FACULDADE DE TECNOLOGIA DE AMERICANA

CURSO ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

Lucas Martins Menezes

ANÁLISE DO DESENVOLVIMENTO DE SOFTWARE PARA UMA OFICINA MECÂNICA

Americana, SP

2016

FACULDADE DE TECNOLOGIA DE AMERICANA

CURSO ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

Lucas Martins Menezes

ANÁLISE DO DESENVOLVIMENTO DE SOFTWARE PARA UMA OFICINA MECÂNICA

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do Curso de Análise e desenvolvimento de sistemas sob a orientação do Prof. Mestre Anderson Luiz Barbosa

Área de concentração: Engenharia de Software

Americana, SP

2016

Lucas Martins Menezes

ANÁLISE DO DESENVOLVIMENTO DE SOFTWARE PARA UMA OFICINA MECÂNICA

Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Análise e desenvolvimento de sistemas pelo CEETEPS/Faculdade de Tecnologia - FATEC/Americana.

Área de concentração: Engenharia de Software

Americana, 05 de dezembro de 2016.

Banca Examinadora:

Prof. Mestre Anderson Luiz Barbosa

Prof. Mestre Kleber de Oliveira Andrade

Prof. Mestre Wladimir da Costa

FICHA CATALOGRÁFICA – Biblioteca Fatec Americana - CEETEPS
Dados Internacionais de Catalogação-na-fonte

M511a MENEZES, Lucas Martins
Análise do desenvolvimento de software para uma oficina mecânica / Lucas Martins Menezes. – Americana: 2016.
83f..

Monografia (Curso de Tecnologia em Análise e Desenvolvimento de Sistemas). - - Faculdade de Tecnologia de Americana – Centro Estadual de Educação Tecnológica Paula Souza.
Orientador: Prof. Ms. Anderson Luiz Barbosa

1. Engenharia de software 2. Desenvolvimento de software I. BARBOSA, Anderson Luiz II. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana.

CDU: 681.3.05

AGRADECIMENTOS

Em primeiro lugar gostaria de agradecer à minha família que desde que me entendo por gente têm me incentivado para que eu tenha uma boa formação acadêmica.

Aos meus amigos por entenderem os meus sonhos e planejamentos futuros e me apoiarem.

Aos professores por toda a atenção e suporte desde que comecei minha jornada acadêmica e também ao senhor Airton pela confiança postada em mim para o desenvolvimento deste projeto.

DEDICATÓRIA

Aos meus pais, José e Syrlei, e irmã, Elizabete, as pessoas mais importantes que tenho na minha vida. Estes que nunca mediram esforços para proporcionar o melhor para mim. O presente projeto é apenas uma pequena parte de tudo que devo retribuir à estas pessoas.

RESUMO

Atualmente a ciência tecnológica e a informação são peças fundamentais da vida humana na sociedade global. A aceleração das comunicações e dos transportes diminuíram as distâncias geográficas, o capital se internacionalizou e rompeu fronteiras. Cada vez mais os Sistemas de Informação vêm assumindo um papel estratégico nas organizações, exigindo que as empresas utilizem tecnologias para realizar transações e para estruturar a comunicação com seus públicos. Cabe aos administradores utilizar-se dos sistemas de informação para conseguir rapidez nas negociações, mantendo sua empresa competitiva. De fato, a tecnologia é a marca do nosso tempo. Tendo em vista o cenário atual, o objetivo do presente trabalho é abordar o campo da Engenharia de Software aplicado de uma forma prática à análise e desenvolvimento de software possibilitando assim um estudo e acompanhamento dos processos de desenvolvimento de um sistema de gestão para uma oficina mecânica visando a melhoria do ambiente de negócios do cliente.

Palavras Chave: Engenharia de Software; Oficina Mecânica; Sistema de Gestão.

ABSTRACT

Currently, the technological science and information are fundamental parts of human’s life in global society. The acceleration of communications and transport has been reduced geographic distances, the capital has been internationalized and it is breaking borders. Increasingly, Information Systems are assuming a strategic role in organizations, requiring companies to use technologies to carry out transactions and to structure communication with their audiences. It is up to the administrators to use the information systems to achieve speed in the negotiations, keeping their company competitive. In fact, technology is the hallmark of our time. In the current scenario, the objective of the present work is to approach the field of Software Engineering applied in a practical way to the analysis and development of software enabling a studying and monitoring of the development and management system processes for a mechanical workshop with a view to improving the customer's business environment.

Keywords: Software Engineering; Mechanical workshop; Management system

SUMÁRIO

LISTA DE FIGURAS E TABELAS

1. INTRODUÇÃO

Sommerville (2007, p. 3) assegura que quase que todos os países necessitam e utilizam sistemas informatizados complexos para tratar de seus negócios. Sendo assim a importância de um bom sistema funcionando dentro de um processo empresarial tem se tornado cada vez mais importante.

Os sistemas de informação possuem papel estratégico nas organizações, sendo utilizados principalmente pelas grandes empresas para a realização de transações e para estruturar a comunicação com seus públicos.

“Sistema de informação é um conjunto de componentes inter-relacionados numa ordem específica que coletam e manipulam dados, para então gerar informações” (Côrtes, 2008, p. 24). Essas informações são então utilizadas pelos usuários para a tomada de decisões.

Cada vez mais os Sistemas de Informação vêm assumindo um papel estratégico nas organizações, exigindo que as empresas utilizem tecnologias para realizar transações e para estruturar a comunicação com seus públicos. Cabe aos administradores utilizar-se do recurso para conseguir rapidez nas negociações, mantendo sua empresa competitiva.

A empresa, Mecânica Moderna, de origem familiar, possui uma grande credibilidade com os clientes da região. Possui dois funcionários, estes os quais realizam diversos tipos de serviços relacionados a mecânica em geral. Não há fornecedor fixo para os produtos, os mesmos são adquiridos de acordo com a necessidade. Fica ao encargo do Sr. Airton, todo o serviço de gestão de negócios bem como a compra dos produtos necessários nos serviços prestados e também a supervisão do trabalho realizado pelos seus dois funcionários.

Sommerville (2007, p. 4) distingue software em dois tipos de produto: Produto genérico, que como o próprio nome diz, é um software mais abrangente que busca atender a maior gama possível de negócios. E Produto sob encomenda, do caso este software é desenvolvido especialmente para um cliente em especifico.

Atualmente, Sr. Airton, o proprietário, gerencia todas as suas atividades empresariais, bem como as contas, orçamentos, ordens de serviço e entre outras funções, com uma combinação de dois sistemas ERP genéricos: O ERP SIC (Sistema Integrado Comercial) da empresa Softmakers Tecnologia de Soluções

Ltda., ERP Millenium da empresa Alfatest ind. e com de produtos eletrônicos S/A. Além dos sistemas ainda existem documentos manuscritos.

O sistema de gestão SIC é um sistema com diversas funcionalidades, estas nas quais algumas são necessárias e outras não tem utilidade no modelo de negócio do cliente. Existe uma versão gratuita dele, em termos de funcionalidades é igual à versão paga que custa R$ 175,00, porém ao expirar o tempo de testes o software já não funciona mais.

O sistema de gestão Millenium também possui uma versão paga e uma gratuita, o cliente utiliza a versão gratuita que possui apenas o modulo de cadastros e a gestão das ordens de serviço, o que serve apenas para armazenar e gerar relatórios dos cadastros sem envolver de fato os processos de negócio que a empresa trata, sendo assim não há como gerar informação qualitativa para o negócio. As opções oferecidas pela versão grátis desde software são: Cliente, veiculo, serviço, peças, oficina, técnicos, ordens de serviço e um conjunto de relatórios que são: Lista de clientes, aniversariantes, serviços, peças, mala-direta e faturamento.

Para Sommerville (2007, p. 5), o principal objetivo de um bom software é proporcionar ao usuário a funcionalidade e desempenho e que ele necessita. Com funcionalidade e desempenho entregues satisfatoriamente a tendência é que o software ajude a reduzir horas de trabalho. Muitas tarefas manuais podem ser substituídas facilmente e deixam mais tempo livre para serem realocados em outras áreas.

Devido a existência de um número excessivo ferramentas gerenciadoras bem como o excesso de funcionalidades nelas. Há um grande bloqueio no armazenamento, processamento e transformação dos dados em informação útil.

Nota-se uma grande confusão no que se refere à organização. O setor mais afetado é o de vendas, mais precisamente os registros de serviços prestados ou orçados, ou seja, as ordens de serviço e os orçamentos. Estas, atividades de gerenciamento, indispensáveis, pois são os processos diretamente ligados a captação de recursos da empresa.

Tendo em vista o cenário atual, esse trabalho estabelece objetivos em duas perspectivas diferentes uma vez que há dois valores gerados para diferentes “pessoas”. A perspectiva acadêmica onde se gera valor para a instituição de ensino, o objetivo é propor um sistema para a oficina Mecânica

Moderna do Sr. Airton, desenvolve-lo e analisar os processos de engenharia de software envolvidos neste. Na perspectiva do desenvolvimento de software em si, onde se gera valor especificamente para o cliente, no caso Sr. Airton, o objetivo é entender o problema dele e formular um sistema que atenda às suas necessidades da maneira mais pontual possível no intuito de agilizar seus processos administrativos através de uma ferramenta de gerenciamento única para todo o negócio.

Este projeto justifica seu valor uma vez que boa parte dos conceitos teóricos aprendidos em sala de aula ganham aspecto prático na real aplicação dos conhecimentos ao desenvolvimento de um sistema para um cliente real. Este também possibilita uma maior compreensão e dos processos envolvidos do desenvolvimento de um sistema bem como a importância dos mesmos para o sucesso e qualidade na execução de um projeto. Acrescenta-se mais valor a partir do momento em que se gera valor além do ambiente acadêmico. Havendo um cliente para receber os frutos de uma boa base teórica aprendida do ambiente acadêmico e aplicada na pratica para gerar valor ao seu negócio, pode-se dizer que é um fato que reafirma a qualidade da instituição de ensino tanto em questões de estrutura quanto de profissionais.

No capítulo 2, Fundamentação teórica, serão abordados todos os conteúdos teóricos necessários para a compreensão dos capítulos seguintes, como processos e metodologias de desenvolvimento de software, UML, banco de dados e testes de software. Todas as abordagens são de grande utilidade para o entendimento principalmente o capítulo 3: O sistema.

Avançando ao capítulo 3, O sistema, o objetivo é descrever o sistema como um todo, desde a análise de requisitos, passando pelos diagramas UML e de banco de dados até casos de teste. Este capitulo é o principal, pois abrange todas as informações a respeito do sistema proposto.

O capítulo 4 trata dos resultados obtidos com o projeto, desde as propostas feitas no início do projeto até a eficiência nos modelos de processo de desenvolvimento, diagramas e modelagens.

Por fim, nas Considerações finais, é feito o fechamento de todo o raciocínio desenvolvido ao longo do projeto, as suposições serão “amarradas” e concluídas a fim de esclarecer se a proposta de solução foi suficiente para sanar os problemas do cliente.

O resultado esperado para este projeto é demonstrar todo o processo de engenharia de software envolvido no processo de desenvolvimento de uma ferramenta de gestão de negócios com ênfase em controle de ordens de serviços e manutenção de clientes voltado para empresas de mecânica automotiva.

Concomitantemente o sistema para gestão de oficina de mecânica automotiva deverá prover todas a ferramentas necessárias para uma gestão de qualidade e incluirá as funcionalidades de cadastro de veículos, clientes, funcionários, ordens de serviço, funcionalidades específicas como por exemplo a função “transferir posse do veículo” e “histórico completo do veículo.

2. FUNDAMENTAÇÃO TEÓRICA

Segundo Pressman (2007, p. 13), “software é um conjunto de instruções para computadores que quando executadas tem a função de manipular dados adequadamente para se obter informação”. Em outras palavras, software pode ser considerado um conjunto de algoritmos que estabelecem um conjunto de instruções a serem executadas, estas na quais tem o objetivo de realizar algum procedimento.

Sommerville (2007, p. 3) afirma que os softwares estão em todos os lugares, praticamente todo equipamento eletrônico possui mesmo que pequeno um software o operando. O software tem ajudado em operações de produção, escolas, pesquisas entre muitos outros.

Dada a importância devido a tamanha participação dos sistemas informatizados no cotidiano, é preciso que haja técnicas para que desenvolvam sistemas o mais próximo possível do que o cliente necessita. É neste campo que a Engenharia de software se faz necessária, pois ela fornece todas as métricas para especificação, desenvolvimento, gerenciamento e evolução do sistema.

Como afirma Magela (2006, p. 24), “nossas técnicas de levantamento e modelagem não são formais o suficiente para garantir a coerência do produto final tendo em vista o atendimento das reais necessidades dos usuários”. A real importância da Engenharia de Software está diretamente relacionada a saber a real necessidade do usuário para que então isto possa ser atendido. Atender de forma correta a demanda é uma questão essencial para a qualidade de software pois de nada adianta atender vários processos de desenvolvimento que tornam a produção mais eficiente reduzindo tempo e gastos se não se entrega o produto correto ao cliente.

Após ter em mente muito bem definido o que o cliente deseja entram os processos de software que são tão importantes quanto, pois definirão um conjunto de atividades e resultados que quando associados geram um produto de software (SOMMERVILLE, 2007, p. 6). Dependendo do modelo que se adota a questão de saber ou não ao exato o que o cliente quer pode se tornar um grande problema ou um pequeno problema que pode ser sanado sem muitos prejuízos.

2.1. Modelos de processos de software

Há vários modelos de processos de software, como já dito, este será um conjunto de atividades que serão realizadas, Sommerville (2007, p. 6) cita quatro processos fundamentais, estes nos quais são comuns a todos os modelos e estão representados na Tabela1:

Tabela 1 – Atividades de processo de software fundamentais

Atividade	Descrição
Especificação do software	A funcionalidade do software e as restrições em sua operação devem ser definidas.
Desenvolvimento do software	O software deve ser produzido de modo que atenda às suas especificações.
Validação do software	O software deve ser validado para garantir que ele faz o que o cliente deseja.
Evolução do software	O software deve evoluir para atender às necessidades mutáveis do cliente.

Fonte: Adaptado de Sommerville, 2007, p. 6

Dentre todas estas atividades, o que varia entre um modelo e outro são os níveis de detalhamento das atividades e os prazos estipulados para cada uma delas. Analisando de maneira geral, não há modelos ruins, apenas situações diferentes onde cada uma delas se aplica melhor, então nesse caso o ideal é que se conheça vários para então decidir qual o mais indicado para o tipo de software que se deseja desenvolver.

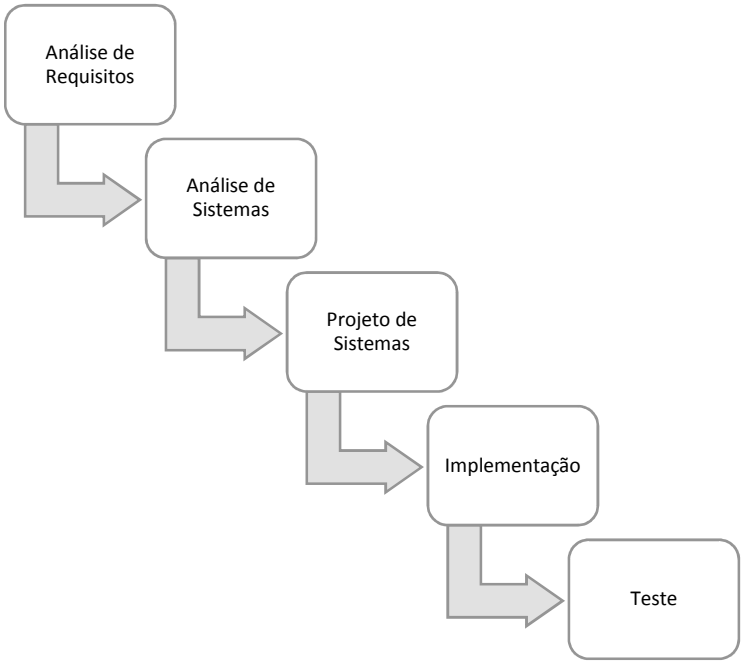
2.1.1. Modelo Cascata (Waterfall)

Sommerville (2007, p. 7) ressalta que no modelo Cascata, são consideradas todas as atividades fundamentais já citadas na Tabela 1, estas são representadas como fases separadas serão submetidas a uma aprovação para que então o desenvolvimento prossiga.

Waterfall pode ser considerado um modelo de certa forma inflexível pois como representado na Figura 1, é sequencial, e não há tanto espaço para retrabalhos, logo, num projeto onde não se tem completa compreensão dos requisitos do cliente.

Este modelo pode se tornar um grande problema, já que os acordos sobre requisitos são feitos no estágio inicial do processo, não havendo assim grande possibilidade de atender requisitos extra vindos do cliente.

Figura 1 - Modelo Waterfall



Fonte: Adaptado de Magela, 2006, p. 29

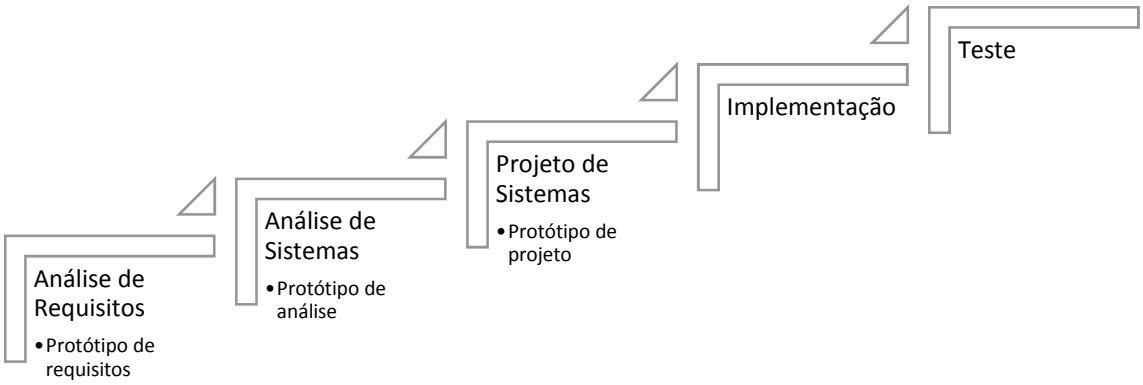
2.1.2. Modelo baseado em protótipo

Magela (2006, p. 29) ressalta que “uma vantagem do modelo baseado em protótipo em relação ao modelo cascata, pois neste conceito entende-se a necessidade de diminuir riscos e incertezas presentes na construção de um software”.

Entende-se que no modelo protótipo há uma atenção maior em relação ao que o cliente deseja, neste modelo há de certa forma uma variação de um

ponto fundamental em todos os modelos, o teste. Assim como representado na Figura 2, há uma espécie de “teste” para cada uma das etapas fundamentais, com exceção das duas últimas que são implementação e teste. Esta etapa constituída por uma homologação do que se foi produzido em determinada fase do projeto, eis então a grande diferença de Homologação e Teste Segundo Magela (2006, p. 29) “O Teste pode ser realizado e ter uma aprovação de 100% de suas funcionalidades, contudo a Homologação avalia o produto gerado no intuito de certificar que ele atende ao usuário. Portando um software não significa nada se não atende ao usuário”.

Figura 2 – Modelo baseado em protótipo



Fonte: Adaptado de Magela, 2006, p. 30

Como se pode notar o modelo baseado em protótipo realmente é muito eficiente definindo O QUE será construído, de fato ter este aspecto bem definido agrega muito valor ao projeto pois há uma diminuição significativa nos riscos durante o desenvolvimento. Magela (2006, p. 29) faz subentender que este é um bom modelo, entretanto adverte: “sua natureza pode levar a definição correta de O QUE se quer, mas COMO construir pode ser altamente penalizado, sem o apoio das devidas técnicas de construção de software”.

2.1.3. Modelo Espiral

O modelo de processo de desenvolvimento em espiral ilustrado na Figura 3, originalmente foi proposto por Boehm em 1988. Neste modelo o processo de sequência de atividades é representado de forma espiral de modo que permita o retorno entre as atividades (MAGELA 2006, p. 45).

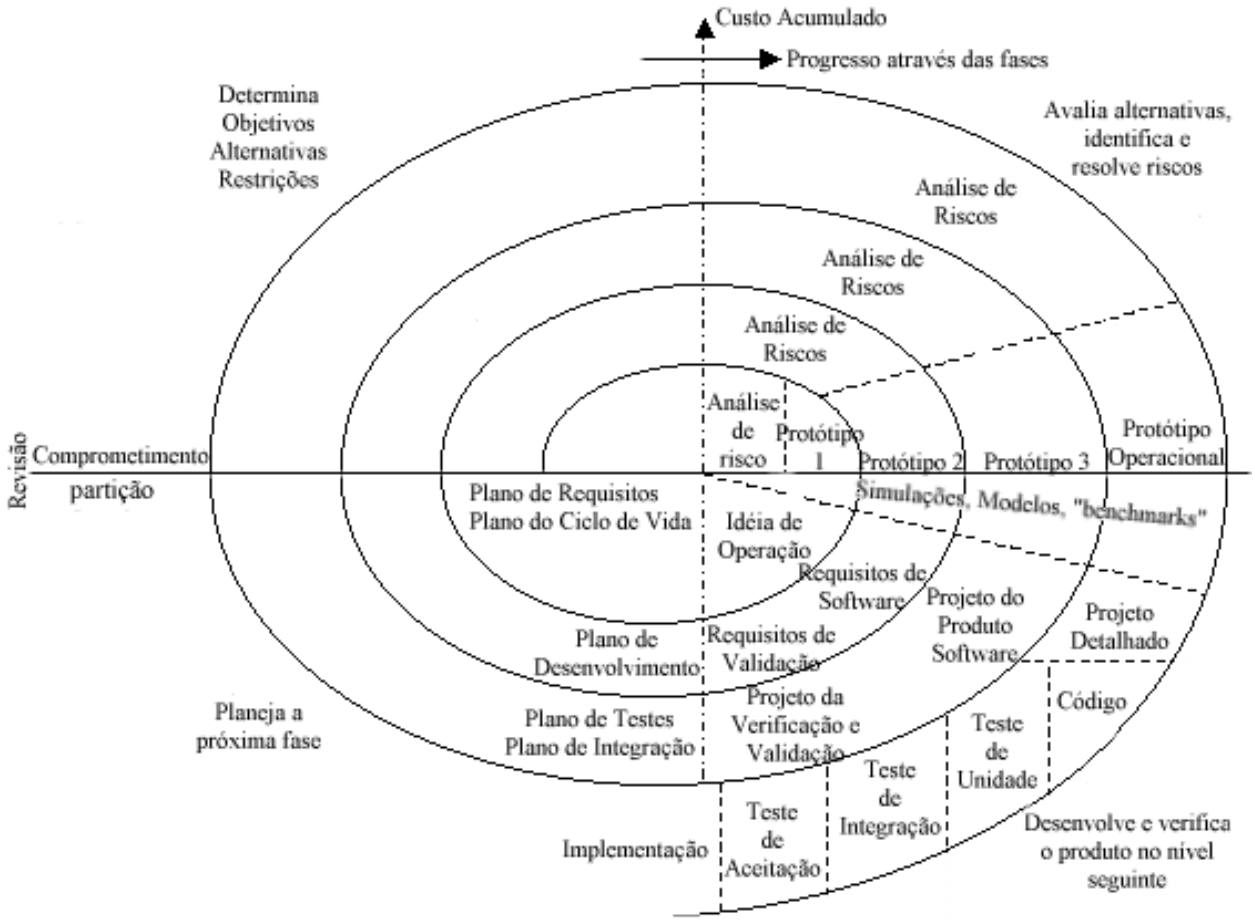


Figura 3 – Modelo Espiral

Fonte: Sommerville, 2007, p. 49

“A importante distinção entre o modelo espiral e outros modelos de processo de software é a explícita consideração dos riscos no modelo em espiral” (SOMMERVILLE, 2007, p. 49). Tendo em vista a afirmação do autor, entende-se que o modelo Espiral pode ser considerado bastante conservador a

partir do momento em que se assume todo e qualquer risco no projeto como sendo de suma importância.

Sommerville (2007, p. 48) afirma que cada loop na espiral representa uma fase do processo de software. Como pode ser visto na tabela 2, cada loop de espiral é dividido em quatro setores.

Tabela 2 – Principais setores do Modelo Espiral

Atividade	Descrição
Definição de objetivos	A partir dos objetivos específicos definidos nessa etapa são identificados os riscos e possíveis planos de ação para eles.
Avaliação e redução de riscos	Consiste numa análise detalhada dos riscos e das providencias que deverão ser tomadas em relação a eles.
Desenvolvimento e validação	Um modelo de desenvolvimento de sistema é escolhido.
Planejamento	Todo o projeto é revisto para que haja uma tomada de decisão de continuar no próximo loop da espiral. Então os planos serão traçados.

Fonte: Adaptado de Sommerville, 2007, p. 48

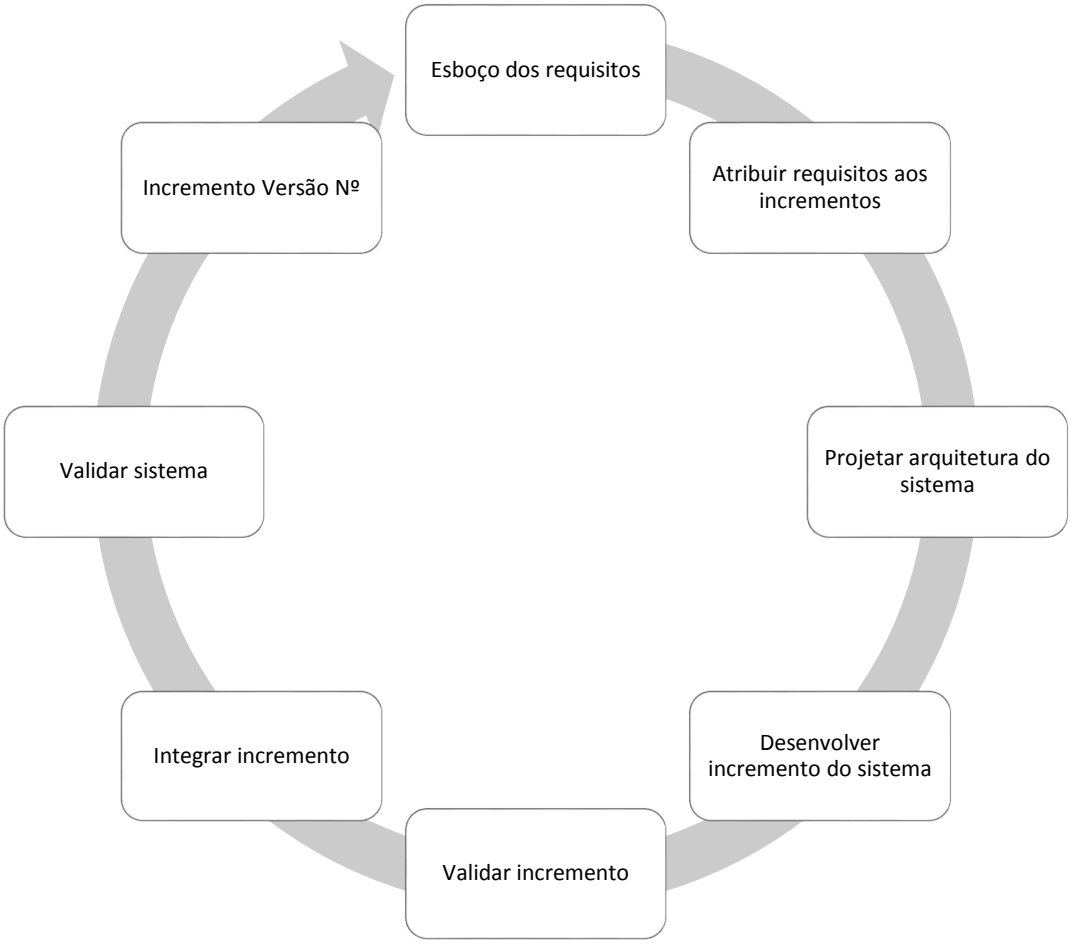
Magela (2006, p. 30) adverte que apesar de neste modelo o alinhamento do que precisa ser feito estar sempre muito bem atualizado um software mutante que a todo momento esteja sujeito a alterações pode ser uma sobrecarga para o projeto e em casos extremos pode até mesmo não ser suportado já que a constante análise de riscos exigiria estudos mais detalhados no projeto, mais protótipos seriam construídos e consequentemente mais tempo seria gasto.

2.1.4. Modelo Incremental

“A abordagem do desenvolvimento incremental foi sugerida por Mills como um meio de reduzir o retrabalho no processo de desenvolvimento” (SOMMERVILLE, 2007, p. 47).

Sommervile (2007, p. 47) afirma que no modelo de desenvolvimento incremental ilustrado na Figura 4 os clientes definem um esboço de funcionalidades e definem prioridades para os mesmos e em seguida são definidos uma serie se estágios de entrega onde cada um conterá parte das funcionalidades totais do sistema, as funcionalidades prioritárias deverão ser entregues primeiro.

Figura 4 – Modelo de desenvolvimento Incremental



Fonte: Adaptado de Sommerville, 2007, p. 47 e Magela, 2006, p. 31

Magela (2006, p. 31) afirma da maneira que este modelo de desenvolvimento propõe, os riscos vão sendo eliminados de forma gradativa e o usuário consegue visualizar valor no produto rapidamente uma vez que o mesmo não terá que aguardar um grande período de tempo para ter a primeira interação

com o sistema. Entretanto a abordagem em questão pode acarretar divergências e até mesmo “catástrofes” visto que em um software as funcionalidades podem ter ligações consideráveis entre si, desta forma não dever ser separadas.

2.2. Escolha do modelo de desenvolvimento de sistema

A escolha do modelo de desenvolvimento mais adequado ao desenvolvimento desde projeto foi baseada nos objetivos iniciais para com o cliente, onde a maior das intenções é atender as necessidades do cliente da maneira mais pontual possível, logo todo o foco “cai” em cima das funcionalidades.

Para que seja possível uma maior precisão na solução do problema do cliente, é preciso sempre atualizar constantemente os requisitos e projeto como no geral, e o modelo escolhido é o modelo que mais possibilita trabalhar desta forma, devido as entregas gradativas de valor ao cliente e também o contato constante com o mesmo.

2.2.1. SCRUM

SCRUM, segundo Schwaber e Sutherland (2013, p. 3) é “Um framework dentro do qual pessoas podem tratar e resolver problemas complexos e adaptativos, enquanto produtiva e criativamente entregam produtos com o mais alto valor possível”.

Logo entende-se que este, diferente da proposta do modelo cascata por exemplo, onde tudo é projetado e depois tudo é desenvolvido, a ideia é entregar valor constantemente e o mais rápido possível.

Schwaber e Sutherland (2013, p. 5) afirmam que “O Time SCRUM é composto pelo *Product Owner*, o Time de Desenvolvimento e o SCRUM Master” Os times SCRUM podem ser considerados auto organizáveis e multifuncionais uma vez que há divisões de papeis, mas isso não impossibilita que um membro do time realize alguma outra tarefa fora da função dele.

O responsável pelo gerenciamento das tarefas a serem feitas é o *Product Owner este também* “é o responsável por maximizar o valor do produto e do

trabalho do Time de Desenvolvimento” (SCHWABER; SUTHERLAND. 2013. p. 5).

Quem realmente “coloca a mão na massa” é o time de desenvolvimento estes são os “profissionais que realizam o trabalho de entregar uma versão usável que potencialmente incrementa o produto “Pronto” ao final de cada Sprint” (SCHWABER; SUTHERLAND. 2013. p. 6).

A pessoa responsável por manter os princípios SCRUM alinhados é o SCRUM Master, ele “faz isso para garantir que o Time SCRUM adere à teoria, práticas e regras do SCRUM. O SCRUM Master é um servo-líder para o Time SCRUM” (SCHWABER; SUTHERLAND. 2013. p. 7).

Com relação aos eventos em SCRUM, todos são criados e planejados com data, horário e prazo de duração, tudo isso no intuito de gerar uma rotina para os integrantes da equipe.

A *Sprint* é o principal evento do SCRUM, este engloba uma série de outros eventos. De acordo com Schwaber e Sutherland (2013, p. 8) “as *Sprints* são compostas por uma reunião de planejamento da Sprint, reuniões diárias, o trabalho de desenvolvimento, uma revisão da Sprint e a retrospectiva da *Sprint*”.

2.2.2. Modelo Iterativo

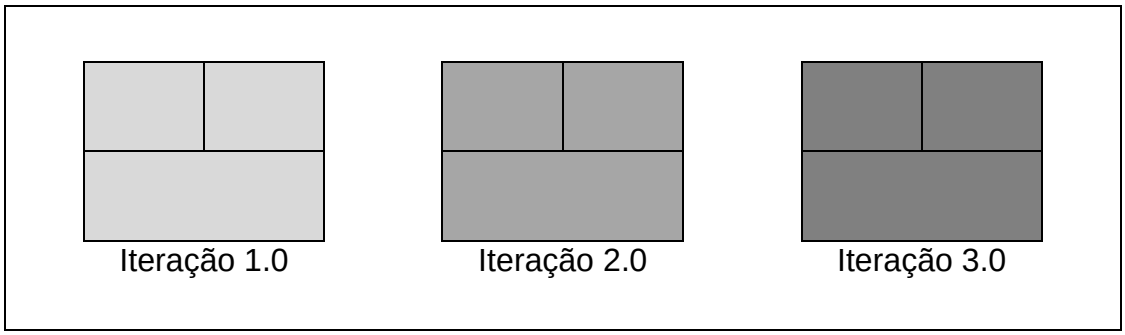
Segundo Bezerra (2002, p. 33), o modelo iterativo surge como uma ideia para solucionar os problemas do modelo cascata. Numa ótima analogia feita por Magela (2006, p. 31), o autor compara os modelos de processo de desenvolvimento de software com a pintura de uma casa e com uma instalação elétrica de uma casa também, onde o modelo cascata iniciaria a pintura apenas quando toda a casa estivesse terminada, o modelo incremental iniciaria a pintura quando a primeira parede já estivesse pronta, esta parede seria pintada completamente e com a tinta pretendida. Por fim o modelo iterativo iniciaria a pintura quando algumas paredes já estivessem prontas, entretanto não aplicaria a tinta pretendida de início, por enquanto aplicaria cal, pois tem noção de que a construção das outras paredes pode sujar a parede pintada. Na instalação elétrica de uma casa o modelo cascata colocaria a fiação toda de uma vez, o modelo incremental colocaria a fiação na medida que os cômodos fossem ficando parcialmente prontos e por fim o modelo iterativo assim como o modelo

incremental também colocaria a fiação na medida que os cômodos fossem ficando parcialmente prontos, entretanto deixaria alguns fios soltos para sanar possíveis eventualidades.

Com base na analogia entende-se que a ideia do modelo iterativo é melhorar pouco a pouco o sistema através iterações (repetições) como representado na Figura 5. Apesar de o desenvolvimento do sistema ser é divido em pequenas partes o mesmo é sempre olhado como um todo, nunca se finaliza por completo uma funcionalidade sem deixar de certa forma uma “válvula de escape” para possíveis alterações do projeto ou mudanças provocadas por outras funcionalidades desenvolvidas no futuro. “O modelo iterativo liberaria todas as funcionalidades de uma vez, entretanto deixando a desejar em todas elas, aumentando seu suporte a cada versão” (MAGELA, 2006, p. 32).

Magela (2006, p. 32) adverte que todo o levantamento de requisitos e arquitetura são um grande desafio e definitivamente definirão o sucesso ou fracasso do projeto de software. Entretanto recomenta fortemente este modelo: “Em resumo, sempre que possível, a abordagem iterativa deveria ser utilizada” (MAGELA, 2006, p. 32).

Figura 5 – Modelo Iterativo



Fonte: Adaptado de Magela, 2006, p. 2006

2.3. Conceitos de *Unified Modeling Language* (UML)

De acordo com Guedes (2007, p. 13) “UML é uma linguagem visual utilizada para modelar sistemas computacionais por meio do paradigma de

Orientação a Objetos". Esta linguagem possui nove tipos de diagramas que são usados para documentar e modelar diversos aspectos dos sistemas.

2.3.1. Diagrama de caso de uso

O diagrama de caso de uso, de acordo com Guedes (2007, p. 15), "é utilizado principalmente para auxiliar no levantamento e análise dos requisitos [...]". Em outras palavras, ele descreve as principais funcionalidades do sistema e a interação dessas funcionalidades com os usuários do mesmo sistema.

Neste diagrama os usuários são chamados de "atores", aqueles que interagem diretamente com o sistema.

Os círculos ovais são os casos de uso, ou seja, tarefas ou funcionalidades realizadas pelo ator. A comunicação é representada pelos traços simples que interligam os atores e os casos de uso.

As setas pontilhadas denominadas "include" e "*extend*" representam as ligações entre casos de uso, uma sequência de eventos que em alguns casos obrigatoriamente acontecem em outros casos opcionalmente acontecem quando um usuário interage com o sistema.

2.3.2. Diagrama de classes

Guedes (2007, p. 17), afirma que este é o diagrama mais importante da UML e complementa: "Como o próprio nome diz, esse diagrama define estrutura das classes utilizadas pelo sistema, determinando os atributos e métodos possuídos por cada classe [...]".

2.3.3. Diagrama de sequência

De acordo com Guedes (2007, p. 20), "o diagrama de sequência preocupa-se com a ordem temporal em que as mensagens são trocadas entre os objetos envolvidos em um determinado processo". De certa forma este

digrama coletaria informações dos diagramas de caso de uso e de classes para então definir uma sequência para os seus objetos.

2.3.4. Diagrama de atividade

Segundo Guedes (2007, 23), “o diagrama de atividade se preocupa em descrever passos a serem percorridos para a conclusão de uma atividade específica”. Então de certa forma esse diagrama, assim como o diagrama de sequência descreve o fluxo do programa, mas de uma forma menos detalhada.

2.4. Conceitos de Banco de dados

Segundo Silberschatz (2006, p. 1) “um sistema de gerenciamento de banco de dados (SGBD) é uma coleção de dados inter-relacionados e um conjunto de programas para acessar esses dados”. Se atentando a definição do autor é possível entender que não só no presente software proposto, mas também, de certa forma em grande parte dos softwares comerciais atuais, o real “coração” do sistema é o banco de dados. Tudo gira em torno deles, tanto que projetamos e construímos ferramentas para melhor manipula-los e armazena-los.

Durante a análise para levantamento, identifica-se as principais partes e objetos envolvidos, suas possíveis ações e responsabilidades, suas características e como elas interagem entre si. A partir das informações obtidas, pode-se desenvolver um modelo conceitual que será utilizado para orientar o desenvolvimento propriamente dito, fornecendo informações sobre os aspectos relacionados ao domínio do projeto em questão.

2.4.1. Entidade

Entidade pode ser definida como “uma ‘coisa’ ou ‘objeto’ no mundo que real que é distinguível de todos os outros objetos” (SILBERSCHATZ, 2006, p. 135).

A partir da definição do autor, pode se interpretar que um objeto a ser armazenado pode ser definido como uma entidade quando este possuir informações qualitativas pertencentes somente a ele mesmo, desta forma o objeto precisa se distinguir dos outros, sendo assim será fixada a sua singularidade através de um identificador.

Uma pessoa, por exemplo, é uma entidade, porque ela contém informações ímpares em relação as outras entidades. Seguindo o mesmo exemplo, de acordo com Silbeschatz (2006, p. 135), um conjunto de entidades seria formado por um conjunto de pessoas, pois estes são do mesmo tipo e podem compartilhar das mesmas propriedades ou atributos. Num conjunto de entidades onde todos componentes são muito parecidos haverá pelo menos um atributo identificador para cada um destes, este será responsável por individualizar uma entidade da outra dentro de um conjunto de entidades.

2.4.2. Relacionamento

Em banco de dados um relacionamento é “uma associação entre várias entidades“. O relacionamento entre tabelas é um conceito fundamental para o Modelo Relacional de Dados. Num banco de dados, embora as informações estejam separadas em cada uma das tabelas, na prática devem existir relacionamentos entre elas. O controle dos relacionamentos será feito através das cardinalidades.

2.4.1. Modelo entidade/relacionamento (MER)

De acordo com Silberschatz (2006, p. 5), O modelo entidade/relacionamento foi desenvolvido para facilitar o projeto de banco de

dados, permitindo especificação de um esquema de empresa que representa de forma abstrata a estrutura que possuirá o banco de dados da aplicação.

2.4.2. Diagrama entidade/relacionamento

A função do diagrama de entidade relacionamento seria “expressar graficamente toda a estrutura lógica geral de um banco de dados”. A partir de um diagrama bem detalhado pode-se desenvolver de fato a estrutura de banco de dados, encarando o mesmo como um “escopo” de banco de dados.

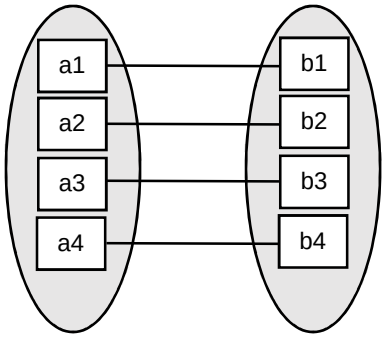
2.4.2.1. Cardinalidade

Em conformidade com Silberschatz (2006, p. 139), “as cardinalidades de mapeamento, ou fatores de cardinalidade expressam o número ao qual outra entidade pode ser associada por um conjunto de relacionamento”.

Silberschatz (2006, p. 140), descreve os tipos de cardinalidade como sendo os seguintes:

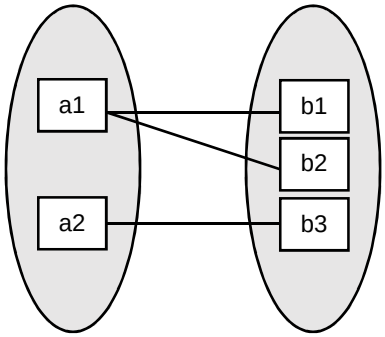
- **Um-para-um:** Uma entidade em A é associada a no máximo uma entidade em B, e uma entidade em B é associada a no máximo uma entidade em A (Figura 6).
- **Um-para-muitos:** Uma entidade em A qualquer número de entidades (zero ou mais) em B., entretanto, uma entidade em B pode ser associada a no máximo uma entidade em A (Figura 7).
- **Muitos-para-um:** Uma entidade em A é associada a no máximo em entidade em B., entretanto, uma entidade em B pode ser associada a qualquer número de entidades (zero ou mais) em A (Figura 8).
- **Muitos-para-muitos:** Uma entidade em A é associada a qualquer número de entidades (zero ou mais) em B e uma entidade em B pode ser associada a qualquer número de entidades (zero ou mais) em A (Figura 9).

Figura 6 – Cardinalidade Um-para-um



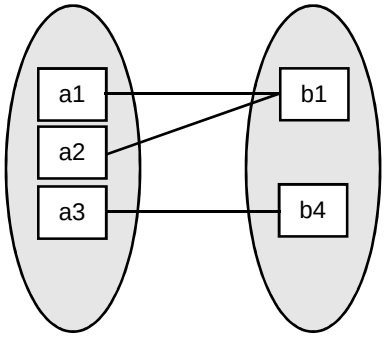
Fonte: Silberschatz (2006, p. 140)

Figura 7 – Cardinalidade Um-para-muitos



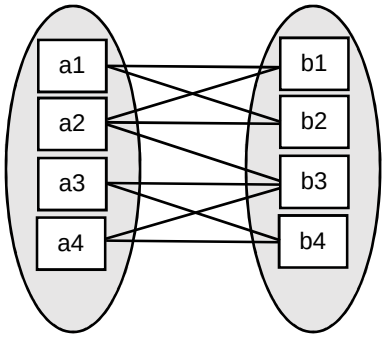
Fonte: Adaptado de Silberschatz (2006, p. 140)

Figura 8 – Cardinalidade Muitos-para-um



Fonte: Adaptado de Silberschatz (2006, p. 140)

Figura 9 – Cardinalidade Um-para-muitos



Fonte: Silberschatz (2006, p. 140)

2.5. Conceitos de Testes de software

Segundo Rios (2006, p. 1), “Em um mundo globalizado, onde a Internet é um importante suporte aos negócios, um teste mal feito pode significar um caminho aberto para diversos problemas gravas como por exemplo: fraudes, incorreções e bloqueios do ‘site’”. Num projeto de software requisitos são pedidos pelo cliente e os mesmos são prometidos a eles. Entregar os requisitos são significa simplesmente entregar algo que está aparentemente funcionando, mas funcionando por completo nos mínimos detalhes, pois como o próprio autor citado anteriormente alerta que as consequências de um software mal testado podem ser grandes.

Para Rios (2006, p. 8) “em essência, teste de software é processo que visa a execução de forma controlada, com o objetivo de avaliar o seu comportamento baseado no que foi especificado. A execução dos testes é considerada um tipo de validação”, ou seja, testar o software não necessariamente significa deixa-lo completamente livre de bugs, mas sim faze-lo atender a necessidade do cliente, e obviamente que isto implica nos processos críticos ou principais do sistema onde o nível de bugs deve ser reduzido ao máximo.

2.5.1. Estágios de teste

“Cabe esclarecer que, muitas vezes, os tipos de testes se sobrepõem, sendo as próprias definições abrangentes ou especificas, conforme a sua execução” (RIOS, 2006, p. 13).

2.5.1.1. Teste unitário

De acordo com Rios (2006, p. 13) “os testes unitários são aplicados verificam o funcionamento de um pedaço do sistema ou software isoladamente ou que possam ser testados separadamente [...]. Na grande maioria dos casos estes testes são realizados pelos próprios desenvolvedores”.

O teste unitário pode ser considerado como o primeiro e mais simples instrumento de teste, a primeira avaliação, por isso pode ser feito pelo próprio desenvolvedor.

2.5.1.2. Teste Integração

Para Rios (2006, p. 13) os testes de integração “são executados em uma combinação de componentes para verificar se eles funcionam corretamente juntos, ou seja, assegurar que as interfaces funcionem corretamente e que os dados são processados de forma correta, conforme as especificações”.

Rios (2006, p. 14) ressalta que os testes são feitos prioritariamente pela pelos desenvolvedores ou por uma equipe de testes de forma incremental onde os módulos vão sendo inclusos gradativamente.

As seguintes estratégias, de acordo com Rios (2006, p.13) são utilizadas neste nível de teste:

> ***Botton-up*:** os testes começam agrupando os componentes, programas módulos ou subsistemas do mais baixo nível para formar novos módulos ou subsistemas em níveis superiores.

> ***Top-down*:** o inverso da estratégia anterior. Uma vantagem desta estratégia é que alguns resultados podem ser apresentados para os usuários antes de completada a construção de componentes, programas módulos, ou subsistemas de níveis mais baixos. É claro que artifícios devem ser criados para simular a ausência destes níveis.
>
> **Dados de Fluxo:** a integração dos componentes, programas, módulos ou subsistemas, é feita pelo desenho do fluxo de dados
>
> **Funcional:** a integração é feita baseada na junção de componentes, programas, módulos ou subsistemas que produzam um resultado funcional significativo para os usuários.
>
> **"*Big-bang*":** combinam todos os componentes, programas, módulos ou subsistemas de uma só vez.
>
> **Testes de sistema:** são realizados pela equipe de testes, visando a execução do sistema como um todo ou um subsistema (parte do sistema), dentro de um ambiente operacional controlado, para validar a exatidão e perfeição na execução de suas funções.

2.5.2. Testes por estágio

Para cada um nos níveis de teste apresentados pode ser feito um processo de testes de caixa preta e caixa branca.

2.5.2.1. Teste Caixa Preta (Black Box)

Segundo Rios (2006, p. 13) os testes de caixa preta "visam verificar a funcionalidade e aderência dos requisitos em uma ótica externa ou do usuário, sem se basear em qualquer conhecimento do código e da lógica interna do componente testado".

Entende-se que o teste de caixa preta verifica a saída dos dados usando entradas de vários tipos, a variedade nos dados de entrada é que define a ótica externa ou do usuário citada pelo autor, desta forma há uma espécie de simulação de como o sistema se portaria diante do uso cotidiano de um usuário.

Teoricamente um sistema com todas as validações para evitar a entrada de dados errados passaria no teste de caixa preta.

2.5.2.2. Teste Caixa Branca (White Box)

De acordo com Rios (2006, p. 13) os testes de caixa branca “visam avaliar as cláusulas de código, a lógica interna do componente codificado, as configurações e outros elementos técnicos”.

Interpreta-se que que os testes de caixa branca exigem um bom conhecimento técnico, pois de certa forma o objetivo é otimizar o código para que processe os dados de forma mais eficiente, por isso neste caso a perspectiva interna significa basicamente o código fonte.

3. O SISTEMA

Por se tratar de atividades que influenciam diretamente o financeiro e a principal ferramenta de geração de recursos da empresa, é muito importante que haja um Sistema integrado de gestão empresarial, onde todas as atividades possam ser geridas da maneira mais organizada possível. Com isso, o negócio tende a ser muito mais organizado e consequentemente mais rentável. A empresa necessita de um ERP completo, porém para este caso, será tratado apenas a solução para o gerenciamento de orçamentos e ordens de serviços, estes nos quais estarão englobados pelo módulo de vendas.

O módulo de vendas é destinado ao controle dos processos de venda, desde o orçamento, a aprovação do mesmo e então a ordem de serviço que irá conter serviços e/ou produtos.

Com relação ao orçamento, tendo em vista as regras de negócio da oficina Mecânica Moderna, objeto de estudo deste, a melhor definição a ser feita é que orçamento é uma ferramenta financeira que tem como objetivo listar o que será adquirido, seja produto ou serviço, podendo assim ter uma previsão das despesas futuras se caso a compra for efetuada.

A Ordem de serviço terá o objetivo de formalizar o serviço prestado a um cliente. Também indicará os materiais e a mão de obra que serão utilizados para a execução de um determinado serviço prestado. Após o orçamento, o cliente tem a opção de solicitar a realização do serviço, então será gerada uma ordem de serviço a partir do orçamento. “

3.1. Requisitos

Em conversa com o cliente foram levantados os principais requisitos do sistema, que servirão como base dos demais estágios da criação do sistema. Como já foi dito, atualmente o cliente utiliza dois softwares de prateleira (SIC e Millenium) e por estes sistemas ainda atenderem parte das necessidades do cliente, alguns dos itens foram construídos com base nestes sistemas. **RF01 -** Para que seja feito qualquer tipo de serviço em algum veículo, o mesmo deve ser cadastrado previamente e associado a um cliente.

3.1.1. Requisitos funcionais

- **RF01 -** Cadastro de clientes deve possuir as mesmas informações do software que ele utiliza atualmente.
- **RF02 -** Cadastro de técnicos, os técnicos são os funcionários da empresa, o responsável pelo serviço deve ser referenciado na ordem de serviço correspondente.
- **RF03 -** Ordem de serviço, é o cadastro das atividades da empresa, se utilizará das informações de todos os outros cadastros como cliente, veiculo, orçamento, funcionários e produto. Porém, esta operação deve pertencer ao veículo e não ao cliente, desta forma, haverá cada veículo possuirá um histórico de serviços realizados mesmo que este seja posse de outro cliente.
- **RF04 -** Consulta de clientes, deve oferecer ao menos os filtros placa, CPF e nome.
- **RF05 -** Função de transferir posse do veículo, um veículo já cadastrado deve possuir a opção de trocar de dono mantendo o histórico de ordens de serviço.
- **RF06 -** AS ordens de serviço devem se tornar contas a receber. Também deve ser possível escolher a forma de pagamento “Fiado”, com dia agendado para pagamento.

3.1.2. Requisitos não funcionais

- **RNF01 –** O sistema deverá atender alguns requisitos de usabilidade como designe de tela simplificado
- **RNF02 –** O sistema deverá ser compatível com os sistemas operacionais Windows 7, 8, 8.1 e 10.
- **RNF03 –** O sistema deve atender padrões de segurança como login padronizado e senha com número mínimo de caracteres.
- **RNF04 –** Devem haver padrões de interface que abriguem lógicas de tela, validação de campos e acionamento de comandos.

- **RNF05 –** O sistema deve trabalhar de acordo com as regras de negócio da empresa.

3.2. Escopo

Para que fosse possível incluir os processos necessários para garantir que o projeto inclui todo e somente o trabalho necessário para terminar o mesmo com sucesso e ao mesmo tempo garantindo que o foco esteja principalmente em definir e controlar o que está ou não incluso no projeto, o seguinte escopo foi construído.

3.2.1. Cadastros

Os cadastros terão objetivo de registrar todos os dados no banco de dados para que então possam ser usados dos nas funcionalidades de serviço e consequentemente a geração de relatórios.

3.2.1.1. Cadastro de Pessoa

Tela de cadastro de pessoas contendo os seguintes campos: Código, Nome, Tipo de Pessoa, RG, CPF, CNPJ, Telefone residencial, Telefone comercial, Celular, E-mail, Data de nascimento, CEP, Endereço, Número, Complemento, Bairro, Cidade e Estado.

3.2.1.2. Cadastro de Veículo

Tela de cadastro de veículos contendo os seguintes campos: Código, Marca, Modelo, Placa, RENAVAM.

3.2.1.3. Cadastro de Produto/Serviço

Tela de cadastro de produtos contendo os seguintes campos: Código, Descrição e Valor Unitário.

3.2.2. Serviços

Os serviços terão objetivo de realizar os processos de prestação de serviços da oficina mecânica, calculando valores e os registrando.

3.2.2.1. Geração de Orçamento

Tela de cadastro de orçamento, contendo os seguintes campos: Código, Cliente, Veículo, Funcionário, Data de criação, Informações do produto e Quantidade.

3.2.2.2. Cadastro e geração de Ordem de Serviço

Tela de registro de ordens de serviço contendo os seguintes campos: Código, Cliente, Veiculo, Serviço, Funcionário.

3.2.2.3. Conta

Tela de registro de contas contendo os seguintes campos: Código, Data de entrada, Forma de Pagamento, Número de Parcelas, Valor da conta, Data de Vencimento, Status da Conta e Observação.

3.2.3. Consultas

As telas de consulta terão objetivo de retornar os dados de acordo com os parâmetros específicos de pesquisa inseridos pelo usuário para que o mesmo possa obter informação que possibilite tomada de decisão.

3.2.3.1. Consulta de Pessoa

Consulta de pessoa com opções para os parâmetros: Nome, RG e CPF.

3.2.3.2. Consulta de Veículo

Consulta de veículo com opções para os parâmetros: placa do carro e RENAVAM.

3.2.3.3. Consulta de Ordem de serviço

Consulta de ordem de serviço com opções para os parâmetros: nome do cliente, placa do veículo, data de registro.

3.2.4. Relatórios

Os relatórios serão responsáveis pela organização dos dados pertinentes para necessidade especifica de geração de informação a respeito da prestação de serviços acordadas entre oficina e cliente.

3.2.4.1. Ordem de Serviço

Relatório de ordem de serviço por cliente, constando o nome do cliente, data do serviço, funcionário, veículo e valor.

3.3. Diagramas UML

Para este projeto, dos nove diagramas propostos pelo UML são apresentados 3: diagrama de caso de uso, classes e sequencia.

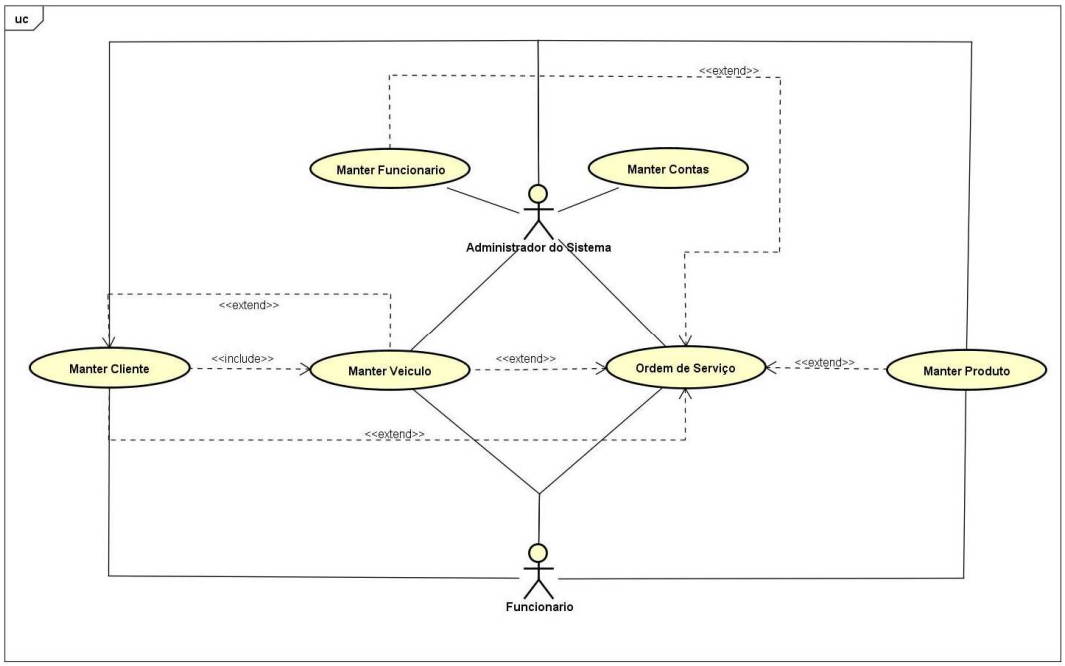
3.3.1. Diagrama de caso de uso

No diagrama de caso de uso, todos os casos de uso iniciados com “Manter” como manter: funcionário, contas, cliente, veículo e produto, significam que o ator poderá realizar as três operações básicas de inclusão, alteração e exclusão. Quanto às ordens de serviço, o diagrama representa ligações com ambos os atores, isto quer dizem que os dois podem gerar ordens de serviço.

As funcionalidades “Manter Cliente” e “Manter Veículo” se associam, pois, um cliente sempre deve estar relacionado a um veículo. Não deve haver cliente sem veículo ou veículo sem dono.

Ao gerar uma ordem de serviço o usuário há a opção de inserir um novo cliente, veículo ou produto se caso estes não estiverem cadastrados. O diagrama descrito pode ser melhor ilustrado na Figura 10.

Figura 10 – Diagrama de Caso de Uso



Fonte: Autoria Própria

3.3.1.1. Descrição dos Casos de Uso

A descrição dos casos de uso nas tabelas 3 a 8 tem como objetivo informar os atores que interagem com o sistema e as etapas que devem ser executadas pelo ator e pelo sistema para que o caso de uso execute a sua função.

Os casos de uso “Manter Cliente” e “Manter Funcionário” são uma subdivisão de um caso de uso maior que poderia se chamar “Manter Pessoa”, mas devido a questões de acesso as funcionalidades que são diferentes entre os “tipos” de pessoa, decidiu-se que será melhor apresenta-las em duas subdivisões, que são aquelas com que o sistema trabalha.

O caso de uso “Manter Funcionário”, ilustrado na tabela 3, faz o gerenciamento de todos os funcionários da empresa. Registra as informações da pessoa para que a mesma possa ser associada venda de produtos e serviços e também a realização de serviços.

Tabela 3 – Descrição do caso de uso “Manter Funcionário”

NOME DO CDU	MANTER FUNCIONÁRIO
CDU GERAL	-
ATOR	Administrador do Sistema
RESUMO	Este Caso de Uso descreve as possíveis atividades de manutenção do cadastro de funcionários, ou seja, permite incluir, alterar ou consultar funcionários.
PRÉ-CONDIÇÕES	-
PÓS CONDIÇÕES	-
FLUXO	Passo 1: O usuário seleciona um tipo de pessoa, insere o nome, RG, CPF, Data de Nascimento, cargo, telefone comercial, telefone residencial, telefone celular, e-mail, endereço, CEP, número, complemento, bairro, cidade, estado. Passo 2: O sistema verifica a existência de um funcionário cadastrado com o CPF informado. Passo 3: O sistema apresenta os dados do funcionário, caso encontre. Então o usuário pode alterar os dados do funcionário. Passo 4: Caso o sistema não encontre um funcionário cadastrado com o CPF informado então será inserido um novo.
RESTRIÇÕES/VALIDAÇÕES	Restrição 1: O campo CPF deve ser validado. Restrição 2: Os campos tipo de pessoa, nome, endereço, número, bairro, cidade, estado e CPF são obrigatórios.

Fonte: Autoria Própria

O caso de uso Manter Contas, descrito na tabela 4 gerencia todas as contas da empresa, essas contas podem ser contas a pagar e podem ser contas a receber.

As contas são resultantes das vendas realizadas, todas elas serão registradas para que possa ser feito um controle das vendas a prazo e para controlar o fluxo de caixa.

Para cada conta haverá um campo identificador para indicar seu status que pode ser “Fechado” para contas já pagas e “Aberto” para contas que ainda não foram pagas por completo.

Tabela 4 – Descrição do caso de uso “Manter Contas”

NOME DO CDU	MANTER CONTAS
CDU GERAL	-
ATOR	Administrador do Sistema
RESUMO	Este Caso de Uso descreve as possíveis atividades de manutenção do cadastro das contas a pagar e contas a receber ou seja, permite incluir, alterar ou consultar contas.
PRÉ-CONDIÇÕES	-
PÓS CONDIÇÕES	-
FLUXO	Passo 1: O usuário insere uma descrição para a conta, forma de pagamento, número de parcelas, valor total da conta, data de vencimento, status da conta e observações Passo 2: O sistema verifica a existência de uma conta cadastrada com o número de documento informado. Passo 3: O sistema apresenta os dados da conta caso encontre. Então o usuário pode alterar ou excluir os dados da conta. Passo 4: Caso o sistema não encontre uma conta cadastrada com o número do documento informado então será inserido um novo.
RESTRIÇÕES/VALIDAÇÕES	Restrição 1: O campo data de vencimento deve ser validado. Restrição 2: Os campos tipo de pessoa, devedor, credor, forma de pagamento, número do documento, valor total da conta e data de vencimento são obrigatórios.

Fonte: Autoria Própria

O caso de uso “Manter Cliente”, ilustrado na tabela 5, gerencia todos os clientes da empresa, uma vez que um dos requisitos básicos para se realizar qualquer tipo de ordem de serviço ou orçamento, antes é preciso cadastrar um

cliente (pessoa), veículo e associar os dois, além disso, em caso de ordem de serviço também é necessário já ter cadastrado pelo menos um funcionário (pessoa), pois este é quem ficará responsável pelo serviço.

O cliente terá todas as suas informações básicas cadastradas, juntamente com todos os produtos que adquiriu e todos os serviços que contratou.

O cadastro dos clientes também supre as relações de contas a receber pois uma venda realizada sempre estará relacionada com algum cliente.

Tabela 5 – Descrição do caso de uso “Manter Cliente”

NOME DO CDU	MANTER CLIENTE
CDU GERAL	-
ATOR	Administrador do Sistema e Funcionário
RESUMO	Este Caso de Uso descreve as possíveis atividades de manutenção do cadastro de clientes, ou seja, permite incluir, alterar ou consultar clientes.
PRÉ-CONDIÇÕES	-
PÓS CONDIÇÕES	É necessário associar um veículo ao cliente
FLUXO	Passo 1: O usuário seleciona um tipo de pessoa, insere o nome, RG, CPF, Data de Nascimento, telefone comercial, telefone residencial, telefone celular, e-mail, endereço, CEP, número, complemento, bairro, cidade, estado. Passo 2: O sistema verifica a existência de um cliente cadastrado com o CPF informado. Passo 3: O sistema apresenta os dados do cliente, caso encontre. Então o usuário pode alterar ou excluir os dados do cliente. Passo 4: Caso o sistema não encontre um cliente cadastrado com o CPF informado então será inserido um novo.
RESTRIÇÕES/VALIDAÇÕES	Restrição 1: O campo CPF deve ser validado. Restrição 2: Os campos tipo de pessoa, nome, endereço, número, bairro, cidade, estado e CPF são obrigatórios. Restrição 3: Um cliente não poderá ser excluído.

Fonte: Autoria Própria

O caso de uso “Manter Veiculo”, representado na tabela 6, controla todo o cadastro de veículos, as informações do mesmo, as informações mais importantes são a placa e o dono deste veículo.

Cada veículo também possuirá o próprio registro de atividades, desta forma, se o cliente vender o veículo, os serviços realizados se mantem registrado e relacionados ao antigo dono facilitando assim a realização de serviços futuros para um possível novo proprietário do veículo.

Tabela 6 – Descrição do caso de uso “Manter Veiculo”

NOME DO CDU	MANTER VEICULO
CDU GERAL	-
ATOR	Administrador do Sistema e Funcionário
RESUMO	Este Caso de Uso descreve as possíveis atividades de manutenção do cadastro de veículos, ou seja, permite incluir, alterar, excluir ou consultar veículos.
PRÉ-CONDIÇÕES	-
PÓS CONDIÇÕES	-
FLUXO	Passo 1: O usuário insere o cliente, marca, modelo, placa e RENAVAM. Passo 2: O sistema verifica a existência de um cliente cadastrado com a placa informada. Passo 3: O sistema apresenta os dados do veículo, caso encontre. Então o usuário pode alterar os dados do veículo. Passo 4: Caso o sistema não encontre um funcionário cadastrado com o CPF informado então será inserido um novo.
RESTRIÇÕES/VALIDAÇÕES	Restrição 1: O campo Placa deve ser validado. Restrição 2: Os campos cliente, marca, modelo, placa e RENAVAM. Restrição 3: Um veículo não poderá ser excluído se estiver relacionado com outras operações.

Fonte: Autoria Própria

O caso de uso “Ordem de Serviço”, descrito na tabela 7 gerencia o que podemos chamar de função “coração” do sistema, ou seja, é a função principal, então basicamente todas as outras funcionalidades trabalham em prol das ordens de serviço, sendo assim e requisito que os outros cadastros como cliente, funcionário, produtos e fornecedores tenham sido feitos para que possam apenas serem selecionados.

Ordens de serviço resultam em contas, no caso um crédito para a empresa em relação ao cliente que contratou um serviço.

Tabela 7 – Descrição do caso de uso “Ordem de Serviço”

NOME DO CDU	MANTER ORDEM DE SERVIÇO
CDU GERAL	-
ATOR	Administrador do Sistema e Funcionário
RESUMO	Este Caso de Uso descreve as possíveis atividades de manutenção do cadastro de ordens de serviço, ou seja, permite incluir, alterar, excluir ou consultar ordens de serviço.
PRÉ-CONDIÇÕES	É necessário possuir clientes, funcionários e veículos cadastrados
PÓS CONDIÇÕES	-
FLUXO	Passo 1: O usuário seleciona um cliente, veículo e funcionário Passo 2: O usuário seleciona produtos que deseja adicionar
RESTRIÇÕES/VALIDAÇÕES	Restrição 1: Os campos quantidade, data de entrada e data de saída devem ser validados. Restrição 2: Os campos cliente veículo e funcionário são obrigatórios.

Fonte: Autoria Própria

Dado o requisito solicitado pelo cliente com base no modelo de negócio utilizado pelo mesmo atualmente, os produtos a serem utilizados nos serviços contratados pelos clientes eram adquiridos dos fornecedores segundo a demanda, sendo assim não há estoque para registrar. Logo o cadastro de produtos tem a função apenas de registrar produtos e associar valores aos mesmos a fim de facilitar o processo de abertura e fechamento das ordens de serviço. A tabela de descrição de caso se uso pode ser vista na tabela 8.

Tabela 8 – Descrição do caso de uso “Manter Produto”

NOME DO CDU	MANTER PRODUTO
CDU GERAL	-
ATOR	Administrador do Sistema e Funcionário
RESUMO	Este Caso de Uso descreve as possíveis atividades de manutenção do cadastro de produto, ou seja, permite incluir, alterar, excluir ou consultar produtos.
PRÉ-CONDIÇÕES	O produto a ser cadastrado precisa ter um fornecedor relacionado a ele
PÓS CONDIÇÕES	-
FLUXO	Passo 1: O usuário insere a descrição do produto e a valor unitário do mesmo.
RESTRIÇÕES/VALIDAÇÕES	Restrição 1: O campo Valor Unitário deve ser validado. Restrição 2: Os campos Descrição e Valor Unitário são obrigatórios.

Fonte: Autoria Própria

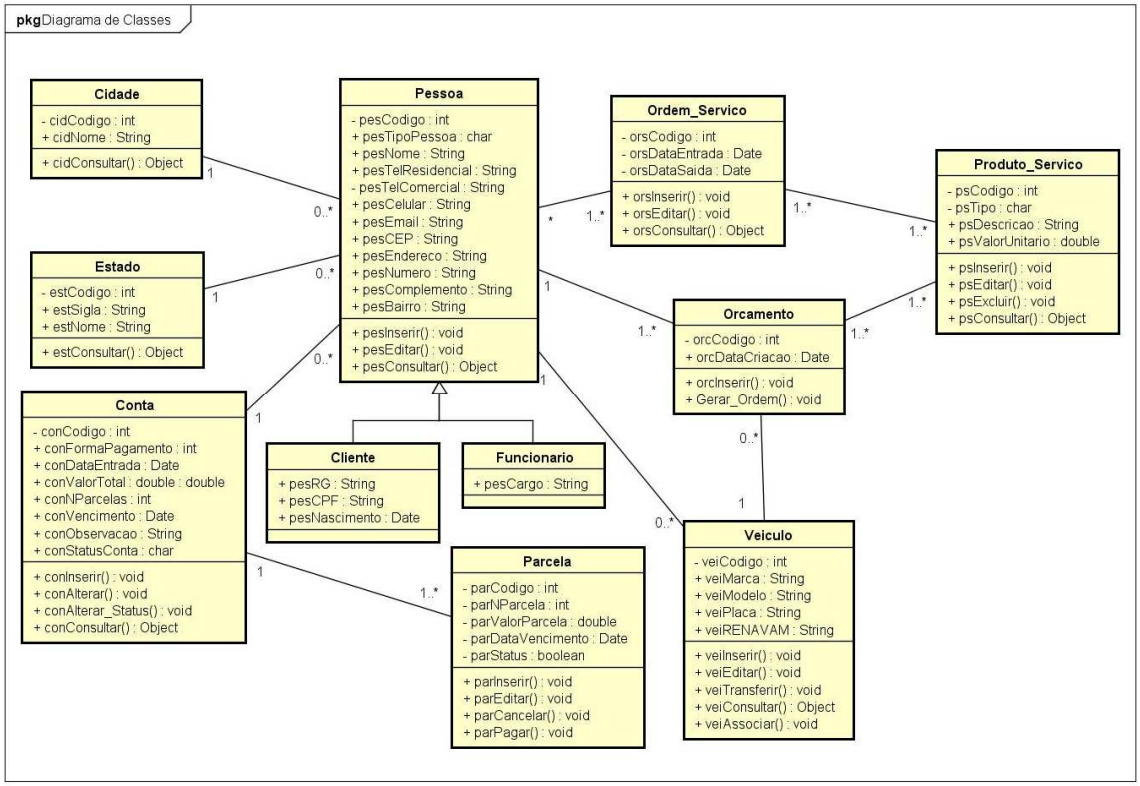
3.3.2. Diagrama de Classes

A classe “Pessoa” representa uma generalização em relação a todos os tipos de “sujeitos” inclusos no sistema. Estes podem ser clientes ou funcionários.

Um orçamento pode resultar numa ordem de serviço, esta é a primeira interação com o cliente, é o documento que dá uma noção básica dos produtos e serviços que ele necessita e então quanto tudo isso vai custar, se o cliente resolver dar seguimento e contratar os serviços da mecânica então o orçamento se tornará uma ordem de serviço, então neste caso a ordem de serviço receberá todas as informações de produtos e serviços necessários e acrescentará o funcionário responsável pelo serviço. Também há possibilidade de o cliente solicitar diretamente a ordem de serviço, neste caso o processo é o mesmo, porém sem a transição de informações de um processo para o outro.

Após a ordem de serviço, registros de contas serão feitos para registrar a partir da venda feita o que há para receber do cliente. O diagrama de classes completo pode ser visto na figura 11.

Figura 11 – Diagrama de classes



Fonte: Autoria Própria

3.3.2.1. Descrição do diagrama de classes

A descrição do diagrama de classes nas tabelas 9 a 19 tem como objetivo informar interações entre as classes e descrever a função de cada método do sistema.

A classe Pessoa, descrita na Tabela 9 centraliza todos os tipos de "indivíduos" que participam das operações do sistema armazenando todos os atributos em comum entre eles.

Tabela 9 – Descrição da classe "Pessoa"

Nome da classe			Pessoa		
Classe geral			-		
Atributos	**Tipo**	**Descrição**	**Métodos**	**Tipo**	**Descrição**
pesCodigo	int	Código identificador do registro de pessoa	pesInserir	void	Permite inserir um novo registro no cadastro de pessoa
pesTipoPessoa	char(3)	Código identificador do tipo de pessoa	pesEditar	void	Permite editar dados de pessoas cadastradas
pesTelResidencial	string	Telefone residencial da pessoa	pesConsultar	Object	Retorna todos os dados de pessoas cadastradas
pesTelComercial	string	Telefone comercial da pessoa			
pesCelular	string	Telefone celular da pessoa	-	-	-
pesEmail	string	E-mail da pessoa	-	-	-
pesCEP	char(8)	CEP da pessoa	-	-	-
pesEndereco	string	Endereço da pessoa	-	-	-
pesNumero	char(5)	Número da residência da pessoa	-	-	-
pesComplemento	string	Complemento da residência da pessoa	-	-	-
pesBairro	string	Bairro onde se localiza a residência da pessoa	-	-	-

Fonte: Autoria própria

A classe Cliente (Tabela 10) é uma extensão da classe Pessoa, ela recebe todos os atributos desta classe e adiciona os atributos “pesRG” e “pesCPF”.

Tabela 10 – Descrição da classe “Cliente”

Nome da classe	Cliente				
Classe geral	Pessoa_Fisica				
Atributos	**Tipo**	**Descrição**	**Métodos**	**Tipo**	**Descrição**
pesRG	char(9)	Nome da pessoa	-	-	-
pesCPF	char(11)	CPF da pessoa	-	-	-
pesNascimento	date	Data de nascimento da pessoa	-	-	-

Fonte: Autoria própria

A classe “Funcionario” (Tabela 11) é uma extensão da classe Pessoa, ela recebe todos os atributos desta classe e adiciona o atributo “pesCargo”.

Tabela 11 – Descrição da classe “Funcionario”

Nome da classe	Funcionario				
Classe geral	Pessoa_Fisica				
Atributos	**Tipo**	**Descrição**	**Métodos**	**Tipo**	**Descrição**
pesCargo	string	Descrição do cargo do funcionário	-	-	-

Fonte: Autoria própria

A classe “Cidade” (Tabela 12) auxilia a classe “Pessoa” fornecendo a ela todos os a cidades existentes no Brasil.

Tabela 12 – Descrição da classe “Cidade”

Nome da classe	Cidade				
Classe geral	-				
Atributos	**Tipo**	**Descrição**	**Métodos**	**Tipo**	**Descrição**
cidCodigo	int	Código identificador da cidade	cidConsultar	Object	Retorna o nome da cidade e o seu codigo identificador
cidNome	string	Nome da cidade	-	-	-

Fonte: Autoria própria

A classe “Estado” (Tabela 13) auxilia a classe “Pessoa” fornecendo a ela todos os estados existentes no Brasil.

Tabela 13 – Descrição da classe “Estado”

Nome da classe			Estado		
Classe geral			-		
Atributos	**Tipo**	**Descrição**	**Métodos**	**Tipo**	**Descrição**
estCodigo	int	Código identificador do estado	estConsultar	Object	Retorna o nome do estado e o seu código identificador
estNome	string	Nome do estado	-	-	-
estSigla	char(2)	Sigla do estado	-	-	-

Fonte: Autoria própria

A classe “Conta” (Tabela 14) recebe todos os atributos referentes as contas e controla o recebimento de todas a possíveis parcelas de cada serviço realizado.

Tabela 14 – Descrição da classe “Conta”

Nome da classe			Conta		
Classe geral			-		
Atributos	**Tipo**	**Descrição**	**Métodos**	**Tipo**	**Descrição**
conCodigo	int	Código identificador da conta	conInserir	void	Permite inserir um novo registro no cadastro de conta
conFormaPagamento	int	Forma de pagamento	conAlterar	void	Permite alterar um registro já cadastrado
conDataEntrada	date	Data de cadastro da conta	conAlterar_Status	void	Permite alternar status entre pago e devendo
conValorTotal	double	Valor total da conta	conConsultar	void	Permite consultar as contas cadastradas
conNParcelas	int	Número total de parcelas da conta	-	-	-
conVencimento	date	Data de vencimento geral da conta	-	-	-
conObservacao	string	Observações gerais da conta	-	-	-
conStatusConta	char(1)	Status da conta	-	-	-

Fonte: Autoria própria

A classe “Parcela” (Tabela 15) auxilia a classe “Conta” armazenando todo o parcelamento gerando para cada conta resultante de um serviço realizado.

Tabela 15 – Descrição da classe “Parcela”

Nome da classe		Parcela			
Classe geral		-			
Atributos	**Tipo**	**Descrição**	**Métodos**	**Tipo**	**Descrição**
parCodigo	int	Código identificador da parcela	parInserir	void	Permite inserir uma parcela
parNParcela	int	Número da parcela	parEditar	void	Permite editar uma parcela cadastrada
parValorParcela	double	Valor da parcela	parCancelar	void	Cancela a parcela
parDataVecimento	date	Data de vencimento da parcela	parPagar	Object	Alterar o status para pago
parStatus	boolean	Define se a parcela foi paga ou não	-	-	-

Fonte: Autoria própria

A classe “Veiculo” (Tabela 16) registra todas as informações relacionadas ao veículo incluindo o seu proprietário.

Tabela 16 – Descrição da classe “Veiculo”

Nome da classe		Veiculo			
Classe geral		-			
Atributos	**Tipo**	**Descrição**	**Métodos**	**Tipo**	**Descrição**
veiCodigo	int	Código identificador do veiculo	veiInserir	void	Permite inserir um novo registro no cadastro do veiculo
veiMarca	string	Marca do veiculo	veiEditar	void	Permite editar dados do veículo cadastrado
veiModelo	string	Modelo do veiculo	veiTransferir	void	Permite transferir a posse do veículo cadastrado
veiPlaca	string	Placa do veiculo	veiConsultar	Object	Retorna todos os dados de veículos cadastrados
veiRENAVAM	string	RENAVAM do veiculo	veiAssociar	void-	Permite associar o veículo a uma pessoa já cadastrada

Fonte: Autoria própria

A classe orçamento (Tabela 17) é composta pelos atributos das classes “Pessoa”, “Veiculo” e “Servico”. Seus atributos próprios são apenas o código identificador do registro a data de cadastro do orçamento.

Tabela 17 – Descrição da classe “Orcamento”

Nome da classe		Orcamento			
Classe geral			-		
Atributos	**Tipo**	**Descrição**	**Métodos**	**Tipo**	**Descrição**
orcCodigo	int	Código identificador do orçamento	orcInserir	void	Permite inserir um novo registro no cadastro do veiculo
orcData	date	Data de registro do orçamento	orcGerar Ordem	void	Permite gerar uma nova ordem de serviço a partir dos dados de orçamento
-	-	-	-	-	-
-	-	-	-	-	-

Fonte: Autoria própria

A classe “Produto_Servico” (Tabela 18) dá suporte as classes “Orcamento” e “Ordem_Servico”. Esta classe guarda os atributos básicos dos produtos e serviços oferecidos pela empresa, a função principal é registrar o os nomes dos produtos e serviços e quanto deve ser cobrado por eles.

Tabela 18 – Descrição da classe “Produto_Servico”

Nome da classe		Produto_Servico			
Classe geral			-		
Atributos	**Tipo**	**Descrição**	**Métodos**	**Tipo**	**Descrição**
psCodigo	int	Código identificador do produto ou serviço	psInserir	Void	Permite inserir um novo serviço no cadastro dos produtos ou serviços
psTipo	char(1)	Identifica produto ou serviço	psEditar	Void	Permite editar dados do produto ou serviço cadastrado
psDescricao	string	Descrição do serviço	psExcluir	Void	Permite excluir um produto ou serviço cadastrado
psValor	double	Valor a ser cobrado pelo serviço	psConsultar	Object	Retorna todos os dados de produtos ou serviços cadastrados

Fonte: Autoria própria

A classe “Ordem_Servico” (Tabela 19) é composta pelos atributos das classes “Pessoa”, “Veiculo”, “Servico” e em alguns casos também a classe “Orcamento”. Seus atributos próprios são apenas o código identificador do registro a data de cadastro da ordem de serviço.

Tabela 19 – Descrição da classe Ordem_Servico

Nome da classe		Ordem_Servico			
Classe geral			-		
Atributos	**Tipo**	**Descrição**	**Métodos**	**Tipo**	**Descrição**
orsCodigo	int	Código identificador da ordem de serviço	Inserir	void	Permite inserir uma nova ordem de serviço
orsData	date	Data de registro da ordem de serviço	Editar	void	Permite editar dados de uma ordem de serviço cadastrada
-	-	-	Consultar	Object	Retorna todos os dados de ordens de serviço cadastrados

Fonte: Autoria própria

3.3.3. Diagrama de sequencia

Os diagramas que serão apresentados se referem a dois processos básicos: o cadastro de um cliente seguido do cadastro de seu veículo e um pedido de orçamento seguido de um pedido de ordem de serviço. Estes são os processos fundamentais ao sistema, todo o fluxo básico está nele.

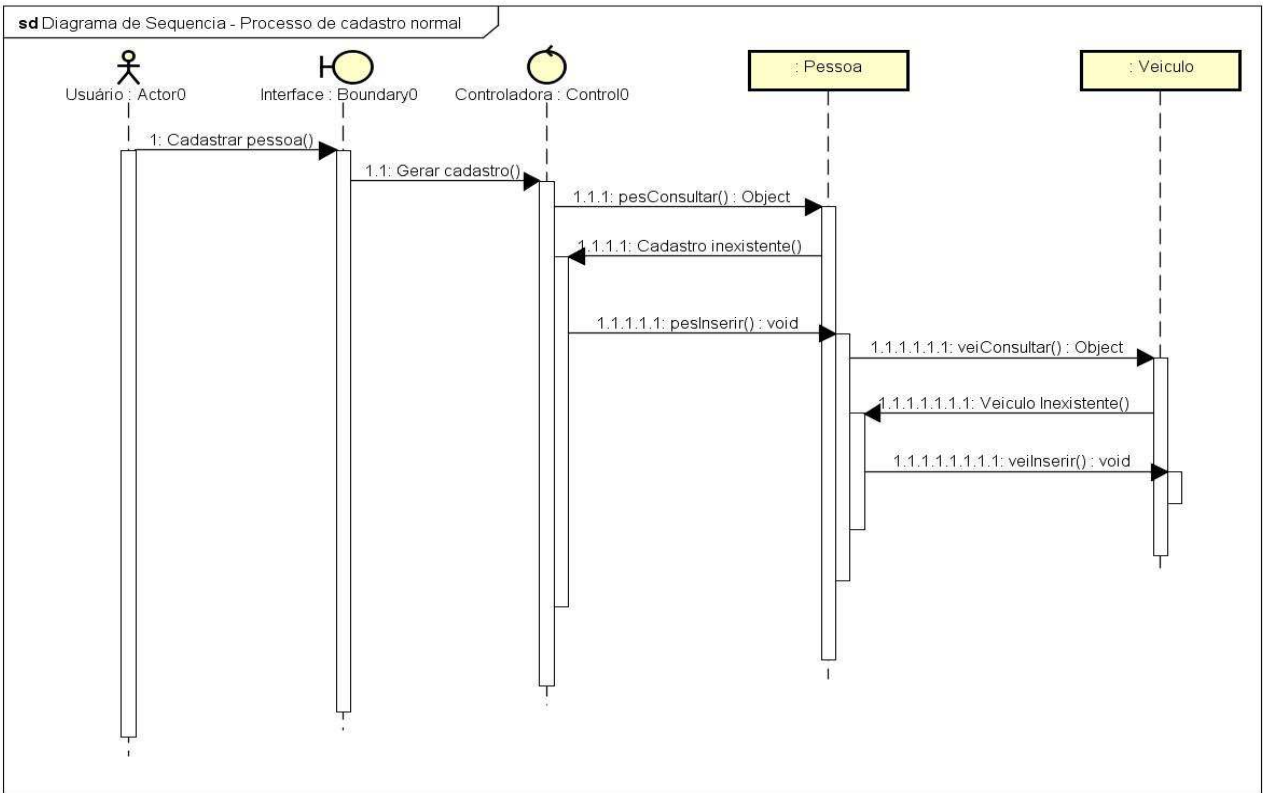
O cadastro de pessoa engloba dois participantes diferentes dentro do sistema, o cliente e o funcionário. O cliente aparecerá sempre como comprador e solicitante de uma ordem de serviço ou orçamento, enquanto o funcionário terá participação apenas ao final do processo que seria de fato uma das finalizações dos processos do sistema que seria o registro de ordem de serviço, onde a mesma será associada a um funcionário e esta pessoa ficará responsável pelo serviço.

O cadastro de veículos está diretamente associado ao cadastro de pessoa porque um veículo sempre tem que possuir um proprietário, entretanto com a entrada da funcionalidade de transferência de posse de veículo, que está melhor representada na Figura 14, este registro para a se tornar, de certa forma

independente, pois possui seu próprio histórico de serviços realizados, sem pouco fazer diferença em quem era o proprietário do veículo no momento.

No processo de cadastro de pessoa e veiculo podem haver duas variações. O processo normal como representa a Figura 12, o sistema consultaria ambas as tabelas de Pessoa e Veiculo e não encontraria resultados iguais em relação ao que se pretende cadastrar, então o processo vai até a tabela Pessoa, faz o cadastro, vai até a tabela Veiculo faz o cadastro e também associa a pessoa cadastrada como sendo proprietária do veículo também cadastrado.

Figura 12 – Diagrama de sequência: Cadastro de pessoa - Fluxo normal

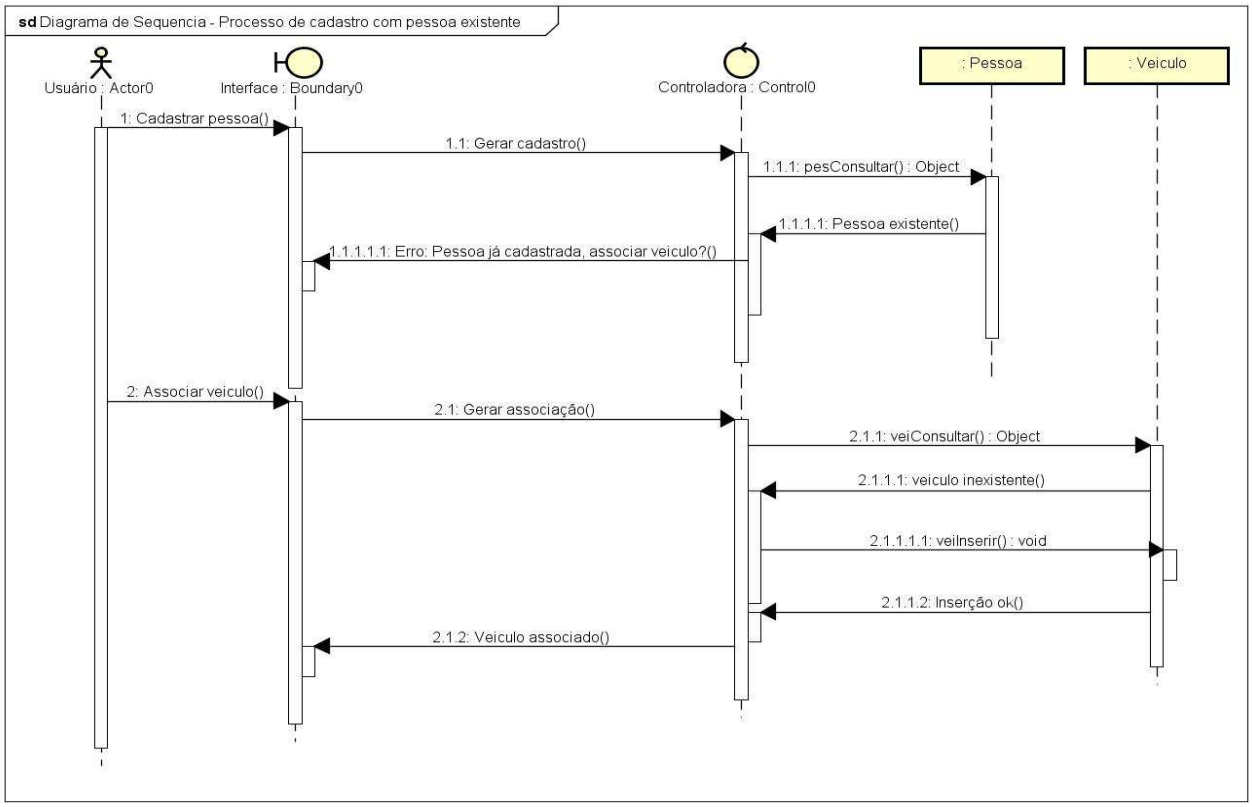


Fonte: Autoria própria

A primeira variação do processo de cadastro de pessoa como ilustra a Figura 13, é quando a pessoa que se pretende cadastrar já foi inserida no sistema, neste caso o usuário tem a opção de apenas associar um novo veículo a esta pessoa que já está cadastrada.

Esta funcionalidade atende um requisito especifico solicitado pelo cliente, este seria chamado de transferência de posse de veículo, a principal vantagem é evitar duplicidade de registros, principalmente para o cadastro de veículos e ainda assim manter todo o histórico já realizado em determinado veículo, independentemente se a posse do mesmo foi alterada.

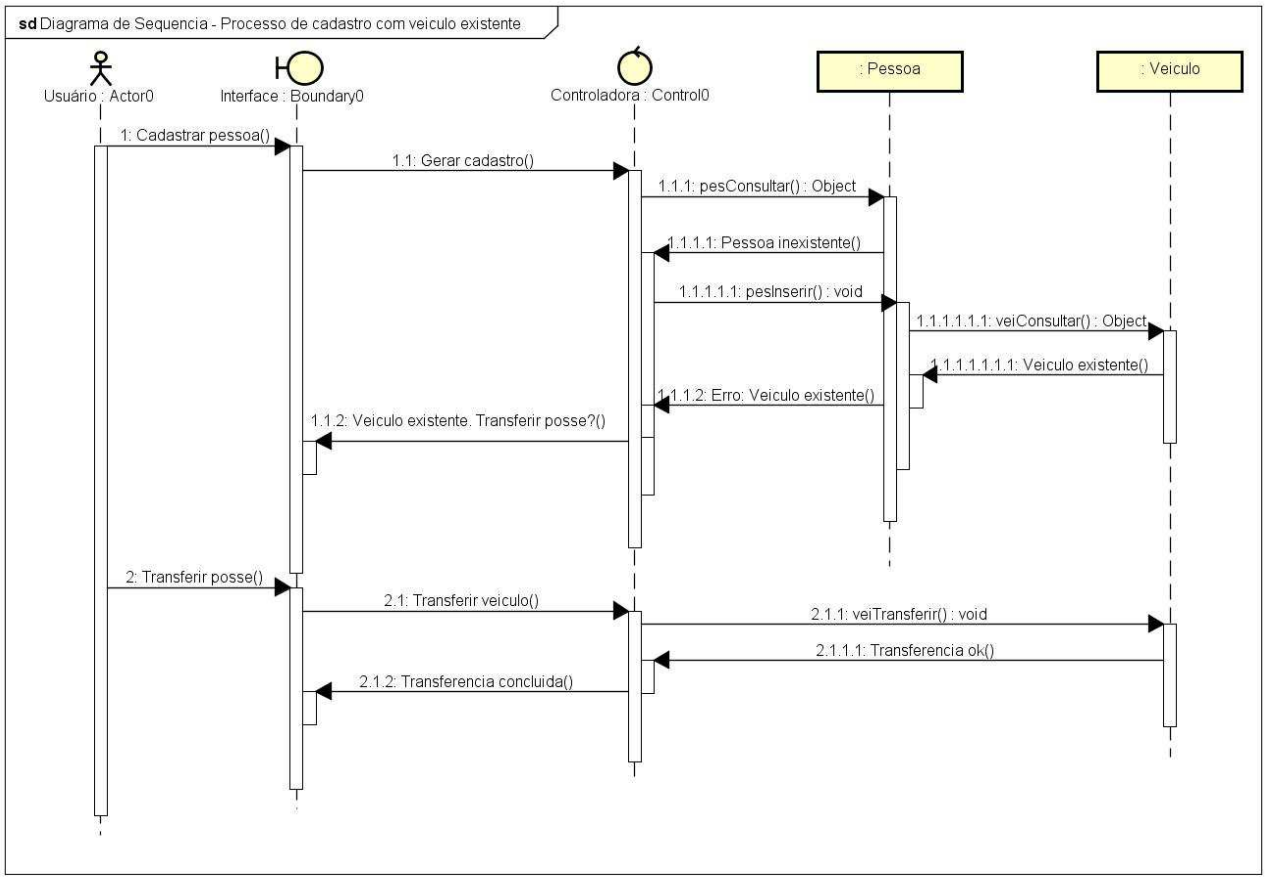
Figura 13 - Diagrama de Sequência: Cadastro de pessoa - Variação 1



Fonte: Autoria própria

A segunda variação do processo de cadastro de pessoa como representado na Figura 14, é quando o veículo que se pretende cadastrar já foi inserido no sistema, neste caso o usuário tem a opção de transferir a posse desse veículo para outra pessoa cadastrada, no caso a pessoa que foi inserida naquele momento.

Figura 14 - Diagrama de Sequência: Cadastro de pessoa - Variação 2

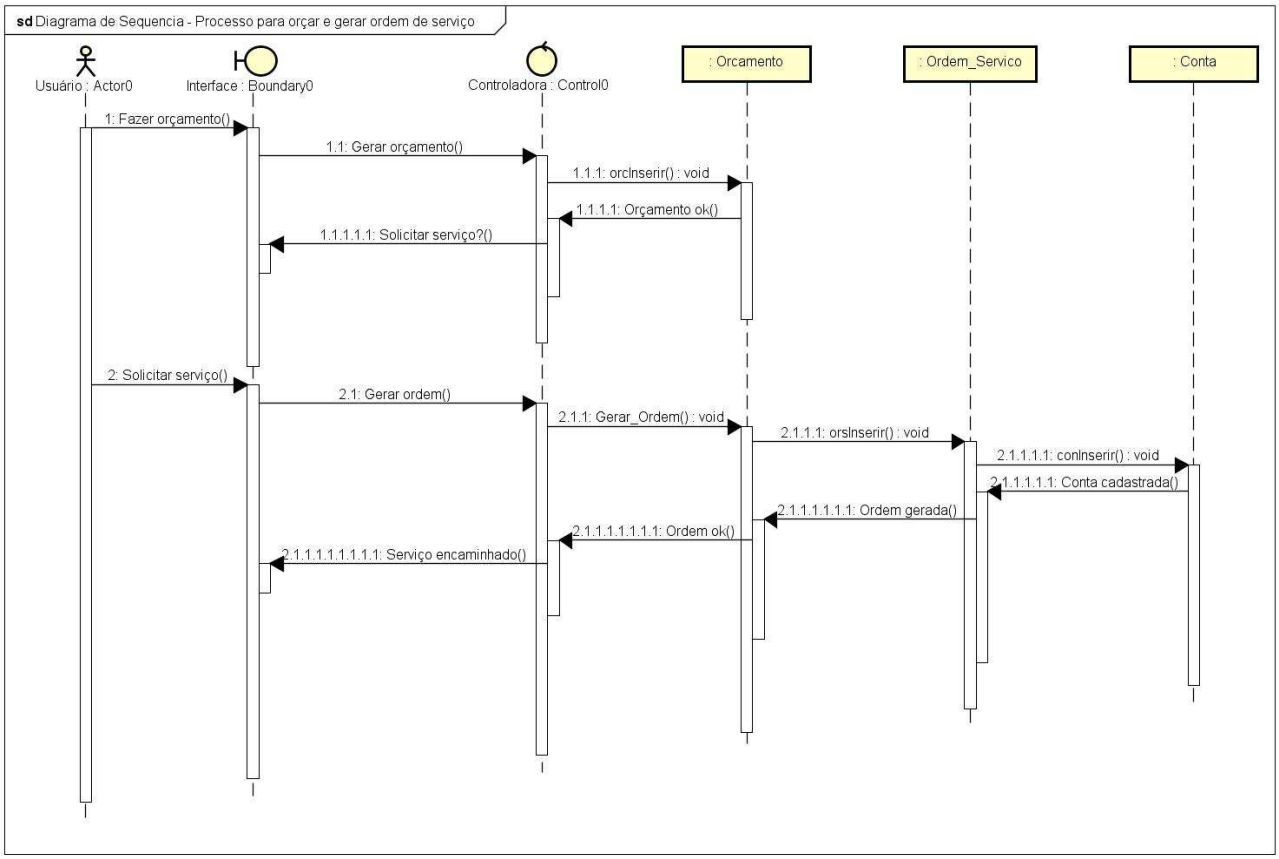


Fonte: Autoria própria

No processo de orçamento não é feito tipo algum de registro no banco de dados, uma vez que este processo serve apenas para informar ao cliente quanto custa determinado serviço, então são feitas apenas consultas e relações sistêmicas entre uma classe e outra para que esta informação seja trazida ao cliente.

Ao receber o orçamento, o cliente tem opção de gerar uma ordem de serviço o que quer dizer que o pedido feito apenas em formato de consulta agora será de fato executado o que também influenciará em um cadastro de conta referente ao pagamento pelo serviço contratado. Todo este processo pode ser melhor visualizado no diagrama de sequência da Figura 15.

Figura 15 – Diagrama de Sequência: Orçamento e Ordem de serviço



Fonte: Autoria própria

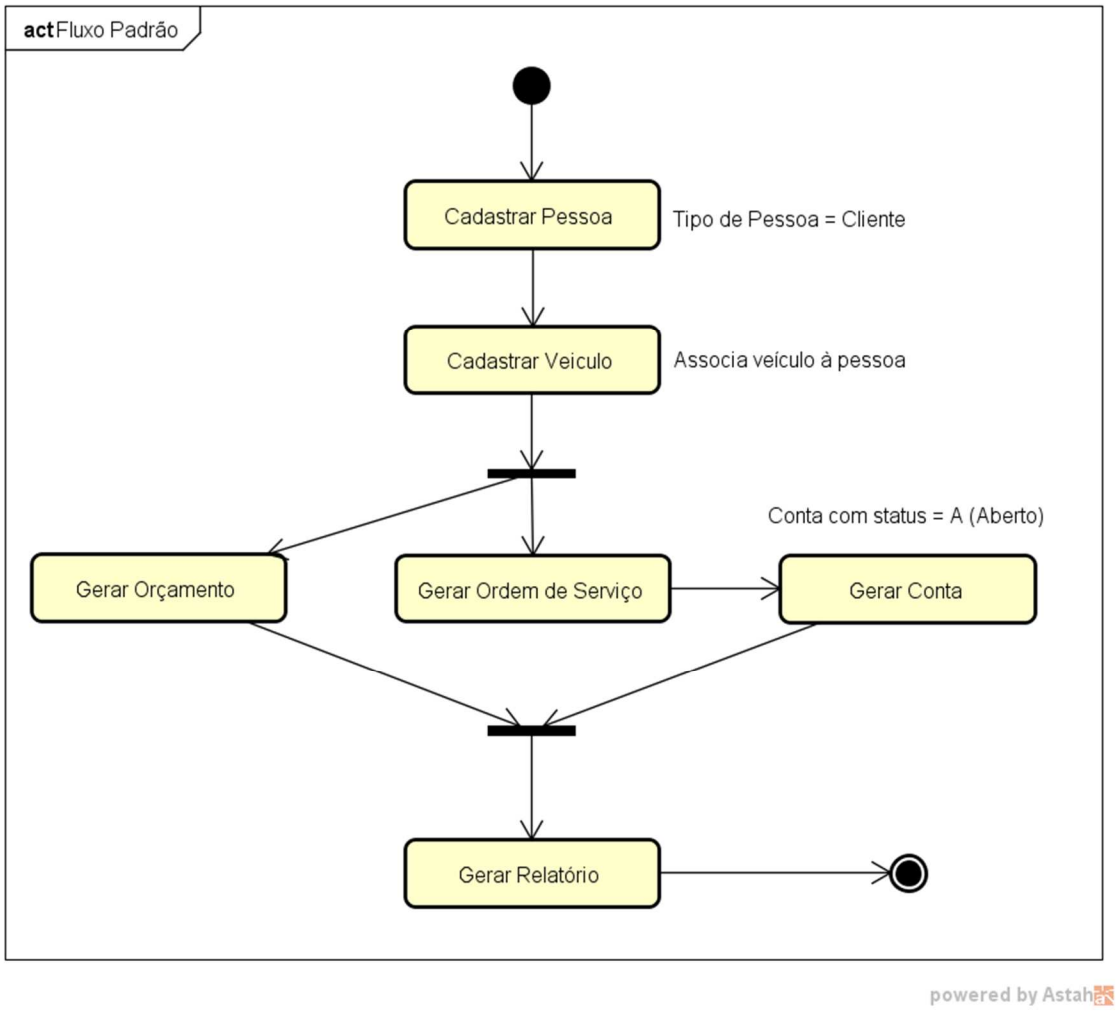
3.3.4. Diagrama de atividade

O diagrama de atividade descrito na Figura 16 representa o fluxo básico entre as funcionalidades mais importantes do sistema. O cadastro de pessoas, esta pode ser cliente ou funcionário, mas nesse caso a situação mais pertinente se aplica para cliente, então esta funcionalidade é a primeira a ser utilizada.

Seguindo o fluxo, após cadastrar pessoa (cliente), deve-se cadastrar veículo para então associar este à pessoa (cliente) que foi cadastrada. Estes dois são os cadastros mínimos para o próximo passo do fluxo.

O próximo passo do sistema executa um processo dentro no sistema, este pode ser a geração de um orçamento ou de uma ordem de serviço, ambos geram relatórios após processados, mas apenas a ordem de serviço gera uma conta em aberto.

Figura 16 – Diagrama de Atividade



Fonte: Autoria própria

3.4. Modelagem de Banco de dados

Segundo Silberschatz (2006, p. 1) “um sistema de gerenciamento de banco de dados (SGBD) é uma coleção de dados inter-relacionados e um conjunto de programas para acessar esses dados”. Se atentando a definição do autor é possível entender que não só no presente software proposto, mas também, de certa forma em grande parte dos softwares comerciais atuais, o real “coração” do sistema é o banco de dados. Tudo gira em torno deles, tanto que projetamos e construímos ferramentas para melhor manipula-los e armazena-los.

3.4.1. Diagramas de Banco de Dados

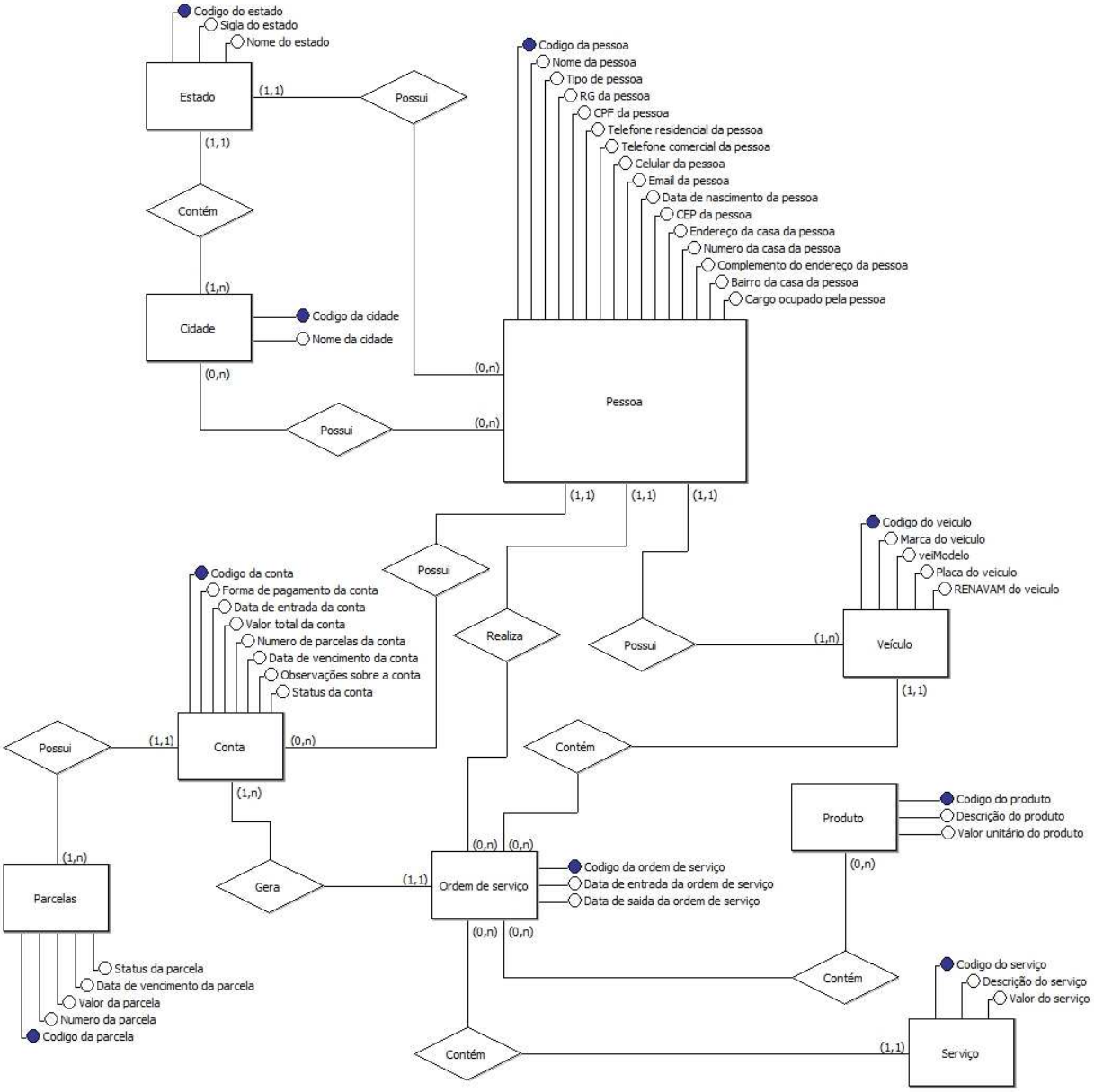
Existe uma serie de ferramentas conceituais para projetar e descrever um banco de dados e suas relações. Como representantes desse conjunto de ferramentas conceituais, serão apresentados dois diagramas, estes nos quais representam o banco de dados em sua totalidade, descrevendo cada uma das relações estabelecidas entre as classes.

O primeiro, ilustrado na Figura 17, apesar de evidentemente ser um diagrama, considera-se apenas como uma representação do Modelo Entidade/Relacionamento (MER), pois a intenção no mesmo é ser propositalmente o mais informal possível, pode-se dizer que este é o rascunho do que viria a se tornar de fato o diagrama de banco de dados, este foi feito para que se pudesse adequar todos os requisitos sem gastar muito tempo pensando em detalhes e ao mesmo tempo fazendo de forma que um leigo também entenda, no caso o próprio cliente.

O segundo diagrama, representado pela Figura 18, é o Diagrama Entidade/Relacionamento (DER) e é de fato o diagrama oficial e técnico, com todos os detalhes possíveis, de cardinalidade, chaves estrangeiras e tabelas auxiliares.

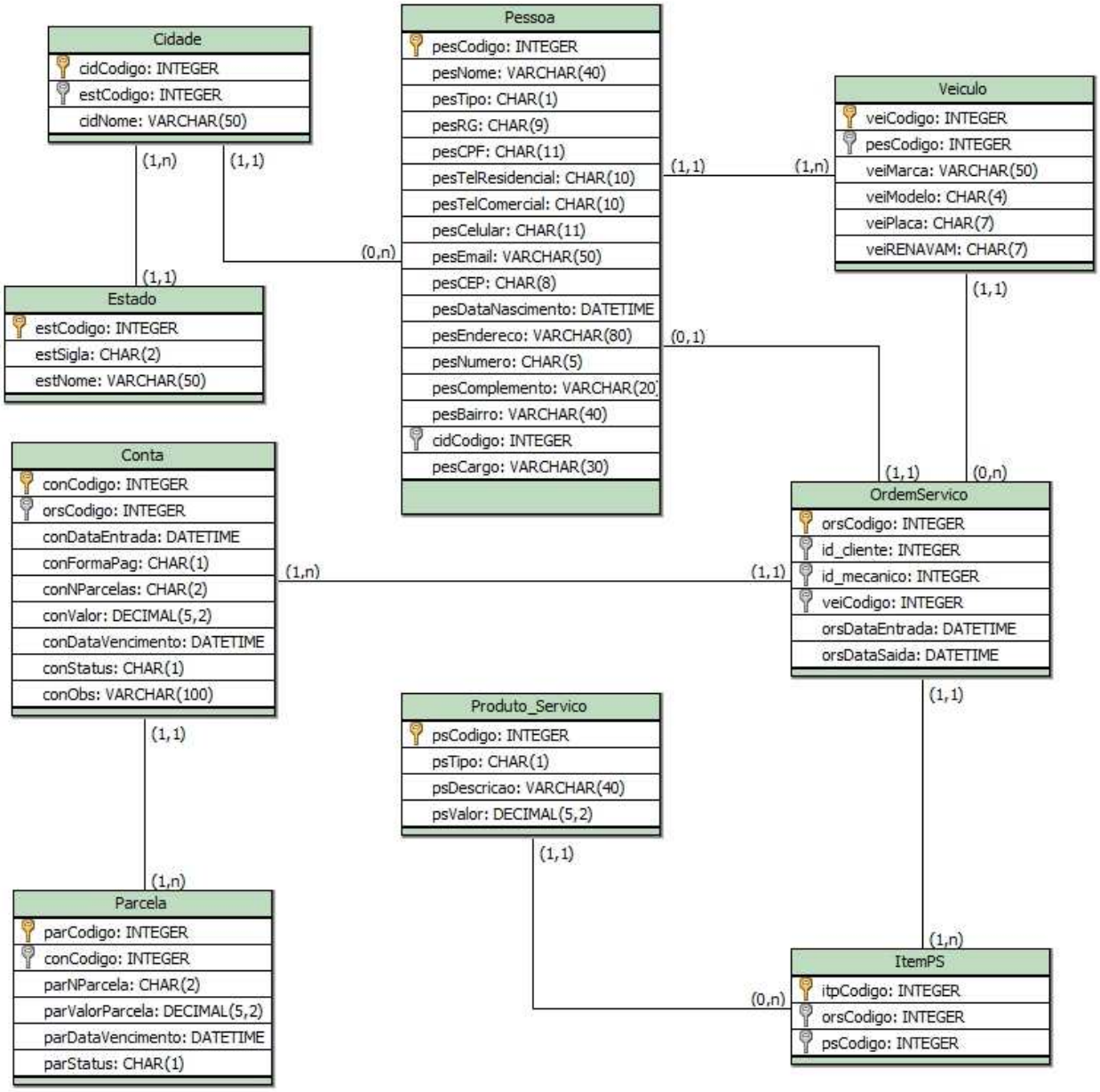
Tanto neste diagrama (Figura 18) quanto no MER (Figura 17), eliminou-se o objeto “Orçamento” presente no Diagrama de classes ilustrado na Figura 11. Apesar se ser parte do processo da empresa, este objeto faz apenas uma simulação de preços através de consultas no banco de dados, então como não há informações que necessitem ter armazenadas, logo este objeto não pode ser considerado uma entidade e não sendo uma entidade não faz parte das relações de banco de dados, sendo considerado apenas um processo sistêmico.

Figura 17 - Modelo Entidade/Relacionamento (Conceitual)



Fonte: Autoria própria

Figura 18 - Diagrama Entidade/Relacionamento (Lógico)



Fonte: Autoria própria

3.5. Ferramentas de desenvolvimento

Todas as ferramentas para o desenvolvimento do sistema foram escolhidas principalmente devido a minha preferência particular, por serem ferramentas que eu considero eficientes e mais fáceis de trabalhar

3.5.1. Ambiente de Desenvolvimento Integrado (IDE)

A IDE escolhida para o desenvolvimento deste projeto do o Visual Studio 2015.

Além da preferência pessoal, há outros motivos para escolher o Visual Studio como IDE a ser utilizada como a orientação em tempo real enquanto se escreve o código. A grande variedade de linguagens de programação para se trabalhar como C#, Visual Basic, F#, C++, HTML, JavaScript, TypeScript, Python entre outras.

Facilidade e eficiência na depuração do código encontrando bugs no código rapidamente. Depuração em vários idiomas, seja local ou remotamente. Diagnóstico de problemas de desempenho sem deixar o fluxo de trabalho de depuração.

3.5.1.1. Linguagem de programação

O C# foi a linguagem de programação escolhida por possuir grandes vantagens, uma delas é a possibilidade de maior interatividade com os internautas, que passam a poder postar informações a qualquer momento, com o C# controlando o envio e recebimento de dados, principalmente a partir da mesma página. Uma outra vantagem na utilização de C# é programação orientada a objeto, não possíveis em C e em Visual Basic.

Uma outra vantagem é o recurso do ambiente Windows. No C# o desenvolvimento é muito semelhante com aplicações desktop, utilizando recursos do ambiente Windows. Assim, alguns dos controles que somente eram encontrados em componentes são agora nativos. Outros aspectos vantajosos no

C#, é que toda *Data Definition Language* (DLL) ou componente não precisa mais ser registrado, extingue-se o papel do *registry* e com os novos controles *validator*, torna-se mais fácil consistir e validar dados na web, bastando apenas invocar o componente, estabelecer a propriedade e vincular a algum controle. Além disso, tratar erros também é muito mais fácil, por meio do *Exception*, o próprio C# identifica e trata o erro

3.5.2. Sistema Gerenciador de Banco de dados (SGBD)

O sistema gerenciador de banco de dados escolhido foi o MySQL, por muitas razões e uma delas seria a sua natureza de código aberto. É livre e pode ser usado por qualquer pessoa, sem qualquer licença ou autorização. E sob a GNU General *Public License*, o código fonte está disponível sob o domínio. Isto permite aos desenvolvedores personalizar o código-fonte de acordo com as suas necessidades e modificar o banco de dados para o seu uso.

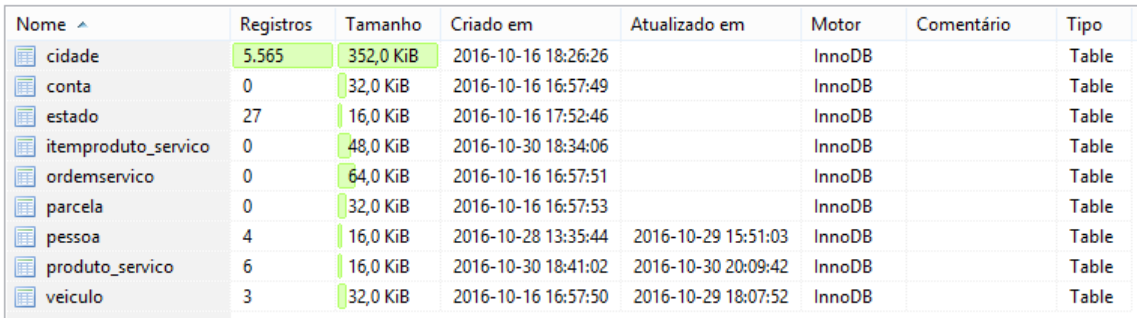
Esta base de dados é fácil de usar e funciona muito rápido. Como é, basicamente, uma versão modificada do SQL, um conhecimento geral do SQL é suficiente para trabalhar de forma eficiente com o MySQL.

3.6. Visual do sistema

3.6.1. Banco de dados finalizado

Como pode ser visto na Figura 19, todo a estrutura de banco de dados foi construída exatamente de acordo com a modelagem definida pós levantamento, organização e seleção de requisitos. O banco tem como objetivo ser o mais minimalista possível, atendendo apenas os requisitos necessários, mas ainda assim permitindo possíveis alterações em iterações futuras.

Figura 19 – Versão final do banco de dados



Nome	Registros	Tamanho	Criado em	Atualizado em	Motor	Comentário	Tipo
cidade	5.565	352,0 KiB	2016-10-16 18:26:26		InnoDB		Table
conta	0	32,0 KiB	2016-10-16 16:57:49		InnoDB		Table
estado	27	16,0 KiB	2016-10-16 17:52:46		InnoDB		Table
itemproduto_servico	0	48,0 KiB	2016-10-30 18:34:06		InnoDB		Table
ordemservico	0	64,0 KiB	2016-10-16 16:57:51		InnoDB		Table
parcela	0	32,0 KiB	2016-10-16 16:57:53		InnoDB		Table
pessoa	4	16,0 KiB	2016-10-28 13:35:44	2016-10-29 15:51:03	InnoDB		Table
produto_servico	6	16,0 KiB	2016-10-30 18:41:02	2016-10-30 20:09:42	InnoDB		Table
veiculo	3	32,0 KiB	2016-10-16 16:57:50	2016-10-29 18:07:52	InnoDB		Table

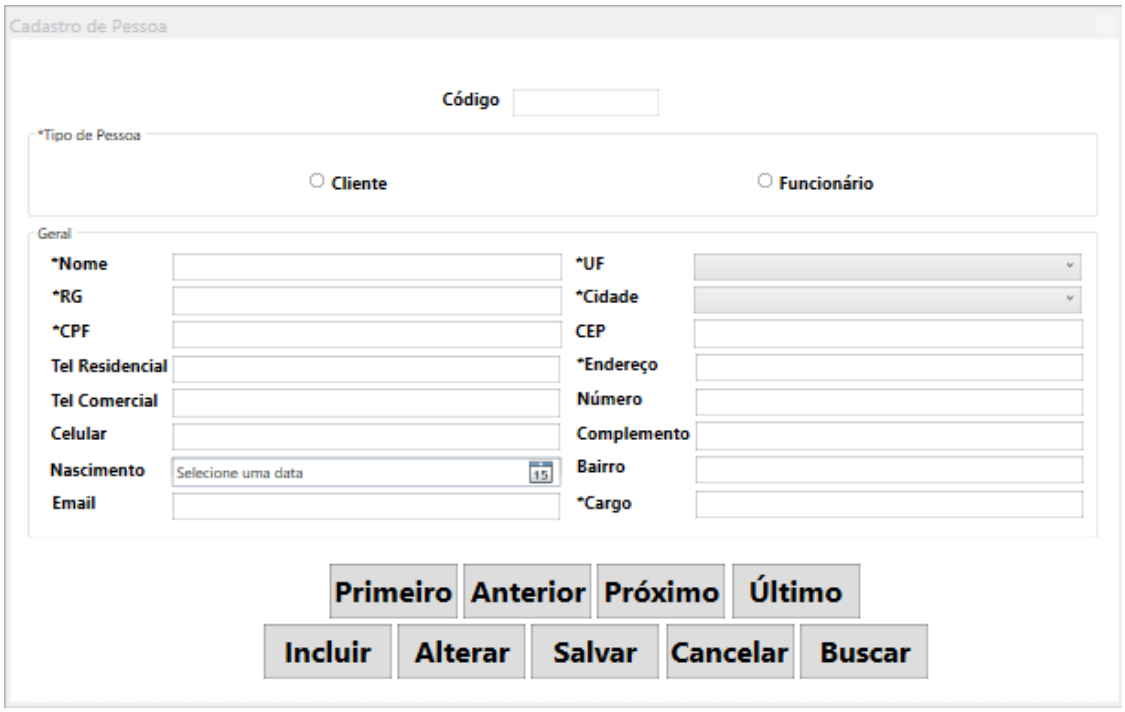
Fonte: Autoria própria

3.6.2. Telas do sistema

Para todas as telas foram adotados padrões de designe, onde todos os campos obrigatórios são identificados com o caractere “*”, o campo de código não é editável, as cores de tela e botões são sempre os mesmos, em campos onde é possível foram adicionados componentes que permitem que o usuário insira informações de forma padronizada no banco de dados. O tamanho da tela também é fixo, o usuário não pode maximiza-la ou minimiza-la, apenas fecha-la. Em telas auxiliares como a tela de Busca de registro, o botão padrão para fechar a tela foi substituído por um botão personalizado que fecha a tela atual e retorna a tela de cadastro.

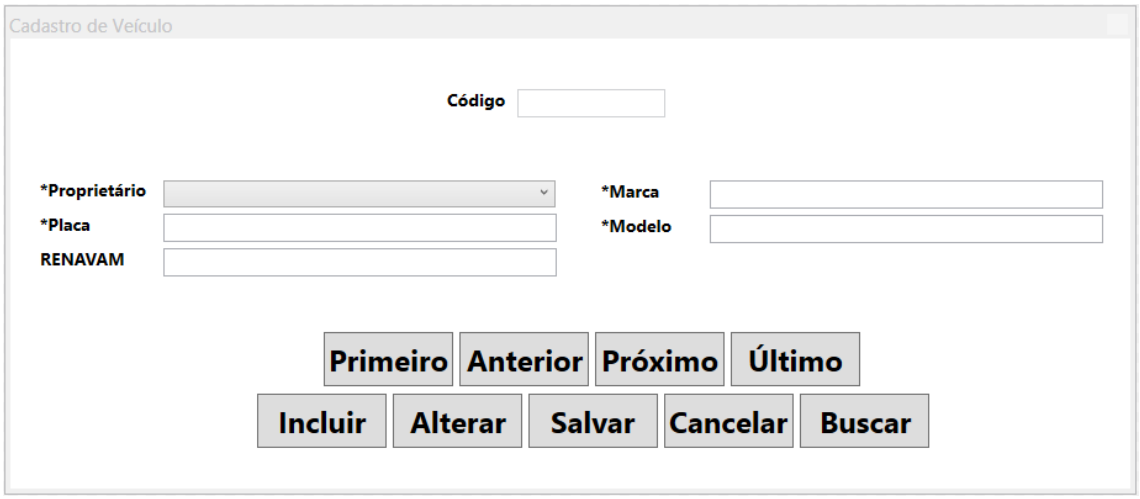
Nas telas de Cadastros em geral como: Cadastro de Pessoa (Figura 20), Veículo (Figura 21) e Produto/Serviço (Figura 22) é possível cadastrar e alterar um registro além de navegar entre eles utilizando os botões Primeiro, Anterior, Próximo e Ultimo. Se o registro for grande também é possível fazer uma busca rápida através do Código, para isso há uma tela auxiliar (Figura 23), basta preencher o código do registro que se deseja buscar e ela tentará localiza-lo.

Figura 20 - Tela de Cadastro de Pessoa



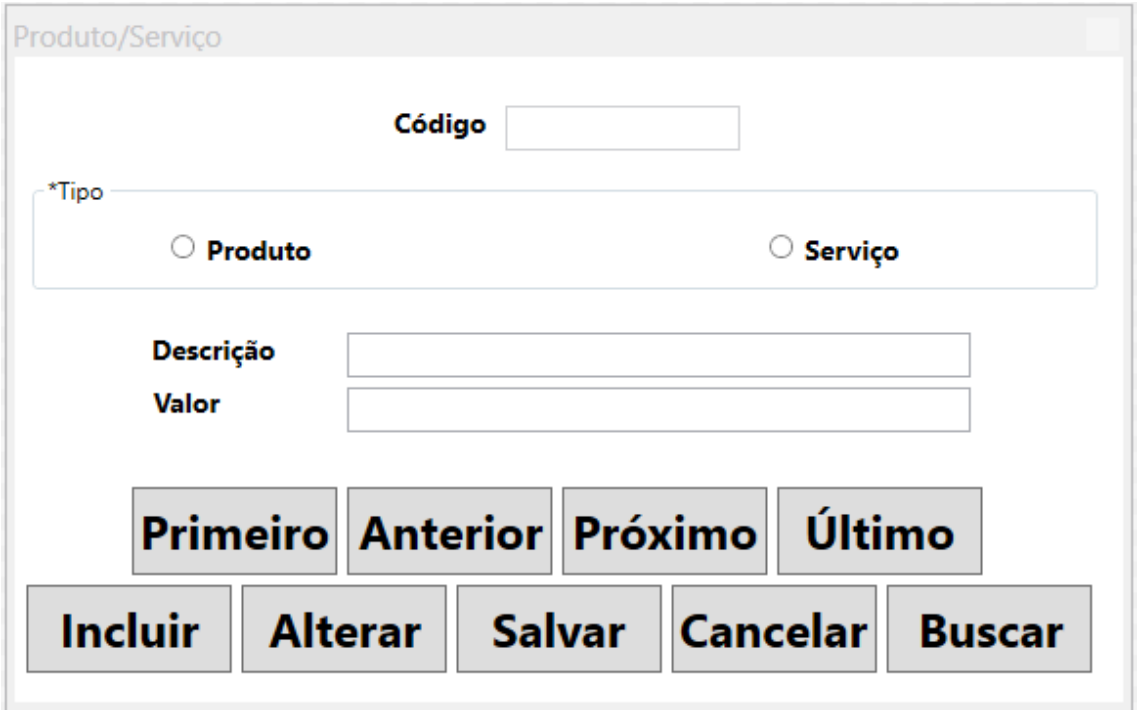
Fonte: Autoria própria

Figura 21 - Tela de Cadastro de Veiculo



Fonte: Autoria própria

Figura 22 - Tela de Cadastro de Produto/Serviço



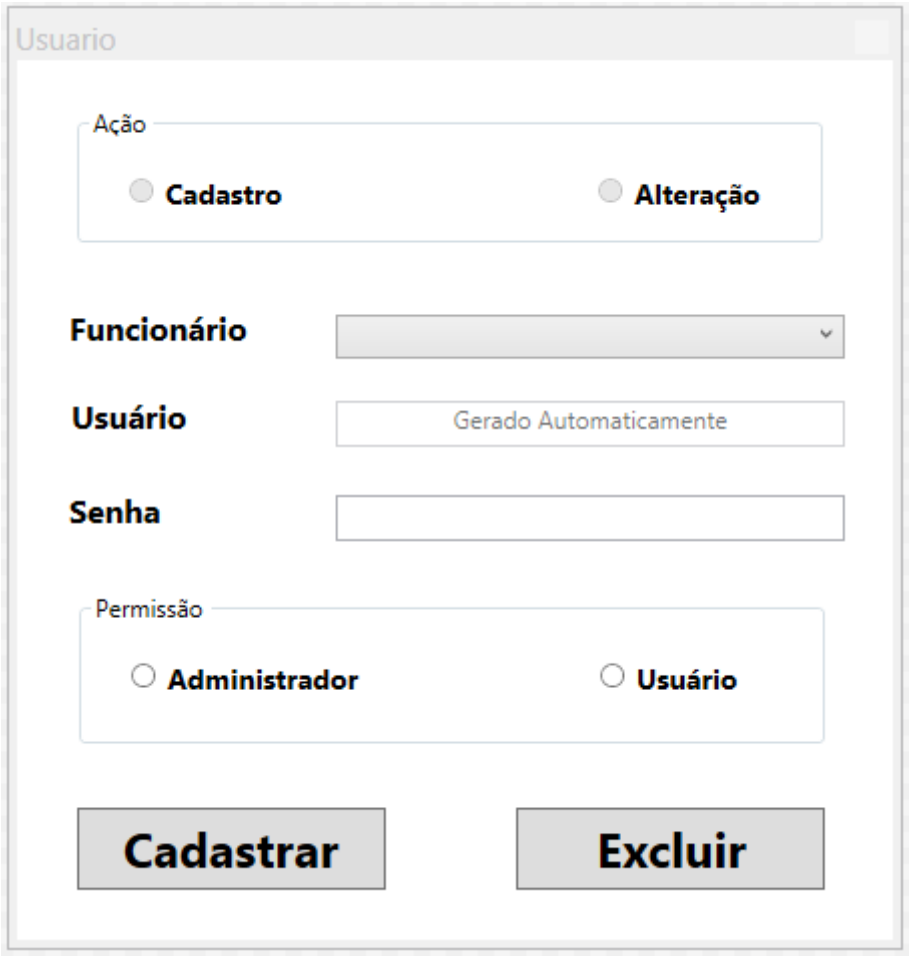
Fonte: Autoria própria

A tela de Usuários, representada na Figura 23, segue basicamente o mesmo padrão das telas de cadastro, mas não há botões de navegação entre os registros.

Nesta tela, se tiver permissão de administrador, o usuário poderá visualizar o registro de cada funcionário cadastrado podendo criar um login, se caso este ainda não tiver, alterar sua senha, tipo de acesso ou até mesmo excluir seu acesso. O login é gerado de forma padronizada, com as 3 primeiras letras do usuário cadastrado, o código dele, o símbolo “@” e por último a palavra mecânica (ex. luc2@mecanica).

Os usuários com permissão simples terão limitações quanto a administração desta tela, estes poderão apenas visualizar o próprio cadastro e alterar a própria senha.

Figura 23 – Tela de Usuários



Fonte: Autoria própria

Na Tela Ordem, representada na Figura 24, são proporcionadas duas funções, nesta tela é possível fazer ordens de serviço e orçamentos. A diferença entre eles é que apenas as ordens de serviço são armazenadas. No processo adiciona-se um cliente, seu veículo e o técnico responsável pelo serviço, após isso os itens da ordem podem ser adicionados, estes podem ser produtos ou/e serviços.

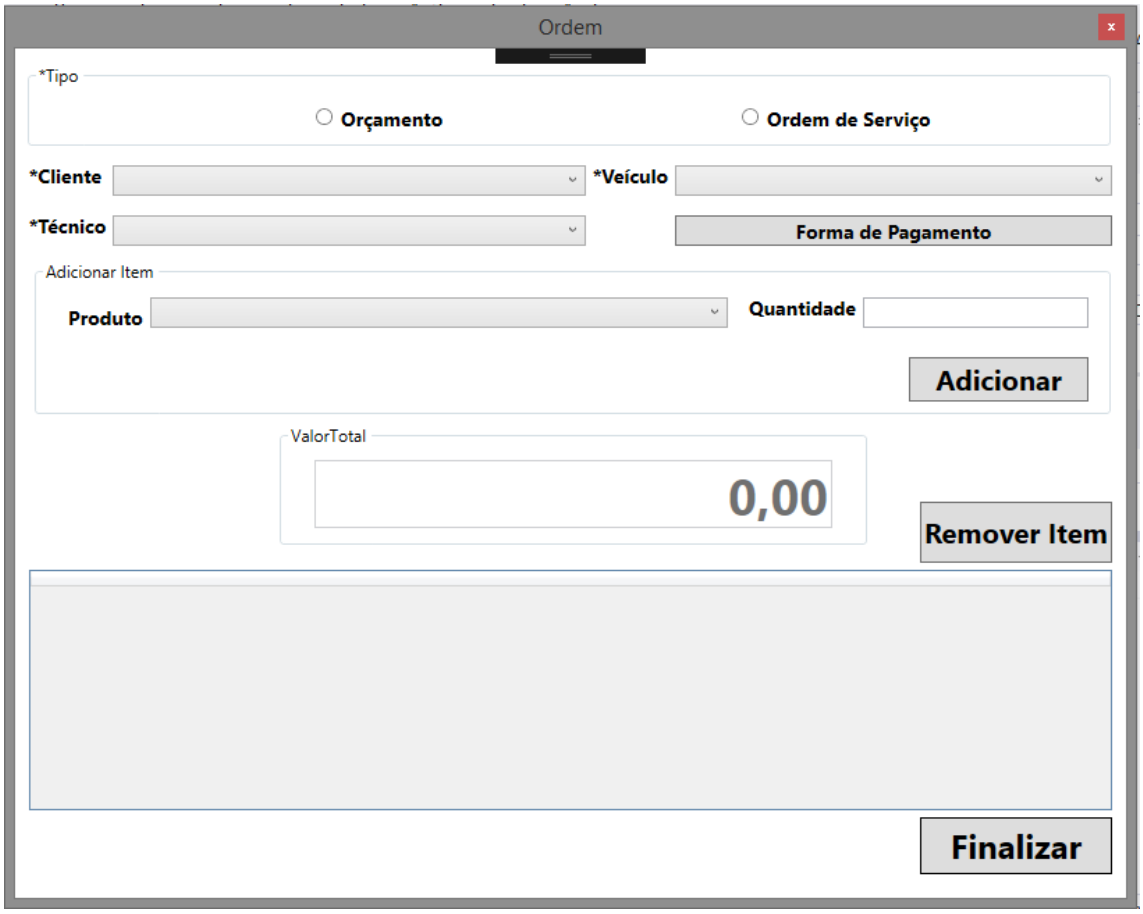
Após adicionar todos os itens, o usuário escolhe uma forma de pagamento acessando a Tela Conta (Figura 25) onde ele selecionará pagamento à vista, ou parcelado, se for parcelado escolherá o número de parcelas e então a data de cobrança da primeira parcela.

Tendo escolhido uma forma de pagamento basta finalizar a ordem de serviço ou orçamento. Se for uma ordem de serviço será gerado um registro de

conta que será fechado somente após o pagamento total pelo serviço. Esta conta também sempre terá um registro de parcelas associado a ela, mesmo que o cliente escolha pagamento a vista, haverá um registro com uma parcela pelo menos. Quando o serviço for finalizado e o veículo entregue ao cliente, então a ordem também será fechada com a última informação que é a data de entrega do veículo com o serviço feito. Este registro, a pedido do cliente, ficará armazenado permanentemente para consultas em futuras novas ordens de serviço no mesmo veículo.

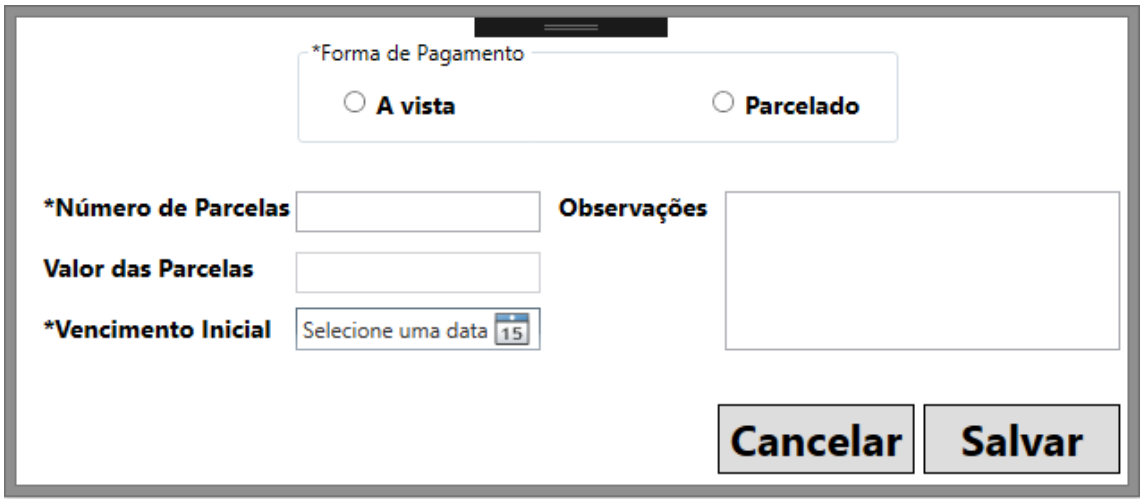
O relatório representado na Figura 26 é um exemplo gerado de um orçamento, esta é uma saída do sistema e um comprovante de fechamento de acordo de prestação de serviços entre oficina e cliente.

Figura 24 – Tela de Ordem



Fonte: Autoria própria

Figura 25 – Tela Auxiliar de forma de pagamento



Fonte: Autoria própria

Figura 26 – Relatório de Serviços (exemplo)

```
--------------------------------------------------
Mecânica Moderna
Av. Francisco Bertolli, nº 34
Pq.das Nações -Sumaré / SP 13181 - 130
Telefone: (19) 3864-9725 Celular: (19)7804-8659
Email: mecanica-moderna@hotmail.com
--------------------------------------------------
Cliente: Joaquim Maldonado (id:20)
Endereço: Rua Jose Marques, nº 658
Telefone: 1936568944 / 19869987456
CPF: 23658945685
Mecânico: Maria Joana Aquino (id:2)
Veiculo: Ford Fiesta, 2010 (id:4)
--------------------------------------------------
Tipo de Documento: Orçamento
Data: 27/11/2016
--------------------------------------------------
Item                        | Valor | Qtd | Total
Pastilha de Freio            35,00   4     140,00
Revisão                      310,00  1     310,00

--------------------------------------------------

Valor Total: 450,00 R$

Forma de Pagamento: Parcelado em 2 X de 225,00 R$

--------------------------------------------------


      _____                 _____
        Cliente               Responsável
```

Fonte: Autoria própria

3.7. Estrutura de Testes

O teste de software a ser implementado será o teste de Caixa Preta, visando uma perspectiva funcional do sistema mediante aos requisitos já definidos no levantamento de requisitos feito no início do projeto.

O Teste de Caixa Preta será associado ao Teste Unitário, que tem como objetivo analisar as funcionalidades do sistema de forma avulsa, os testes serão divididos por telas onde em cada uma delas haverá uma expectativa de funcionamento que será verificada de acordo com o roteiro de testes.

3.7.1. Caso de Teste Cadastro de Pessoa

O primeiro caso de teste se refere a tela de cadastro de pessoa onde os campos Tipo de Pessoa (que pode ser Cliente ou Funcionário), Nome, RG, CPF, UF, Cidade e Endereço são obrigatórios e para Tipo de Pessoa Funcionário o campo Cargo também é obrigatório.

O sistema também é preparado para verificar se o CPF e o e-mail (que não é obrigatório) informados são válidos. Não é possível salvar o registro antes que todos os campos obrigatórios sejam preenchidos com os valores válidos.

O fluxo de dados normal pode ser visto na Tabela 20, neste caso todos os dados foram inseridos corretamente e o registro no banco de dados foi feito com sucesso.

Tabela 20 - Caso de teste: Cadastro de Pessoa -Fluxo Básico

Cadastro de Pessoa				
Fluxo Básico				
ID	**Objetivo**	**Dados de Entrada**	**Procedimento**	**Resultado Esperado**
01	Registrar no banco de dados na tabela pessoa as informações de pessoa	Tipo de Pessoa: Marcar "Cliente" ou "Funcionário" Nome (Exemplo): João Silva RG (Exemplo): 911225341 CPF (Exemplo): 48512650346 Tel. Residencial (Exemplo): 1938667899 Tel. Comercial (Exemplo): 1938745000 Celular (Exemplo): 19988775663 Nascimento (Exemplo): 10/10/1990 E-mail (Exemplo): mecanica@gmail.com UF (Exemplo): 25 | São Paulo | SP Cidade (Exemplo): 5.096 | Sumaré CEP (Exemplo): 13180698 Endereço (Exemplo): Rua Jose silva Número (Exemplo): 2597 Complemento (Exemplo): Casa B Bairro (Exemplo): Campo Belo Cargo (Se Tipo de Pessoa = "Funcionário" - Exemplo): Mecânico	Preencher todos os dados de pessoa de acordo com as métricas de validação	Salvar no banco de dados todos os dados preenchidos. E exibi-los como parte do cadastro.

Fonte: Autoria própria

O fluxo na Tabela 21 representa um fluxo alternativo para cadastro de pessoas que é ativado quando um ou mais campos obrigatórios estão vazios no momento em que o usuário tenta salvar o registro. Neste caso nada será salvo, e o sistema exibirá uma mensagem informando ao usuário que os campos identificados com o caractere “*” são obrigatórios.

O terceiro caso de teste referente ao Cadastro de Pessoa, está representado na Tabela 22, este segundo fluxo alternativo é ativado quando no campo CPF é inserido um valor inválido, neste caso, assim como no caso de teste de fluxo alternativo 1 (Tabela 21), o registro não será salvo e o usuário receberá uma mensagem o informando que o campo CPF está inválido.

Tabela 21 - Caso de teste: Cadastro de Pessoa -Fluxo Alternativo 1

Cadastro de Pessoa				
Fluxo Alternativo 1				
ID	**Objetivo**	**Dados de Entrada**	**Procedimento**	**Resultado Esperado**
02	Garantir que os campos obrigatórios sejam preenchidos.	Tipo de Pessoa: <Branco> Nome (Exemplo): <Branco> RG (Exemplo): <Branco> CPF (Exemplo): <Branco> UF (Exemplo): <Branco> Cidade (Exemplo): <Branco> Endereço (Exemplo): <Branco> Cargo (Se Tipo de Pessoa = "Funcionário" - Exemplo): <Branco>	Deixar campos obrigatórios em branco.	Exibir mensagem: Os campos identificados com * são obrigatórios.

Fonte: Autoria própria

Tabela 22 - Caso de teste: Cadastro de Pessoa -Fluxo Alternativo 2

Cadastro de Pessoa				
Fluxo Alternativo 2				
ID	**Objetivo**	**Dados de Entrada**	**Procedimento**	**Resultado Esperado**
03	Garantir que o campo CPF seja preenchido corretamente.	Tipo de Pessoa: Marcar "Cliente" ou "Funcionário" Nome (Exemplo): João Silva RG (Exemplo): 911225341 CPF (Exemplo): 0000 UF (Exemplo): 25 | São Paulo | SP Cidade (Exemplo): 5.096 | Sumaré Endereço (Exemplo): Rua jose silva Cargo (Se Tipo de Pessoa = "Funcionário" - Exemplo): Mecânico	Preencher os campos obrigatórios, e preencher CPF com valor inválido.	Exibir mensagem: O campo CPF está inválido.

Fonte: Autoria própria

3.7.2. Caso de Teste Cadastro de Veículo

O segundo caso de teste se refere ao cadastro de veículo, o que seria, de certa forma, o processo seguinte ao cadastro de pessoa. Este processo depende que uma pessoa já tenha sido cadastrada, uma vez que uma das informações que esta tela armazena no banco é a pessoa proprietária do veículo.

O fluxo normal para este caso de teste, representado na Tabela 23, exige que haja uma pessoa previamente cadastrada para associar ao veículo que se pretende cadastrar, e que todas as informações obrigatórias sejam preenchidas como Placa, Marca e Modelo.

O fluxo alternativo, representado na Tabela 24, prevê a possibilidade de o usuário deixar de preencher um ou mais campos obrigatórios, estes que são basicamente todos com exceção do campo RENAVAM, neste caso o sistema são registrará as informações no banco de dados e exibirá uma mensagem alertando o usuário que os campos identificados com “*” obrigatoriamente tem que ser preenchidos.

Tabela 23 - Caso de teste: Cadastro de Veículo -Fluxo Básico

Cadastro de Veículo				
Fluxo Básico				
ID	**Objetivo**	**Dados de Entrada**	**Procedimento**	**Resultado Esperado**
04	Registrar no banco de dados na tabela veículo as informações de veículo	Proprietário (Exemplo): 01 | João José Placa (Exemplo): XXX4455 RENAVAM (Exemplo): 9112253 Marca (Exemplo): Ford Ka Modelo (Exemplo): 2000	Preencher todos os dados de veículo de acordo com as métricas de validação	Salvar no banco de dados todos os dados preenchidos. E exibi-los como parte do cadastro.

Fonte: Autoria própria

Tabela 24 - Caso de teste: Cadastro de Veículo -Fluxo Alternativo 1

Cadastro de Veículo				
Fluxo Básico				
ID	**Objetivo**	**Dados de Entrada**	**Procedimento**	**Resultado Esperado**
05	Garantir que os campos obrigatórios sejam preenchidos.	Proprietário (Exemplo): <Branco> Placa (Exemplo): <Branco> Marca (Exemplo): <Branco> Modelo (Exemplo): <Branco>	Deixar todos os campos em branco.	Exibir mensagem: Os campos identificados com * são obrigatórios.

Fonte: Autoria própria

3.7.3. Caso de Teste Ordem

O terceiro caso de teste de refere a tela Ordem, esta é responsável por gerenciar duas funções dentro do sistema, Orçamentos e Ordens de Serviço. Estas duas funções são quase iguais, tendo apenas uma diferença: os Orçamentos não são registrados no banco de dados, eles são apenas uma simulação de quanto vai custar para solicitar determinados serviços com determinados produtos.

No fluxo normal, representado na Tabela 25, o usuário entra na tela Ordem, escolhe o tipo de ordem (orçamento ou ordem de serviço), seleciona um cliente e um dos seus veículos que já foram previamente cadastrados. Em seguida adiciona produtos e serviços necessários. For fim finaliza o orçamento ou a ordem de serviço.

O primeiro fluxo alternativo previsto representado na Tabela 26, garante que nenhum registro seja salvo sem antes preencher as informações obrigatórias que são o tipo de ordem (orçamento ou ordem de serviço), cliente e veículo. A ordem também tem que obrigatoriamente ter ao menos um produto ou serviço. Do contrário a tela Ordem não armazena registros e informa o usuário que os campos identificados com “*” tem que obrigatoriamente serem preenchidos e que a ordem não pode ser salva sem itens.

O segundo fluxo alternativo, representado na Tabela 27, impede que um produto ou serviço seja adicionado sem sua respectiva quantidade, o que implica no cálculo de valor a ser pago por cada produto ou serviço e também o valor total da ordem de serviço ou orçamento, logo tendo o usuário clicado no botão Adicionar e estando o campo quantidade em branco ou com um valor inválido, o que seria qualquer coisa diferente de um número maior ou igual a 1, ou seja números negativos, letras, caracteres especiais serão todos barrados nesta validação. Assim como no fluxo alternativo 1 (Tabela 26), nenhum registro será armazenado e o sistema exibirá uma mensagem ao usuário informando que um item não pode ser adicionado a ordem sem antes atribuir uma quantidade ao mesmo.

Tabela 25 - Caso de teste: Ordem - Fluxo Básico

Ordem				
Fluxo Básico				
ID	**Objetivo**	**Dados de Entrada**	**Procedimento**	**Resultado Esperado**
06	Registrar no banco de dados na tabela ordemservico ou mostrar os dados de orçamento ao usuário.	Tipo (Exemplo): Marcar "Orçamento" ou "Ordem de Serviço" Cliente (Exemplo): 01 | João José Veículo (Exemplo): 01 | Ford Ka 2010 Produto (Exemplo): 01 | Troca de Óleo Quantidade (Exemplo): 1	Preencher todos os dados de veículo de acordo com as métricas de validação	Salvar no banco de dados na tabela ordemservico todos os dados preenchidos. E abrir tela de Contas.

Fonte: Autoria própria

Tabela 26 - Caso de teste: Ordem - Fluxo Alternativo 1

Ordem				
Fluxo Alternativo 1				
ID	**Objetivo**	**Dados de Entrada**	**Procedimento**	**Resultado Esperado**
07	Garantir que os campos obrigatórios sejam preenchidos.	Tipo (Exemplo): <Desmarcado> Cliente (Exemplo): <Branco> Veículo (Exemplo): <Branco> Produto (Exemplo): <Branco> Quantidade (Exemplo): <Branco>	Deixar todos os campos em branco	Exibir mensagem: Os campos identificados com * são obrigatórios.

Fonte: Autoria própria

Tabela 27 - Caso de teste: Ordem - Fluxo Alternativo 2

Ordem				
Fluxo Alternativo 2				
ID	**Objetivo**	**Dados de Entrada**	**Procedimento**	**Resultado Esperado**
08	Registrar no banco de dados na tabela ordemservico ou mostrar os dados de orçamento ao usuário.	Tipo (Exemplo): Marcar "Orçamento" ou "Ordem de Serviço" Cliente (Exemplo): 01 | João José Veículo (Exemplo): 01 | Ford Ka 2010 Produto (Exemplo): 01 | Troca de Óleo Quantidade (Exemplo): <vazio>	Preencher todos os dados de veículo e tentar adicionar item sem quantidade	Exibir mensagem: Um item não pode ser adicionado sem quantidade.

Fonte: Autoria própria

4. ANÁLISE DE RESULTADOS

Como dito no início deste trabalho, a proposta do mesmo é demonstrar todo o processo de engenharia de software envolvido no de desenvolvimento de uma ferramenta de gestão de negócios com ênfase em controle de ordens de serviços e manutenção de clientes voltado para empresas de mecânica automotiva.

As fases do processo de engenharia de sistemas descritas por Sommerville (2007, p.17) seriam: Definição de requisitos, Projeto do sistema, desenvolvimento de subsistemas, integração, instalação e evolução do sistema. Por fim haveria a desativação do sistema após findado seu ciclo de vida.

A modelagem de software contribuiu muito para o desenvolvimento do projeto, permitindo de certa forma uma simplificação da realidade, esta que começou no primeiro contato com cliente, onde ele expos a sua realidade, e mais importante que isso, as suas necessidades. No início não é possível compreender um sistema complexo como este em sua totalidade, mas nas primeiras atividades já é possível estruturar um esboço de como o sistema deve ser para que as vontades do cliente sejam atendidas. Além de ajudar a planejar, os procedimentos promoveram a importância da documentação de todas as decisões tomadas.

Uma vez que os procedimentos de coleta de requisitos e modelagem permitem visualizar apenas um esboço do que realmente virá a se tornar o sistema, a melhor opção de modelo de processo de desenvolvimento de software realmente foi o modelo iterativo. Este permitiu uma melhora constante no sistema visando atender de melhor forma as necessidades do cliente, e ao mesmo tempo controlando os riscos do projeto já que os requisitos estavam em constante atualização.

Como já previsto, a tendência do sistema é se tornar mais complexo ao longo do tempo, precisando de atualização e aperfeiçoamento. Portanto, à medida que o sistema evoluir, tendo armazenadas documentação e modelagem, o trabalho será muito menor e com um risco bem reduzido. Isto tudo fortalecido pelo modelo iterativo, que permite deixar sempre algumas “pontas soltas” para que haja sempre a possibilidade de personalizar o sistema.

Os diagramas foram um segundo nível de requisitos onde foram estruturadas as informações que haviam sido coletadas, estes por serem mais visuais permitem uma compreensão maior e mais rápida dos requisitos, e melhor que isso, não necessariamente se exige um alto conhecimento técnico para entende-los. Os diagramas serviram não só para estruturar as informações coletadas, mas também para cadenciar o desenvolvimento do sistema, estabelecendo métricas e padrões para os elementos internos ao sistema.

Os testes permitiram fazer uma revisão geral em praticamente todos os aspectos do sistema, além de encontrar bugs o objetivo principal foi verificar se as funcionalidades do sistema estavam de fato atendendo as necessidades do cliente, isto parte do visual das telas até a sequência das funcionalidades, tipos de dados coletados e informações geradas.

Tendo em vista as fases propostas, todas foram cumpridas desde a primeira fase de levantamento de requisitos até as fases de teste. Uma vez que o modelo de processo de desenvolvimento de software adotado foi o iterativo, há de se considerar o trabalho desenvolvido neste projeto como sendo a primeira versão de software que será submetida à uma serie de melhorias no futuro.

O protótipo de sistema será utilizado inicialmente à fim de realizar testes práticos no cotidiano de trabalho do cliente, desta forma espera-se detectar bugs dos tipos mais simples aos mais complexos dentro dos processos de negócio, agilizando assim a construção do escopo com suas respectivas prioridades nas iterações seguintes.

Como já citado no capítulo introdutório, Sommerville (2007, p. 5), destaca que principal objetivo de um bom software é proporcionar ao usuário a funcionalidade e desempenho e que ele necessita. Com funcionalidade e desempenho entregues satisfatoriamente a tendência é que o software ajude a reduzir horas de trabalho. Muitas tarefas manuais podem ser substituídas facilmente e deixam mais tempo livre para serem realocados em outras áreas.

Uma vez que o sistema foi desenvolvido para gerenciar os negócios de um cliente em especifico, o software fornecerá apenas o que for necessário, isso significa que no sistema proposto não há funcionalidades ou informações sobressalentes. É evidente que quando se tem em mãos apenas as ferramentas que serão utilizadas, a agilidade no controle dos negócios aumenta muito, já que tudo está mais organizado e consequentemente mais acessível.

Parte dos processos de contas era feito manualmente, agora todo este processo é informatizado e praticamente automatizado de ponta a ponta. Isto é mais uma evidencia de que a redução nas horas de trabalho realmente aconteceu.

A proposta do sistema, numa perspectiva geral, é otimizar os processos dentro do ambiente de trabalho, e não necessariamente resolver todos os problemas de processos e negócios da empresa, isso ficaria ao encargo de outras áreas do conhecimento, como administração, contabilidade e marketing, mas uma vez que o sistema foi completamente desenvolvido para se adaptar os processos de negócios existentes e também otimiza-los, se estes sofrerem alterações a ferramenta poderá acompanha-las através de modificações e personalizações nas iterações futuras sem dificuldade alguma.

5. CONSIDERAÇÕES FINAIS

Neste trabalho foi abordado todo o processo de desenvolvimento de software para a oficina Mecânica Moderna. Uma empresa de origem familiar que possui uma grande credibilidade com os clientes da região de Sumaré/SP. Esta possui dois funcionários, estes nos quais realizam diversos tipos de serviços relacionados a mecânica geral de carros. Fica ao encargo do Sr. Airton, proprietário da empresa, todo o serviço de gestão de negócios, como a compra de produtos necessários nos serviços prestados e também a supervisão do trabalho realizado pelos seus dois funcionários. Todo este processo de administração é feito através de três ferramentas, duas delas informatizadas e uma manual que seria um grande caderno de anotações. Como relatado pelo cliente, Sr. Airton, três ferramentas diferentes para administrar processos de mesma origem afetam muito a organização e produtividade da empresa de maneira geral, uma vez que ele, como proprietário, possui uma série de outras atividades a serem realizadas, estas que muitas vezes são prejudicadas pelo tempo gasto para coordenar a administração do negócio em três ferramentas diferentes que possuem diversas funcionalidades, muitas que ele não necessita.

Tendo em vista a adversidade do Sr. Airton, o objetivo geral deste trabalho foi organizar os processos administrativos da empresa Mecânica Moderna através do desenvolvimento de um sistema de gestão personalizado. Toda a análise e desenvolvimento do sistema foi descrita com foco nos procedimentos de engenharia de software visando o conhecimento e entendimento dos mesmos bem como a sua importância para a qualidade de um sistema de gestão.

Todos os objetivos propostos foram cumpridos. Houve um acompanhamento minucioso em todos os processos de engenharia de software envolvidos, desde a primeira coleta de requisitos até o último teste de software. Desde o início se manteve uma rígida “vigilância” para que se mantivessem os processos do modelo de desenvolvimento de software adotado, o modelo Iterativo, este que procura entregar valor ao cliente gradativamente, sempre verificando se o que está sendo desenvolvido é realmente o que o cliente deseja.

Procurou-se detalhar o máximo possível todos os documentos que envolviam informações a respeito dos requisitos e funcionalidades do sistema. Tudo isso visando o real cumprimento dos requisitos do cliente e a viabilização

de melhorias para o sistema no futuro. Todos os diagramas relacionados a requisitos e modelagem foram construídos com o maior número de detalhes possível, o que agilizou muito a construção do banco de dados e sistema. Os testes foram realizados nas principais funcionalidades do sistema com objetivo de verificar validações e o fluxo interno do sistema. Os processos administrativos foram todos otimizados, isto consequentemente agilizou todo o trabalho de gestão do negócio, uma vez que boa parte dos processos foram automatizados, não necessitando assim de muito tempo do usuário para manipular o sistema.

Os trabalhos futuros, ainda seguindo a metodologia de desenvolvimento adotada, serão conduzidos através de novas iterações, o cliente ficará com um protótipo do sistema, e no começo usará o mesmo como uma ferramenta secundária, ou seja, as ferramentas que o cliente já utilizava, por fins de segurança aos seus dados, continuarão sendo usadas temporariamente, este processo inicial terá como objetivo corrigir possíveis falhas no sistema que talvez tenham passado desapercebidas nos processos de teste, findado o período de testes do cliente (aproximadamente um mês), haverá uma nova reunião. Esta reunião basicamente seria uma nova coleta de requisitos para as modificações do sistema, dando prioridade à correção de bugs e então as modificações solicitadas pelo cliente.

Atualmente já existem itens no escopo das novas iterações, estes já foram acordados com o cliente antes mesmo da entrega do protótipo, estas seriam: Personalização nos relatórios para que sejam gerados no formato PDF (atualmente gerado em TXT), exibindo alguma imagem no cabeçalho representando a empresa; Backup automático com armazenamento na nuvem; Notificações de vencimento de conta enviados no e-mail do cliente se o mesmo possuir um cadastrado; Adicionar teclas de atalho para todos os botões de navegação.

Este trabalho foi muito importante para o meu aprofundamento nos conhecimentos de engenharia de software, passando por cada um dos processos necessários na modelagem e construção de um sistema. Houve uma grande contribuição no que se refere a engenharia de software aprendida no ambiente acadêmico aplicada de forma pratica ao cotidiano de um analista de sistemas formado que estou prestes a me tornar, principalmente o contato direto e constante com o cliente.

REFERÊNCIAS

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 6023:** informação e documentação: Referências: elaboração. Rio de Janeiro, 2002. 24p.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 10520:** informação e documentação: Citações: elaboração. Rio de Janeiro, 2002. 7p.

BEZERRA, Eduardo. **Princípio de Análise e Projeto de Sistemas com UML.** 3ª. ed. São Paulo: Campus, 2002.

CORTES, Pedro Luiz. **Administração de sistemas de informação**. São Paulo: Saraiva, 2008.

GUEDES, Gilleanes T. A. **UML - Uma abordagem prática**. 1ª. ed. São Paulo: Novatec, 2010.

MAGELA, Rogério. **Engenharia de software aplicada**. Rio de Janeiro: Alta Books, 2006.

O'REILLY, Tim. **What Is Web 2.0:** Design Patterns and Business Models for the Next Generation of Software. Blog O'Reilly, 30 set. 2005. Disponível em: <http://www.oreilly.com/pub/a/web2/archive/what-is-web-20.html>. Acesso em: 25 mar. 2016, às 17h01min.

PILONE, Dan; MILES, Russ. **Use a cabeça:** Desenvolvimento de Software. Trad. Aldir José Coelho; Marcelo Santos. São Paulo: Alta Books, 2008.

PRESSMAN, R. S. **Engenharia de Software**. 3ª. ed. São Paulo: Person Makron Books, 2007.

RIOS, Emerson; MOREIRA, Trayahú. **Teste de Software**. 2ª. ed. Rio de Janeiro: Alta Books, 2006.

SANTOS, Andressa Schaurich dos. et al. **A Importância de Sistemas de Informação em Pequenas Empresas:** um Estudo de Caso em uma Agência de Publicidade. Associação Educacional Dom Bosco, 2012. Disponível em:

<http://www.aedb.br/seget/arquivos/artigos12/21616171.pdf>. Acesso em: 26 mar. 2016, às 10h05min.

SCHWABER, Ken; SUTHERLAND, Jeff. **Guia do SCRUM**: Um guia definitivo para o SCRUM. Trad. Fábio Rodrigues Cruz; Rafael Sabbagh. 2013. Disponível em: <http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-Portuguese-BR.pdf>. Acesso em 09 dez. 2016, às 12h31min

SEVERINO, Antônio Joaquim. **Metodologia do trabalho científico**. 23ª. ed. São Paulo: Cortez Editora, 2007.

SILBERSCHATZ, Abraham; KORTH, Henry; SUDARSHAN, S. **Sistema de Banco de Dados** Trad. Daniel Vieira. 5ª. ed. São Paulo: Elsevier, 2006.

SOMMERVILLE, Ian. **Engenharia de Software**. Trad. Profª. Dra. Selma Shin Shimizu Melnikoff; Prof. Dr. Reginaldo Arakaki; Prof. Dr. Edílson de Andrade Barbosa. 8ª.ed. São Paulo: Addison Wesley, 2007.

TURBAN, Efraim. **Tecnologia da Informação para Gestão:** Em Busca de um Melhor Desempenho Estratégico e Operacional 7ª ed. Porto Alegre: Bookman.

WEBER, Kival Chaves; ROCHA, Ana Regina Cavalcanti da; NASCIMENTO, Célia Joseli do. **Qualidade e Produtividade em Software**. 4ª.ed. São Paulo: Makron Books, 2001.