

**INSTITUTO DE DESENVOLVIMENTO EDUCACIONAL DO ALTO
URUGUAI - FACULDADE IDEAU
CURSO DE SISTEMA DE INFORMAÇÃO**

**SISTEMA PARA CONTROLE DE ESTOQUE DE
UMA AUTOPEÇAS INTEGRADO COM A
GERAÇÃO DE O.S. PARA UMA OFICINA
MÊCANICA**

TRABALHO DE CONCLUSÃO DE CURSO II

THIAGO VELEDA IANZER RODRIGUES

**BAGÉ/RS
2022**

THIAGO VELEDA IANZER RODRIGUES

**SISTEMA PARA CONTROLE DE ESTOQUE DE UMA
AUTOPEÇAS INTEGRADO COM A GERAÇÃO DE
O.S. PARA UMA OFICINA MÊCANICA**

Trabalho de Conclusão de Curso de Graduação, do
Curso Superior de Sistemas de Informação, do Instituto
de Desenvolvimento Educacional do Alto Uruguai,
Faculdade IDEAU, Campus Bagé, como requisito
parcial para obtenção do título de Bacharel.

Orientador: Prof. Salvador Loní Tadeo Camargo

**BAGÉ /RS
2022**

**INSTITUTO DE DESENVOLVIMENTO EDUCACIONAL DO ALTO
URUGUAI - FACULDADE IDEAU
CURSO DE SISTEMA DE INFORMAÇÃO**

**A comissão examinadora, abaixo assinada aprova o Trabalho de
Conclusão de Curso**

**SISTEMA PARA CONTROLE DE ESTOQUE DE UMA
AUTOPEÇAS INTEGRADO COM A GERAÇÃO DE
O.S. PARA UMA OFICINA MÊCANICA**

THIAGO VELEDA IANZER RODRIGUES

como requisito parcial para obtenção do grau de

Bacharel em Sistemas de Informação

Aprovado em: __/__/__

COMISSÃO EXAMINADORA:

Orientador Prof. Salvador Loní Tadeo Camargo

Prof. Rafael Rodrigues Bastos

Coordenador do Curso Prof. Marcus Vinícius

**BAGÉ /RS
2022**

AGRADECIMENTOS

À minha mãe, que fez de tudo o que estava ao seu alcance para me auxiliar na caminhada, com apoio, amor, compreensão, carinho e companheirismo.

Ao meu pai, por todo apoio, conselhos, incentivo, inspiração e compreensão durante todo percurso do curso.

À minha avó Antonieta e o meu avô Amintas por todo suporte, carinho e dedicação, os quais sempre me deram forças para superar os desafios.

Aos professores Salvador Loní Tadeo Camargo e Rafael Rodrigues Bastos, por toda colaboração, compreensão, paciência, zelo e dedicação ao repassarem conhecimento durante todo o período curso.

A todos os meus colegas de curso pelo apoio e experiências compartilhadas nesses anos de aprendizado.

Ao meu colega Gabriel pela sua contribuição no trabalho.

A Faculdade IDEAU por todo suporte e por todas as oportunidades oferecidas durante o curso.

*"A maioria dos bons programadores
fazem programação não porque eles
esperam ser pagos ou obter
Adulação pelo público, mas porque
é divertido de programar."*

(Linus Torvalds)

RESUMO

Este trabalho descreve um Sistema de Informação para gestão de estoques e serviços, de uma auto peças e oficina mecânica, e compreende as etapas do desenvolvimento, com o foco no controle de peças de reposição e acessórios para automóveis, bem como, o gerenciamento dos serviços mecânicos de oficina. O estudo foi realizado no estabelecimento que tem o seu enfoque na comercialização de peças e acessórios de automóveis anexo a uma oficina mecânica prestadora de serviços de reparo. O trabalho possui como principal domínio o armazenamento e o tratamento dos dados referentes ao estoque, fornecedores, clientes, veículos e serviços prestados. O armazenamento desses dados é de suma importância para que o estabelecimento tenha um controle adequado do seu estoque de mercadorias, permitindo que os colaboradores possam executar as suas atividades de forma planejada e eficiente. Para atender as premissas em questão foram construídos modelos gráficos em linguagem UML, que servem de base para construir e documentar o sistema, de forma a tornar possível avaliar o comportamento. Após a conceitualização do sistema partiu-se para a fase de construção, a qual compreende a construção do esquema do banco de dados sob o paradigma relacional e implementação do código fonte, na linguagem de programação C++, com suporte do framework Qt. Dentre as principais funcionalidades estão os recursos de armazenamento de transações e consultas a base de dados no tocante a estoques, serviços, lista de fornecedores, clientes e veículos, com opções para o seu gerenciamento, e também geração de relatório de vendas e ordens de serviço, garantindo a tomada de decisão em condições reais e abrangente.

Palavras-chave: Gestão de estoques e serviços. Linguagem C++. Banco de dados.

ABSTRACT

This work describes an Information System for the management of inventories and services of an auto parts and mechanic shop and comprises the stages of development, with a focus on controlling spare parts and accessories for automobiles and on the management of mechanic shop services. The study was carried out in the establishment that focuses on the commercialization of car parts and accessories, attached to a mechanical repair shop. Its main domain is the storage and treatment of stock data, suppliers, customers, vehicles and services provided. The storage of this data is of paramount importance for the establishment to have adequate control of its stock of goods, allowing employees to carry out their activities in a planned and efficient way. To meet the premises in question, graphical models were built in UML language, which serve as a basis for building and documenting the system, to make it possible to evaluate the behavior. After the conceptualization of the system, the construction phase started, which comprises the construction of the database schema under the relational paradigm and implementation of the source code, in the C++ programming language, supported by the Qt framework. The main functionalities are the storage of transactions and consultations to the database regarding stocks, services, list of suppliers, customers and vehicles, with options for their management, and also the generation of sales reports and work orders, guaranteeing the decision-making in real and comprehensive conditions.

Keywords: Inventory and service management. C++ language. Database.

LISTA DE FIGURAS

Figura 1 Classe e objeto.....	17
Figura 2 Interface do Qt Creator.....	25
Figura 3 Tela de inicialização do pgAdmin4.....	25
Figura 4 Prateleiras de peças e acessórios.....	28
Figura 5 Ficha de identificação da peça.....	29
Figura 6 Planilhas.....	29
Figura 7 Ordem de serviço.....	30
Figura 8 Diagrama de Caso de Uso (parte I).....	31
Figura 9 Diagrama de Caso de Uso (parte II).....	32
Figura 10 Diagrama de Classes (parte I).....	34
Figura 11 Diagrama de Classes (parte II).....	35
Figura 12 Diagrama de Classes (parte III).....	36
Figura 13 Diagrama de Classes (parte IV).....	36
Figura 14 Diagrama ER (parte I).....	38
Figura 15 Diagrama ER (parte II).....	38
Figura 16 Estrutura de arquivos.....	40
Figura 17 Tela de login.....	44
Figura 18 Tela de abertura do sistema.....	45
Figura 19 Tela de principal do sistema.....	47
Figura 20 Tela de gestão de usuários.....	50
Figura 21 Tela de gestão de clientes.....	53
Figura 22 Tela de gestão de fornecedores.....	55
Figura 23 Tela de veículos de clientes.....	57
Figura 24 Tela de seleção de marcas e modelos.....	57
Figura 25 Tela de gestão de estoque.....	59
Figura 26 Tela de vendas.....	60
Figura 27 Registrando vendas.....	60
Figura 28 Tela de gestão de vendas.....	61
Figura 29 Relatório de vendas.....	62
Figura 30 Tela de agendamento de serviços.....	63
Figura 31 Tela de Ordem de serviço.....	65
Figura 32 Tela sobre.....	66
Figura 33 Instalador do sistema.....	68

LISTA DE QUADROS

Quadro 1 Versões das ferramentas.....	27
Quadro 2 Classes.....	37
Quadro 3 Tabelas do banco de dados.....	39
Quadro 4 Classe Conexao.....	42
Quadro 5 Abrindo conexão no construtor da classe.....	43
Quadro 6 Fechando conexão no destrutor da classe.....	43
Quadro 7 Operação de login no sistema.....	45
Quadro 8 Trecho de código-fonte para chamada dos formulários.....	47
Quadro 9 Trecho do código-fonte da consulta em linguagem SQL.....	48
Quadro 10 Trecho do código-fonte em linguagem C++ de verificação de acesso.....	49
Quadro 11 Trecho do código-fonte do filtro de registros.....	51
Quadro 12 Código-fonte do método que implementa a integração com o CEP V2.....	53
Quadro 13 Código-fonte da chamada do formulário.....	58
Quadro 14 Atualizando os registros dos serviços.....	63
Quadro 15 Adicionando produtos ao arquivo PDF da O.S.....	65

LISTA DE ABREVIATURAS E SIGLAS

AOO	Análise Orientada a Objetos
API	<i>Application Programming Interface</i>
CPF	Cadastro de Pessoas Físicas
CRUD	<i>Create, Read, Update and Delete</i>
GNU	<i>General Public Licence</i>
HTTP	<i>Hypertext Transfer Protocol</i>
HTTPS	<i>Hypertext Transfer Protocol Secure</i>
IDE	<i>Integrated Development Environment</i>
OO	Orientado a Objetos
O.S.	Ordem
PDF	<i>Portable Document Format</i>
POO	Programação Orientada a Objetos
SGBD	Sistema Gerenciador de Banco de Dados
SQL	<i>Structured Markup Language</i>
SOA	<i>Service-Oriented Architecture</i>
UML	<i>Unified Modeling Language</i>
XML	<i>Extensible Markup Language</i>

SUMÁRIO

1. INTRODUÇÃO	12
1.1 Motivação	12
1.2 Justificativa	13
1.3 Objetivos.....	13
1.3.1 Objetivo geral	13
1.3.2 Objetivos específicos	13
1.4 Organização do trabalho.....	14
2. REFERÊNCIAL TEÓRICO	14
2.1 Gestão de estoque.....	14
2.2 Gestão e tecnologia de informação.....	15
2.3 Ordem de serviço.....	15
2.4 Banco de dados.....	16
2.5 Orientação a objetos.....	16
2.5.1 Princípios básicos da OO.....	18
2.5.1.1 Abstração.....	18
2.5.1.2 Encapsulamento.....	18
2.5.1.3 Herança.....	18
2.5.1.4 Polimorfismo.....	19
2.5.2 Modelagem de software.....	19
2.5.3 Modelagem de um software orientado a objetos.....	19
2.6 UML.....	20
2.7 Web services.....	21
2.8 API.....	21
3 MATERIAIS E MÉTODOS.....	22
3.1 Ferramentas.....	22
3.1.1 Draw.io.....	22
3.1.2 Linguagem C++.....	22
3.1.3 Framework Qt.....	24
3.1.4 Qt Creator.....	24
3.1.5 PostgreSQL.....	25
3.1.6 Brasil API.....	26
3.1.7 Git e GitHub Desktop.....	26
3.1.8 Versões das ferramentas.....	27
3.2 Metodologia.....	27
3.2.1 Domínio contextual.....	27
3.3 Resultados e discussões.....	31
3.3.1 Modelagem do sistema.....	31
3.3.2 Diagramas de Caso de Uso.....	32
3.3.3 Descrição dos Casos de Uso.....	33
3.3.3.1 Logar.....	33
3.3.3.2 Cadastrar e gerenciar usuários.....	33
3.3.3.3 Cadastrar e gerenciar clientes.....	33
3.3.3.4 Cadastrar e gerenciar fornecedores.....	34
3.3.3.5 Cadastrar veículos.....	34
3.3.3.6 Cadastrar e gerenciar estoque de mercadorias.....	34
3.3.3.7 Realizar operações de venda.....	34
3.3.3.8 Gerenciar vendas.....	34

3.3.3.9 Agendar e gerenciar serviços.....	34
3.3.3.10 Cadastrar e gerenciar O.S.....	35
3.3.4 Diagrama de classes.....	35
3.3.5 Diagrama ER.....	38
3.4 Desenvolvimento do sistema.....	40
3.4.1 Estrutura do projeto.....	40
3.4.2 Padrão de nomenclatura de arquivos e variáveis.....	42
3.4.3 Criação do banco de dados.....	42
3.4.4 Classe conexao.....	43
3.4.5 Classes e formulários do sistema.....	44
3.4.5.1 Classe e formulário de login.....	44
3.4.5.2 Classe e formulário principal.....	47
3.4.5.3 Classe e formulário de gestão de usuários.....	50
3.4.5.4 Classe e formulário de gestão e clientes.....	53
3.4.5.5 Classe e formulário de gestão de fornecedores.....	55
3.4.5.6 Classe e formulário de gestão de marcas.....	57
3.4.5.7 Classe e formulário de gestão de modelos.....	57
3.4.5.8 Classe e formulário de gestão de veículos de clientes.....	57
3.4.5.9 Classe e formulário de gestão de estoque.....	59
3.4.5.10 Classe e formulário de nova venda.....	60
3.4.5.11 Classe e formulário de gestão de vendas.....	62
3.4.5.12 Classe e formulário de agendamento de serviços.....	63
3.4.5.13 Classe e formulário de ordem de serviço.....	65
3.4.5.14 Formulário sobre.....	67
3.4.5.15 Instalador do sistema.....	68
4. CONCLUSÃO.....	68
4.1 Trabalhos futuros.....	68
REFERÊNCIAS.....	69

1 INTRODUÇÃO

De modo a atender o mercado empresarial da atualidade, que se caracteriza pela extrema competitividade, é necessário que as empresas absorvam a enorme quantidade de dados geradas a todo instante pelos mais diversos setores que a mesma abrange, tendo em vista, que será a partir deles que serão produzidas as informações de suporte, as decisões pertinentes a gestão e ao desenvolvimento de suas atividades.

Será a partir desses dados armazenados e organizados que a empresa realizará as atividades de forma eficiente, com a finalidade primordial de alcançar êxito e satisfação junto aos seus clientes, corrigindo também os problemas ocasionados por falta de informações cruciais, como: falta de controle no estoque de mercadorias ou a demora no recebimento da encomenda.

Diante da necessidade de acesso a informações precisas e imediatas, faz-se necessário que a empresa invista em soluções que permita sanar a dificuldade, e possa realizar o serviço demandado pelo cliente. Para alcançar esse objetivo a empresa deve aplicar em seu cotidiano soluções e tecnologias que permitam a automatização de suas atividades e ao mesmo tempo auxiliem em sua organização.

Os sistemas de informação vêm ao encontro dessa necessidade de acompanhar as crescentes evoluções tecnológicas no qual o mercado está inserido, sendo através da sua utilização que a empresa consiga melhorar os seus processos de desenvolvimento de atividades e gestão, possibilitando, dessa forma, superar a concorrência.

Buscando atender a necessidade tecnológica atual, este trabalho identifica e investiga as principais causas de ineficiência e também de alguma insatisfação de clientes, com a impossibilidade por vezes de resolver de pronto seus problemas, melhorando o armazenamento dos dados do negócio de forma a propiciar uma visão mais adequada e abrangente.

1.1 Motivação

A motivação veio através da oportunidade de colocar o conhecimento em prática e desenvolver uma solução que permitisse que a empresa pudesse aprimorar o desempenho de suas atividades, refletindo também essas melhoras aos colaboradores e clientes. Por meio de uma análise realizada sobre o negócio, foram detectados diversos pontos que poderiam ser melhorados, sendo possível desenvolver modelos adequados para sanar algumas dificuldades, tais como a automação do registro das operações, ou a eliminação do registro das operações de forma manual.

1.2 Justificativa

Visto a necessidade de melhorar a gestão do estoque de peças e acessórios, bem como, a organização dos serviços a serem realizados, após toda a análise dos aspectos a serem considerados, optou-se pela implementação de um sistema que atendesse os principais requisitos do estabelecimento, atentando-se aos aspectos que envolvem as operações do negócio.

Para que o controle das operações envolvendo as suas mercadorias possam auxiliar os processos do estabelecimento, foram identificadas uma série de processos que podem ser automatizados, como: cadastro de clientes, fornecedores e veículos, cadastro de peças automotivas, controle de vendas, agendamento de serviços e armazenamento de dados dos serviços realizados e geração de OS (ordem de serviço)

Para integrar as operações da empresa a um sistema, foi concebida a implementação de um sistema que possa auxiliar no gerenciamento dessas informações, propiciando ao estabelecimento um meio de realizar o controle das suas atividades diárias de uma maneira mais eficiente, agilizando os processos, evitando erros e permitindo que os processos internos sejam aperfeiçoados, trazendo também a melhora dos serviços oferecidos aos seus clientes, podendo atendê-los de uma forma mais organizada e precisa.

1.3 Objetivos

1.3.1 Objetivo Geral

Construir um Sistema de Informação para gestão de estoques de uma auto peças integrado com ordem de serviço.

1.3.2 Objetivos Específicos

- Manter o cadastro dos veículos com acesso e aquisição sob demanda;
- Manter cadastro do cliente com individualização dos veículos;
- Cadastro da O.S.;
- Agenda de serviços da oficina;
- Cadastro de fornecedores;
- Registro de observações do veículo, como arranhões, amassados e outras partes danificadas.

1.4 Organização do trabalho

Este trabalho está organizado em capítulos, onde são apresentados os seus objetivos e fases de desenvolvimento do projeto, permitindo o acesso de forma objetiva aos seus dados iniciais, seus materiais e métodos, abordando os conceitos e tecnologias utilizadas para o desenvolvimento, sendo relatado ao final a conclusão alcançada expondo as considerações finais.

No capítulo 2 será apresentado o referencial teórico do trabalho, com foco nos principais conceitos utilizados para o desenvolvimento. No capítulo 3 contém os materiais e o métodos utilizados para o desenvolvimento do sistema. Por fim, o capítulo 4 apresenta a conclusão do trabalho. Após são apresentadas as referências bibliográficas utilizadas como base para os textos.

2 REFERENCIAL TEÓRICO

2.1 Gestão de estoque

De uma forma geral, conforme diz Ballou (2006), estoques podem ser matérias primas, suprimentos, componentes, materiais em processo e produtos acumulados em diversos pontos do processo produtivo e da cadeia logística das empresas. Para Chopra e Meindl (2004), o estoque pode ser entendido como um dos principais fatores geradores de custos em uma cadeia de suprimentos. Caracterizando-se com afirmações de que os estoques cíclicos existem, pois a produção ou compra em grandes lotes permite que sejam obtidas economias de escala.

Segundo Borges et. al (2010), um bom gerenciamento de estoques garante a minimização de investimentos que eventualmente podem ficar imobilizados por períodos de tempo que excedem prazos razoáveis, o que repercute no custo do capital e virtualmente em comprometimento do fluxo de caixa. Para o autor, as empresas trabalham com estoques de diferentes tipos e necessidades desde administração, o que impõe necessidade de atenção constante dada sua repercussão.

Wernke et al. (2011), narra que o acirramento da competição mercadológica induz os administradores a buscar uma produtividade maior dos ativos a disposição, buscando sempre o equilíbrio entre eficiência e eficácia, para que uma empresa possa ser competitiva de maneira constante. Cita que uma estrutura enxuta permite produzir mais com aplicação de menos recursos, sendo esse fato um poderoso diferencial competitivo. Um conceito importante para o gestor no tocante ao uso adequado de recursos é de que se tentar continuamente fazer menos com menos acabará fazendo nada com nada.

Por isso, torna-se muito válida a ideia de Taylor (2006), em que salienta o grau de relevância para a saúde da organização que uma boa gestão de estoques minimiza as incertezas

econômico-financeiras dado que, estes aspectos são extremamente dinâmicos e de alta instabilidade.

2.2 Gestão e tecnologia de informação

Os Sistemas de Informação são mecanismos cuja função é coletar, guardar e distribuir informações para suportar as funções gerenciais e operacionais das organizações. Um Sistema de Informação abrange entradas, saídas e transformação através do processamento, que dentro da dinâmica de uma organização encadeia-se em um fluxo cíclico, recebendo a contribuição de diferentes usuários.

Para Will e Ross (2006), a informação e a TI estão entre os principais ativos de uma organização e são os menos entendidos, sendo também áreas com crescimento em investimentos e evidências, e serem um segmento que possibilita aumento de lucros das organizações. Para os autores as empresas que realmente utilizam a TI de uma forma correta, possuindo conhecimento na área são 20% mais lucrativa do que seus concorrentes, sendo os Sistemas de Informações considerados como componentes fundamentais para o desempenho da organização.

Para uma gestão de estoque eficiente é de primordial importância o gerenciamento da informação. Dentre as vantagens decorrentes da gestão e do compartilhamento da informação, destacam a redução do custo de processamento de pedidos e do nível de estoque. Todo sistema, usando ou não recursos da tecnologia de informação, que manipula e gera informação, podem ser genericamente considerados sistema de informação.

2.3 Ordem de serviço

O controle das atividades de reparos de terceiros se dá através de ordens de serviço, que é o instrumento pelo qual se determina ou regula procedimentos para a execução de serviços, fixa comandos de trabalho, imposições de cunho administrativo específicas e relativas pessoa. O sistema de ordem de serviço tem como objetivo gerenciar as informações geradas no dia a dia dos funcionários em forma de apontamento de serviço. Também arquiva todas as informações dos serviços que estão sendo executados, encerrados ou interrompidos na empresa.

Segundo Rezende e Abreu (2001), relata que toda empresa preocupada com a sua perenidade e a sua competitividade deve focar seus esforços na atuação e organização das atividades e armazenamento de todas as etapas dos serviços e materiais utilizados, bem como, no planejamento de sistemas de informação com foco na gestão de tais informações.

Na ordem de serviço devem ser registrados a descrição do serviço executado, os materiais utilizados em tal serviço e as observações pertinentes a execução do serviço. A O.S é uma solicitação de uma atividade de serviços a ser executada em um objeto de manutenção, em uma sociedade do cliente em uma determinada data. É utilizada para documentação de acompanhamento das atividades de um determinado serviço prestado.

2.4 Banco de Dados

Segundo Heuser (2009), banco de dados é o conjunto de dados integrados que tem como objetivo atender a uma comunidade de usuários. É o conjunto de arquivos integrados que atendem a um conjunto de sistemas, esse compartilhamento tem como objetivo evitar a redundância não controlada de informações, onde nessa forma de processamento cada informações é armazenada uma única vez, sendo acessado pelos vários sistemas que dela necessitam.

Heuser (2009) conceitua SGBD como software que incorpora as funções de definição, recuperação e alteração de dados em um banco de dados. Ainda conceitua modelo de dados como a descrição formal da estrutura de um banco de dados.

O modelo de banco de dados é a descrição formal da estrutura de um banco de dados, normalmente são considerados dois níveis de abstração de modelos de dados:

- 1) Modelo conceitual: descreve a estrutura de um banco de dados de forma independente de um SGBD particular. A técnica dessa modelagem conceitual é abordagem entidade-relacionamento (ER), onde nesta técnica, um modelo conceitual é usualmente representado através de um diagrama de entidade-relacionamento (DER);
- 2) Modelo lógico: é uma descrição de um banco de dados no nível de abstração, visto pelo usuário do SGBD, assim este modelo é dependente do tipo particular de SGBD que está sendo usado.

O projeto de um banco de dados ocorre em duas fases:

- 1) Modelagem conceitual: nesta fase é construído um modelo conceitual, na forma de um diagrama ER, este modelo captura as necessidades da organização em termos de armazenamento de dados de forma independente de implementação;
- 2) Projeto lógico: está etapa objetiva transformar o modelo conceitual obtido na primeira fase em um modelo lógico. O modelo lógico definido como o banco de dados será implementado em um SGBD específico. O projeto lógico é adequado para a construção de um novo banco de dados.

2.5 Orientação a objetos

Para Farinelli (2007), a orientação a objetos é uma tecnologia que enxerga os sistemas como sendo coleção de objetos integrantes. Ela permite melhorar a reusabilidade e a extensibilidade do software. Tal tecnologia é fundamentada no que chamamos de modelos de objetos, que engloba os princípios da abstração, hierarquização, encapsulamento, classificação, modularização, relacionamento, simultaneidade e persistência.

A proposta da orientação a objetos é representar o mais fielmente possível as situações do mundo real nos sistemas computacionais. Consiste em considerar os sistemas computacionais não como uma coleção estruturada de processos, mas sim como uma coleção de objetos que interagem entre si.

Para Farinelli (2007), os programas orientados a objetos são programas estruturados em módulos que agrupam um estado e operações sob esse estado. Apresentam ênfase em reutilização de código. Um dos grandes diferenciais de tal programação em relação a outros paradigmas de programação, que também permitem a definição de estruturas e operações sobre essas estruturas, está no conceito de herança, mecanismo através do qual definições existentes podem ser facilmente estendidas.

Os principais conceitos de POO são:

a) Objetos: representa um determinado elemento do mundo real. É uma entidade do mundo real que merece representação para o ambiente estudado. As características que descrevem um objeto são chamadas de atributos, e as ações que um objeto pode executar são chamados de métodos ou serviços. Chamamos de interface o conjunto de métodos disponíveis em um objeto;

b) Classes: uma classe representa o conjunto de objetos que possuem características e comportamentos comuns. Uma classe representa um gabarito para muitos objetos e descreve como esses objetos estão estruturados internamente.

Conforme Farinelli (2007), na Figura 1 é apresentado um exemplo de classes e objetos.

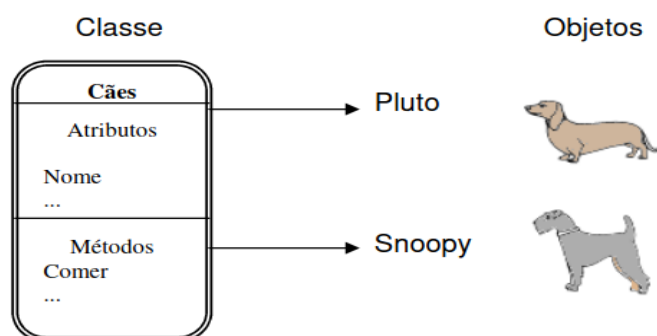


Figura 1 – Classe e objeto

2.5.1 Princípios básicos da OO

Farinelli (2007), relata que os princípios básicos da OO são: abstração, encapsulamento, herança e polimorfismo.

2.5.1.1 Abstração

Trata-se do princípio pelo qual cada componente deve manter oculta sob sua guarda uma decisão de projeto única. Para utilização desse componente, apenas o mínimo necessário para a sua operação deve ser revelado.

Através do princípio da abstração isolamos os objetos que queremos observar do ambiente complexo em que se situam, e nesses objetos representamos somente as características que são relevantes para o problema em questão.

O uso da abstração permite que determinadas partes do problema ganhem maior ou menor peso, e dessa maneira detalhes podem ser desconsiderados em determinados momentos para que outros sejam ressaltados.

2.5.1.2 Encapsulamento

É o princípio de projeto pelo qual cada componente de um programa deve agregar toda a informação relevante para a sua manipulação como uma unidade. Aliado ao conceito de ocultamento de informação, é um poderoso mecanismo da POO.

O encapsulamento é um dos grandes trunfos da POO em relação a programação tradicional, sendo que a sua vantagem é que disponibiliza o objeto com toda sua funcionalidade sem que seja preciso saber como ele funciona internamente e nem como é armazenado internamente os dados recuperados.

2.5.1.3 Herança

É o mecanismo pelo qual uma classe obtém as características e métodos de outra para expandi-la ou especializá-la de alguma forma, ou seja, uma classe pode herdar características, métodos e atributos de outras classes.

Da mesma forma uma classe transmite suas características para outras classes, tornando aquelas que recebem suas características suas herdeiras. Herança significa que todos os atributos

e métodos programados no ancestral já estarão automaticamente presentes em seus descendentes sem necessidade de reescrevê-los.

Temos como forma de herança:

- a) Herança Simples: denominada simples quando uma classe herda características simples de uma superclasse;
- b) Herança Múltipla: quando uma classe herda características de duas ou mais superclasses.

2.5.1.4 Polimorfismo

É a capacidade de uma variável se referir em tempo de execução a objetos de diversas classes. É o nome dado a capacidade que objetos diferentes possuem de responder a uma mesma mensagem. Uma mesma mensagem pode apresentar formas de execução diferentes, próprias de cada objeto.

O usuário pode enviar uma mensagem genérica e abandonar detalhes sobre a exata implementação sobre o objeto receptor. O polimorfismo ocorre quando um método que já foi definido no ancestral é redefinido no descendente com comportamento diferente.

2.5.2 Modelagem de software

Deitel (2006), relata que para criar soluções avançadas, deve seguir-se um processo detalhado para analisar os requisitos do projeto, isto é, determinar o que o sistema deve fazer e desenvolver um design que atenda esses requisitos, ou seja, decidir como o sistema deve fazê-lo. Idealmente, os passos do processo de projeto em sequência são: modela-se e revisa-se o design antes de escrever qualquer código.

Se o paradigma em uso for o de orientação a objetos, então a análise, projeto e implementação devem ser consistentemente aplicados. A importância de usar-se o paradigma OO permite escalabilidade da solução tanto a nível de tecnologia quanto de recursos humanos.

2.5.3 Modelagem de um software orientado a objetos

Segundo Farinelli (2007), a modelagem baseada em objetos é um meio para se descrever os sistemas do mundo real. Um objeto representa uma abstração de uma entidade do mundo real, combinando, em um mesmo elemento informação e comportamento.

Uma vantagem da modelagem orientada a objeto é o fato de que um mesmo objeto concebido em fase de análise, passa com as mesmas características desde o usuário até o programador que será responsável pela sua codificação final.

Os passos básicos a serem observados para o desenvolvimento de um modelo orientado a objeto, normalmente compreendidos são:

- a) Identificar as classes e os objetos;
- b) Identificar as estruturas e os relacionamentos entre os objetos;
- c) Identificar os atributos e os métodos importantes.

O foco principal da análise orientada a objetos não está centrado na estrutura ou nos dados do sistema, mas sim na composição das classes e objetos do sistema.

2.6 UML

Segundo Pereira (2011), a UML foi concebida com intuito de estabelecer um padrão único a ser usado para especificação das características de sistemas computacionais projetados para atender as necessidades desses sistemas.

A UML serve para construir modelos concisos, precisos, completos e sem ambiguidades, tendo de maneira geral, as seguintes características:

- a) Modela os aspectos estruturais e comportamentais do sistema, ou seja, pode especificar os conceitos do negócio, seus relacionamentos e os estados, sequências de atividades e de colaborações;
- b) Provê uma linguagem que permite o entendimento e a utilização por humanos e a leitura por máquinas;
- c) Provê elementos de notação para modelar todos os tipos de sistemas de computação;
- d) Permite a modelagem do conceito ao artefato executado, utilizando técnicas orientadas a objetos;
- e) Embora a especificação já contenha elementos de notação que permitem o atendimento a um grande número de situações e propósitos, a linguagem é extensível e adaptável a necessidades específicas;
- f) Contempla as necessidades de produção de modelos pequenos e simples a grandes e complexos;
- g) Modela processos manuais ou automatizados, independente da tecnologia que usa;
- h) É uma linguagem para visualização do modelo, facilitando o entendimento pelas equipes de análise de negócio, desenvolvimento de sistemas e pelos clientes;

- i) Serve para construir código de computador, embora não seja uma linguagem de programação de computadores;
- j) A UML é o padrão de fato usado em análise e projeto de sistema de informática orientado a objetos.

Conforme Pereira (2011), a UML, sendo uma linguagem gráfica, conta ainda com a facilidade de emprego, pois os elementos da notação são símbolos gráficos que podem ser compostos com a ajuda de ferramentas gráficas interativas que garantem a corretude e a consistência do modelo.

2.7 Web services

Web services são Sistemas de Informação que utilizando um conjunto de tecnologias que são padrão de internet, permitem a integração entre outros sistemas de informação, através de troca de mensagens. O funcionamento da Web Services tem como base os padrões *eXtensible Markup Language* (XML) e *Simple Object Access Protocol* (SOAP), o transporte dos dados é realizado através do protocolo HTTP ou HTTPS.

Conforme Cruz (2010), os dados com operações, mensagens, parâmetros, dentre outros, são transferidos no formato XML, os quais são encapsulados pelo protocolo SOAP e descritos através da linguagem *Web Services Description Language* (WSDL). O protocolo *Universal Description, Discovery and Integration* (UDDI) é utilizado para os processos de publicação, pesquisa e descoberta de *Web Services*. Para o autor, *Web Services* são diversas tecnologias fundamentais para um SOA e sendo muito utilizadas para o desenvolvimento de aplicações pelo fato de obterem respostas positivas, adaptáveis e flexíveis.

2.8 API

Para Jacobson et al. (2011), existem duas ramificações para definição de API: uma técnica e uma de negócios. A definição técnica refere-se a um sistema que proporciona uma maneira para que duas aplicações computacionais possam se comunicar entre si. Estas usam um meio de comunicação comum, normalmente a internet, para a troca de informações, proporcionando, deste modo, uma linguagem na qual ambos os sistemas conectados possam entender. Uma API tem a função de promover uma interface de comunicação entre diferentes sistemas através do fornecimento de serviços estabelecidos por padrões ou regras de programação definidos pelo fornecedor da API.

No que se refere a área de negócios API, é uma ponta que liga desenvolvedores e usuários aos serviços promovidos por uma empresa, e estes serviços são disponibilizados via internet por

um provedor, possibilitando, deste modo, o desenvolvimento de aplicativos a partir destes serviços.

Podemos definir dois tipos distintos de APIs: públicas e privadas, onde neste contexto consideramos as APIs privadas mais importantes. As APIs públicas se tornam uma peça importante para qualquer aplicação da Web.

Para Jacobson et al. (2011), uma API pública pode ser definida em resumo como aquela que é exposta para um conjunto de pessoas, tendo pouca ou nenhuma relação contratual entre as partes.

Uma API privada tem como característica ser um conjunto de funcionalidades e serviços que são utilizadas apenas dentro das corporações como auxílio, na criação de seus produtos ou execução de operações.

De acordo Jacobson et al. (2011), API privada providencia uma interface independente de linguagem que é disponibilizada através de protocolos web e tem seu acesso limitado a apenas um conjunto específico de desenvolvedores ou organizações, o que faz com que não seja comercializado para o público geral, restringindo o acesso as funcionalidades da API.

3 MATERIAIS E MÉTODOS UTILIZADOS

A seguir serão descritos as ferramentas e os métodos que foram utilizados para a modelagem e desenvolvimento do sistema.

3.1 Ferramentas

3.1.1 Draw.io

Os modelos do sistema foram construídos utilizando a ferramenta Draw.io, que permite criação de diagramas como fluxogramas, *wireframes*, organogramas e diagramas UML, bem como, diversos outros tipos de diagramas de sistemas (DIAGRAMS.NET, 2022).

O Draw.io é de licença gratuita e possui versões para web e desktop, possuindo ainda recursos para a criação de diversos tipos de diagramas e até mesmo protótipos de diferentes níveis. A ferramenta foi utilizada para o desenvolvimento dos diagramas de casos de uso e de classes.

3.1.2 Linguagem C++

Segundo Bjarne Stroustrup (2014), C++ é uma linguagem compilada. Para que um programa seja executado, seu texto fonte deve ser processado por um compilador, produzindo

arquivos de objeto, que são combinados por um *linker* produzindo um programa executável. Um programa C++ normalmente consiste de muitos arquivos de código-fonte (geralmente chamados simplesmente de arquivos-fonte).

Um programa executável é criado para uma combinação específica de hardware/sistema, não é portátil, digamos, de um Mac para um PC com Windows. Quando falamos de portabilidade de programas C++, geralmente queremos dizer portabilidade de código-fonte, ou seja, o código-fonte pode ser compilado e executado com êxito em vários sistemas.

Enfatizando a qualidade da linguagem C++ destacam-se três características diferenciadas:

- a) Programa fonte;
- b) Programa objeto compilado;
- c) Programa executável resultado da linkedição do código objeto, onde está agregado a parte de *run-time* da aplicação.

A linguagem C++ foi uma evolução da linguagem C desenvolvido sob os paradigmas estruturado e funcional. O C possui uma menor abstração e ferramentas práticas do que o C++, visto que, não utiliza os benefícios do paradigma orientado a objetos.

Os componentes da biblioteca padrão são códigos C++ perfeitamente comuns, fornecidos por cada implementação C++, ou seja, a biblioteca padrão C++ pode ser implementada no próprio C++, e é com usos muito menores de máquinas código para coisas como troca de contexto de thread. Isso implica que C++ é suficientemente expressivo e eficiente para as tarefas de programação de sistemas mais exigentes.

O C++ é uma linguagem de tipagem estática, ou seja, o tipo de cada entidade (por exemplo, objeto, valor, nome e expressão) deve ser conhecido pelo compilador em seu ponto de uso. O tipo de um objeto determina o conjunto de operações que lhe são aplicáveis.

Para Loureiro (2019), O C++ é uma linguagem que permite a automatização de tarefas que envolvem objetos, sendo um objeto uma combinação de código e dados que pode ser manipulada como uma unidade. Exemplos de objetos são: ficheiros, diretórios, janelas de linha de comandos, imagens e controles típicos de interface de utilizador (tais como: botões, caixas de seleção e grupos de opções).

A escolha do C++ deu-se pela familiaridade com a linguagem, levando os benefícios de sua robustez, estabilidade e desempenho. O C++ pode ser considerada uma linguagem que se situa entre o alto e o baixo, sendo multiparadigma, com ênfase na orientação a objetos a tecnologia de ampla aceitação na comunidade tecno-científica, e possuindo diversas bibliotecas com recursos que permitem abstrações para facilitar os desenvolvedores, mas também mantém as características

de uma linguagem de baixo nível que permite que se tenha um maior controle do hardware da máquina e por consequência disso um desempenho excepcional.

3.1.3 Framework Qt

As tecnologias de desenvolvimento mais modernas contam com um framework, que são um conjunto de bibliotecas para uma linguagem em específico que trazem diversos recursos dos mais variados tipos de dados. Cada framework possui uma finalidade específica.

A escolha do Qt foi também uma consequência do C++, ele permite o desenvolvimento de aplicações multiplataforma com interface gráfica de usuário, fornecendo diversas bibliotecas, elementos de interface e recursos, que permitem o desenvolvimento de aplicações de alto nível, i.e. soluções não a nível primário.

A portabilidade de aplicações é uma das principais vantagens do Qt, pois é possível que aplicações desenvolvidas em uma plataforma funcione em outra, bastando para isso recompilar a aplicação.

3.1.4 Qt Creator

O Qt Creator é uma IDE multiplataforma de desenvolvimento de software que utiliza o framework Qt, sendo compatível com os sistemas operacionais Windows, Linux e macOS. A IDE possui um editor de código avançado que conta com diversas ferramentas, sendo compatível com uma série de linguagens de programação como: Python, Javascript dentre outras, permitindo a construção de softwares nas plataformas desktop, móveis e embarcadas (QT COMPANY, 2022).

Na Figura 2 é possível visualizar a interface do Qt Creator, no editor de texto encontra-se um trecho de código de uma função de validação de CPF, em linguagem C++, do sistema.

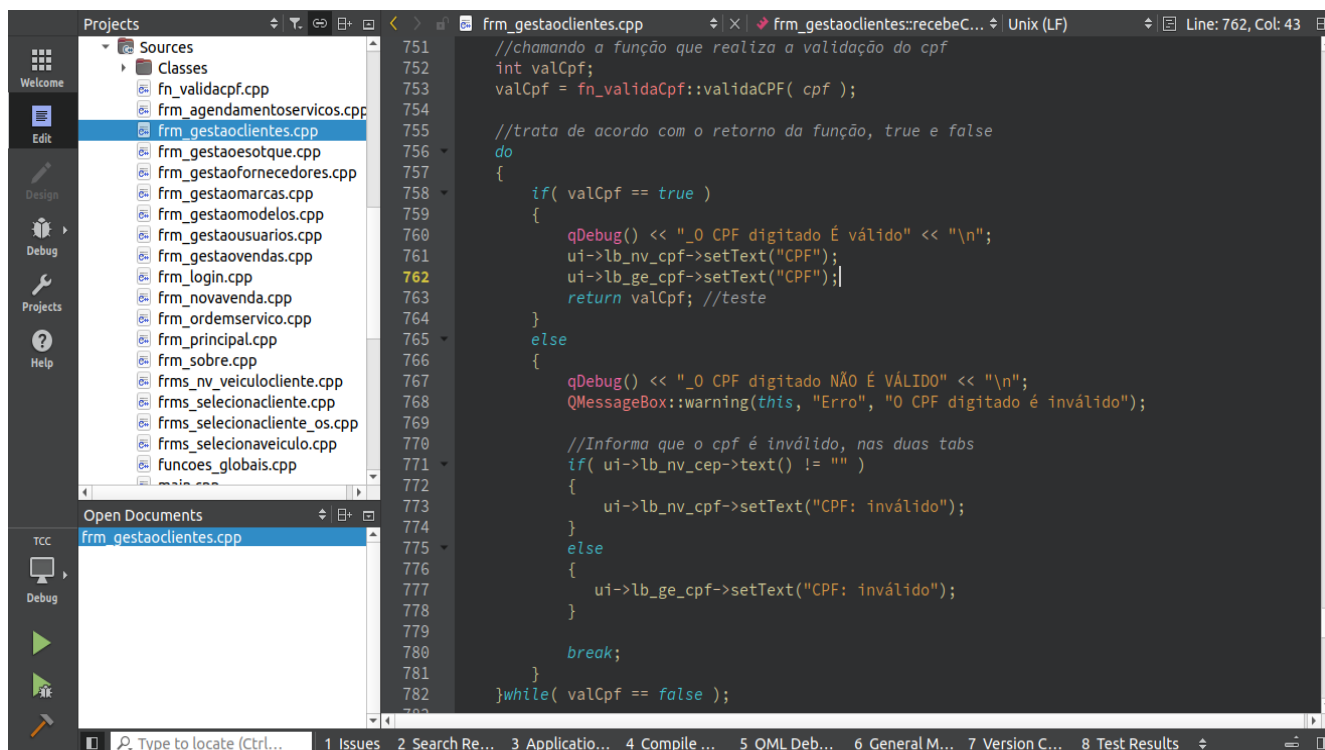


Figura 2 – Interface do Qt Creator

3.1.5 PostgreSQL

O PostgreSQL é um sistema de bancos de dados relacional. Utiliza a linguagem SQL combinada com diversos recursos que armazenam e dimensionam com segurança as cargas de trabalho de dados mais complexas (POSTGRES GLOBAL DEVELOPMENT GROUP, 2022).

A escolha desse banco para o desenvolvimento do sistema foi devido a familiaridade e experiência na sua utilização em outros projetos e compatibilidade com as bibliotecas do QT. Foi utilizado em conjunto com o pgAdmin4, ferramenta que permite o acesso a diversos recursos do PostgreSQL através de uma interface gráfica, comportando diversos scripts SQL prontos para funcionalidades importantes do banco, destacando-se pela capacidade de geração de Diagramas ER a partir do banco de dados desenvolvido.

A Figura 3 apresenta a tela de inicialização do pgAdmin4:



Figura 3 - Tela de inicialização do pgAdmin4

3.1.6 Brasil API

A Brasil API é um projeto experimental que tem como objetivo centralizar e disponibilizar *endpoints* modernos com baixíssima latência utilizando tecnologias como *Vercel Smart CDN* responsável por fazer o cache das informações em atualmente 23 regiões distribuídas ao longo do mundo (Brasil API, 2022).

Foram utilizados os serviços CEP V2 e CNPJ, disponibilizados para consulta e validação de CEP e CNPJ. O CEP V2 permite validar o CEP e buscar informações de localidade, auxiliando no cadastro de clientes, onde foi facilitado o cadastro de clientes, buscando os seus principais dados de forma automática. O serviço CNPJ permite validar o CNPJ e buscar muitas informações da empresa, através desse serviço foi facilitado o cadastro de fornecedores, buscando os seus principais dados de forma automática.

3.1.7 Git e GitHub Desktop

O Git é um sistema de controle de versões distribuído (DVCS) distribuído gratuito e de código aberto, ele permite o versionamento de projetos de diferentes escalas e tecnologias de uma forma prática e eficiente. Tem como principal função registrar alterações feitas em cima de um código, armazenando todas as suas informações e possibilitando, se necessário, regredir a versões anteriores dos arquivos de uma aplicação, de forma simples e rápida (GIT, 2022).

O GitHub Desktop é um aplicativo multiplataforma que tem como função fornecer uma interface gráfica para o Git, e permite a interação com a plataforma GitHub. Através dessa ferramenta é possível utilizar as principais funções do Git de uma maneira simples, fornecendo uma interface de visualização para as alterações realizadas no código de projetos, com acesso ao histórico de modificações, tudo isso funcionando integrada com a plataforma GitHub, que permite hospedar repositórios de projetos privados ou públicos, trazendo a possibilidade de interação com a comunidade de desenvolvedores que utilizam a plataforma (GitHub 2022).

A ferramenta tornou-se de vital importância para o desenvolvimento do trabalho, pois o versionamento de todo o código fonte do projeto é realizado através dela, contribuindo para uma maior organização e segurança, contando com todo o histórico de suas modificações e também permitindo reverter alterações que ocasionam erros no sistema.

Destaca-se outro fator importante, que é a hospedagem dos repositórios do projeto na plataforma GitHub, possibilitando ter uma cópia do código fonte acessível de qualquer dispositivo, e também a disponibilização do projeto, visto que se encontra sob licença GNU (*General Public License*), tornando a disponibilização do seu código fonte obrigatório.

3.1.8 Versões das ferramentas

As versões das ferramentas utilizadas para o desenvolvimento do trabalho estão descritas no Quadro 1.

Quadro 1 – Versões das ferramentas

Ferramenta	Versão
Draw.io	13.9.9
C++	17
Qt Creator	6.2.2
PostgreSQL	13
Brasil API	1.0.0
Git	2.25.1
GitHub Desktop	3.0

Fonte: primária

3.2 Metodologia

Para a realização deste projeto foram realizadas análises no ambiente do estabelecimento e entrevistas com os proprietários, de forma a entender as regras de negócio do estabelecimento e também levantar e definir os requisitos para o sistema.

Para obter uma maior compreensão do tema, foram realizadas pesquisas bibliográficas referentes a sistemas de gestão de estoque, permitindo uma maior noção das variáveis e por menores que envolvem esse tipo de sistema.

Definidos os requisitos, foi possível estabelecer as funcionalidades que o sistema deveria ter, sempre respeitando as regras de negócio do estabelecimento. Com as funcionalidades definidas partiu-se para a fase de modelagem conceitual do sistema.

3.2.1 Domínio contextual

A organização física do estoque é dividida em prateleiras que possuem divisórias, permitindo, desta forma, que as peças sejam armazenadas e separadas por grupos de modelos semelhantes. As prateleiras são identificadas por grupos, sendo estes devidamente identificados por letras. As divisórias são numeradas de forma crescente da esquerda para a direita, com o intuito de facilitar a localização de cada peça.

Referente aos blocos das prateleiras, estes são identificados por uma etiqueta com a sua letra correspondente e suas divisórias por números, como por exemplo A001, onde a letra indica a prateleira e o número identifica uma divisória na prateleira, permitindo que as peças sejam armazenadas e localizadas de forma prática. Em cada divisória poderá ser armazenada uma ou várias peças, normalmente sendo armazenadas conforme a marca e modelo dos veículos a que se destinam, contudo, podem ser armazenadas peças de marcas diferentes, porém do mesmo tipo.

O estabelecimento opera como oficina mecânica e auto peças multimarcas, onde as peças podem ser vendidas ao público em geral ou utilizadas nos serviços prestados aos clientes da oficina. Importante ressaltar que a empresa só trabalha com peças originais ou paralelas de primeira linha, sendo as peças identificadas pelo código fornecido pelas montadoras e indústrias. As peças são diversificadas e possuem diferentes aplicações, requerendo uma maior atenção, pois apesar de serem nomeadas, existem várias peças com o mesmo nome.

As peças são identificadas através das fichas que possuem as informações conforme descritas em suas embalagens. Dentre as informações principais temos: o nome da peça, o seu código, os modelos, marcas e anos inicial e final dos veículos em que podem ser aplicadas.

É necessário ter atenção, pois diferentes peças com o mesmo nome são destinadas a veículos de anos de fabricação diferentes, e caso a identificação não esteja clara pode ocasionar erros durante a venda e na aplicação das mesmas nos serviços.

Para elucidar de forma mais clara o descrito acima foi incluso imagens referentes às prateleiras e fichas das peças, como demonstrado na Figura 4 e Figura 5.

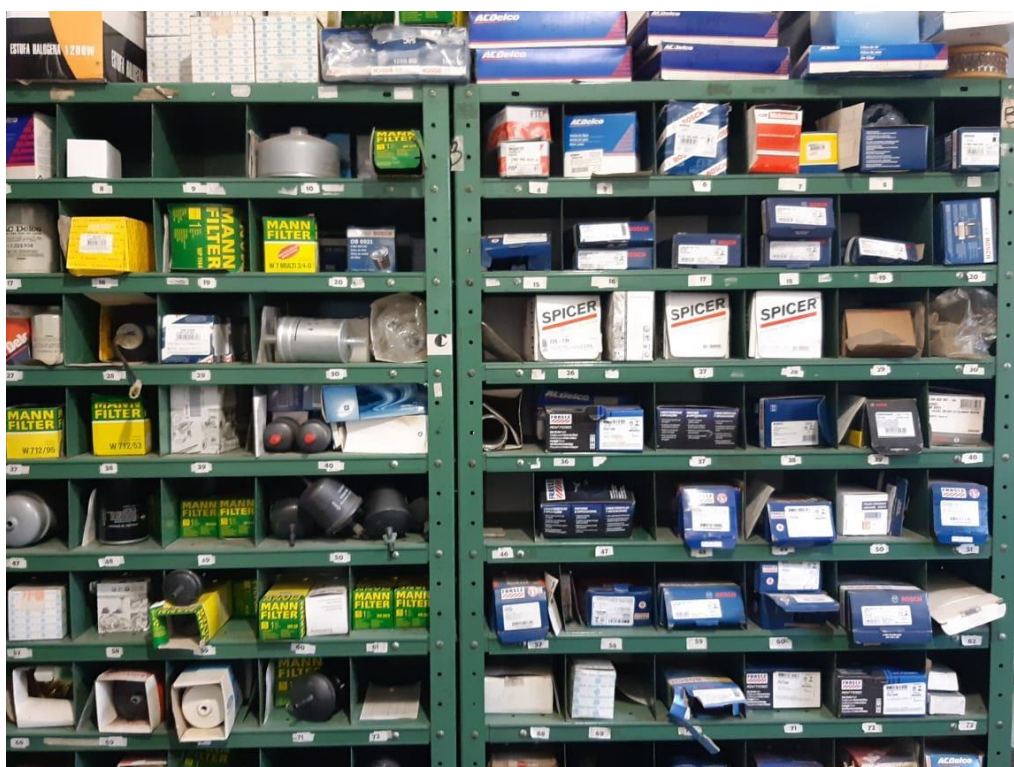


Figura 4 – Prateleiras de peças e acessórios

[illegible]

Figura 5 – Ficha de identificação da peça

Esta ficha de identificação da peça em função da instalação do sistema tornou-se obsoleta e representa uma sensível redução de trabalho já que os registros eram feitos todos de forma manual.

O estabelecimento utiliza planilhas, onde são armazenados os dados de locação da peça, código da peça, descrição da peça, quantidade da peça, custo da peça e o valor total das peças, com a finalidade de realizar a gestão de seu estoque de peças. A Figura 6 apresenta as planilhas utilizadas pelo estabelecimento.

	A	B	C	D	E	F	G
39	A006	93213067	PIVO INFERIOR ACD20 PS-707	2	75,73		=(E39*F39)
40	A007	52252687	TERMINAL DIANTEIROSQ.(N365)	1	23,92		=(E40*F40)
41	A007	7324540	TERMINAL DIR.INTERNO N317	2	7,58		=(E41*F41)
42	A008	52252688	TERMINAL DIANTEIRO DIR.(N364)	1	24,82		=(E42*F42)
43	A010	52259125	PIVO SUSPENSÃO SUPL.(n318)	4	12,08		=(E43*F43)
44	A011	90223846	PIVO ESQUERDO OMEGA N376	1	37,88		=(E44*F44)
45	A012	90334035	TERMINAL ESQ. OMEGA(N373)	1	13,43		=(E45*F45)
46	A013	7329356	PIVÔ ARTICULADOR CHEVETTE N3005	2	6,56		=(E46*F46)
47	A015	93314496	KIT JUNTA ESFERICA INF.	2	34		=(E47*F47)
48	A016	90373519	TERMINAL ESQ.(N387)	2	14,08		=(E48*F48)
49	A016	90373520	TERMINAL DIREÇÃO LD(N386)	1	14,66		=(E49*F49)
50	A017	90334032	TERMINAL DIREÇÃO ÔMEGA(372)	1	17,93		=(E50*F50)
51	A017	90334031	TERMINAL DIREÇÃO ÔMEGA(N374)	1	17,83		=(E51*F51)
52	A018	90092492	BRAÇO AUX.OMEGA 4CIL	1	33,02		=(E52*F52)
53	A018	90343907	ALGEMA	1	29,89		=(E53*F53)
54	A019	90495045	BIELETA DA BARRA ESTABILIZADORA	4	9,19		=(E54*F54)
55	A020	90334075	TIRANTE DO ESTABILIZADOR (N370)	2	23,13		=(E55*F55)
56	A021	N3045	ARTICULADOR PICK-UP CORSA 95/98	1	11,92		=(E56*F56)
57	A022	37902743-1	BIELETA BARRA ESTABILIZADORA DAINT	2	3,81		=(E57*F57)
58	A023	ZBA407365	ARTICULADOR TRANS.PASSAT N135	1	20		=(E58*F58)
59	A024	823407365E	ARTICULADOR	4	24		=(E59*F59)
60	A025	325407365-2	ARTICULADOR I.F. N187	2	20,29		=(E60*F60)

Figura 6 – Planilhas

O estabelecimento além de possuir a modalidade de venda de peças diretamente ao consumidor, utiliza as peças na prestação de serviços nos automóveis dos clientes da oficina, onde os dados são transcritos em uma ordem de serviço com detalhamento das peças e dos serviços a serem realizados, bem como, a descrição dos veículos e proprietários dos mesmos. A Figura 7 contém um exemplo do modelo de ordem de serviço do estabelecimento.

[illegible]

Figura 7 – Ordem de serviço

3.3 Resultados e discussões

As principais funções do sistema centram-se na gestão do estoque de mercadorias e geração de ordens de serviço. Para atender essas necessidades o sistema tem a tarefa de controlar as atividades de cadastro e gerenciamento das mercadorias do estoque, incluindo os seus fornecedores e veículos para as quais estão destinadas.

Com isso é possível realizar operações de vendas e geração de ordens de serviço de uma forma que os requisitos solicitados sejam devidamente atendidos, armazenando as informações das vendas e serviços realizados, podendo emitir relatórios sobre essas operações. O sistema também possibilita agendar e gerenciar serviços, possibilitando uma maior organização das atividades da empresa.

3.3.1 Modelagem do sistema

Como um meio de documentação e auxílio para entender melhor os requisitos do sistema, são desenvolvidos modelos que permitem ter uma noção do funcionamento do mesmo, demonstrando os principais dados tratados, de forma a facilitar a implementação e para isso são utilizadas técnicas de modelagem de software na construção de diagramas, utilizando ferramentas adequadas.

3.3.2 Diagrama de Casos de Uso

Segundo Booch et al. (2006), os diagramas de casos de uso têm como objetivo a visualização, especificação e documentação do comportamento de um dos elementos do sistema em uma perspectiva externa, de usuário, para tal mostra conjuntos de casos, atores e os relacionamentos entre eles.

Abaixo, nas Figuras 8 e 9, encontram-se os diagramas de casos de uso do sistema.

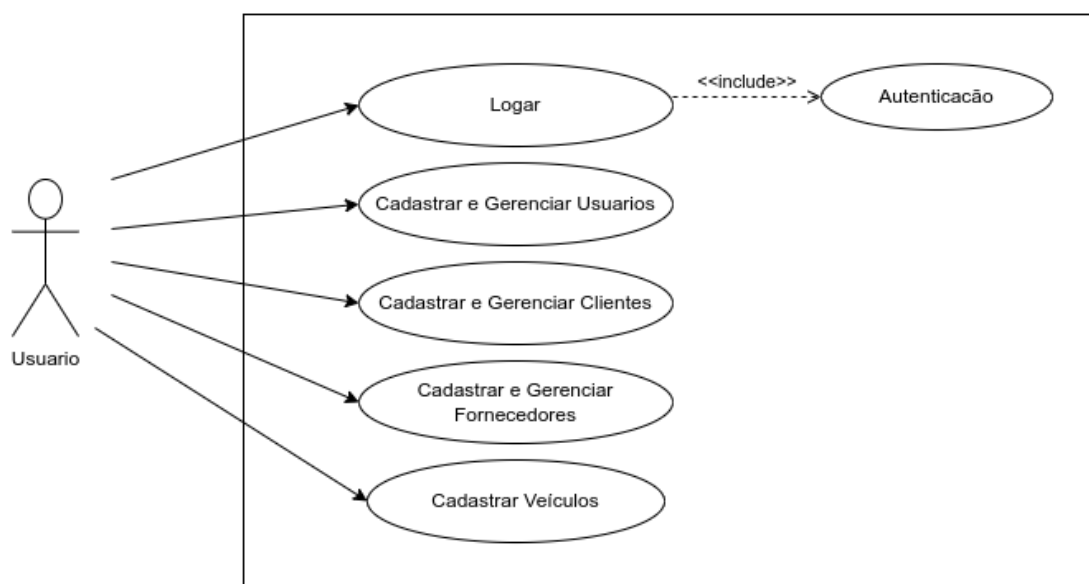


Figura 8 - Diagrama de Caso de Uso (parte I)

Conforme a Figura 8, o ator usuário pode realizar as operações de cadastro e gerenciamento de usuários, clientes, fornecedores e veículos, sendo necessário estar logado no sistema para a sua utilização.

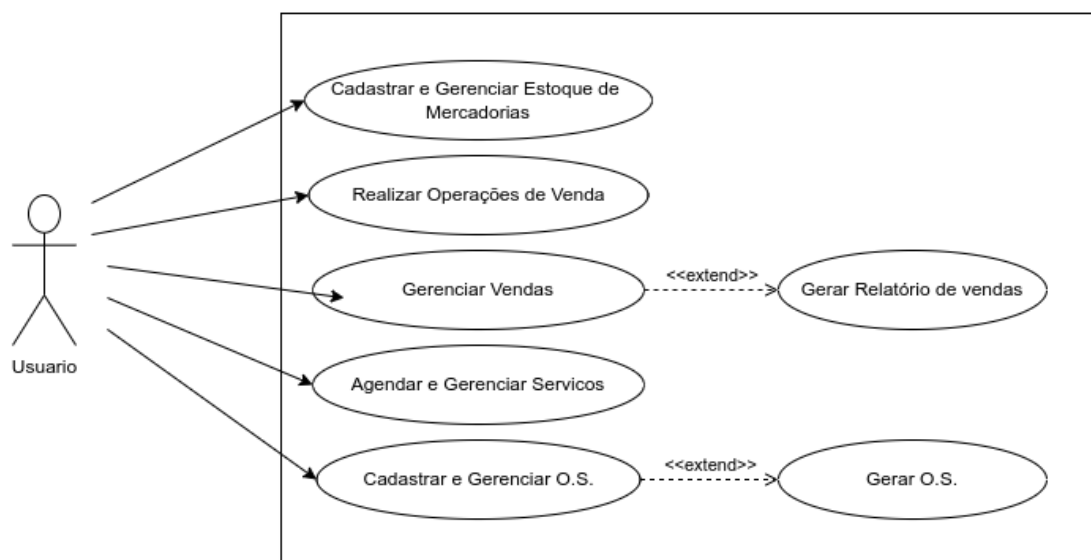


Figura 9 - Diagrama de Caso de Uso (parte II)

A figura 9 demonstra as operações de gestão de estoque, operações de vendas, gerenciamento de vendas com geração de relatórios, agendamento e gestão de serviços, cadastro, gestão e geração de O.S.

3.3.3 Descrição dos Casos de Uso

Serão descritos os casos de uso que representam as principais funcionalidades do sistema de uma forma detalhada nos tópicos abaixo. Os casos de uso suportam as principais operações de CRUD.

3.3.3.1 Logar

Para utilizar o sistema é necessário realizar uma autenticação, com usuário e senha cadastrados no sistema. Estando autenticado é possível acessar o formulário principal do sistema, contendo o seu menu que dá acesso a todas as suas funcionalidades.

3.3.3.2 Cadastrar e Gerenciar Usuários

O cadastro e gestão de usuários podem ser feitos no sistema, para isso sendo necessário que o usuário tenha a permissão de administrador para acessar esse recurso.

3.3.3.3 Cadastrar e Gerenciar Clientes

Os usuários podem cadastrar e gerenciar clientes no sistema. O cadastro de veículos para os clientes não é algo obrigatório, visto que, alguns clientes utilizam somente os serviços de autopeças. O cadastro de veículos de clientes é uma funcionalidade utilizada na realização de serviços de reparo realizados pela oficina.

3.3.3.4 Cadastrar e Gerenciar Fornecedores

Os usuários podem cadastrar fornecedores no sistema. Os fornecedores de mercadorias são registrados no sistema para haver um melhor controle, sendo relacionados as mercadorias fornecidas.

3.3.3.5 Cadastrar Veículos

Os usuários podem cadastrar veículos no sistema, consistindo em: cadastro de marcas, modelos e veículos. O sistema possui uma série de marcas e modelos cadastrados por padrão na base de dados. Os dados dos veículos podem estar associados as peças e acessórios com os quais são compatíveis e clientes aos quais pertencem.

3.3.3.6 Cadastrar e Gerenciar Estoque de Mercadorias

Os usuários podem cadastrar e gerenciar o estoque de mercadorias. As mercadorias estão associadas a um fornecedor e modelo de veículo.

3.3.3.7 Realizar Operações de Venda

Os usuários podem realizar operações de vendas. As operações de venda de mercadorias destinam-se ao setor de autopeças.

3.3.3.8 Gerenciar Vendas

Os usuários podem acessar as informações das vendas realizadas, podendo visualizar dados como: usuário que realizou a venda, mercadoria, quantidade e valor, sendo possível também, a geração de um relatório em PDF da venda selecionada.

3.3.3.9 Agendar e Gerenciar Serviços

Os usuários podem agendar e gerenciar serviços a serem realizados pelo setor de oficina. Os registros dos serviços possuem uma coluna de status para que seja possível visualizar a sua situação.

3.3.3.10 Cadastrar e Gerenciar O.S

Os usuários podem cadastrar, gerenciar e gerar ordens de serviço. As ordens de serviço contêm os principais dados dos serviços prestados, estando ligadas aos clientes, veículos e peças utilizadas na prestação do serviço pelo setor de oficina.

3.3.4 Diagrama de Classes

Segundo Guedes (2018), os diagramas de classes têm como objetivo a visualização estática das classes do sistema, com seus atributos e métodos e a relação entre as classes existentes.

Com os requisitos do sistema definidos é possível definir as suas classes, atributos, métodos e relacionamentos, tomando como base a forma que o Qt estrutura as classes dos projetos. A Figura 10 ilustra a primeira parte do diagrama de classes, contendo as classes responsáveis pelo login e menu principal do sistema, cadastro e gestão dos usuários.

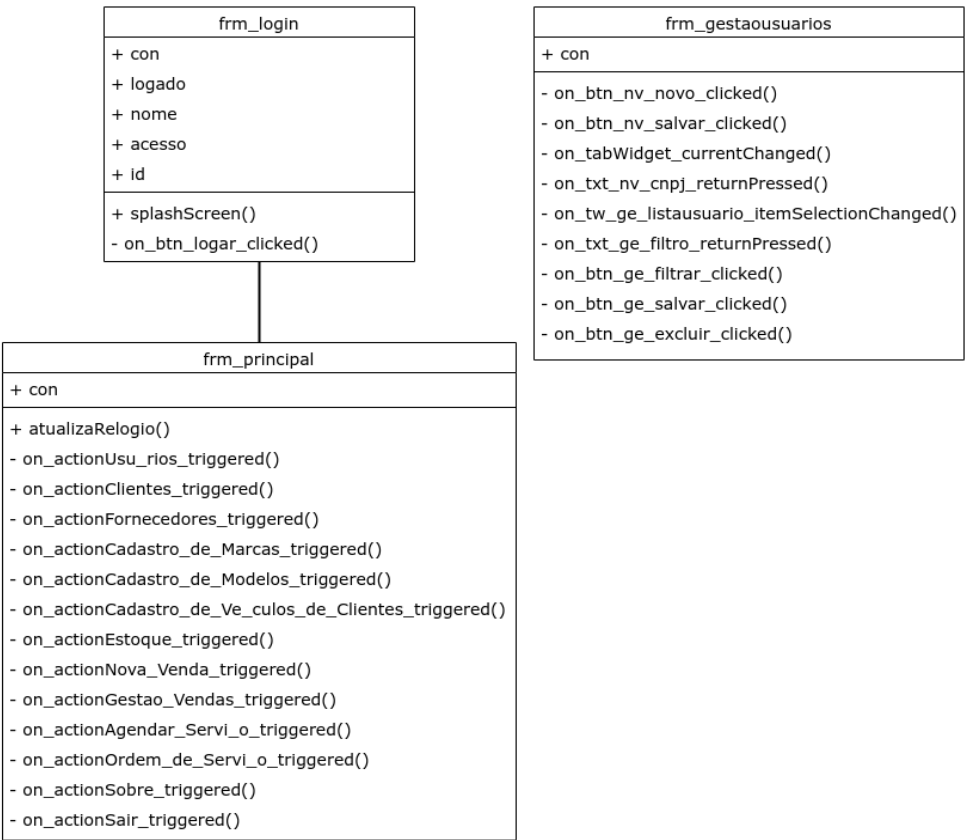


Figura 10 - Diagrama de Classes (parte I)

A Figura 11 representa a segunda parte do diagrama de classe. As classes de cadastro e gestão fornecedores, clientes, marcas, modelos e veículos do sistema, contendo nos seus métodos as operações de CRUD realizadas. As operações dessas classes consistem na inserção, visualização, edição, atualização e deleção de dados na base de dados, bem como, integrações com as APIs, de micro serviços que permitem fazer as consultas e validações do CEP e CNPJ.

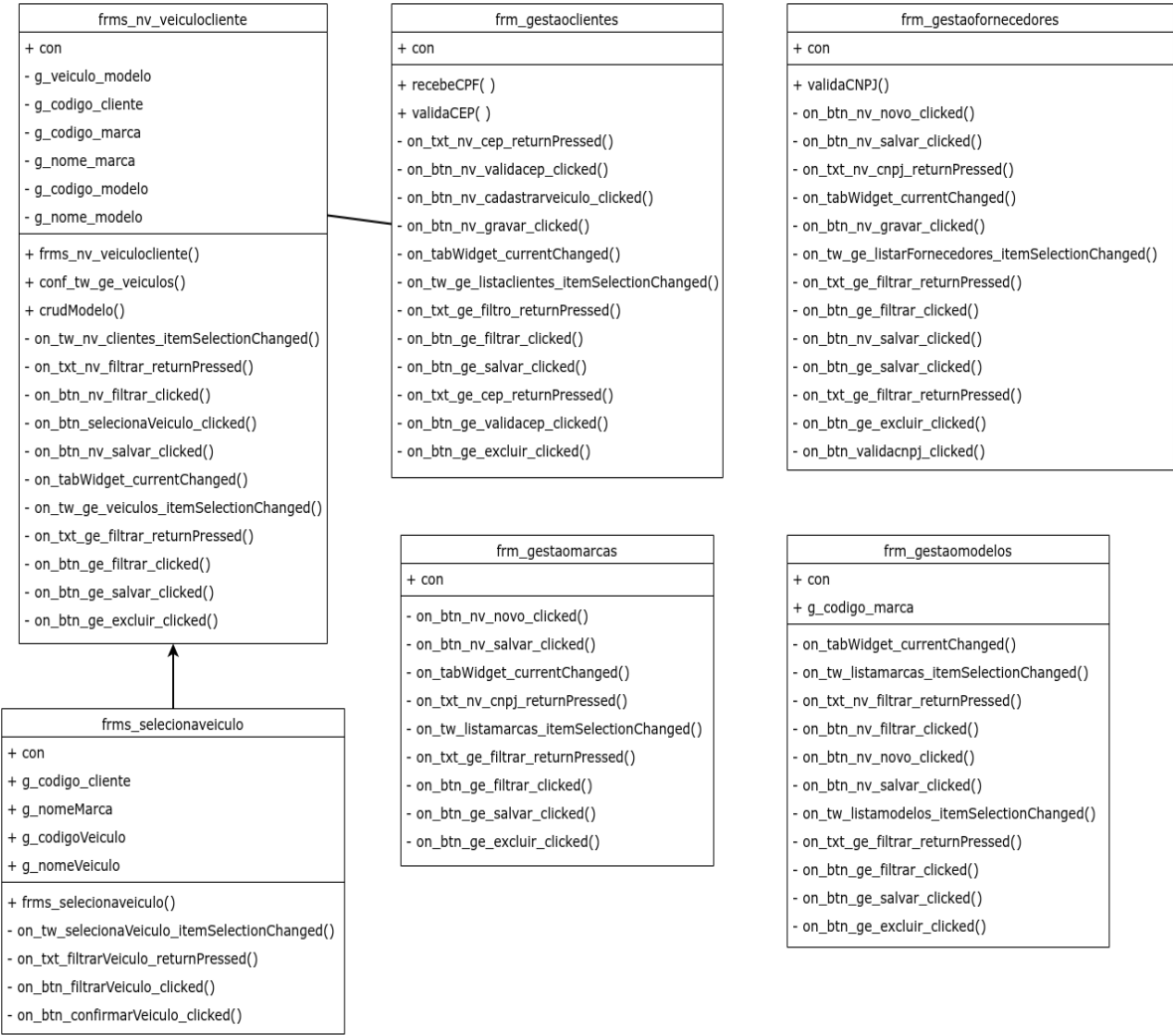


Figura 11 - Diagrama de Classes (parte II)

A Figura 12 representa a terceira parte do diagrama de classes, ilustra as classes responsáveis pelo cadastro e gestão do estoque de mercadorias, realização de vendas, verificação das vendas realizadas e relatórios de vendas.



Figura 12 - Diagrama de Classes (parte III)

Na Figura 13 encontram-se as classes responsáveis pelo agendamento e gestão de serviços, cadastro, gestão e geração de ordens de serviço.

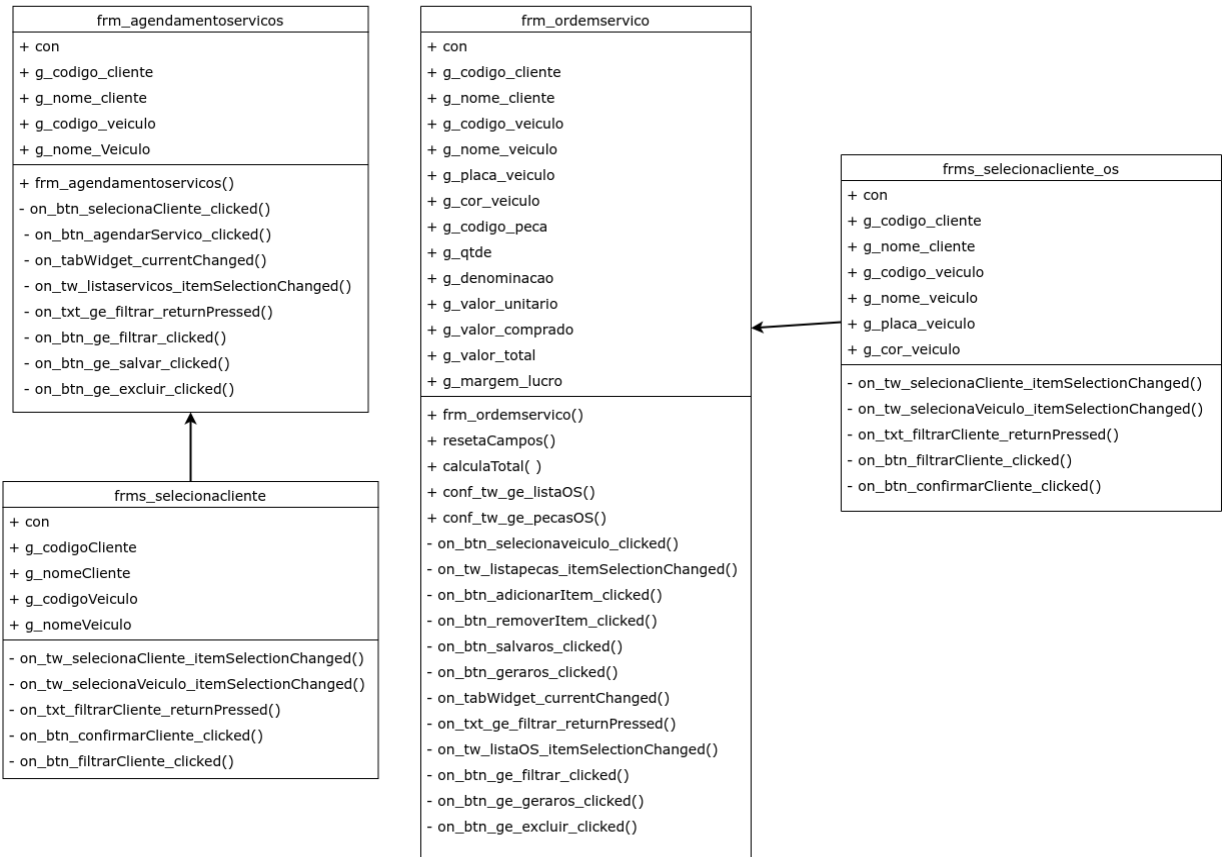


Figura 13 - Diagrama de Classes (parte IV)

No Quadro 2 encontram-se as descrições das classes.

Quadro 2 – Classes

Classe	Descrição
frm_login	Representa o formulário de login do sistema
frm_principal	Representa o formulário principal do sistema, o seu menu
frm_gestaousuarios	Representa o formulário de cadastro e gestão de usuários
frm_gestaoclientes	Representa o formulário de cadastro e gestão de clientes
frm_gestaofornecedores	Representa o formulário de cadastro e gestão de fornecedores
frm_gestaomarcas	Representa o formulário de cadastro e gestão de marcas de veículos
frm_gestaomodelos	Representa o formulário de cadastro e gestão de modelos de veículos
frms_nv_veiculocliente	Representa o formulário de cadastro e gestão de veículos de clientes
frms_selecionaveiculo	Representa um formulário de seleção de marcas e modelos, sendo um subformulário para seleção de marca e modelo de veículo utilizado no cadastro de veículos de clientes
frm_gestaoestoque	Representa o formulário de cadastro e gestão de estoque
frm_novavenda	Representa o formulário de vendas
frm_gestaovendas	Representa o formulário de gestão de vendas
frm_agendamentoservicos	Representa o formulário de agendamento de serviços
frms_selecionacliente	Representa um formulário de seleção de clientes, com veículos, sendo utilizado no agendamento de serviços
frm_ordemservico	Representa o formulário de cadastro, gestão e geração de ordens de serviço
frms_selecionacliente_os	Representa um formulário de seleção de clientes, com veículos, sendo utilizado pelo formulário de ordem de serviço

Fonte: primária

3.3.5 Diagrama ER

O diagrama, gerado a partir do pgAdmin4 serve como base para o desenvolvimento banco de dados, e também para documentar a estrutura. O diagrama pode ser visualizado nas Figuras 14 e 15.

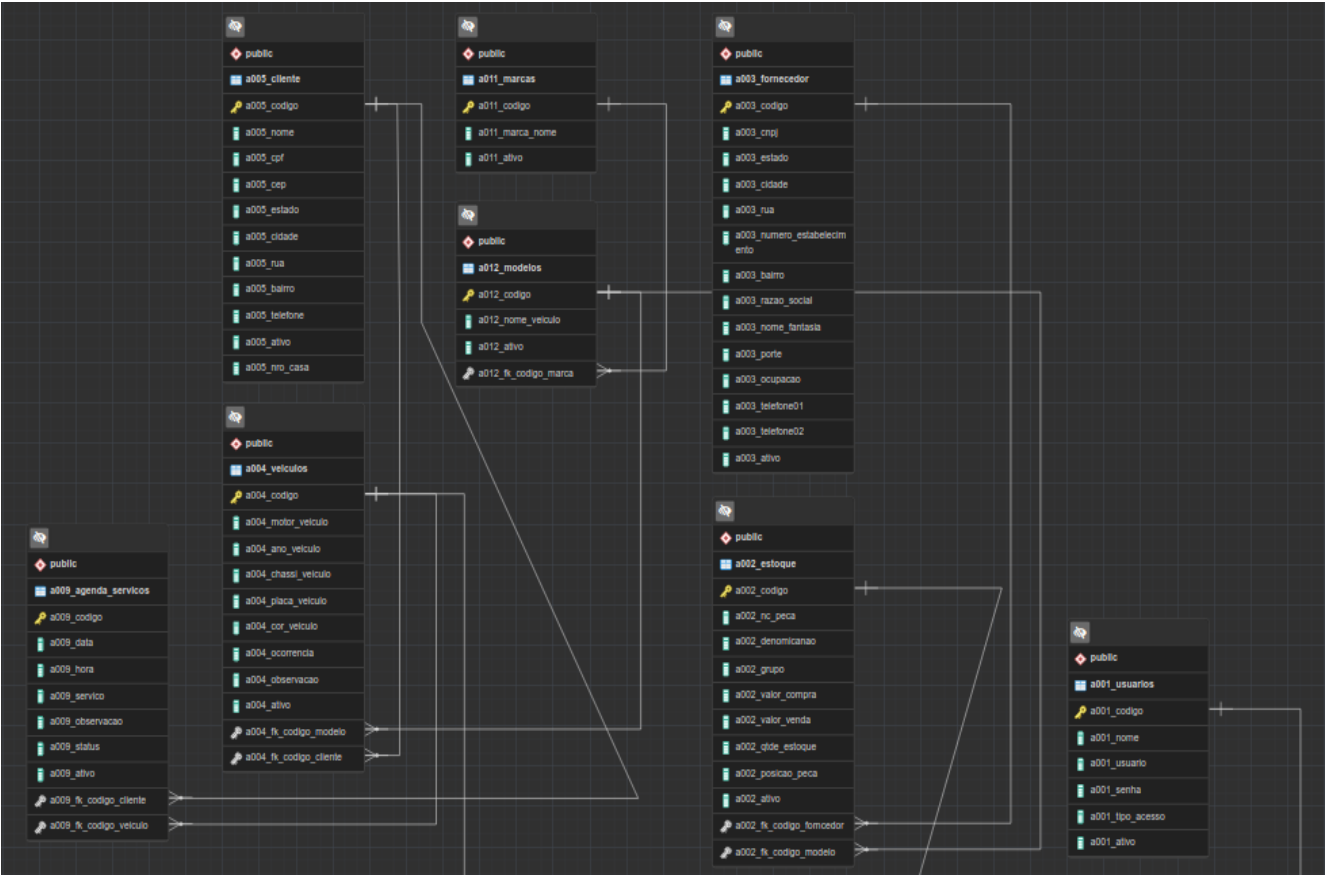


Figura 14 - Diagrama ER (parte I)

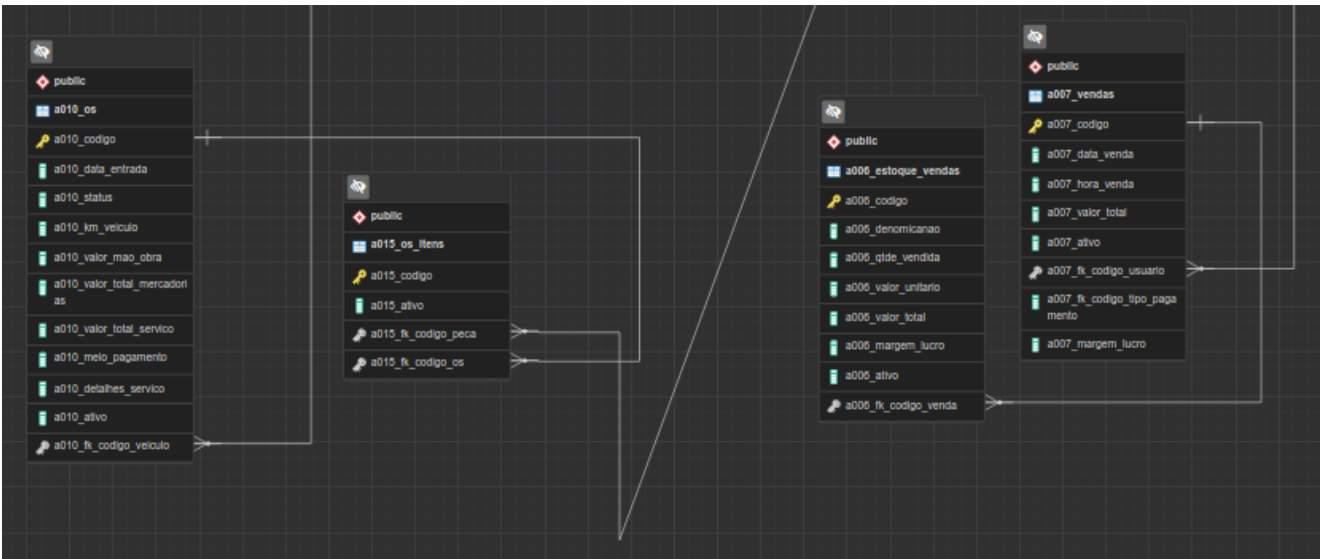


Figura 15 - Diagrama ER (parte II)

O banco é composto por 12 tabelas, bem como, suas colunas, que são numeradas e seguem um padrão de nomeação que foi adotado para uma maior organização, escolhido devido a

familiaridade na utilização. Como uma rotina de segurança o sistema adota o conceito de status de utilização mediante ao uso de uma variável booleana.

O Quadro 3 fornece uma descrição das tabelas do banco.

Quadro 3 – Tabelas do banco de dados	
Tabela	Descrição
a001_usuario	Armazena os dados dos usuários do sistema
a002_estoque	Armazena os dados do estoque de mercadorias
a003_fornecedor	Armazena os dados dos fornecedores de mercadorias
a004_veiculos	Armazena os dados dos veículos dos clientes
a005_cliente	Armazena os dados dos clientes da empresa
a006_estoque_vendas	Armazena os dados de mercadorias das vendas realizadas
a007_vendas	Armazena os dados das vendas realizadas
a009_agenda_serviços	Armazena os dados referentes ao agendamento de serviços
a010_OS	Armazena os dados das ordens de serviços
a011_marcas	Armazena os dados das marcas dos veículos
a012_modelos	Armazena os dados dos modelos de veículos
a015_os_itens	Armazena os dados dos itens incluídos nas ordens de serviço

Fonte: primária

3.4 Desenvolvimento do sistema

Neste capítulo será apresentado o desenvolvimento do sistema, o processo de criação do banco de dados, bem como, o funcionamento de suas classes e configurações para a comunicação com a base de dados PostgreSQL, através das funções disponibilizadas pelo framework Qt.

3.4.1 Estrutura do projeto

O framwework Qt é orientado a objetos, fazendo com que a construção dos formulários da interface gráfica seja implementada através de classes. Na criação de um projeto são gerados quatro tipos arquivos:

- a) Arquivo com extensão “.pro” que fica na pasta raiz do projeto;
- b) A arquivos de cabeçalho com extensão “.h” armazenados na pasta “*Headers*”;

- c) Arquivo principal com a extensão “.cpp” armazenados na pasta “*Sources*”;
- d) Arquivo onde são gerados os formulários com extensão “.ui” armazenados na pasta “*Forms*”.

Esses arquivos são criados e armazenados nas respectivas pastas pela IDE na criação de uma nova classe de formulário, ilustrado na Figura 16.

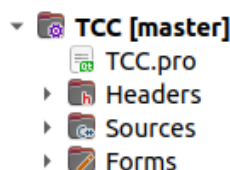


Figura 16 – Estrutura de arquivos

O arquivo com extensão “.pro” é o arquivo do automatizador de compilação do projeto, contendo configurações do projeto, como versão da linguagem de programação, ferramentas e arquivos de códigos que são utilizados durante a compilação do mesmo. Essas configurações também permitem configurar e portar os projetos para diferentes plataformas.

A classe possui como padrão os seus métodos construtores, chamados na inicialização da classe, e destrutores, chamados na finalização da classe. Os atributos da classe e protótipos dos métodos são declarados no arquivo de cabeçalho da classe, já a implementação do código encontra-se no arquivo principal, onde são implementados os métodos prototipados no arquivo de cabeçalho.

O arquivo que contém a interface gráfica é construído por meio do Qt Designer, ferramenta contida na IDE Qt Creator, que possui diversas opções de componentes de interface, configurações e também suporte à CSS para personalização das telas e componentes. Essa ferramenta permite abstrair o processo de construção das telas, onde as telas são construídas por meio do QML (*Qt Modeling Language*), linguagem de marcação para interfaces de usuário.

As classes do projeto definem os formulários da interface gráfica de usuário e as suas funcionalidades. Os arquivos de código fonte estão estruturados pela hierarquia de arquivos demonstrada na Figura 6. Cada formulário contém um conjunto de atributos e métodos para diferentes operações no sistema. As operações de CRUD são realizadas por meio da instanciação da classe de conexão, que contém os drivers, configurações de conexão e métodos.

O projeto também conta com alguns arquivos de funções e variáveis globais para a reutilização em diferentes partes do código fonte. O arquivo “funcoes_globais.h” contém funções que servem para auxiliar em validações, configurações dos formulários e operações realizadas com

frequência. O Diretório “Resources” contém imagens que são utilizadas em algumas telas do sistema.

3.4.2 Padrão de nomenclatura de arquivos e variáveis

Foram adotados alguns padrões de nomenclatura e desenvolvimento, para organizar o projeto, estabelecendo regras para nomeação de arquivos de código, variáveis, funções, *templates* de comentários para funções (contendo as informações de autor, data, propósito e chamada) e também tabulação do SQL nas operações de CRUD.

Os arquivos de código-fonte referentes as classes dos formulários, bem como, as classes, foram nomeados inteiramente com letras minúsculas utilizando o padrão “*Sake Case*”. Os formulários principais possuem “frm_” no início da sua nomenclatura, já os subformulários (criados para pequenas atividades de auxílio aos formulários principais) possuem “frms_” no início da sua nomenclatura.

Os arquivos de classes que não são formulários foram nomeados inteiramente com letras minúsculas, utilizando o padrão “*Snake Case*”. Já as classes foram nomeadas com a letra inicial maiúscula seguindo também o padrão “*Snake Case*”. Esses arquivos e as suas classes iniciam as suas nomenclaturas com “Cl”.

Os campos visuais da interface foram nomeados de acordo com o componente utilizado (“cb_” – *ComboBox*; “txt_” – *Text Box*; “lb_” – *Label*; “tw_” – *Table Widget*; “btn_” – *Button*). Através desse padrão foi possível identificar os diferentes componentes de interface utilizados no sistema.

As variáveis e atributos eram nomeados seguindo o padrão “*Snake Case*” e “*Pascal Case*”. As variáveis globais e atributos de classe iniciam a nomenclatura com “g_”, variáveis utilizadas nos construtores de classe iniciam a nomenclatura com “c_”. As funções e métodos foram nomeados adotando o padrão “*Pascal Case*”.

3.4.3 Criação do banco de dados

Após analisar todos os requisitos necessários do sistema, foi possível estruturar a base de dados e posteriormente construí-la, utilizando o pgAdmin 4, plataforma de administração e desenvolvimento para o PostgreSQL. A criação das tabelas e colunas, bem como, o diagrama do banco, foram feitas utilizando essa ferramenta. Ela dispõe de diversos recursos para gestão do banco de dados. As instruções SQL foram criadas e testadas na ferramenta.

3.4.4 Classe Conexao

A classe “Conexao” foi desenvolvida utilizando bibliotecas que o Qt fornece para integração e manipulação de bancos de dados, e contém os atributos de configuração do banco de dados, como driver e instâncias de conexão. As operações de abrir e fechar o banco de dados foram implementadas através de métodos, utilizando os atributos definidos. O código-fonte do arquivo da classe, em linguagem C++ pode ser visualizado na Quadro 4 .

Quadro 4 - Classe Conexao

```
class Conexao
{
    public:
        QSqlDatabase bancoDeDados;
        QString banco_host = "localhost";
        QString banco_usuario = "postgres";
        QString banco_senha = "12meurex";
        QString banco_nome = "dbAmincar";

        Conexao()
        {
            //drive do postgre
            bancoDeDados = QSqlDatabase::addDatabase("QPSQL");
        }
        inline void fechar() //fecha a conexao com o banco
        {
            bancoDeDados.close();
        }
        bool abrir() //abre a conexao com o banco
        {
            //fazendo conexao com o PostgreSQL
            bancoDeDados.setHostName( banco_host );
            bancoDeDados.setUserName( banco_usuario );
            bancoDeDados.setPassword( banco_senha );
            bancoDeDados.setDatabaseName( banco_nome );

            //verifica se o banco de dados foi aberto
            if( !bancoDeDados.open() )
            {
                return false;
            }
            else
```

```

        {
            return true;
        }
    }
}

```

Fonte: primária

Através dessa classe é realizada a conexão com a base de dados, bastando incluir o seu arquivo de cabeçalho e instancia-la nas classes dos formulários para se conectar a base através os métodos fornecidos, O código utilizado nos formulários para a abertura e fechamento da conexão com o banco de dados, onde o método de abertura é chamado no construtor da classe do formulário, e o método de fechamento de conexão é chamado no destrutor da classe, chamado quando o formulário é fechado. Nos Quadros 5 e 6 encontram-se ilustrados os métodos de abertura e fechamento da conexão no código-fonte.

Quadro 5 - Abrindo conexão no construtor da classe

```

if( !con.abrir() ) //verificando se a conexao foi aberta
{
    if( !con.abrir() )
    {
        QMessageBox::warning(this, "ERRO", "Erro ao abrir banco de dados");
    }
}

```

Fonte: primária

Quadro 6 - Fechando conexão no destrutor da classe

```

con.fechar(); //fechando conexao com o banco de dados
delete ui;

```

Fonte: primária

3.4.5 Classes e formulários do sistema

Nesta seção são abordadas e explicadas as funcionalidades do sistema, descrevendo a sua implementação. Os formulários são criados através de classes e sua parte visual encontra-se no arquivo com extensão “.ui”. Já o código-fonte responsável pelas instruções encontra-se nos arquivos com extensão “.h” e “.cpp”.

3.4.5.1 Classe e formulário de login

Para acessar as funcionalidades do sistema é necessário estar devidamente autenticado, requisito fundamental para montar a segurança dos dados. Para atender esse requisito foi construído um formulário de verificação e validação de login, apresentada na Figura 17.

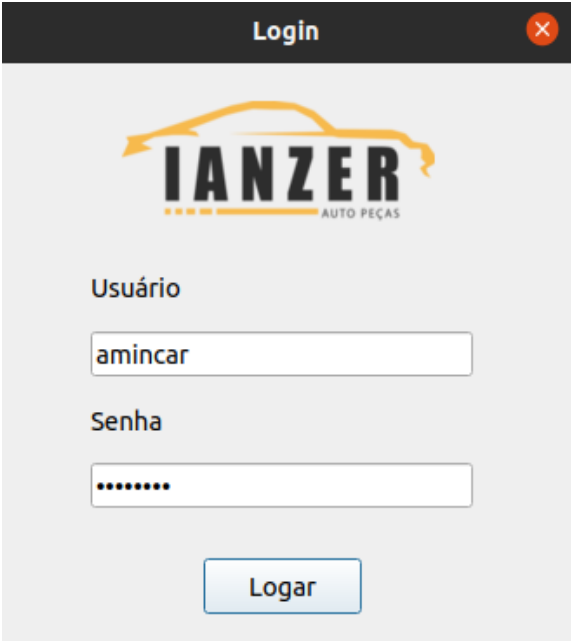
A imagem mostra uma interface de login para o sistema "IANZER AUTO PEÇAS". No topo, há uma barra preta com o título "Login" e um ícone de fechar (X) em um círculo vermelho. Abaixo, o logotipo da empresa é exibido. O formulário principal contém dois campos de entrada: "Usuário" com o texto "amincar" preenchido, e "Senha" com pontos para ocultar o texto. Um botão "Logar" está posicionado abaixo dos campos.

Figura 17 – Tela de login

Nesta tela o usuário do sistema, é preciso informar o seu nome de usuário e senha devidamente cadastrados, para que o acesso as funcionalidades do sistema sejam liberadas. Este recurso garante segurança e controle no acesso ao sistema, e que os dados do sistema estejam protegidos, pois somente os usuários autenticados podem manipular os dados.

Quando o usuário informar os seus dados credenciais e clicar no botão “Logar”, será realizada uma consulta no banco de dados para verificar se os dados informados correspondem a algum registro, se não corresponderem é emitida uma mensagem em uma caixa de texto com a mensagem “Usuário ou senha não encontrados”. Caso os dados correspondam a algum registro será carregada uma tela de abertura e o usuário será redirecionado para a tela principal do sistema.

A Figura 18 demonstra a tela de abertura do sistema.



Figura 18 – Tela de abertura do sistema

O Quadro 7 ilustra como é feito esse processo no código-fonte do sistema, através de uma consulta em linguagem SQL na base de dados, na classe do formulário de login.

Quadro 7 - Operação de login no sistema

```
if( !con.abrir() ) //verifica a conexao com o banco de dados
{
    QMessageBox::warning(this, "ERRO", "Não foi possível se conectar ao banco
de dados");
}
else //login com o banco de dados
{
    QString usuario = ui->txt_usuario->text();
    QString senha = ui->txt_senha->text();
    QSqlQuery query;
    query.prepare("SELECT * FROM a001_usuarios "
                  "WHERE a001_usuario = '"+usuario + "'"
                  "AND a001_senha = '"+senha + "'");

    if(query.exec()) //executa a query
    {
        query.first(); //pega o primeiro retorno da consulta
        if(query.value(1).toString() != "")
        {
            variaveis_globais::logado = true;
            variaveis_globais::nome_colab = query.value(1).toString();
            variaveis_globais::username_colab = query.value(2).toString();
        }
    }
}
```

```

        variaveis_globais::id_colab = query.value(0).toInt();
        variaveis_globais::acesso_colab = query.value(4).toString();

        con.fechar();
        close();
        splashScreen();
    }
    else
    {
        QMessageBox::warning(this, "ERRO", "Usuário ou senha inválido");
    }
}
else
{
    QMessageBox::warning(this, "ERRO", "Falha no login");
}
}
con.fechar(); //fecha a conexao com o banco de dados após o login

```

Fonte: primária

3.4.5.2 Classe e formulário principal

O formulário principal do sistema disponibiliza o acesso a todas as funcionalidades na aba de gestão, no menu localizado na parte superior da tela, também são mostradas todos os serviços pendentes no período de 7 dias, para facilitar a visualização dos usuários e permitir que tenham um maior controle dos serviços da semana. Na parte inferior da tela encontra-se a barra de status do sistema, informando a base de dados, usuário, data e hora atuais.

A Figura 19 apresenta a tela principal com o menu do sistema.

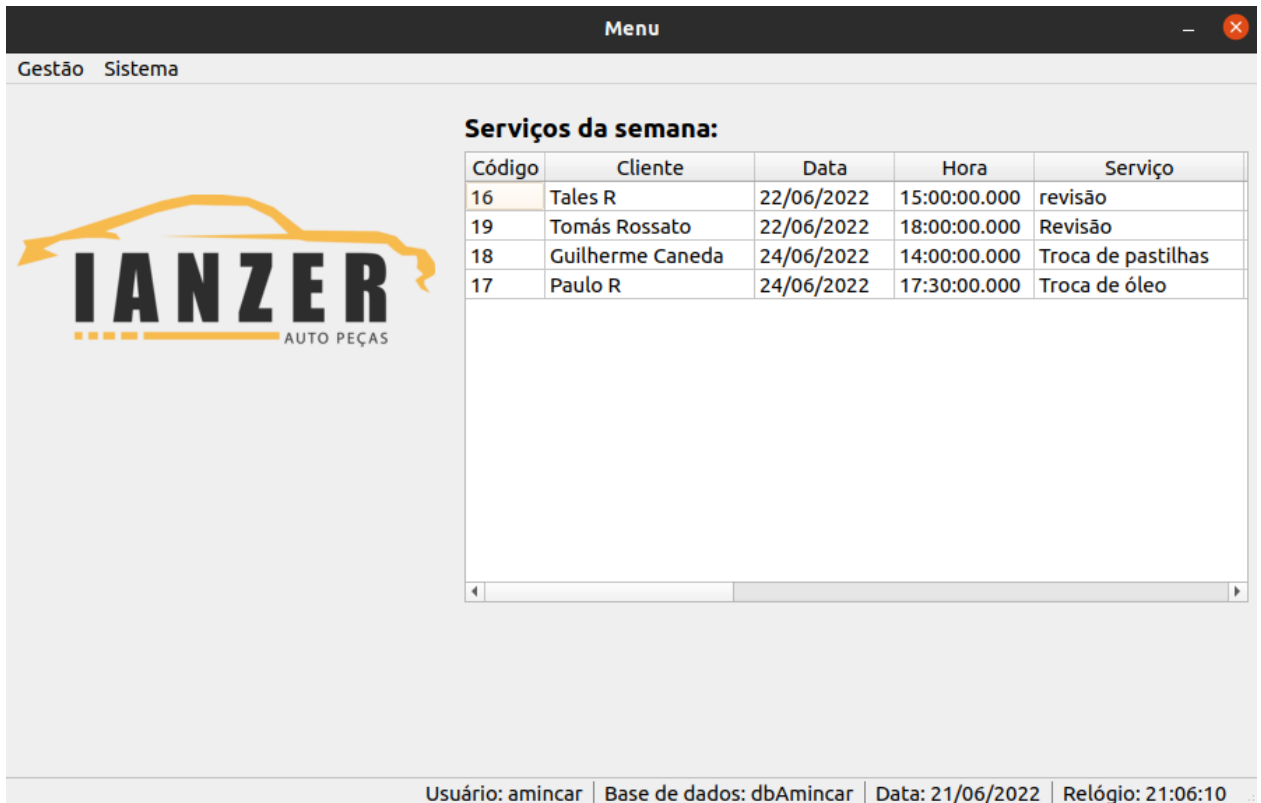


Figura 19 – Tela principal do sistema

O Quadro 8 ilustra uma parte do código-fonte da tela principal, responsável pela chamada dos formulários de gestão de clientes e gestão de fornecedores com as funcionalidades do sistema, que funciona instanciando as classes dos formulários.

Quadro 8 - Trecho de código-fonte para chamada dos formulários

```
#include "frm_gestaoclientes.h"
#include "frm_gestao fornecedores.h"

void Frm_principal::on_actionClientes_triggered()
{
    frm_gestaoclientes *f_gestaoclientes = new frm_gestaoclientes();
    f_gestaoclientes->exec();

    try {
        delete f_gestaoclientes;
        qDebug() << "Deletando ponteiro";
    } catch ( ... ){
        qDebug() << "__Falha ao deletar ponteiro: f_gestaoclientes na tela principal";
    }
}
```



```

void Frm_principal::on_actionFornecedores_triggered()
{
    frm_gestaofores *f_gestaofores = new frm_gestaofores();
    f_gestaofores->exec();

    try {
        delete f_gestaofores;
        qDebug() << "Deletando ponteiro";
    } catch ( ... ) {
        qDebug() << "__Falha ao deletar ponteiro: f_gestaofores na tela principal";
    }
}

```

Fonte: primária

Para acessar atributos e métodos das classes de outros formulários, bem como, funções de arquivos externos, é necessário incluir o arquivo de cabeçalho “.h” no topo do arquivo de cabeçalho da classe ou do arquivo “.cpp”. No caso deste projeto foi adotado como padrão declarar os arquivos de bibliotecas de funcionalidades externas no arquivo “.h” e os arquivos de classes de formulários no arquivo “.cpp”.

O Quadro 9 ilustra uma parte do código-fonte de um método da classe do formulário principal, responsável por exibir os serviços da semana, realizando uma consulta a base de dados. Esse método é chamado no construtor da classe.

Quadro 9 – Trecho do código-fonte da consulta em linguagem SQL

```

QSqlQuery query;
query.prepare("SELECT "
               "a009_codigo                "
               ",a005_nome                  "
               ",TO_CHAR(a009_data, 'DD/MM/YYYY') "
               ",a009_hora                  "
               ",a009_servico               "
               ",a012_nome_veiculo          "
               ",a004_placa_veiculo         "
               ",a009_observacao            "
               ",a009_status                "
               "FROM "
               "a009_agenda_servicos "
               "JOIN a005_cliente ON (a005_codigo = a009_fk_codigo_cliente) "
               "JOIN a004_veiculos ON (a004_codigo = a009_fk_codigo_veiculo) "

```

```
        "JOIN a012_modelos ON (a012_codigo = a004_fk_codigo_modelo)    "
        "WHERE "
        "a009_data BETWEEN CURRENT_DATE AND    CURRENT_DATE + 6 "
        "AND a009_status = 'Pendente' "
        "AND a009_ativo = true "
        "ORDER BY "
        "a009_data ASC, a009_hora ASC");
```

Fonte: primária

3.4.5.3 Classe e formulário de gestão de usuários

O formulário de gestão de usuários só pode ser acessado pelos usuários que tem permissão de administrador. O sistema conta com dois níveis de permissão: administrador e funcionário, criados para limitar os acessos de certos usuários a algumas funcionalidades.

O Quadro 10 ilustra o trecho do código-fonte do formulário principal, utilizado para realizar a verificação da permissão de acesso do usuário para o acesso do formulário.

Quadro 10 – Trecho do código-fonte em linguagem C++ de verificação de acesso

```
void Frm_principal::on_actionUsu_rios_triggered()
{
    //verifica a permissao de acesso do colaborador
    if(variaveis_globais::acesso_colab == 'A')
    {
        frm_gestaousuarios *f_gestaousuarios = new frm_gestaousuarios();
        f_gestaousuarios->exec();

        try
        {
            delete f_gestaousuarios;
        }
        catch ( ... )
        {
            qDebug() << "__Falha ao deletar ponteiro: f_gestaousuarios na tela principal";
        }
    }
    else
    {
        QMessageBox::information(this, "ACESSO", "Acesso não permitido");
    }
}
```

Fonte: primária

Esse formulário fornece as funcionalidades de cadastro e gestão de usuários no sistema. Para realizar o cadastro são necessários os dados de nome, usuário, senha e nível de permissão. Estes dados podem ser visualizados, editados ou excluídos através da aba de gestão do formulário.

A Figura 20 apresenta a aba de gestão da tela de gestão de usuários.

A interface 'Gestão de Usuários' possui duas abas: 'Novo Usuário' e 'Gestão Usuários'. A aba 'Gestão Usuários' contém uma seção de filtro com o texto 'Filtrar por:', um menu suspenso com o valor '-', um campo de texto e um botão 'Filtrar'. Abaixo, há uma tabela com os seguintes dados:

Código	Nome	Usuario
17	funcionario01	func01
14	master	root
13	admin	admin
8	Thiago Ianzer	amincar

Abaixo da tabela, há uma seção 'Dados Usuário' com campos para: Nome (Thiago Ianzer), Usuário (amincar), Senha (mascara de pontos) e Tipo de Acesso (Administrador). No rodapé da interface, há dois botões: 'Salvar' e 'Excluir'.

Figura 20 – Tela de gestão de usuários

De modo a facilitar a busca dos registros foi implementado um filtro. Os registros visualizados podem ser filtrados pelo nome e usuário, selecionados através de um controle combo box. Essa mesma lógica foi utilizada na implementação dos filtros dos outros formulários do sistema.

O Quadro 11 contém um trecho do código-fonte do formulário que implementa a funcionalidade de filtro dos registros.

Quadro 11 – Trecho do código-fonte do filtro de registros

```

QString cb_filtro = ui->cb_ge_filtrar->currentText();
QString txt_filtro = ui->txt_ge_filtrar->text();

QString busca;
QString filtro_sql;

QStringList cb_opc; //Dados do combo box
cb_opc << "Nome" << "Usuário";

funcoes_globais::removerLinhas( ui->tw_ge_listausuario );

//verificando se algo foi digitado no campo de filtro
if( ui->txt_ge_filtrar->text() == "" )
{
    //consulta de acordo com o radio selecionado
    if( cb_filtro == "" )
    {
        busca = "SELECT "
                "a001_codigo      "
                ",a001_nome      "
                ",a001_usuario    "
        "FROM "
        "a001_usuarios "
        "WHERE "
        "a001_ativo = true "
        "ORDER BY "
        "a001_codigo DESC";
    }
    else
    {
        busca = "SELECT "
                "a001_codigo      "
                ",a001_nome      "
                ",a001_usuario    "
        "FROM "
        "a001_usuarios "
        "WHERE "
        "a001_ativo = true "
        "ORDER BY "
        "a001_codigo DESC";
    }
}

```

```

    }
    else
    {
        //consulta de acordo com a seleção do combo box
        switch( cb_opc.indexOf( cb_filtro ) )
        {
            //Nome
            case 0:

                filtro_sql = "a001_nome LIKE '%" +txt_filtro+ "%' ";
                break;
            //Usuário
            case 1:

                filtro_sql = "a001_usuario LIKE '%" +txt_filtro+ "%' ";
                break;
            default:
                qDebug() << "_Houve um problema ao filtrar realizar o filtro(swith
case)";

                break;
        }

        busca = "SELECT "
                "a001_codigo      "
                ",a001_nome      "
                ",a001_usuario    "
                "FROM "
                "a001_usuarios "
                "WHERE "
                + filtro_sql +
                "AND a001_ativo = true "
                "ORDER BY "
                "a001_codigo DESC";
    }

```

Fonte: primária

3.4.5.4 Classe e formulário de gestão de clientes

Esse formulário fornece as funcionalidades de cadastro e gestão de clientes no sistema. Para realizar o cadastro são necessários os dados: nome, CPF, telefone, CEP, estado, cidade, rua, número da residência e bairro.

A aba de gestão clientes permite visualizar, editar e excluir os registros de clientes cadastrados no sistema, contando também com o recurso de filtragem de registros para facilitar a busca.

A Figura 21 ilustra a aba de gestão da tela de gestão de clientes.

Gestão de Clientes

Novo Cliente

Gestão Clientes

Filtrar por:

-

Filtrar

Filtrar

Código	Cliente	CPF	Veículo	Placa	Estado	Cidade
11	Matheus L	15646791014	I30	JAP2762	RS	BAGÉ
9	Tomás Rossato	41946731072	CLIO	IPK2O33	RS	BAGÉ
8	Tales R	55087412084	CAMARO	IPH2K39	RS	BAGÉ
7	Guilherme Caneda	79169688001	350Z	BHC2J65	RS	BAGÉ
6	Gabriel S	96108747030	FOCUS	CHU786	RS	BAGÉ
5	Rodrigo S	58780323090	C5	BHC456	RS	BAGÉ

Dados Cliente

Código

7

Nome

Guilherme Caneda

CPF

79169688001

Telefone

985567315

CEP

96401970

Estado

RS

Cidade

BAGÉ

Rua

AVENIDA GENERAL OSÓRIO 1251

Nro. Casa

135

Bairro

CENTRO

Salvar

Excluir

Figura 22 – Tela de gestão de clientes

De forma a facilitar o preenchimento dos dados dos clientes no cadastro, foi realizada uma integração com a ferramenta CEP V2, disponibilizada pela Brasil API, que permite consultar um CEP, validá-lo e retornar os dados de estado, cidade, rua e bairro. O código-fonte do método responsável pela integração com o CEP V2 pode ser visualizado no Quadro 12.

Quadro 12 - Código-fonte do método que implementa a integração com o CEP V2

```
void frm_gestaoclientes::validaCEP( const QString &cep )
{
    QEventLoop waitLoop;
    QNetworkAccessManager manager;
    QNetworkReply *reply = manager.get(
```

```

        QNetworkRequest(   QUrl("https://brasilapi.com.br/api/cep/v1/"
+ cep) ) );

QObject::connect(reply, SIGNAL(finished()), &waitLoop, SLOT(quit()));
waitLoop.exec();

QJsonDocument doc(QJsonDocument::fromJson(reply->readAll()));
QJsonObject json(doc.object());

//validando o cep, e tratando erros
if( json.find("erro") != json.end() )
{
    json["erro"].toBool() ?
ui->lb_nv_cep->setText("CEP") : ui->lb_nv_cep->setText("CEP: inválido");

}
else
{
    QString cep = json["cep"].toString();
    QString estado = json["state"].toString();
    QString cidade = json["city"].toString();
    QString bairro = json["neighborhood"].toString();
    QString rua = json["street"].toString();

    qDebug() << "Bairro: " << bairro;
    qDebug() << "Rua: " << rua;

    ui->txt_nv_cep->setText( cep );
    ui->txt_nv_estado->setText( estado );
    ui->txt_nv_cidade->setText( cidade );
    ui->txt_nv_bairro->setText( bairro );
    ui->txt_nv_rua->setText( rua );

}
}

```

Fonte: primária

3.4.5.5 Classe e formulário de gestão de fornecedores

Esse formulário fornece as funcionalidades de cadastro e gestão de fornecedores no sistema. Para realizar o cadastro são utilizados os dados de CNPJ, estado, cidade, rua, número do estabelecimento, bairro, razão social, nome fantasia, porte da empresa, ocupação, telefone 1 e telefone 2.

Na aba de gestão fornecedores é possível visualizar, editar e excluir os registros de fornecedores cadastrados no sistema, contando também, com o recurso de filtragem de registros para facilitar a busca.

A Figura 22 ilustra a aba de gestão da tela de gestão de fornecedores.

Gestão de Fornecedores

Novo Fornecedor

Gestão Fornecedores

Filtrar por:

-

Filtrar

Código	Razão Social	Nome Fantasia	CNPJ	Estado	Cidade
22	GUAIBACAR VEICULOS E ...		92661446000185	RS	PORTO ALEGRE
21	TAINA DOS SANTOS LIMA ...	SALAO DE BELEZA	29037197000115	AP	MACAPÁ
14	ANDRE IANZER RODRIGUES	VIDRACARIA SAO PAULO	09477207000126	RS	BAGE

Dados Empresa

CNPJ

92661446000185

Bairro

SANTA MARIA GORETTI

Telefone1

5135841427

RS

RS

Razão Social

GUAIBACAR VEICULOS E PECAS LTDA

Telefone2

5135841425

Cidade

PORTO ALEGRE

Nome Fantasia

Rua

SERTORIO

Porte

GRANDE

Numero

2485

Ocupação

S PARA VEÍCULOS AUTOMOTORES

Salvar

Excluir

Figura 22 – Tela de gestão de fornecedores

Para facilitar o preenchimento dos dados dos fornecedores no cadastro, foi realizada uma integração com a ferramenta CNPJ, disponibilizada pela Brasil API que permite consultar um CNPJ, validá-lo e retornar dados referentes a empresa.

Para este cadastro foram consultados os dados de: estado, cidade, rua, número do estabelecimento, bairro, razão social, nome fantasia, porte, ocupação e telefone 1.

A implementação dessa funcionalidade no código-fonte é feita de uma forma semelhante à do CEP V2, ilustrada no Quadro 12.

3.4.5.6 Classe e formulário de gestão de marcas

Esse formulário fornece as funcionalidades de cadastro e gestão de marcas de veículos no sistema. Para realizar o cadastro é utilizado o nome da marca apenas.

Na aba de gestão marcas é possível visualizar, editar e excluir os registros das marcas de veículos cadastradas no sistema, contando também com o recurso de filtragem de registros para facilitar a busca.

As funcionalidades desse formulário foram implementadas utilizando uma estrutura semelhante à dos formulários descritos anteriormente.

3.4.5.7 Classe e formulário de gestão de modelos

Esse formulário fornece as funcionalidades de cadastro e gestão de modelos de veículos no sistema. Na aba de novo modelo são exibidas as marcas de veículos cadastradas no sistema. É necessário selecionar uma marca e inserir o nome do modelo no campo.

Na aba de gestão modelos é possível visualizar, editar e excluir os registros dos modelos de veículos cadastrados no sistema, contando também com o recurso de filtragem de registros para facilitar a busca.

As funcionalidades desse formulário foram implementadas utilizando uma estrutura semelhante à dos formulários descritos anteriormente.

3.4.5.8 Classe e formulário de gestão de veículos de clientes

Esse formulário fornece as funcionalidades de cadastro e gestão de veículos de clientes no sistema. Para realizar o cadastro de um novo veículo na aba de novo veículo é necessário selecionar um cliente, selecionar a marca e modelo do veículo nos dados veículo. Pressionando o botão ao lado dos campos será aberto um outro formulário, que permite a seleção da marca e modelo do veículo.

Após realizar esse processo, é necessário ser preenchidos os campos de motor, ano, cor, placa, chassi e observação. A Figura 23 e 24 ilustram a aba de novo veículo, no formulário de gestão de veículos de clientes e o formulário de seleção de marca e modelo de veículo.

Veículos de Clientes

Novo Veículo

Gestão Veículos

Filtro:

-

Filtrar

Código	Cliente	CPF	CEP	Estado	Cidade	Rua
12	Thiago Veleda Ianz...	03887161010	96400320	RS	BAGÉ	DOUTOI
11	Matheus L	15646791014	96420220	RS	BAGÉ	TRAVES
10	Paulo R	40757613047	96422160	RS	BAGÉ	RUA CEC
9	Tomás Rossato	41946731072	96408470	RS	BAGÉ	RUA ...
8	Tales R	55087412084	96410380	RS	BAGÉ	RUA CEN
7	Guilherme Caneda	79169688001	96401970	RS	BAGÉ	AVENIDA
6

Carro Cliente

Código Cliente:

12

Dados Veículo

ACURA

NSX

..

Motor

Ano

Cor

v6

1993

vermelho

Placa

Chassi

bhc2j84

pw3h45f

Observação

Salvar

Figura 23 – Tela de veículos de clientes

Seleção de Marca e Modelo

Filtrar:

-

Filtrar

Código	Marca	Modelo
1	ACURA	INTEGRA
2	ACURA	LEGEND
3	ACURA	NSX
4	AGRALE	MARRUA
5	ALFA ROMEO	145
6	ALFA ROMEO	147
7	ALFA ROMEO	155
8	ALFA ROMEO	156
9	ALFA ROMEO	164
10	ALFA ROMEO	166
11	ALFA ROMEO	2300
12	ALFA ROMEO	SPIDER

Confirmar

Figura 24 – Tela de seleção de marcas e modelos

Na aba de gestão modelos é possível visualizar, editar e excluir os registros dos modelos de veículos cadastrados no sistema, contando também com o recurso de filtragem de registros para facilitar a busca.

No Quadro 13 ilustra um trecho do código-fonte que realiza a chamada do formulário de seleção de veículos.

Quadro 13 - Código-fonte da chamada do formulário

```
//btn formulario de seleção de veículos
void frms_nv_veiculocliente::on_btn_selecionaVeiculo_clicked()
{
    QString codigo_cliente = ui->txt_nv_codcliente->text();

    frms_selecionaveiculo *fmSelecionaVeiculo = new frms_selecionaveiculo(
this, codigo_cliente );
    fmSelecionaVeiculo->exec();
    //deletando ponteiro
    try
    {
        delete fmSelecionaVeiculo;
    }
    catch ( ... )
    {
        qDebug() << "__Falha ao deletar ponteiro: fmSelecionaVeiculo na tela
de agendaserviços";
    }
}
```

Fonte: primária

3.4.5.9 Classe e formulário de gestão de estoque

Esse formulário fornece as funcionalidades de cadastro e gestão do estoque de produtos do sistema. Para realizar o cadastro de produtos na aba de novo produto é necessário selecionar um fornecedor e um modelo de veículo, após esse processo os campos de Nc. Peça, Denominação, Grupo, Valor de Compra, Valor de Venda, Qtde e Pos. Prateleira.

A Figura 25 ilustra a aba de novo produto no formulário de gestão de estoque.

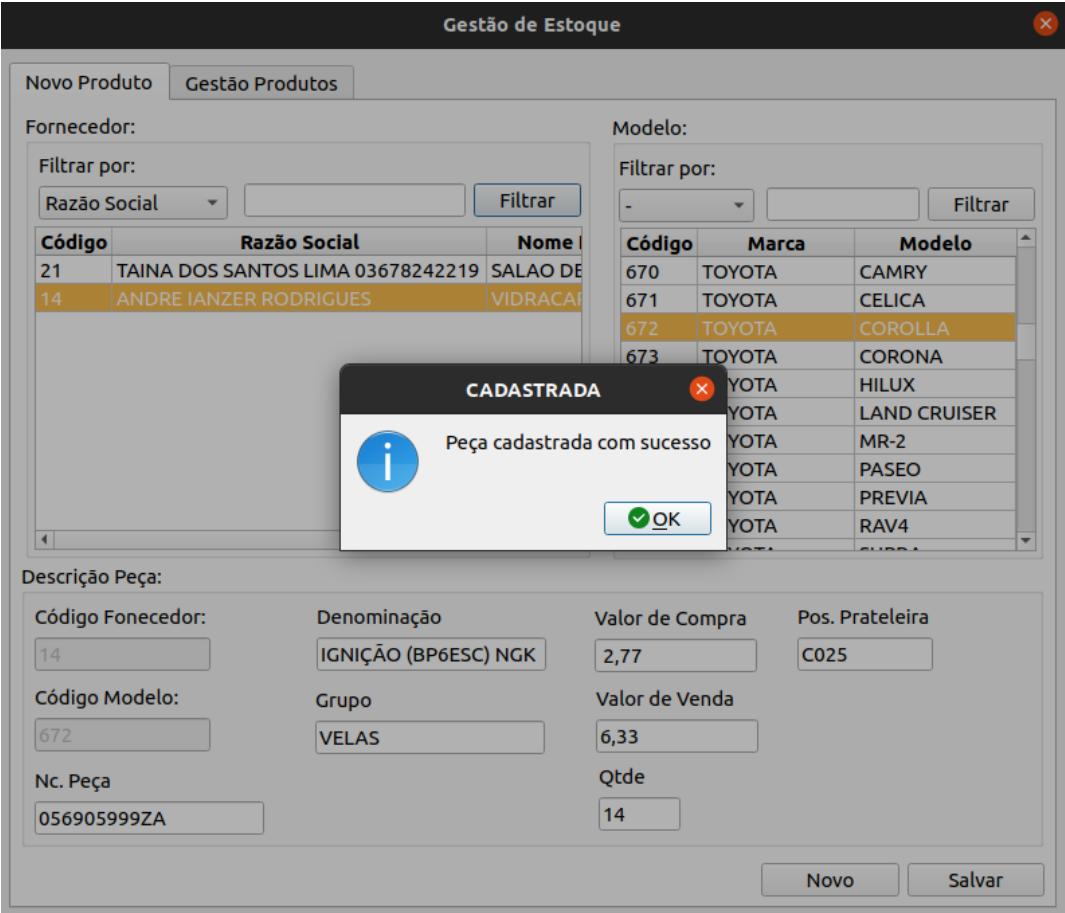


Figura 25 – Tela de gestão de estoque

Na aba de gestão modelos é possível visualizar, editar e excluir os registros dos produtos cadastrados no sistema, contando também com o recurso de filtragem de registros para facilitar a busca.

3.4.5.10 Classe e formulário de nova venda

Esse formulário fornece as funcionalidades para a realização de vendas de produtos. Para a realização de uma venda é necessário selecionar um produto através da tabela de seleção de peças e pressionar o botão “Adicionar” para inseri-lo na tabela de itens venda.

No canto inferior esquerdo da tela é informado o valor total da venda e a margem de lucro da venda. Para concluir a venda é necessária pressionar o botão “Finalizar Venda”.

A Figura 26 ilustra a tela de vendas.

Vendas

Seleção de peças:

Filtrar por:

-

Pesquisar produto

Código	Fornecedor	Modelo	Nc. Peça	Denominação	Grupo	Valor
7	TAINA DOS SANTOS LIMA ...	VECTRA	0	JG.DE ...	PASTILHAS	R\$ 0,00
6	ANDRE IANZER RODRIGUES	RANGER	1234332227	TRANSFOR...	ELETRICA	R\$ 0,00
5	ANDRE IANZER RODRIGUES	SYMBOL	90373520	TERMINAL ...	Direção	R\$ 1.40
4	TAINA DOS SANTOS LIMA ...	IDEA	0	CORREIA ...	CORREIAS	R\$ 0,00
3	AMINTAS PINTO VELEDA	INTEGRA	1212333	bandeja	suspensao	R\$ 100
2	ANDRE IANZER RODRIGUES	F-12000	345656	junta	cabeçote	R\$ 100
1	AMINTAS PINTO VELEDA	NSX	23223	amorte	susp	R\$ 0,00

Código do produto

Qtde. Estoque

Qtde

Adicionar

Remover

Itens venda:

Código	Produto	Valor Un.	Qtde	Total	Lucro
2	junta	200.00	1	200	100
3	bandeja	150.00	1	150	50

Total Venda: R\$ 350,00

Total Lucro: R\$ 150,00

Finalizar Venda

Figura 26 – Tela de vendas

A Figura 27 ilustra um trecho do código-fonte, em linguagem C++ e linguagem SQL, responsável por registrar as vendas realizadas.

```
//verifica se tem produtos no table widget, para realizar uma venda
if( ui->tw_listaprodutos->rowCount() > 0 )
{
    int codigo_venda;

    QString msgFimVenda;
    QString data = QDate::currentDate().toString("yyyy-MM-dd");
    //QString data=QDate::currentDate().toString("dd/MM/yyyy"); //data venda
    QString hora = QTime::currentTime().toString("hh:mm:ss");
    double total = calculaTotal(ui->tw_listaprodutos, 4);
    double lucro = calculaTotal(ui->tw_listaprodutos, 5);

    //inserindo dados da venda na tabela de vendas
    QSqlQuery query;
    query.prepare("INSERT INTO "
                  "a007_vendas(a007_data_venda      "
                  "      ,a007_hora_venda          "
                  "      ,a007_fk_codigo_usuario   "
                  "      ,a007_valor_total         "
                  "      ,a007_margem_lucro)        "
                  "VALUES('"+data                  + "'"
                  "      ,'" +hora                  + "'"
                  "      ,'" +QString::number( variaveis_globais::id_colab ) + "'"
                  "      ,'" +QString::number( total ) + "'"
                  "      ,'" +QString::number( lucro ) + "'"");
```

Figura 27 – Registrando vendas

3.4.5.11 Classe e formulário de gestão de vendas

Esse formulário permite visualizar todas as vendas realizadas. A tabela vendas permite visualizar os dados da venda como: usuário que realizou a venda, data, hora, valor total da venda e margem de lucro.

A tabela de produtos permite visualizar todos os produtos contidos na operação de venda, informando os dados como: produto da operação de venda, quantidade, valor unitário, valor total e margem de lucro.

O botão “Gerar relatório” localizado na parta inferior direita permite a geração de um relatório da venda selecionada. Esse relatório é gerado em arquivo “PDF” e contém todas os dados da venda.

As Figuras 28 e 29 ilustram a tela de gestão de vendas e o relatório de vendas.

Gestão de Vendas

Filtrar:

Usuário

21/06/2022

21/06/2022

Filtrar

Todas as Vendas

Vendas:

Código	Usuário	Data	Hora	V.Total	M.Lucro
12	Thiago Ianzer	21/06/2022	16:27:29.000	350	R\$ 150,00
11	Thiago Ianzer	19/05/2022	00:09:41.000	0	R\$ 0,00
10	Thiago Ianzer	15/05/2022	17:12:23.000	600	R\$ 500,00
9	Thiago Ianzer	15/05/2022	17:05:41.000	400	R\$ 300,00
8	Thiago Ianzer	15/05/2022	16:40:43.000	600	R\$ 500,00
7	Thiago Ianzer	15/05/2022	16:17:38.000	400	R\$ 300,00
6	Thiago Ianzer	13/05/2022	20:36:14.000	200	R\$ 100,00
5	Thiago Ianzer	13/05/2022	20:36:05.000	0	R\$ 0,00

Produtos:

Código Mov	Produto	Qtde	Val.Uni	Val.Total	M.Lucro
8	junta	1	200	200	100
9	bandeja	1	150	150	50

Gerar Relatório

Figura 28 – Tela de gestão de vendas

RELATÓRIO DE VENDAS					
DADOS VENDA					
Código	Usuário	Data	Hora	Val.Total	Margem Lucro
12	Thiago Ianzer	21/06/2022	16:27:29.000	R\$ 350,00	R\$ 150,00
ITENS VENDA					
Código	Produto	Qtde	Val.Uni	Val.Total	M.Lucro
8	junta	1	R\$ 200,00	R\$ 200,00	R\$ 100,00
9	bandeja	1	R\$ 150,00	R\$ 150,00	R\$ 50,00

Figura 29 – Relatório de vendas

3.4.5.12 Classe e formulário de agendamento de serviços

Esse formulário fornece as funcionalidades de agendamento e gestão de serviços. Para realizar o agendamento de um serviço na aba de novo serviço é necessário é necessário selecionar um cliente, selecionar o modelo do veículo, pressionando o botão ao lado dos campos será aberto um outro formulário, que permite a seleção de cliente e veículo.

Após selecionar o cliente e o modelo do veículo é necessário selecionar data, hora e preencher os campos de serviço, que especifica o tipo de serviço, e observação, que permite uma maior descrição do serviço. Realizada essa operação o serviço é registrado como status “pendente”.

Na aba de gestão serviços é possível visualizar, editar, podendo alterar o status do serviço para “realizado” ou “cancelado”, e excluir os registros dos serviços cadastrados no sistema, contando também com o recurso de filtragem de registros para facilitar a busca.

A Figuras 30 ilustra a aba de novo serviço do formulário de agendamento de serviços.

Agendamento de Serviços

Novo Serviço

Gestão Serviços

Dados Serviço:

Dados cliente:

Rodrigo S

CS

..

Data

03/08/2022

Hora

17:00

Serviço

Revisão

Observação

Revisão geral do veículo

Agendar

Figura 30 – Tela de agendamento de serviços

O Quadro 14 contém um trecho do código-fonte em linguagem SQL responsável pela funcionalidade de edição dos dados dos serviços.

Quadro 14 – Atualizando os registros dos serviços

```
query.prepare("UPDATE "
    "a009_agenda_servicos "
    "SET "
    "a009_data          =' " +data          + " '"
    ",a009_hora         =' " +hora          + " '"
    ",a009_servico      =' " +servico        + " '"
    ",a009_observacao   =' " +observacao    + " '"
    ",a009_status       =' " +status        + " '"
    "WHERE "
    "a009_codigo =' " +id+ " '");
```

Fonte: primária

63

3.4.5.13 Classe e formulário de ordem de serviço

Esse formulário fornece as funcionalidades para geração de ordens de serviço. Para criar uma ordem de serviço é necessário selecionar um veículo, contendo os dados de cliente proprietário, modelo, placa e cor, pressionando o botão “Selecionar” é aberto um formulário para selecionar os dados do veículo.

Após selecionar os dados do veículo é necessário preencher os campos de data do serviço, quilometragem do veículo, meio de pagamento, com as opções de: à vista, PIX e cheque, valor de mão de obra, observações, para detalhes do serviço.

Após preencher esses campos o usuário pode selecionar os produtos, que são peças e acessórios de veículos, contidos na ordem de serviço e adicionar a tabela de “Peças O.S” para incluir os itens na O.S, com o valor total das peças sendo exibido.

Quando os dados do veículo são preenchidos esses produtos são filtrados automaticamente pela marca e modelo do veículo, facilitando a identificação das peças e acessórios compatíveis com o modelo do veículo.

Após preencher todos os campos e dados do serviço o usuário tem a opção de salvar os dados da O.S no botão “Salvar O.S.” ou gerar a O.S diretamente no botão “Gerar O.S.”. A ordem de serviço é gerada com os dados do serviço em um arquivo PDF.

A Figura 31 ilustra a tela de nova O.S.

Ordem de Serviço

Nova O.S.

Gestão O.S.

Dados Veículo:

Rogério S

INTEGRA

bch2j84

Branco

Selecionar

Data Entrada

24/06/2022

KM Veículo

73.000

M. Pagamaneto

À VISTA

V. Mão de Obra

400,00

Detalhes

Revisao geral: troca de bandeja de suspensão

Produtos

Peças: INTEGRA

Código	Fornecedor	Nc. Peça	Denominacao
3	AMINTAS PINTO VELEDA	1212333	bandeja

Código do produto

Qtde. Estoque

Qtde

Adicionar

Remover

Peças O.S:

Código	Denominação	Valor Un.	Qtde
3	bandeja	150.00	1

Total Peças: R\$ 150,00

Salvar O.S.

Gerar O.S.

Figura 31 – Tela de Ordem de serviço

Na aba de gestão O.S. é possível visualizar, editar, excluir os registros das ordens de serviços cadastradas no sistema. Também é possível filtrar os registros de ordens de serviço e gerar um arquivo PDF com a O.S. selecionada.

O Quadro 15 ilustra um trecho do código-fonte que insere os dados dos produtos na ordem de serviço gerada em arquivo PDF.

Quadro 15 – Adicionando produtos ao arquivo PDF da O.S

```
painter.drawText(25, 660, "Código");
painter.drawText(130, 660, "Mercadoria");
painter.drawText(295, 660, "Valor Un.");
painter.drawText(415, 660, "Qtde");
painter.drawText(540, 660, "V.Total");

//informando localização, coluna e linha
int linha = 710;
int salto = 20;
```

65

```

for( int i = 0; i < ui->tw_pecasOS->rowCount(); i++ )
{
    painter.drawText(25,linha, ui->tw_pecasOS->item(i, 0)->text());
    painter.drawText(130,linha, ui->tw_pecasOS->item(i, 1)->text());
    painter.drawText(295,linha, "R$ " + ui->tw_pecasOS->item(i, 2)->text());
    painter.drawText(415,linha, ui->tw_pecasOS->item(i, 3)->text());
    painter.drawText(540,linha, "R$ " + ui->tw_pecasOS->item(i, 4)->text());
    linha += salto;
}

```

Fonte: primária

3.4.5.14 Formulário sobre

O formulário sobre acessado no formulário principal na aba sistema do menu. Esse formulário contém os dados como versão do sistema, autor, licença e repositório no GitHub com o código-fonte do sistema, visto que o Qt na sua versão gratuita só pode ser utilizado sobre licença GNU, tornando necessário o compartilhamento do código-fonte.

A figura 32 ilustra a tela sobre do sistema.



Figura 32 – Tela sobre

3.4.5.15 Instalador do sistema

Por fim o sistema contém um instalador que auxilia no processo de implantação do sistema. O instalador permite que sejam realizadas as operações de instalação, atualização e desinstalação, bem como, especificar informações do sistema.

A Figura 33 ilustra o instalador do sistema.

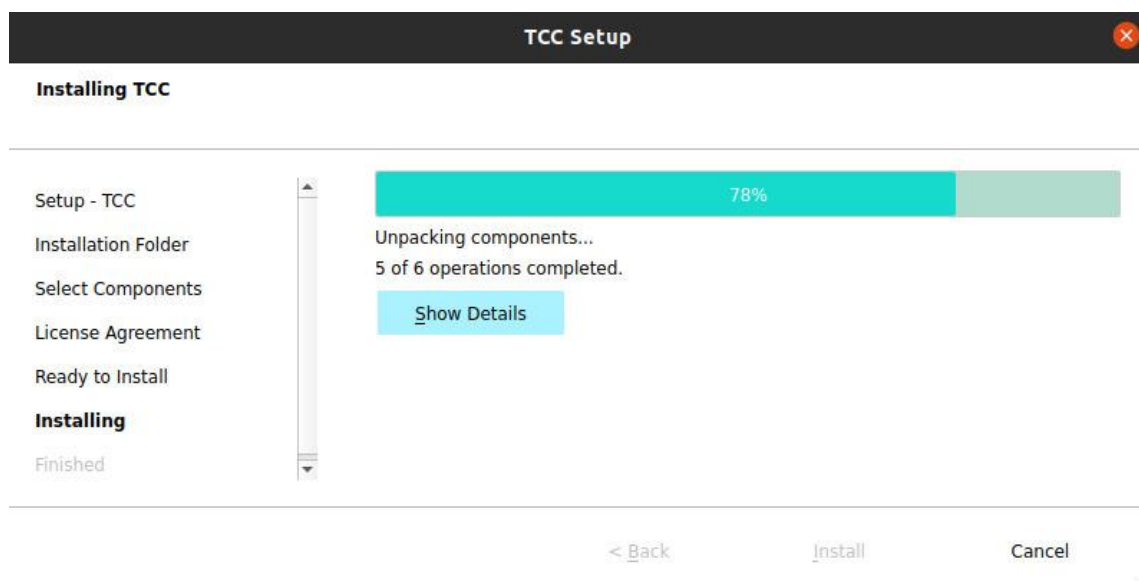


Figura 33 – Instalador do sistema

4 CONCLUSÃO

O presente estudo teórico e prático pauta-se no desenvolvimento de um sistema com a finalidade de melhorar os processos de gestão de estoque integrado a ordem de serviços de uma loja de auto peças anexa a uma oficina mecânica, o método utilizado para elicitação de requisitos consistiu na entrevista com os proprietários e análise do domínio. No decorrer do trabalho ficaram registradas as principais questões técnicas vinculadas a implementação do sistema, permitindo assegurar que o produto atende integralmente a demanda dos usuários.

O sistema resultante deste trabalho provou-se de fácil utilização e rápido aprendizado pelos usuários, o que elimina eventuais dependências ao desenvolvedor como também a substituição de recursos humanos sem grandes traumas.

4.1 Trabalhos futuros

Considerando a área de tecnologia de informação é de consenso geral a contínua evolução decorrente de mudanças no domínio do problema e de novas técnicas e ferramentas, tais como o aperfeiçoamento das interfaces com os fundamentos de UX e que assim justifiquem melhorias para manter o sistema no estado da arte. que considerando as características do paradigma de orientação a objetos utilizado torna-se necessários aprimorá-lo.

REFERÊNCIAS

- BALLOU, R. H. **Gerenciamento da cadeia de suprimentos: Planejamento, organização e logística empresarial**. 5 E.d. Porto Alegre: Bookman, 2006.
- BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML: guia do usuário**. Elsevier, 2006.
- BORGES, S. T.; CAMPOS, S. M.; BORGES, C. E. **Implantação de um sistema para controle de estoques em uma gráfica/editora de uma universidade**. Revista Eletrônica Produção e Engenharia, v. 3, 2010.
- CHOPRA, Sunil; MEINDL, Peter. **Gerenciamento da cadeia de suprimentos: estratégia, planejamento e operação**. São Paulo: Pearson, 2004.
- CRUZ, André Luiz Casetto Viera da. **Soa: concepts and technologies used in service oriented architecture**. Intellectus, 2010.
- DEITEL, H. M. **C++ Como Programar**. São Paulo: Pearson Prentice Hall, 2006.
- FARINELLI, Fernanda. **Conceitos básicos de programação orientada a objetos**. 1 E.d. Minas Gerais, 2007.
- GOMES, Daniel Adorno. **Web services soap em java: Guia prático para o desenvolvimento de web services java**. 2 E.d. Novatec, 2014.
- GUEDES, G. T. **UML 2-Uma abordagem prática**, Novatec, 2018.
- HEUSER, Carlos Alberto. **Projeto de banco de dados**. 6 E.d. Porto Alegre: Bookman, 2009.
- JACOBSON, D. Brasil, G. WOODS, D. **APIs: A strategy guide**. 1 Ed, Gravenstein Highway North, Sebastopol, CA – O’ Reilly Media, Inc, 2011.
- LOUREIRO, Henrique. **C++ Guia Moderno de Programação**. Lisboa: FCA – Editora de Informática, 2019.
- PEREIRA, Luiz Antônio de Moraes. **Análise e Modelagem de Sistemas com a UML**. 1. Ed. Rio de Janeiro, 2011.
- REZENDE, Denis A.; ABREU, Aline F. **Tecnologia de informação: Aplicada a sistemas de informação empresariais**, 2 Ed. São Paulo: Atlas 2001.
- STROUSTRUP, Bjarne. **A Tour of C++**. Michigan: Pearson Education, 2014.
- TAYLOR, A. David. **Logística na cadeia de suprimentos uma perspectiva gerencial**. São Paulo: Pearson, 2006.
- WEILL, P; ROSS, W. J. **Governança de TI – como as empresas com melhor desempenho administram os direitos decisórios de TI na busca por resultados superiores**. 1. Ed. São Paulo: M.Books, do Brasil, 2006.
- WERNK, Rodney; LEMBECK, Marluce; NASCIMENTO, Fábio de Araújo. **Gestão financeira de estoques: estudo de caso e indústria têxtil de médio porte**. Revista brasileira de contabilidade. Rio de Janeiro, 2011.

