sql.Length);
                comando.Parameters.Clear();

                sql.Append("update Servico set nomeServico=@nomeServico, preco=@preco,
");
                sql.Append("observacao = @observacao, marcaModelo=@marcaModelo where
servicoID=@servicoID");

                comando.CommandText = sql.ToString();

                comando.Parameters.AddWithValue("@nomeServico",
objServico._NomeServico);
                comando.Parameters.AddWithValue("@preco", objServico._Preco);
                comando.Parameters.AddWithValue("@servicoID",
objServico._ServicoID.ToString());
                comando.Parameters.AddWithValue("@observacao",
objServico._Observacao);
                comando.Parameters.AddWithValue("@marcaModelo",
objServico._MarcaModelo);
                ConexaoBanco.Crud(comando);
            }
            catch (Exception erro)
            {
                throw new Exception(erro.Message);
            }
        }
        /// <summary>
        ///  Método responsável em fazer a exclusão do serviço no banco.
        /// </summary>
        /// <param name="objServico">Informações do serviço recebida como
parâmetro.</param>
        public void Excluir(Servico objServico)
        {
            try
            {
                sql.Remove(0, sql.Length);
                comando.Parameters.Clear();

                sql.Append("Delete from Servico where servicoID = @servicoID");

                comando.CommandText = sql.ToString();

                comando.Parameters.AddWithValue("@servicoID", objServico._ServicoID);
                ConexaoBanco.Crud(comando);
            }
            catch (Exception erro)
            {
                throw new Exception(erro.Message);
            }
        }

        #endregion
        //---------------------------------///------------------------///------------
```

```
        #region BUSCAS
        /// <summary>
        /// Método responsável em fazer a busca dos serviços solicitados.
        /// </summary>
        /// <param name="servico">Nome do serviço passado com parâmetro dentro do
objeto serviço.</param>
        /// <returns>Este método retorna uma lista de serviço.</returns>
        public IList<Servico> BuscarListaServico(Servico servico)
        {
            try
            {
                sql.Remove(0, sql.Length);
                comando.Parameters.Clear();

                sql.Append("Select * from Servico where nomeServico Like
@nomeServico");
                comando.Parameters.AddWithValue("@nomeServico", "%" +
servico._NomeServico + "%");
                comando.CommandText = sql.ToString();
                SqlDataReader dr = ConexaoBanco.Selecionar(comando);
                listaServico = new List<Servico>();

                if (dr.HasRows)
                {
                    while (dr.Read())
                    {
                        Servico objServico = new Servico();
                        objServico._ServicoID = (int)dr["servicoID"];
                        objServico._NomeServico = (string)dr["nomeServico"];
                        objServico._Preco = (decimal)dr["preco"];
                        objServico._Observacao = (string)dr["observacao"];
                        objServico._MarcaModelo = (string)dr["marcaModelo"];
                        listaServico.Add(objServico);

                    }
                }
                else
                {
                    listaServico = null;
                }
                return listaServico;
            }
            catch (Exception erro)
            {
                throw new Exception(erro.Message);
            }
        }
        /// <summary>
        /// Método responsável em fazer a busca do serviços solicitados.
        /// </summary>
        /// <param name="servico">Id do serviço passado com parâmetro dentro do objeto
serviço.</param>
        /// <returns>Este método retorna um objeto do tipo serviço.</returns>
        public Servico BuscarServico(Servico servico)
        {
            try
            {
                sql.Remove(0, sql.Length);
                comando.Parameters.Clear();

                sql.Append("Select * from Servico where servicoID = @servicoID");
                comando.Parameters.AddWithValue("@servicoID", servico._ServicoID);
```

```
                comando.CommandText = sql.ToString();
                SqlDataReader dr = ConexaoBanco.Selecionar(comando);
                Servico objServico = new Servico();
                if (dr.HasRows)
                {
                    dr.Read();

                    objServico._ServicoID = (int)dr["servicoID"];
                    objServico._NomeServico = (string)dr["nomeServico"];
                    objServico._Preco = (decimal)dr["preco"];
                    objServico._Observacao = (string)dr["observacao"];
                    objServico._MarcaModelo = (string)dr["marcaModelo"];
                }
                else
                {
                    objServico = null;
                }
                return objServico;
            }
            catch (Exception erro)
            {
                throw new Exception(erro.Message);
            }
        }

        #endregion
        //---------------------------------///-------------------------///------------
    }
}
```

4.3 Testes

Teste é o ato de avaliar algo, como por exemplo, o conhecimento, funcionamento e outros, mas quando falamos de software o “Teste de software é o processo de execução de um produto para determinar se ele atingiu suas especificações e funcionou corretamente no ambiente para o qual foi projetado” (Cláudio Dias Neto)

Os quadros (16 a 31) relacionam a sequência dos quadros conforme exemplos apresentados, onde descrevem todos os passos necessários para o desenvolvimento de alterar e excluir uma Ordem de Serviço.

4.3.1 Teste das ações e resultados esperado do botão “Novo”.

Número do Teste:	**01**
Localização:	Ordem Serviço => Novo
Objeto de Teste:	Exibição de tela de cadastro de uma nova ordem de serviço

Caso de Teste:	Botão “Novo”.
Pré-Condição:	a) O usuário deverá está devidamente autenticado b) As informações de Cliente, Funcionário, Moto, Serviços e Produtos já deverão está previamente inseridos no sistema.
Procedimentos:	Clicar no botão novo.
Resultado esperado:	a) Campo data será preenchido automaticamente pelo sistema. b) Os Mecânicos registrados serão exibidos para seleção. c) Todos os campos liberados para inserção. d) Botão Cancelar e Gravar disponível.

Quadro 9. Teste Botão Novo O.S.

4.3.2 Teste das ações e resultados esperado do botão “Buscar Cliente”.

Número do Teste:	**02**
Localização:	Ordem Serviço => Novo => Buscar Cliente
Objeto de Teste:	Testar a busca de Cliente.
Caso de Teste:	Botão “Buscar Cliente”.
Pré-Condição:	a) Ações do Teste 01 deverão já ter sido executadas. b) As informações do Cliente já deverão está inseridas no Banco de Dados.
Procedimentos:	Após a execução da pré-condição escreva o nome do Cliente ou apenas clique em “Buscar Cliente”
Resultado esperado:	a) Se digitado parte de um nome exibirá uma lista resumida dos clientes que conterem parte desses. b) Se apenas Clicado no buscar será exibido uma lista contendo todos os clientes existentes no banco.

Quadro 10. Teste Botão Selecionar Cliente

4.3.3 Teste das ações e resultados esperado do botão “Selecionar Cliente”.

Número do Teste:	**03**
Localização:	Ordem Serviço => Novo => Buscar Cliente => Selecionar Cliente

Objeto de Teste:	Testar selecionar de Cliente
Caso de Teste:	Botão “Selecionar Cliente”
Pré-Condição:	Ações do Teste 01 e 02 deverão já ter sido executadas.
Procedimentos:	Escolha na lista o nome do Cliente e clique em “Selecionar Cliente”
Resultado esperado:	a) Os campos nome do cliente, telefone, CPF/CNPJ, endereço e bairro serão preenchidos com as informações inerentes aos mesmos. b) O campo nome ficará bloqueado e botão limpar informações do cliente tomará o lugar do “Buscar Cliente”

Quadro 11. Teste Botão Buscar Moto

4.3.4 Teste das ações e resultados esperado do botão “Buscar Moto”

Número do Teste:	**04**
Localização:	Ordem Serviço => Novo => Buscar Moto
Objeto de Teste:	Testar “Buscar Moto”
Caso de Teste:	Botão Buscar Moto
Pré-Condição:	a) Ações do Teste 01 deverá já ter sido executado. b) As informações da Moto já deverão estar inseridas no Banco de dados.
Procedimentos:	Após a execução da pré-condição escreva o número da placa da Moto ou apenas clique em “Buscar Moto”
Resultado esperado:	a) Se digitado parte de um número da placa exibirá uma lista resumida dos clientes que conterem parte desses. b) Se apenas estiver Clicado no buscar será exibido uma lista contendo todas as Motos existentes no Banco.

Quadro 12. Teste Botão Selecionar Moto

4.3.5 Teste das ações e resultados esperado do botão “Selecionar Moto”.

Número do Teste:	**05**
Localização:	Ordem Serviço => Novo => Buscar Moto => Selecionar Moto

Objeto de Teste:	Testar Seleção de Moto
Caso de Teste:	Botão “Selecionar Moto”
Pré-Condição:	Ações do Teste 01 e 04 deverão já ter sido executadas.
Procedimentos:	Escolha na lista a placa da Moto e clique no botão “Selecionar Moto”.
Resultado esperado:	a) Os campos Placa Marca/Modelo e ano serão preenchidos com as informações inerentes aos mesmos. b) O campo Placa ficará bloqueado e botão limpar informações da moto tomará o lugar do “Buscar Moto”

Quadro 13. Teste Botão Buscar Produto

4.3.6 Teste das ações e resultados esperado do botão “Buscar” Serviço.

Número do Teste:	**06**
Localização:	Ordem Serviço => Novo => Tab Produto=>Buscar Produto
Objeto de Teste:	Testar de Buscar Produto
Caso de Teste:	Botão Buscar Produto
Pré-Condição:	Ações do Teste 01 deverá já ter sido executadas.
Procedimentos:	a) Após a execução das pré-condições clique no botão “Buscar Produtos”. b) As informações do Produto já deverão estar inseridas no Banco de dados.
Resultado esperado:	a) Se digitado parte do nome exibirá uma lista resumida dos Produtos que conterem parte desse. b) Se apenas estiver Clicado no buscar será exibido uma lista contendo todos os Produtos existentes no Banco.

Quadro 14. Teste Botão Selecionar Produto

4.3.7 Teste das ações e resultados esperado do botão “Selecionar Produto”.

Número do Teste:	**07**
Localização:	Ordem Serviço => Novo => Buscar Produto => Selecionar Produto

Objeto de Teste:	Testar Selecionar Produto
Caso de Teste:	Botão “Selecionar Produto”
Pré-Condição:	Ações do Teste 01, 06 deverão já ter sido executadas.
Procedimentos:	Após a execução das pré-condições clique no botão “Selecionar Produto”.
Resultado esperado:	a) O campo nome do produto ficará bloqueado. b) O botão Limpar Campos de Inserção surgirá no lugar do Botão Buscar Produto. c) O campo Nome do produto e preço será preenchido com suas informações. d) O campo quantidade liberado para inserção de valor.

Quadro 15. Teste Campo Quantidade.

4.3.8 Teste das ações e resultados esperado do “Campo Quantidade”.

Número do Teste:	**08**
Localização:	Ordem Serviço => Novo => Tab Produto=> Campo Quantidade.
Objeto de Teste:	Testar Campo Quantidade.
Caso de Teste:	“Campo Quantidade”
Pré-Condição:	Ações do Teste 01, 06 e 07 deverão já ter sido executadas.
Procedimentos:	Após a execução das pré-condições Insira a “Quantidade” de produtos.
Resultado esperado:	a) Campo quantidade ficará bloqueado. b) Exibirá o botão de limpar quantidade e Inserir o produto.

Quadro 16. Teste Botão Inserir Produto.

4.3.9 Teste das ações e resultados esperado do “Botão Inserir Produto”.

Número do Teste:	**09**
Localização:	Ordem Serviço => Novo => Tab Produto=> Botão Inserir Produto.
Objeto de Teste:	Testar Botão Inserir Produto.
Caso de Teste:	“Botão Inserir Produto”

Pré-Condição:	Ações do Teste 01, 06, 07 e 08 deverão já ter sido executadas.
Procedimentos:	Após a execução das pré-condições clique no “Botão Inserir Produto”.
Resultado esperado:	a) Verifica se existe uma ordem de serviço se não, grava a ordem de serviço. b) Carrega uma lista com os produtos inseridos. c) Soma os valores dos produtos e serviços e mostra nos seus respectivos campos.

Quadro 17. Teste Botão Buscar Serviço

4.3.10 Teste das ações e resultados esperado do botão Buscar Serviço.

Número do Teste:	**10**
Localização:	Ordem Serviço => Novo => tab Serviço => Buscar Serviço
Objeto de Teste:	Testar Buscar Serviço
Caso de Teste:	Botão “Buscar Serviço”
Pré-Condição:	a) Ações do Teste 01 deverá já ter sido executadas. b) Serviço deverão já estar inseridos no Banco.
Procedimentos:	Após a execução das pré-condições clique no botão “Buscar Serviço”.
Resultado esperado:	Exibe uma lista de serviços em um GridView

Quadro 18. Teste Botão Inserir Serviço

4.3.11 Teste das ações e resultados esperado do botão Inserir Serviço.

Número do Teste:	**11**
Localização:	Ordem Serviço => Novo => tab Serviço => Buscar Serviço => Inserir Serviço
Objeto de Teste:	Testar Inserir Serviço
Caso de Teste:	Botão “Inserir Serviço”
Pré-Condição:	Ações do Teste 01 e 10 deverá já ter sido executadas.
Procedimentos:	Após a execução das pré-condições clique no botão “Inserir Serviço”.

Resultado esperado:	a) Exibe uma lista de serviços inseridos em um GridView b) Soma os valores dos Serviços e Produtos, mostra nos seus respectivos campos.

Quadro 19. Teste Botão Gravar O.S.

4.3.12 Teste das ações e resultados esperado do botão Gravar.

Número do Teste:	**12**
Localização:	Ordem Serviço => Novo => Gravar
Objeto de Teste:	Testar gravar ordem de serviço
Caso de Teste:	Botão gravar
Pré-Condição:	Ações do Teste 01, 03 e 05 deverão já ter sido executadas.
Procedimentos:	Após a execução das pré-condições clique no botão “Gravar” Ordem de serviços.
Resultado esperado:	a) Grava no banco todos os dados inseridos b) Redireciona para a tela inicial da ordem de serviço c) Desbloqueia o campo buscar, botão buscar e botão novo. d) Busca todas as ordens de serviço em abertas e mostra na ordem decrescente da última para a primeira em um GridView. e) Oculta os Botões Gravar e Cancelar

Quadro 20. Teste Botão Editar O.S.

4.3.13 Teste das ações e resultados esperado do botão “Editar”.

Número do Teste:	13
Localização:	Ordem Serviço => Selecionar => Editar
Objeto de Teste:	Testar alterar ordem de serviço
Caso de Teste:	Botão Editar
Pré-Condição:	a) O usuário deverá estar devidamente autenticado b) Existir uma ordem de Serviço gravada e não finalizada.
Procedimentos:	a) Selecione Ordem de Serviço no Menu. b) Escolha a Ordem de Serviço que deseja alterar

	na lista de serviços abertas e clique em “Selecionar” c) Clique em editar
Resultado esperado:	a) Habilita os Campos e lista de serviço e produto que tem permissão para ser alterado. b) Mostra os botões Cancelar e Gravar. c) Oculta o botão Editar.

Quadro 21. Teste Botão Excluir O.S.

4.3.14 Teste das ações e resultados esperado do botão “Excluir”.

Número do Teste:	**14**
Localização:	Ordem Serviço => Selecionar => Alterar =>Excluir
Objeto de Teste:	Testar Excluir ordem de serviço
Caso de Teste:	Botão Excluir
Pré-Condição:	a) Selecione uma Ordem de serviço em Ordem de Serviço b) A ordem de Serviço não deve estar Finalizada
Procedimentos:	Selecione o Botão Excluir
Resultado esperado:	a) Exclui todos os itens de serviço b) Exclui a Ordem de Serviço c) Redireciona para a tela inicial da Ordem de Serviço d) Mostra a lista com os serviços em aberto ainda existentes

Quadro 22. Teste Botão Finalizar O.S.

4.3.15 Teste das ações e resultados esperado do botão “Finalizar”.

Número do Teste:	**15**
Localização:	Ordem Serviço => Selecionar =>Finalizar
Objeto de Teste:	Testar Finalizar ordem de serviço
Caso de Teste:	Botão Finalizar
Pré-Condição:	a) A ordem de Serviço não deve estar Finalizada
Procedimentos:	Selecione o Botão Finalizar
Resultado esperado:	a) Preenche na Ordem de Serviço a data de finalização

	b) Fecha a tela de Edição c) Redireciona para a tela inicial da Ordem de Serviço d) Mostra a lista com os serviços em aberto ainda existentes

Quadro 23. Teste Botão Finalizar

4.3.16 Teste das ações e resultados esperado do botão “Cancelar”.

Número do Teste:	**16**
Localização:	Ordem Serviço
Objeto de Teste:	Testar Cancelar ordem de serviço
Caso de Teste:	Botão Cancelar
Pré-Condição:	Estar na Tela de Cadastro de Ordem de Serviço
Procedimentos:	Selecione o Botão Cancelar
Resultado esperado:	a) Redireciona para a tela inicial da Ordem de Serviço b) Mostra a lista com os serviços em aberto ainda existentes

Quadro 24. Teste Botão Cancelar.

5 CONCLUSÃO

A realização deste estudo teve como objetivo demonstrar o processo de desenvolvimento de um sistema de informação para que possibilite a efetivação de sistematização, análise, integração dos dados e registros bem como, executar e controlar o estoque e serviços na empresa Mix Motos.

Pois, se sabe que atualmente o mercado de trabalho cobra, e as empresas se encontram obrigadas a investir em avanços para poder competir com a concorrência, procurando sempre, em um sistema de informação, saídas para suprimir investimentos desnecessários e melhoria na qualidade dos serviços prestados.

Por meio deste estudo, foi desenvolvido um Sistema de Informação para Controle de Serviços e Estoque de Peças – SICSEP, que tem finalidade de controlar os serviços e estoque como também gerenciar informações dos clientes e funcionários através um registro de informações no sistema.

Através das fases da criação deste sistema que se insere um diagnóstico do programa, é que se aceita usar todas as experiências adquirida, para um melhor entendimento de diversas definições utilizada na prática apresentam a real dificuldade que envolve para se desenvolver um sistema específico com todas as fases dentro de uma modelagem.

No entanto, as ferramentas que são utilizadas para se criar e diagnosticar necessitadas no decorrer do sistema é de alta relevância para se alcançar um resultado satisfatório para suprir a necessidade de melhoria e qualidade de serviços prestados pelas empresas, principalmente para atender os anseios daqueles que esperam resultado satisfatório.

Por fim, o estudo em questão auxiliou de forma satisfatória o cotidiano da empresa Mix Motos através de um controle eficaz de seus produtos e serviços, deixando todos os envolvidos no projeto satisfeitos com o resultado.

REFERÊNCIAS

BOOCH, Grady; James Rumbauhk; Ivar Jacobson. **UML; guia do usuário**. – Rio de Janeiro: Elsevier, 2005.

GUEDES, Gilleanes T. A. **UML2: uma abordagem prática**. – São Paulo: Novatec, 2008.

LIMA, Adilson da Silva. **UML 2.0 do Requisito à Solução.** 3ª. ed. Cidade: Érica, 2008.

LAKATOS, Eva Maria. MARCONI Marina de Andrade. **Fundamentos de Metodologia Científica**. – 6. ed - São Paulo: Atlas, 2009.

MARTINS, A. P. **Saúde em rede. Rev. Saúde Business**. Ano 2, n.8. 2009.

SILVA, Ricardo Pereira. **UML: modelagem orientada a objetos**. – Florianópolis: Visual Books, 2007.

WAZLAWICK, Raul Sidnei. **Análise e projeto de sistemas de informação orientados a objeto**. Rio de Janeiro: Elsevier, 2004.

INTERNET

Cláudio Dias Neto, A. C. (s.d.). Introdução a Teste de Software. Disponível no site: http://www.comp.ita.br/~mluisa/TesteSw.pdf. Acesso em 30.06.2014, 18:00.

Introdução ao ***Enterprise Architect***. Disponível no site: ttp://www.sparxsystems.com/enterprise_architect_user_guide/10/standard_uml_models/whatisuml.htm. Acesso em 30.05.2014, 10:44;00.

APÊNDICE A - Ata da Entrevista Qualitativa

Aos 19 dias de outubro 2012, 14h30min às 15h10min na Mix Motos rua plácido de castro, nº 7925, Bairro JK-I o senhor Luiz Costa Veloso, aluno de Sistemas de Informação entrevistou o senhor Flavio de Jesus, proprietário da Empresa Mix Motos onde foi discutido na reunião o funcionamento, como também as formas de realizações dos serviços da empresa. Ficou acordado que seria feito um resumo de tudo para assim marcar uma nova reunião para esclarecimento de eventuais duvidas de funcionamento da empresa.

APÊNDICE B - Ata da Entrevista Quantitativa

Aos 19 dias de junho 2013, 15h30min às 16h00min na empresa Mix Motos rua plácido de castro, nº 7925, Bairro JK-I o senhor Luiz Costa Veloso, aluno de Sistemas de Informação entrevistou o senhor Flavio de Jesus, proprietário da Empresa Mix Motos para tirar algumas duvidas sobre o sistema que vai ser desenvolvido. Ficou acordado que no surgimento de eventuais duvidas o aluno Luiz entraria em contato com o Senhor Flávio para marcar uma nova reunião.

APÊNDICE C - Entrevista Qualitativa

Em entrevista com o Sr. Flávio de Jesus, foram esclarecidas as rotinas de serviços da empresa Mix Motos, a fim de se estabelecerem as funcionalidades e o objetivo do sistema de informação, este relatou que precisa de um sistema que controle os serviços realizados para facilitar a solução de eventual reclamação por parte do cliente, pois quando precisa fazer a conferencia dos produtos e serviços que foram executados na moto, este necessita procurar no bloco de pedido, verificando de um a um, até achar o pedido deste. “Também tenho um problema sério com meu estoque onde tenho uma enorme dificuldade em manter o meu controle, me trazendo vários transtornos no meu dia a dia, como quando vou realizar uma venda ou realizar novos pedidos, que na maioria das vezes tenho de verificar se ainda tenho em estoque, sendo assim se o sistema puder solucionar esses transtornos e melhorar significativamente a agilidade no fechamento das conferencias de ordem de serviço ao final do dia, já seria uma melhora entanto.” Quanto sua rotina, descreveu que o cliente ao chegar à loja vai até o balcão e solicita o produto que lhe interessa ou solicita ao funcionário da loja que verifique o defeito da sua moto. Depois o mecânico informa o tipo de serviço e as peças, então é anotado em blocos de pedidos onde são preenchidas as informações do mecânico, atendente, serviço, produto, valores, moto e cliente. No final do expediente é feito a conferência destes.

APÊNDICE D - Entrevista Quantitativa

1 – Os serviços serão divididos por categorias?

R=> Não.

2 - Quais os dados do cliente e funcionário serão obrigatórios?

R=> Funcionário = nome, endereço, data de nascimento, CPF, telefone e cargo.

R=> Cliente = nome, endereço, data de nascimento, endereço, Bairro, Cidade, CPF/CNPJ e telefone.

3 – Como será definido o nível de permissões para acesso dos usuários?

R=> será dividido em dois níveis onde eu vou ser o nível máster e os funcionários terão acesso à ordem de serviço, moto e cliente podendo deixar livre os produtos e serviços para consulta.

4 – As peças já vêm de fábrica com algum tipo de identificação padrão?

R=> Quase todas vem com código de barra.

5 – O mesmo funcionário que verifica o defeito na moto, é o mesmo que realiza o serviço?

R=> Sim

6 – O sistema terá que identificar os produtos que estão reservados para não serem vendidos?

R=> sim

7 – Há produtos com o mesmo nome, mas com valores diferentes para cada tipo de moto?

R=> Sim.

8 – O sistema será hospedado local ou web?

R=> local.

9 – A ordem de serviço será aberta por veículo ou pelo nome do cliente podendo ter mais de um veículo?

R=> por cliente e deverá ter uma moto em cada ordem de serviço.

10 – Você vai querer agrupar os produtos por categorias ou grupos?

Não.

11- Há serviços com o mesmo nome e com valores diferentes para cada tipo de moto?

R - Sim

12 - Como você prefere definir o usuário do sistema? Exemplo: por um nome ou número do CPF.

R - pode ser por um nome mesmo.