PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA POLITÉCNICA
ENGENHARIA DE COMPUTAÇÃO
COORDENAÇÃO DE TCC

SISTEMA DE INFORMAÇÃO PARA CONTROLE DE VEÍCULOS E VIAGENS DE UMA TRANSPORTADORA

TRABALHO DE CONCLUSÃO DE CURSO

RENATO QUEIROZ SILVA

GOIÂNIA, GO
2022

RENATO QUEIROZ SILVA

SISTEMA DE INFORMAÇÃO PARA CONTROLE DE VEÍCULOS E VIAGENS DE UMA TRANSPORTADORA

Trabalho de Conclusão de Curso apresentado ao curso de Engenharia de Computação da Pontifícia Universidade Católica de Goiás, como parte dos requisitos necessários à obtenção do título de Bacharel em Engenharia de Computação

Orientador: Prof. Me. Olegário Correa da Silva Neto

GOIÂNIA, GO
2022

RENATO QUEIROZ SILVA

SISTEMA DE INFORMAÇÃO PARA CONTROLE DE VEÍCULOS E VIAGENS DE UMA TRANSPORTADORA

Este Trabalho de Conclusão de Curso julgado adequado para obtenção do título de Bacharel em Engenharia de Computação, e aprovado em sua forma final pela Escola Politécnica, da Pontifícia Universidade Católica de Goiás, em ___/___/____.

_____
Prof. Me. Ludmilla Reis Pinheiro dos Santos
Coordenadora de Trabalho de Conclusão de Curso
Escola Politécnica

Banca Examinadora:

_____
Prof. Me. Olegário Correa da Silva Neto
Pontifícia Universidade Católica de Goiás
Orientador

_____
Prof. Dr. José Luiz de Freitas Júnior
Pontifícia Universidade Católica de Goiás

_____
Prof. Me. Eugênio Júlio Messala Cândido Carvalho
Pontifícia Universidade Católica de Goiás

Goiânia
2022

Agradecimentos

Agradeço a Deus primeiramente pela saúde e pela vida

Agradeço ao meu orientador por minhas ausências nas orientações, pela paciência com os maus momentos que tive durante o desenvolvimento deste trabalho.

Foram muitas dificuldades, crises de enxaqueca que duravam uma semana completa, muita correria pelos horários apertados para conciliar a Universidade, o emprego e o trabalho de conclusão.

Agradeço a minha família, principalmente a minha mãe, que sempre esteve ao meu lado, nos momentos mais difíceis e também nos momentos de alegria.

Nada resiste a persistência, insistência e não desistência. Tenha foco e o resultado chegará.

Resumo

Este projeto descreve o desenvolvimento de software de gestão de fretes, tendo como base os requisitos levantados durante a elicitação, utilizando algumas ferramentas como, Python, Django, MySQL, Bootstrap , entre outras.
Palavras-chave: Software. Gestão. Fretes. Logística.

Abstract

This project describes the development of freight management software, based on the requirements raised during the elicitation, using some tools such as Python, Django, MySQL, Bootstrap, among others.
Keywords: Software. Management. freight. Logistics.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ABNT	Associação Brasileira de Normas Técnicas
ANTT	Agência Nacional de Transportes Terrestres
API	Application Programming Interface
CNH	Carteira Nacional de Habilitação
CPF	Cadastro de Pessoas Físicas
CSS	Cascading Style Sheets
DE	Diagnóstico de Enfermagem
DOU	Diário Oficial da União
HTML	HyperText Markup Language
IBM	International Business Machines
IDE	Integrated Development Environment
JPEG	Joint Photographic Experts Group
JS	Javascript
KM	Quilômetro
MYSQL	Banco de dados
PDF	Portable Document Format
PNG	Plano de Negócios e Gestão
PSF	Programa de Saúde Da Família
PYTHON	Linguagem de programação
RF	Radiofrequência
RFN	Requisito Não funcional
SQL	Structured Query Language
SVG	Scalable Vector Graphics
TMS	Transportation Management System
UML	Linguagem de Modelagem Unificada (do inglês, Unified Modeling Language)

Sumário

1 Introdução

A grande capacidade produtiva do agronegócio brasileiro, a dificuldade de armazenagem e a alta demanda de exportação transformaram o modal rodoviário em um dos mais importantes da malha logística brasileira.

A demanda de fretes para escoamento da produção fez com que os operadores logísticos aumentassem suas frotas, tendo cada vez capacidade de movimentação de cargas, oferecendo serviços de boa qualidade aos contratantes e, ao mesmo tempo, tendo que sacrificar um dos pilares mais importantes no desenvolvimento de uma empresa, o gerenciamento de recursos .

O gerenciamento de fretes parece não ser uma tarefa tão complexa, mas por depender de muitas variáveis, a complexidade acaba aumentando significativamente.

Para facilitar o gerenciamento de fretes surge o TMS*(Transportation Management System)*, ou em tradução livre, Sistema de Gerenciamento de Transportes, um sistema completo de gerenciamento de empresas logísticas.

O TMS usa tecnologia para ajudar as empresas a planejar, executar e otimizar a movimentação de mercadorias para garantir a conformidade da remessa e a documentação. Também conhecido como solução de gerenciamento de transporte, um TMS fornece visibilidade das operações de transporte do dia a dia, informações e documentação de conformidade comercial e garante a entrega pontual de cargas e mercadorias.

Neste trabalho desenvolverei um sistema de controle de veículos e fretes para uma transportadora de pequeno porte.

1.1 Objetivos Gerais

Desenvolver um *software* web para gerenciamento de veículos e viagens de uma transportadora de pequeno porte.

1.2 Objetivos Específicos

Entre os objetivos específicos estão:

- Analisar os requisitos levantados, identificar características para desenvolver o software de gerenciamento de fretes.
- Utilizar técnicas e linguagens de programação para desenvolver o software web.
- Utilizar o framework Django para web.
- Utilizar a Linguagem de Programação de Alto Nível Python.

1.3 Justificativa

A justificativa do trabalho foi a necessidade de melhorar e facilitar o gerenciamento da frota e dos fretes da Mota Transportes, empresa de pequeno porte especializada no transporte de cargas ligadas ao agronegócio (soja, milho, calcário, etc), localizada no município de Bom Jesus - GO.

Considerando a falta de métricas para mostrar os resultados da empresa como gastos mensais, consumo de combustível e manutenções. O software proposto por este trabalho será de grande valia para facilitar o dia a dia e a análise de desempenho de cada veículo, considerando quantidade de fretes, valores, custos de manutenções e de combustível, facilitando na solução mais viável.

1.4 Metodologia

1.4.1 Elicitação dos Requisitos

A Elicitação dos requisitos é o processo de reunir informações sobre o sistema requerido e os sistemas existentes e separar dessas informações os requisitos de usuário e de sistema. Fontes de informação durante a fase de descoberta de requisitos incluem documentação, stakeholders do sistema e especificações de sistemas similares. Você interage com os stakeholders por meio da observação e de entrevistas e pode usar cenários e protótipos para ajudar os stakeholders a compreenderem o que o sistema vai ser. (SOMERVILLE, 2011)

1.4.2 Entrevistas

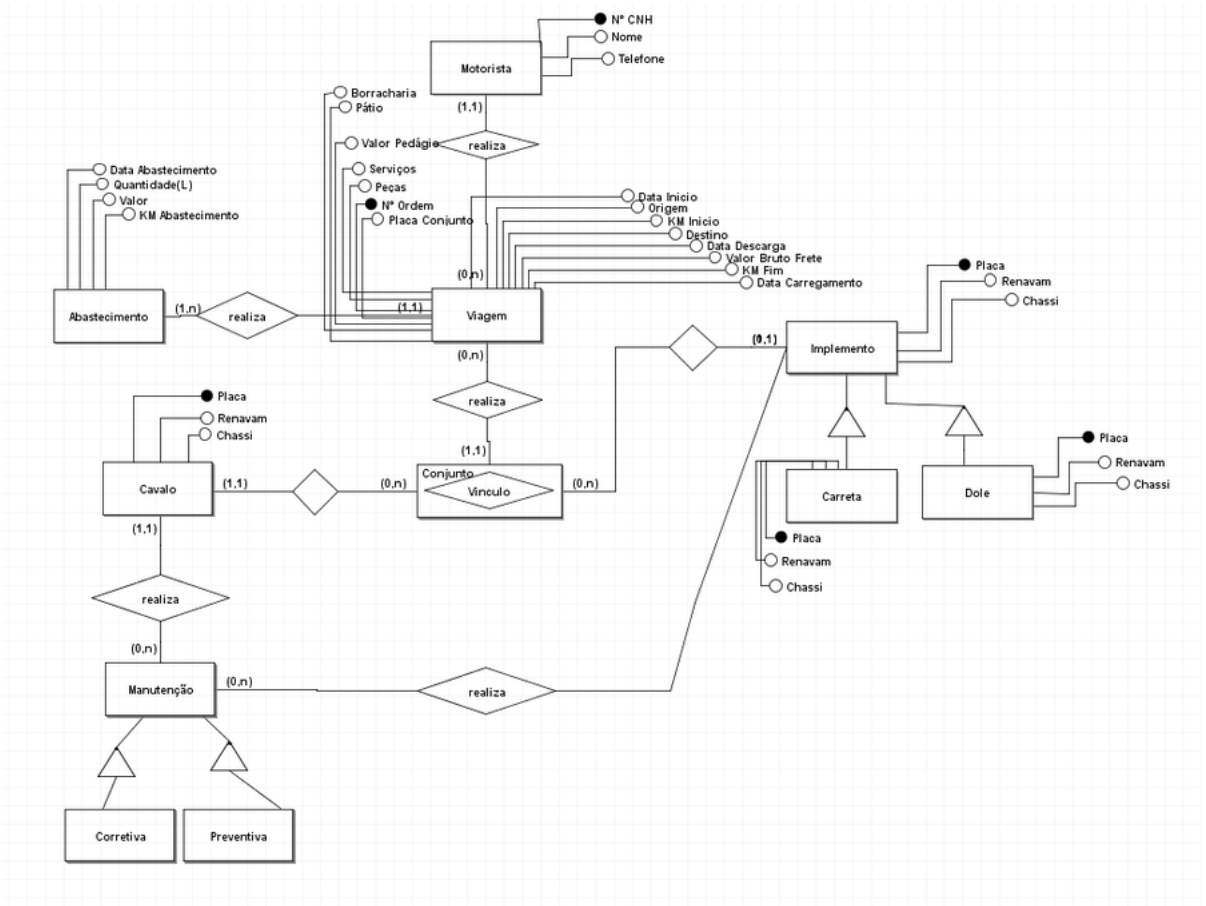
Segundo Somerville, existem duas categorias de entrevistas:

- Entrevistas Abertas: onde a equipe de Engenharia de requisitos explora o conhecimento do stakeholder e fórmula a partir de então, uma visão dos problemas a ser resolvido.
- Entrevistas Fechadas: onde o stakeholder responde perguntas definidas anteriormente pela equipe de Engenharia de Requisitos.

Após as entrevistas com o stakeholder, a conclusão do levantamento do problema em um modelo de dados conceitual será mostrada na imagem abaixo.

Figura 1 - Modelo de Dados Conceitual

Figura 1 – Modelo de Dados Conceitual

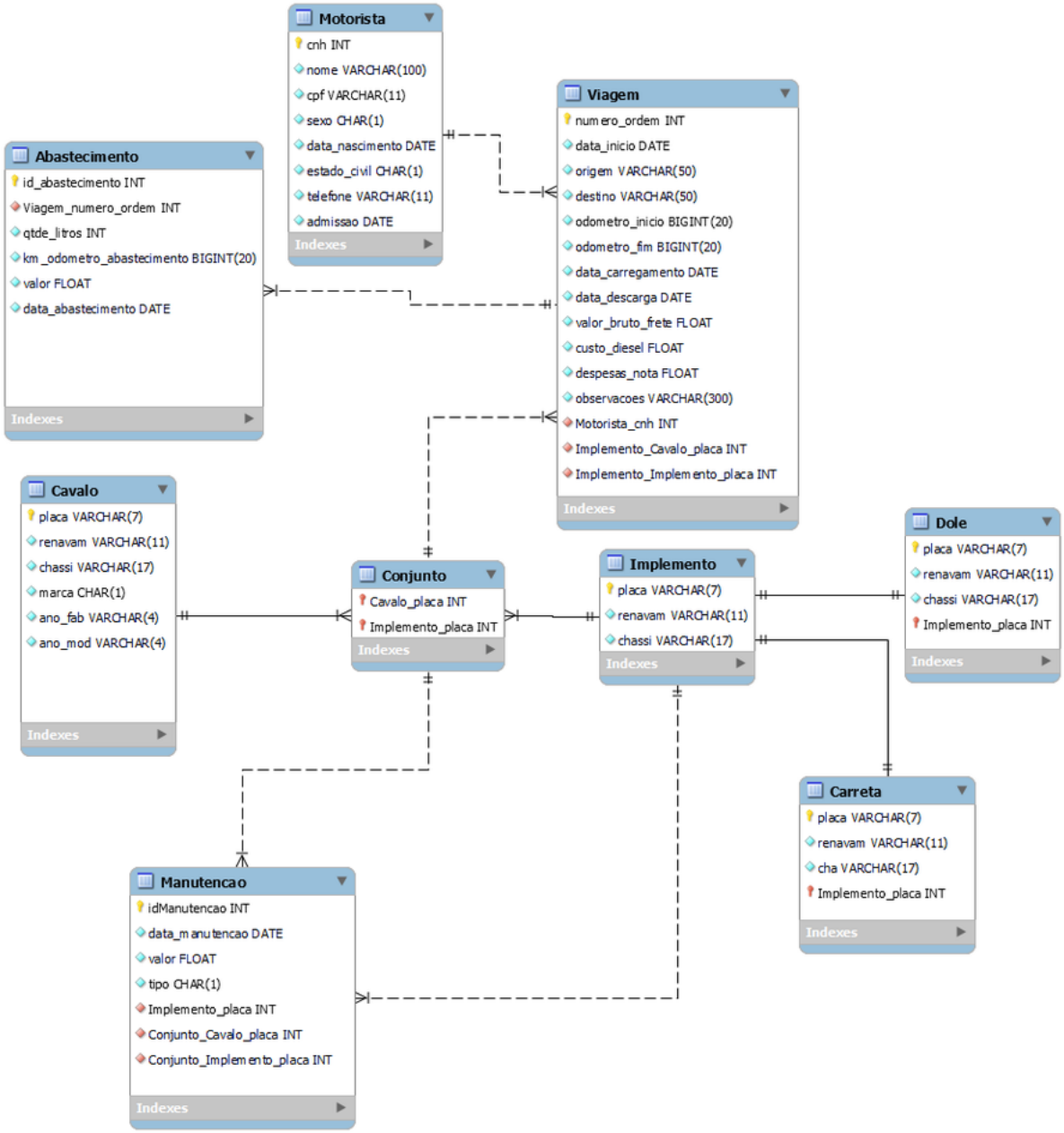


1.4.3 Modelo de Dados Lógico

Após o modelo de dados conceitual, foi gerado o modelo de dados lógico, que será apresentado na figura abaixo.

Figura 2 - Modelo de Dados Lógico

Figura 2 – Modelo de Dados lógico



1.5 Estrutura do Trabalho

O presente trabalho está estruturado em seis capítulos, conforme descrito a seguir.

No Primeiro Capítulo, há a introdução do trabalho, a descrição dos objetivos, gerais, específicos, justificativa e metodologia de trabalho.

No Segundo Capítulo é apresentado o referencial teórico, com a presença dos principais conceitos e trabalhos relacionados ao conteúdo abordado.

O Terceiro Capítulo traz a descrição dos materiais e métodos utilizados ao longo do desenvolvimento do trabalho.

O Quarto Capítulo traz os resultados obtidos no desenvolvimento do trabalho.

O Quinto Capítulo abarca a discussão dos resultados obtidos, conclusão do trabalho e propostas para estudos futuros correlatos.

O Sexto Capítulo traz as referências utilizadas no trabalho.

2 Referencial Teórico

Este capítulo apresenta o referencial teórico do trabalho, trazendo as características e conceitos que serão utilizados para o desenvolvimento do software em questão.

2.1 Mercado Logístico

No primeiro quadrimestre de 2021, o transporte de carga no Brasil teve um aumento de 38% em relação ao mesmo período de 2020, segundo Índice de Movimentação de Cargas do Brasil (Terra, 2021).

O crescimento da movimentação de cargas no Brasil deve-se a diversos fatores, tais como aumento da produção interna, valorização do dólar frente ao real e consequentemente ao aumento das exportações de diversos commodities.

Segundo estimativa da CONAB (Companhia Nacional de Abastecimento), há previsão de aumento de produção de 12,5% na Safra 2021/2022 (GOV, 2022).

Considerando todo o cenário positivo produtivo nacional, a demanda continuará alta para o transporte rodoviário de cargas, fazendo assim, com que os operadores logísticos tenham grandes demandas para realizar.

2.2 Importância do Gerenciamento de Fretes

Com base na tabela de Transporte Rodoviário de Carga por Lotação da Agência Nacional de Transportes Terrestres (ANTT), de 18 de março de 2022, define-se um valor mínimo de frete para tipo de carga, quantidade de eixos, distância em km e nível de periculosidade (DOU, 2022)

Considerando que os preços de fretes do transporte rodoviário multimodal no Brasil é tabelado, sendo regulado via portaria publicado no Diário Oficial da União, o gerenciamento dos custos de fretes tornam-se essenciais.

Com a alta do barril do petróleo nos últimos dois anos, provocados pela pandemia da Covid-19 no mundo, afetando diretamente o preço do óleo diesel, item derivado do petróleo. No Brasil, temos autossuficiência de produção de petróleo, mas não temos capacidade de refino, processo de onde se origina o Diesel.

O gerenciamento frete a frete é importante para demonstrar cada despesa, por exemplo, gasto com diesel em determinada viagem, deixando transparente o lucro líquido a cada viagem realizada.

3 Materiais e Métodos

3.1 Materiais

3.1.1 PyCharm

O PyCharm é um IDE multiplataforma que oferece experiência consistente nos sistemas operacionais Windows, macOS e Linux.

O PyCharm foi projetado por programadores, para programadores, para fornecer todas as ferramentas necessárias para o desenvolvimento produtivo em Python.

O PyCharm fornece conclusão de código inteligente, inspeções de código, realce de erros em tempo real e correções rápidas, com refatorações de código automatizadas e recursos avançados de navegação.(PYCHARM, 2022)

3.1.2 Draw.io

É um software de desenho gráfico de plataforma cruzada gratuita e de código aberto desenvolvido em HTML5 e JavaScript. Sua interface pode ser usada para criar diagramas como fluxogramas, wireframes, diagramas UML, organogramas e diagramas de rede.

O draw.io está disponível on-line como aplicativo da Web entre navegadores e como aplicativo de desktop off-line para Linux, macOS e Windows. Seu aplicativo off-line é construído usando a estrutura Electron. O app da Web não requer login ou registro on-line e pode ser aberto e salvo no disco rígido local. Os formatos de armazenamento e exportação suportados para download incluem PNG, JPEG, SVG e PDF.

3.2 Métodos

3.2.1 Metodologia de Desenvolvimento MTV Django

Um website implementando utilizando framework Django utiliza o modelo MTV.

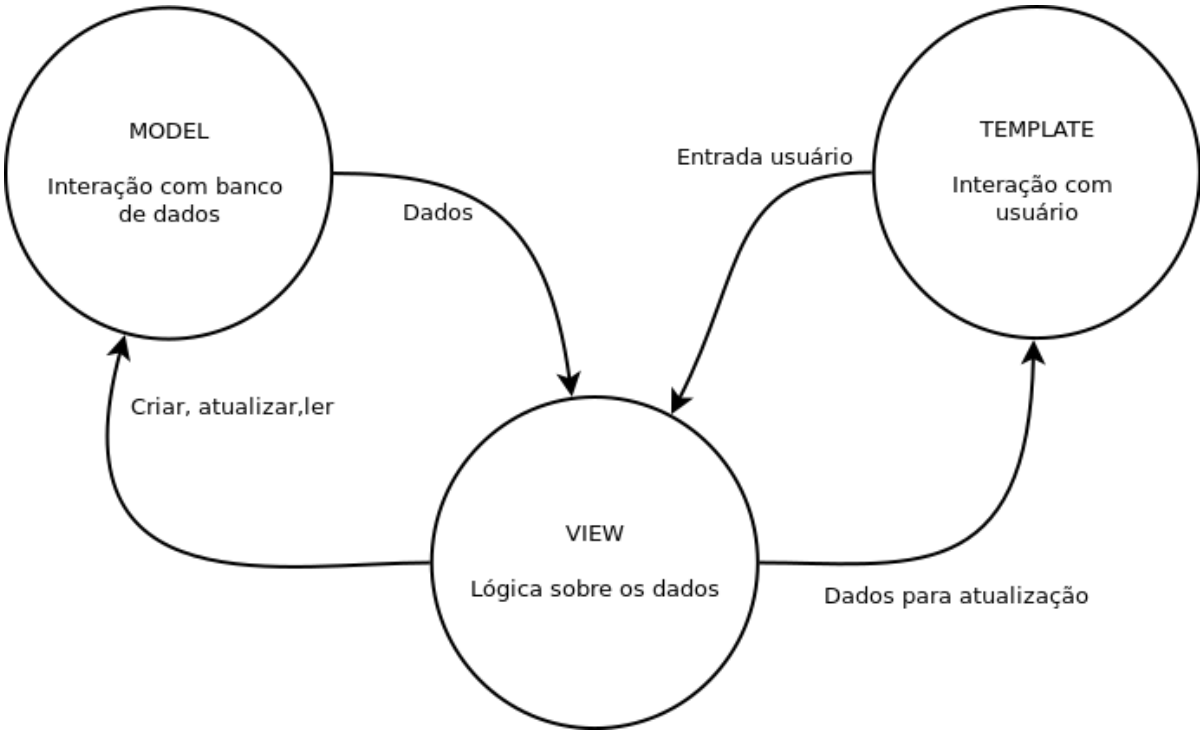
Model(Modelo): o modelo interage diretamente com o banco de dados.

View(Vista): recebe os dados que o modelo recebeu na base de dados, faz o processamento e envia para o Template.

Template(Gabarito): renderiza os dados para o usuário. É através do template que o usuário interage com o site.

Figura 3 - MTV do Django

Figura 3 – MTV do Django



Caderno de Laboratório, 2021. Disponível em: <https://cadernodelaboratorio.com.br/o-modelo-mtv-no-django> </https:>

3.2.2 Python

Python é uma linguagem de programação de alto nível, interpretada de script, imperativa, orientada a objetos, funcional, de tipagem dinâmica e forte. Foi criada por Guido van Rossum em 1991.

Python é uma linguagem de programação que permite trabalhar mais rapidamente e integrar seus sistemas com mais eficiência. (PYTHON, 2022).

A Linguagem Python é mantida por uma organização forte e engajada, a PSF (Python Software Foundation), que tem como função dedicada o avanço tecnológico de código aberto relacionado à linguagem de programação, lançando novas versões e atualizações.

3.2.3 Django

Django é um framework web Python de alto nível que permite o rápido desenvolvimento de sites seguros e de fácil manutenção. Construído por desenvolvedores experientes, o Django cuida de grande parte do trabalho de desenvolvimento web, para que você possa se concentrar em escrever seu aplicativo sem precisar reinventar a roda. É gratuito e de código aberto, tem uma comunidade próspera e ativa, ótima documentação e muitas opções de suporte gratuito e pago. (DJANGO, 2022)

3.2.4 Banco de Dados MYSQL

Banco de dados é um local no qual é possível armazenar informações, para consulta ou utilização, quando necessário. Todos são constituídos por três elementos básicos: campos, registros e tabelas. Cada banco de dados possui complexas estruturas internas de funcionamento, uma diferente da outra, a fim de facilitar o acesso aos elementos do banco de dados foi criada e distribuída pela IBM em 1981 uma linguagem de consulta, o SQL, que se tornou uma linguagem de acesso aos bancos de dados muito articulada e funcional que pode ser empregada em computadores de arquiteturas totalmente diferentes e é a linguagem utilizada pelo MySQL (FERRARI, 2007).

MySQL é o banco de dados de código aberto mais popular do mundo, que possibilita a entrega econômica de aplicações de banco de dados confiáveis, de alto desempenho e escaláveis, com base na Web e incorporadas (ORACLE, 2021).d

Nele ficam armazenadas todas as informações dos usuários, documentos e caminhos de imagens mantidos pelo sistema proposto. As operações suportadas por esse banco são inclusões, exclusões, atualizações de registro, alterações de estrutura e outras.

3.2.5 Bootstrap

Bootstrap é o mais popular framework (Junção e Abstração de códigos com funcionalidades genéricas) open-source de HTML (HyperText Markup Language - em tradução livre, Linguagem de Marcação de Texto), CSS (Cascading Style Sheets - em tradução livre Folhas de estilo em cascata) e JS (Javascript - Linguagem de programação).,

Bootstrap foi criado no Twitter em meados de 2010 por Mark Otto e Jacob Thornton no Twitter para facilitar as manutenções internas e eliminar inconsistências.(BOOTSTRAP, 2022).

3.2.6 Modelagem do Software

3.2.6.1 Especificação

Nesta seção serão apresentados os requisitos funcionais (RF), requisitos não funcionais (RFN), sua rastreabilidade e o seus casos de uso.

Na tabela 1, tem-se descritos os requisitos funcionais do sistema.

Tabela 1 – Requisitos Funcionais

Requisitos Funcionais	**Caso de Uso**
RF01: O sistema deverá permitir a manutenção do cadastro de motorista	**UC01**
RF02: O sistema deverá permitir emitir relatórios de motoristas.	**UC02**
RF03: O sistema deverá permitir a manutenção do cadastro de fretes	**UC03**
RF04: O sistema deverá permitir relatórios dos fretes	**UC04**
RF05: 0 sistema deverá permitir a manutenção do cadastro de abastecimento	**UC05**
RF06:O sistema deverá permitir emitir relatório de abastecimento	**UC06**
RF07: O sistema deverá permitir a manutenção do cadastro de manutenções	**UC07**
RF08: O sistema deverá permitir relatórios de manutenções	**UC08**
RF09:O sistema deverá permitir a manutenção do cadastro de Implemento	**UC09**
RF10: O sistema deverá permitir emitir relatórios dos implementos	**UC10**
RF11: O sistema deverá permitir a manutenção dos cavalos-mecânicos	**UC11**
RF12: O sistema deverá permitir emitir relatórios dos cavalos-mecânicos	**UC12**

Na tabela 2, tem-se descritos os requisitos não funcionais do sistema.

Tabela 2 – Requisitos não funcionais

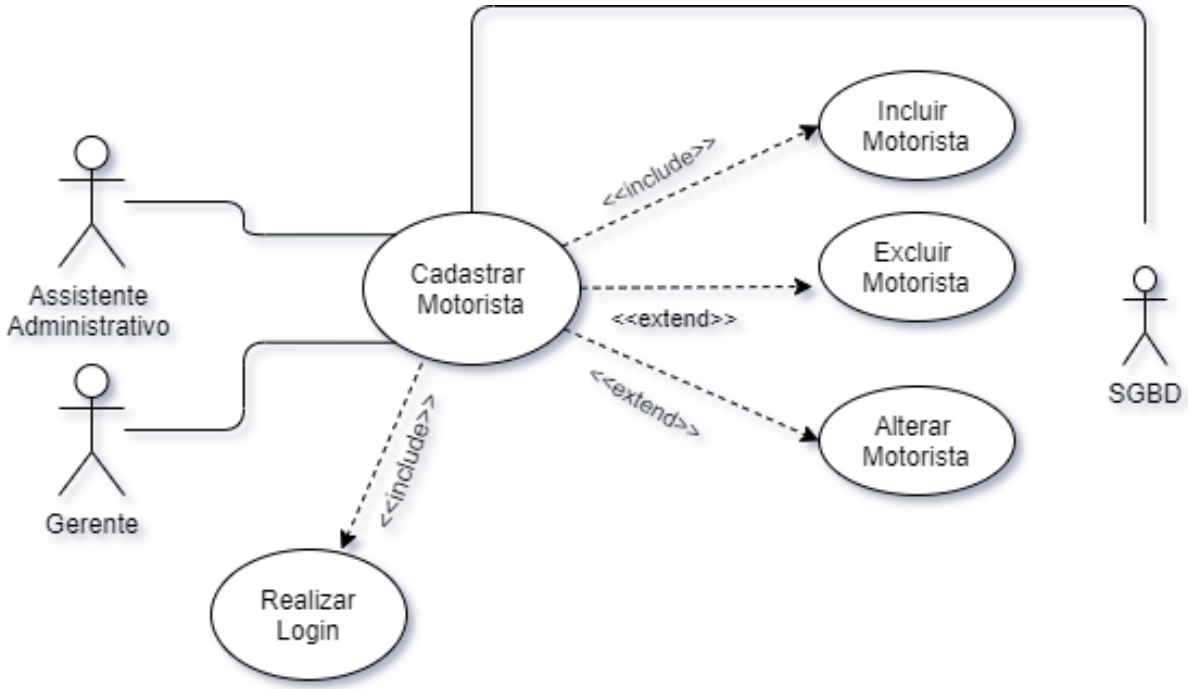
Requisitos não funcionais
RNF01: O sistema deverá executar na web.

3.2.6.2 Diagramas de caso de uso

Esta seção inclui os diagramas de caso de uso do sistema.
Na figura 4, encontra-se o diagrama de caso de uso Cadastrar Motorista.

Figura 4 - Diagrama de Caso de Uso Cadastrar Motorista

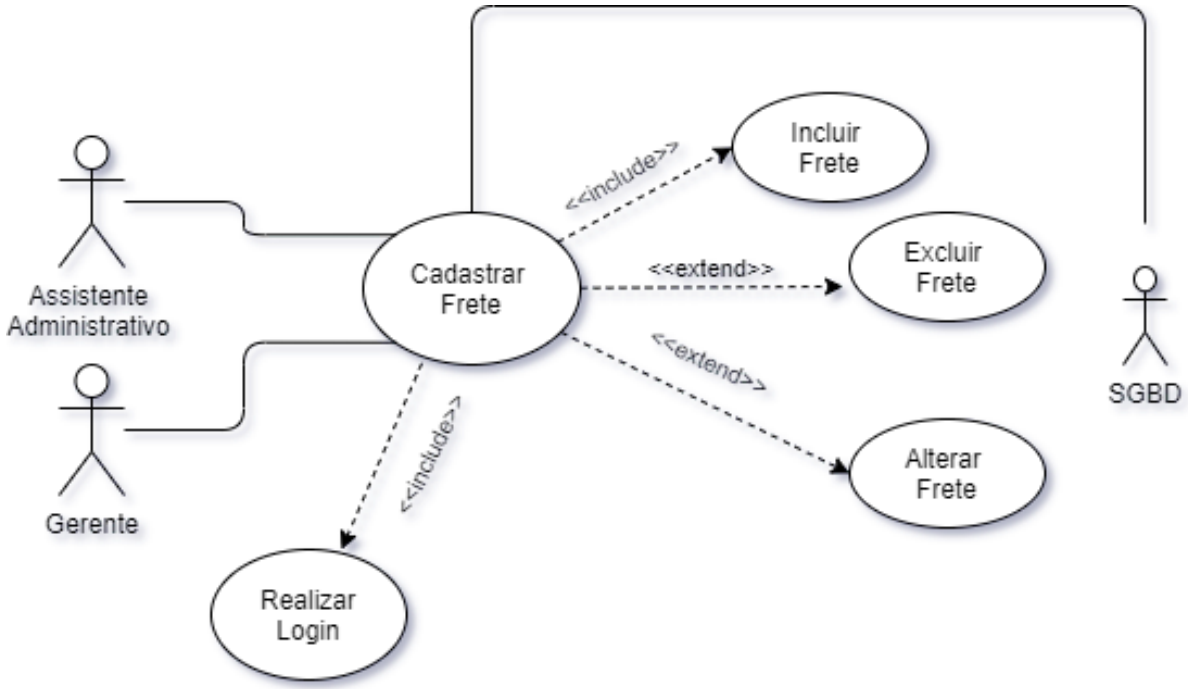
Figura 4 – Cadastrar Motorista



Na figura 5, encontra-se o diagrama de caso de uso Cadastrar Frete.

Figura 5 - Diagrama de Caso de Uso Cadastrar Frete

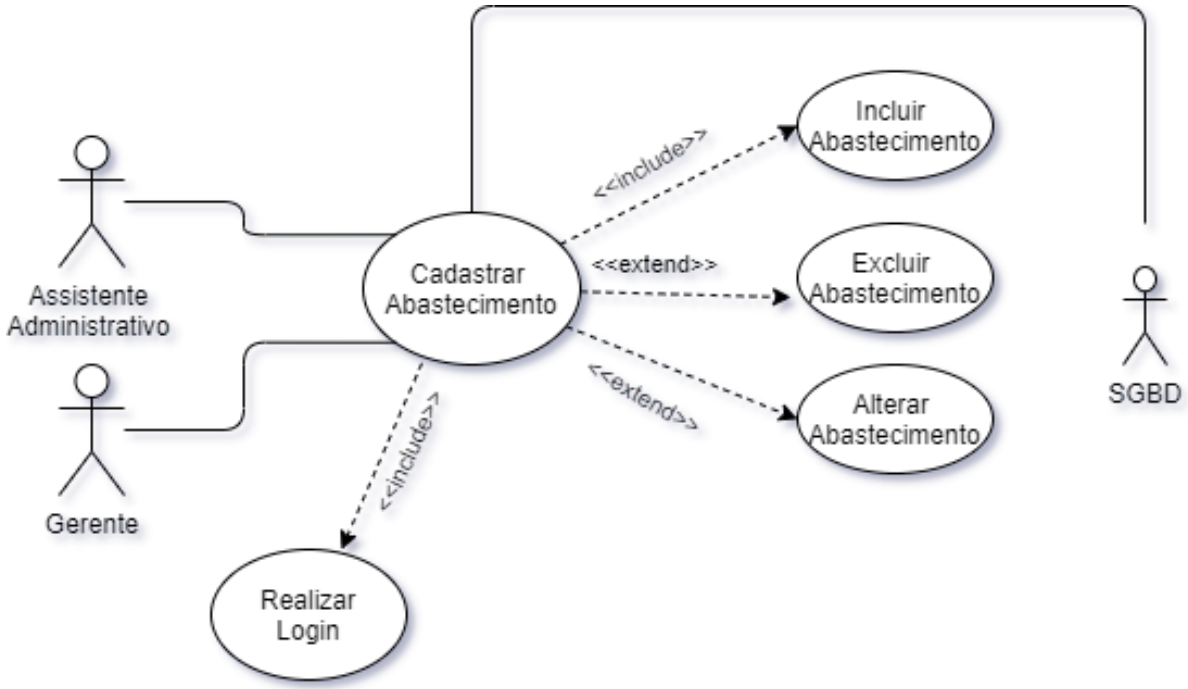
Figura 5 – Cadastrar Frete



Na figura 6, encontra-se o diagrama de caso de uso Cadastrar Abastecimento.

Figura 6 - Diagrama de Caso de Uso Cadastrar Abastecimento

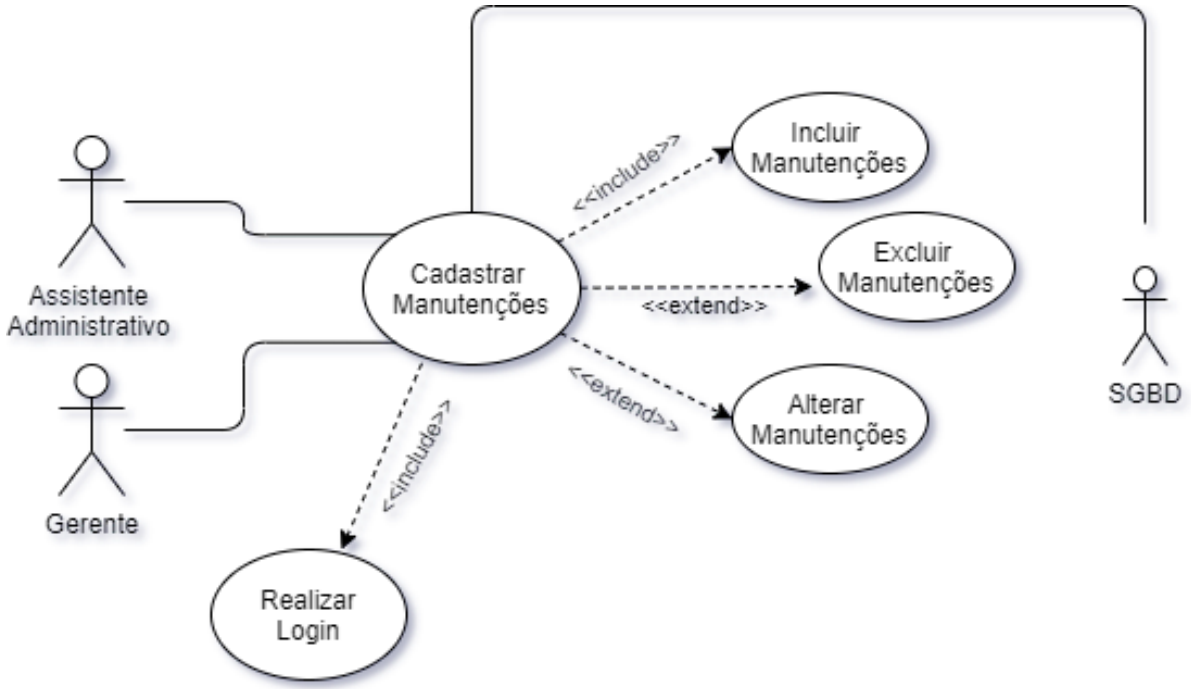
Figura 6 – Cadastrar Abastecimento



Na figura 7, encontra-se o diagrama de caso de uso Cadastrar Frete.

Figura 7 - Diagrama de Caso de Uso Cadastrar Manutenções

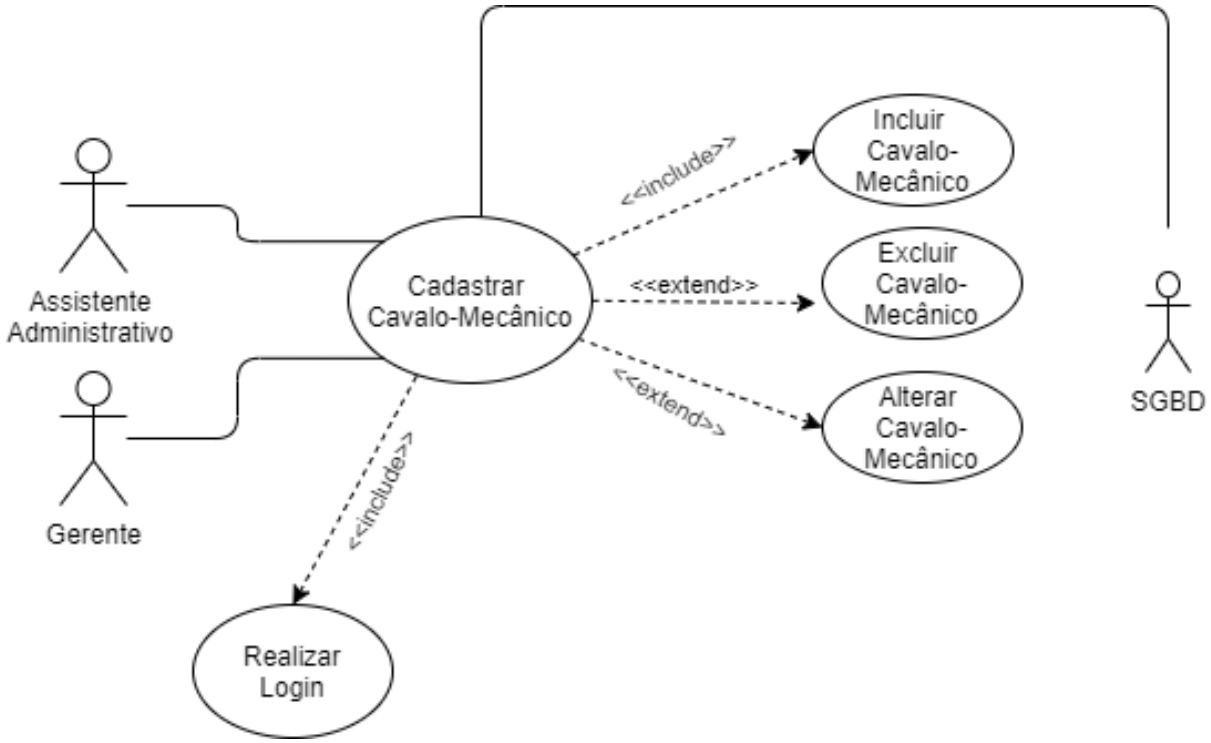
Figura 7 – Cadastrar Manutenções



Na figura 8, encontra-se o diagrama de caso de uso Cadastrar Cavalos-Mecânico.

Figura 8- Diagrama de Caso de Uso Cadastrar Cavalo-Mecânico

Figura 8 – Cadastrar Cavalo-Mecânico



4 Resultados

Esta seção será responsável por descrever os resultados obtidos no desenvolvimento do software proposto por esse projeto.

4.1 Desenvolvimento do software web

O resultado obtido por este trabalho foi o desenvolvimento é um sistema aparentemente simples, que facilitará muito na tomada de decisão da empresa.

As visitas que realizei e a convivência com o gerente no dia a dia da empresa mostrou muitas falhas, que este software web resolverá, assim que implantado, como, por exemplo, qual veículo foi mais econômico em abastecimentos em um mês, ou qual veículo tem menos problemas durante os trajetos das viagens.

O software web ajudará fiscalização por parte do gerente e do dono da empresa, pois, ficará disponível (24h) por dia via internet, de onde estiverem, estarão acompanhando o rendimento e trabalho dos motoristas e suas viagens.

4.2 Principais Interfaces

Figura 9 - Tela Inicial

Figura 9 – Tela Inicial

Figura 10 - Tela Opções do Motorista

Figura 10 – Tela Opções do Motorista

Figura 11 - Cadastrar Motorista

Figura 11 – Tela Cadastrar Motorista

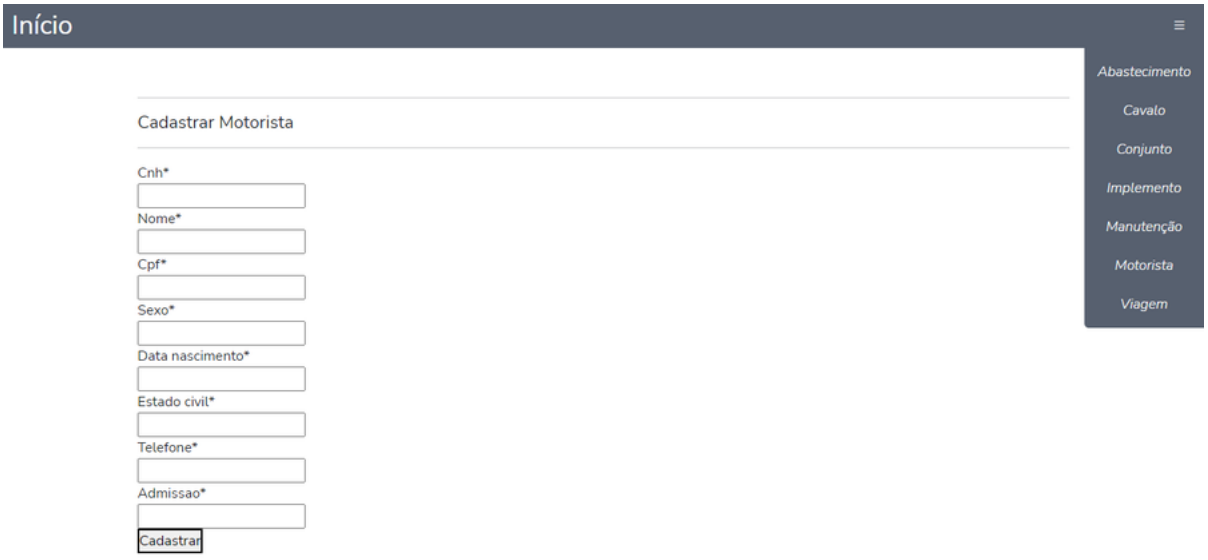


Figura 12 - Listagem Motoristas Cadastrados

Figura 12 – Telas Listagem Motoristas Cadastrados

5 Conclusão

Este trabalho apresenta um sistema de informação com o objetivo de solucionar um problema complexo da Mota Transportes. O sistema partiu dos requisitos levantados “in loco” considerando as dificuldades relatadas pelo stakeholder (gerente) da empresa. A escolha da tecnologia foi feita pensando em melhorias e novas funcionalidades para o sistema. Como o Django oferece a facilidade de integração de novos “apps”, trabalhando de forma modular e de fácil integração, a evolução do software será contínua de acordo com a necessidade.

Conclui-se com este trabalho que, um gerenciamento feito por um software, reúne, facilita e desmistifica uma empresa, seja ela de qual porte e de, qual segmento. Este agregará valor e condições de uma boa gestão para a Mota Transportes, o que pode facilitar a tomada de decisão, o investimento em novos veículos, gerando empregos e renda.

5.1 Trabalhos Futuros

Como proposta de trabalho futuro, seria integrar o módulo de fretes de uma plataforma que terceiriza os fretes para a Mota Transportes, não sendo mais necessário realizar o cadastro manual de fretes no sistema, que seria alimentado por meio de uma API (*Application Programming Interface),* buscando os dados no software de um terceiro, economizando esforço humano para cadastro de viagens para controle interno.

6 Referencias

BOOTSTRAP. **About.**2022. Disponível em: https://getbootstrap.com/docs/5.1/about/overview/. Acesso em: 20 Fev. 2022.

DJANGO. **Documentation**.2022 .Disponível em: https://docs.djangoproject.com/pt-br/4.0/. Acesso em: 05 Jan. 2022

FERRARI, Fabrício Augusto. **Crie banco de dados em MySQL**.São Paulo: Digerati Books,2007.

GOV.BR. **PORTARIA Nº 169, DE 18 DE MARÇO DE 2022.** 2022. Disponível em: https://www.in.gov.br/en/web/dou/-/portaria-n-169-de-18-de-marco-de-2022-387043278. Acesso em: 01 Abr. 2022

GOV.BR. **Produção agrícola brasileira deve crescer 12,5% nesta safra.** 2022. Disponível em: https://www.gov.br/pt-br/noticias/agricultura-e-pecuaria/2022/01/producao-agricola-brasileira-deve-crescer-12-5-nesta-safra#:∼:text=O%20Brasil%20deve%20produzir%20um,em%20rela%C3%A7%C3%A3o%20%C3%A0%20safra%20passada. Acesso em: 25 Fev. 2022.

JETBRAINS.**Install PyCharm.** 2022. Disponível em: https://www.jetbrains.com/help/pycharm/installation-guide.html. Acesso em: 02 Mar. 2022.

PYTHON. **About.** 2022**.** Disponível em: https://www.python.org/about/. Acesso em: 03 Mar. 2022

SOMMERVILLE, Ian. **Engenharia de Software.** 9. ed. São Paulo: Pearson, 2011. 900 p.

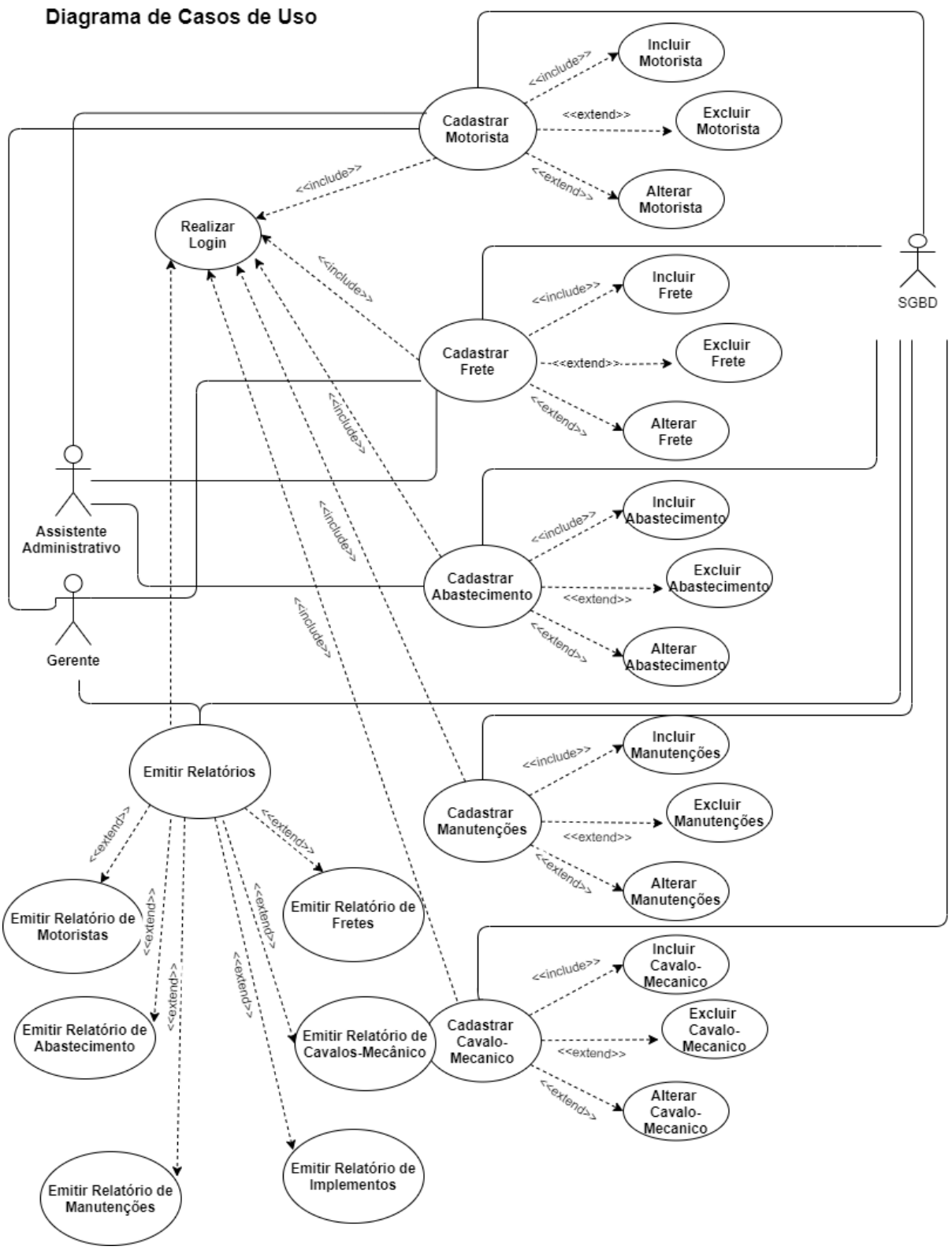
TERRA. **Transporte cresce 38% em 2021 e logística 4.0 é considerada fator de crescimento. 2021. Disponível em:**https://www.terra.com.br/noticias/transporte-cresce-38-em-2021-e-logistica-40-e-considerada-fator-de-crescimento,5fcc4d71fab9d0e0005cdef1ca5d9ddchx2ldh4q.html. Acesso em: 20 de Nov. 2021

Apêndices

APÊNDICE A - Detalhamento dos Casos de Uso

Na figura 13 está descrito todo o diagrama de casos de uso e detalhado caso a caso.

Figura 13 – Diagrama Casos de Uso



Na tabela 3, apresenta-se o caso de uso Cadastrar Motorista

Tabela 3 – Cadastrar Motorista

UC01	O sistema deverá permitir o cadastro de motoristas
Descrição	Permite ao gerente incluir, alterar ou excluir um motorista
Ator	Gerente
Pré - Condição	Gerente fazer *login* no sistema.
Fluxo Principal	1. Sistema solicita ao gerente (nome de usuário, senha, etc) 2. Sistema valida os dados informados pelo gerente. 3. Gerente no botão “Salvar” 4. Sistema persiste os dados no banco de dados. 5. Sistema mostra mensagem de sucesso “Motorista cadastrado com sucesso”
Cenário - Visualização	Sistema mostra lista demotoristas cadastrados atualmente.
Cenário - Edição	1. Gerente seleciona motorista a ser editado. 2. Sistema mostra dados do motorista selecionado. 3. Gerente altera os dados e clica no botão “Salvar” para atualizar os dados. 4. Sistema mostra mensagem de “Motorista Alterado com sucesso”

UC01	O sistema deverá permitir o cadastro de motoristas
Cenário - Inclusão	1. Gerente incluir um novo motorista 2. Sistema mostra motorista incluído
Cenário - Exclusão	1. Sistema mostra registros cadastrados; 2. Gerente seleciona um registro para exclusão; 3. Sistema exclui o registro e mostra os registros restantes.
Pós-condição	Gerente visualizou, editou, incluiu ou excluiu um motorista

Na tabela 4, apresenta-se o caso de uso Emitir Relatório de Motoristas

Tabela 4 – Emitir Relatório de Motoristas

UC02	O sistema deverá permitir ao gerente emitir relatórios de motoristas
Descrição	Permite ao gerente gerar um relatório com informações dos motoristascadastrados
Ator	Gerente

Na tabela 5, apresenta-se o caso de uso Cadastrar Fretes

Tabela 5 – Cadastrar Fretes

UC03	O sistema deverá permitir o cadastro de fretes
Descrição	Permite ao gerente e ao auxiliar administrativo incluir, alterar ou excluir um frete
Ator	Gerente, Auxiliar Administrativo
Pré-condição	Ator deve fazer *login* no sistema.
Fluxo principal	1. Ator informa dados solicitados ao sistema 2. Sistema valida dados informados 3. Ator solicita o salvamento do frete via botão “Salvar” 4. Sistema persiste os dados no banco de dados 5.Sistema mostra mensagem de sucesso “Frete cadastrado com sucesso”
Cenário - Visualização	1. Sistema mostra fretes cadastrados
Cenário - Inclusão	1. Ator incluir um novo frete 2. Sistema verifica a existência do frete no sistema. 3. Sistema mostra frete incluído
Cenário - Exclusão	1. Sistema mostra registros cadastrados; 2. Ator seleciona um registro para exclusão; 3. Sistema exclui o registro e mostra os registros restantes.
Pós-condição	Atores atualizaram, incluíram, editaram ou excluíram um frete

Na tabela 6, apresenta-se o caso de uso Emitir Relatório de Fretes

Tabela 6 – Emitir Relatório de Fretes

UC04	O sistema deverá permitir a emissão de relatório de fretes
Descrição	Permite ao gerente emitir relatórios com informações dos fretes

UC04	O sistema deverá permitir a emissão de relatório de fretes
Ator	Gerente

Na tabela 7, apresenta-se o caso de uso Cadastrar Abastecimento

Tabela 7 – Cadastrar Abastecimento

UC05	O sistema deverá permitir o cadastro de abastecimento
Descrição	Permite ao gerente e ao auxiliar administrativo incluir, alterar ou excluir um abastecimento
Ator	Gerente, Auxiliar Administrativo
Pré-condição	Ator deve fazer *login* no sistema.
Fluxo principal	1. O Ator informa os dados solicitados pelo sistema 2. O sistema valida os dados 3.Ator solicita o cadastramento via botão “Cadastrar Abastecimento” 4. O sistema mostra “Abastecimento Cadastrado com Sucesso”
Cenário - Visualização	1. Sistema mostra abastecimentos cadastrados.
Cenário - Edição	1.Gerenteseleciona abastecimento a ser editado. 2. Sistema mostra dados do abastecimento selecionado. 3.Gerentealtera os dados e clica no botão “Salvar” para atualizar os dados. 4. Sistema mostra mensagem de “Abastecimento alterado com sucesso”

UC05	O sistema deverá permitir o cadastro de abastecimento
Cenário - Exclusão	1) Gerenteseleciona abastecimento a ser excluído. 2) Gerente solicita exclusão pelo botão“Excluir” para atualizar os dados. 3) Sistema mostra mensagem de “Abastecimento excluído com sucesso”
Pós-condição	Atores atualizaram, incluíram, editaram ou excluíram um abastecimento

Na tabela 8, apresenta-se o caso de uso Emitir Relatório de Abastecimentos

Tabela 8 – Emitir Relatório de Abastecimentos

UC06	O sistema deverá permitir a emissão de relatório de abastecimento
Descrição	Permite ao gerente emitir relatórios com informações dos fretes
Ator	Gerente

Na tabela 9, apresenta-se o caso de uso Cadastrar Manutenções.

Tabela 9 – Cadastrar Manutenções

UC07	O sistema deverá permitir o cadastro de manutenções
Descrição	Permite aogerentee ao auxiliar administrativo incluir, alterar ou excluir uma manutenção.
Ator	Assistente Administrativo, Gerente
Pré-condição	Ator deve fazerloginnosistema.
Fluxo principal	1. Ator fornece os dados solicitados pelo sistema 2. Sistema valida os dados 3. Ator solicita o cadastramento via botão “Cadastrar Manutenção” 4. Sistema retorna mensagem “Manutenção Cadastrada com Sucesso”
Cenário - Visualização	1. Sistema mostra todas as manutenções cadastradas.
Cenário - Edição	1. Ator fornece os dados solicitados pelo sistema 2. Sistema valida os dados 3. Ator solicita a edição via botão “Editar Manutenção” 4. Sistema retorna mensagem “Manutenção Editadacom Sucesso“
Cenário - Inclusão	1. Ator fornece os dados solicitados pelo sistema 2. Sistema valida os dados 3. Ator solicita a inclusão via botão “Incluir Manutenção” 4. Sistema retorna mensagem “Manutenção Incluída com Sucesso”

UC07	O sistema deverá permitir o cadastro de manutenções
Cenário - Exclusão	1. Ator seleciona Manutenção a ser excluída. 2. Sistema valida os dados 3. Ator solicita a edição via botão “Editar Manutenção” 4. Sistema retorna mensagem “Manutenção Editadacom Sucesso“
Pós-condição	Atores atualizaram, incluíram, editaram ou excluíram uma manutenção

Na tabela 10, apresenta-se o caso de uso Emitir Relatório de Manutenções.

Tabela 10 – Emitir Relatório de Manutenções

UC08	O sistema deverá permitir a emissão de relatório de manutenções
Descrição	Permite aogerentee ao auxiliar administrativo emitir relatórios de manutenções cadastradas no sistema
Ator	Assistente Administrativo, Gerente

Na tabela 11, apresenta-se o caso de uso Emitir Relatório Implemento

Tabela 11 – Emitir Relatório Implemento

UC10	O sistema deverá permitir a emissão de relatórios de implementos
Descrição	O sistema disponibilizará emissão de relatórios dos implementos cadastrados no sistema
Ator	Assistente Administrativo, Gerente

Na tabela 12, apresenta-se o caso de uso Cadastrar Cavalo Mecânico

Tabela 12 – Cadastrar Cavalo Mecânico

UC11	O sistema deverá permitir a emissão de relatórios de implementos
Descrição	O sistema disponibilizará emissão de relatórios
Ator	Assistente Administrativo, Gerente
Pré-condição	Ator fazer *login* no sistema.
Fluxo principal	1. Ator informa os dados solicitados pelo sistema 2. Sistema valida dados informados 3. Ator solicita cadastramento via botão “Cadastrar Cavalo Mecanico” 4.Sistema retorna mensagem “Cavalo Mecânico cadastrado com sucesso”
Cenário - Visualização	
Cenário - Edição	
Cenário - Inclusão	
Cenário - Exclusão	
Pós-condição	

Na tabela 13, apresenta-se o caso de uso Cadastrar Cavalo Mecânico

Tabela 13 – Emitir Relatório Cavalo Mecânico

UC12	O sistema deverá permitir a emissão de relatórios dos Cavalos-mecânicos
Descrição	O sistema disponibilizará emissão de relatórios para cada cavalo-mecanico
Ator	Assistente Administrativo, Gerente

APÊNDICE B - Dicionário de dados

Tipo de Entidade: Motorista

Conceito: Representa o funcionário responsável por dirigir um conjunto;

Especializado em:

Sinônimos: Motorista do Conjunto

Atributos: CNH, Nome, CPF, Sexo, Data de Nascimento, Estado Civil, Telefone, Data de Admissão

Tabela 14 – Dicionário de Dados - Motorista

Nome	Conceito	Tipo	Tam.	Classificação	Componentes	Restrição de Domínio
CNH	Código Identificador do Usuário	Numérico	11	obr.s, mono, id		
Nome	Nome de Usuário	Alfanumérico	100	obr.s, mono, id		
CPF	Certificado de Pessoa Física	Alfanumérico	14	obr.s, mono, id		
Sexo	Sexo do Usuário	Alfanumérico	1	obr.s, mono, id		M (masculino) F (feminino)
Data de Nascimento	Data de Nascimento do Usuário	Data		obr.s, mono, id		
Estado Civil	Estado Civil do Usuário	Alfanumérico	1	obr.s, mono, id		C (casado) S (solteiro) D (divorciado) V (viúvo)
Telefone	Númer(o)s de telefone do usuário	Numérico	10	obr.s, mult, id		
Data de Admissão	Data de Contratação do Funcionário	Data		obr.s, mono, id		

Nome	Conceito	Tipo	Tam.	Classificação	Componentes	Restrição de Domínio

Tipo de Relacionamento: Realiza; Motorista realiza viagem

- –
  - Conceito: relacionamento necessário para saber qual motorista dirige em determinada Viagem; um Motorista realiza Viagem; um Motorista pode fazer uma ou nenhuma Viagem; uma viagem só pode ter um Motorista
  - Atributos: não tem atributos relevantes

Tipo de Entidade: Viagem

Conceito: Representa a viagem a ser realizada por um Motorista;

Especializado em:

Sinônimos: Viagem para um Motorista

Atributos: Carta frete, Data Inicio, Data de Carregamento, Origem, Destino, KM Inicio, Valor Frete, Custo Diesel, Despesas com Nota, Obsevações

Tabela 15 – Dicionário de Dados - Viagem

Nome	Conceito	Tipo	Tam.	Classificação	Componentes	Restrição de Domínio
Carta Frete	Código Identificador da Viagem	Numérico	10	obr,s,mono,id,p		

Nome	Conceito	Tipo	Tam.	Classificação	Componentes	Restrição de Domínio
Data Inicio	Data do Início da Viagem	Data		obr,s,mono,id,p		
Data do Carregamento	Data que o Conjunto foi carregado	Data		obr,s,mono,id,p		
Origem	Cidade onde foi realizado o carregamento	Alfanumérico	50	obr,s,mono,id,p		
Destino	Cidade onde será realizada a descarga	Alfanumérico	50	obr,s,mono,id,p		
KM Inicio	KM no Odômetro do Cavalo Trator Inicio da Viagem	Numérico	7	obr,s,mono,id,p		
KM Fim	KM no Odômetro do Cavalo Trator Fim da Viagem	Numérico	7	obr,s,mono,id,p		
Valor Frete	Valor Bruto do Frete	Numérico	7	obr,s,mono,id,p		
Custo Diesel	Despesa com Diesel na Viagem	Numérico	7	obr,s,mono,id,p		
Despesas com Nota	Outras despesas necessárias durante a Viagem	Numérico	7	obr,s,mono,id,p		
Observações	Descrição de alguma informação relevante	Alfanumérico	500	obr,s,mono,id,p		

Tipo de Relacionamento: Contém; Uma Viagem contém um conjunto

- – * Conceito: relacionamento necessário para saber qual conjunto está relacionado a viagem; uma Viagem contém um Conjunto; um Conjunto

pode fazer nenhuma Viagem ou várias Viagens; uma viagem só pode ter um Motorista

* Atributos: não tem atributos relevantes

Tipo de Entidade: Conjunto
Conceito: Representa o Conjunto que realiza uma viagem;
Especializado em:
Sinônimos: Conjunto para Viagem
Atributos: Número do Conjunto

Tabela 16 – Dicionário de Dados - Conjunto

Nome	Conceito	Tipo	Tamanho	Classificação	Componentes	Restrição de Domínio
Número do Conjunto	Código Identificador do Conjunto	Numérico	5			

- –
  * Conceito: relacionamento necessário para saber qual conjunto está relacionado a viagem; uma Viagem contém um Conjunto; um Conjunto pode fazer nenhuma Viagem ou várias Viagens; uma viagem só pode ter um Motorista
  * Atributos: não tem atributos relevantes

Tipo de Entidade: Implemento
Conceito: Representa o Implemento a compor o Conjunto;
Especializado em: Carreta e Dole
Sinônimos: Implemento do Conjunto
Atributos: Placa

Tabela 17 – Dicionário de Dados - Implemento

Nome	Conceito	Tipo	Tam.	Classificação	Componentes	Restrição de Domínio
Placa	Código identificador do Implemento	Alfam.	8	obr,s,mono,id,p		

- –
  - Conceito: relacionamento necessário para saber do que é composto o Implemento; uma Implemento pode compor nenhum ou vários Conjuntos; um Implemento pode ter uma ou mais carretas e um dole;
  - Atributos: não tem atributos relevantes

Tipo de Entidade: Carreta
Conceito: Representa a Carreta que compõe o Implemento;
Especializado em:
Sinônimos: Carreta do Implemento
Atributos: Placa, Renavam, Chassi, Tipo

Tabela 18 – Dicionário de Dados - Carreta

Nome	Conceito	Tipo	Tam.	Classificação	Componentes	Restrição de Domínio
Placa	Código identificador da Carreta	Alfanum.	8	obr,s,mono,id,p		

Nome	**Conceito**	**Tipo**	**Tam.**	**Classificação**	**Componentes**	**Restrição de Domínio**
Renavam	Identificador	Alfanum.	20	obr,s,mono,nid,p		
Chassi	Identificador do Chassi	Alfanum.	20	obr,s,mono,nid,p		
Tipo	Tipo de Carreta	Alfanum.	1	obr,s,mono,nid,p		G(Graneleiro) C(Caçamba)

- –
  - Conceito: especialização necessária para saber do que tipo de carreta compõe o Implemento; uma Carreta pode compor nenhum ou vários Implementos; um Implemento pode ter uma ou mais carretas;
  - Atributos: não tem atributos relevantes

Sinônimos: Dole do Implemento
Atributos: Placa, Renavam, Chassi

Tabela 19 – Dicionário de Dados - Dole

Nome	**Conceito**	**Tipo**	**Tam.**	**Classificação**	**Componentes**	**Restrição de Domínio**
Placa	Código identificador da Carreta	Alfanum.	8	obr,s,mono,id,p		

Nome	**Conceito**	**Tipo**	**Tam.**	**Classificação**	**Componentes**	**Restrição de Domínio**
Renavam	Identificador	Alfanum.	20	obr,s,mono,nid,p		
Chassi	Identificador do Chassi	Alfanum.	20	obr,s,mono,nid,p		

- –
  - Conceito: especialização necessária para saber do que tipo de dole compõe o Implemento; um Dole pode compor nenhum ou vários Implementos; um Implemento pode ter apenas um Dole;
  - Atributos: não tem atributos relevantes

Tipo de Entidade: Cavalo
Conceito: Representa o Cavalo que compõe o Conjunto;
Especializado em:
Sinônimos: Cavalo do Conjunto
Atributos: Placa, Renavam, Chassi, Marca, Ano Fabricação, Ano Modelo

Tabela 20 – Dicionário de Dados - Cavalo-Mecânico

Nome	**Conceito**	**Tipo**	**Tam.**	**Classificação**	**Componentes**	**Restrição de Domínio**
Placa	Código identificador da Carreta	Alfanum.	8	obr,s,mono,id,p		

Nome	**Conceito**	**Tipo**	**Tam.**	**Classificação**	**Componentes**	**Restrição de Domínio**
Renavam	Identificador	Alfanum.	20	obr,s,mono,nid,p		
Chassi	Identificador do Chassi	Alfanum.	20	obr,s,mono,nid,p		
Marca	Marca do Cavalo	Alfanum.	1	obr,s,mono,nid,p		V(Volvo) S(Scania) D(Daf) M(Mercedes)
Ano Fabricação	Ano de Fabricação	Data				
Ano Modelo	Ano do Modelo					

- –
  - Conceito: relacionamento necessário para saber que Cavalo compõe o Conjunto; uma Cavalo pode compor nenhum ou vários Implementos; um Implemento pode ter uma ou mais carretas;
  - Atributos: não tem atributos relevantes

APÊNDICE C - Códigos de desenvolvimento

Models

Figura 14 – Model Abastecimento

abastecimento > models.py > ...

```python
from django.db import models
from viagem.models import Viagem

class Abastecimento(models.Model):
    qtde_litros = models.IntegerField()
    km_odometro_abastecimento = models.BigIntegerField()
    valor = models.FloatField()
    data_abastecimento = models.DateField()
    viagem_numero_ordem = models.OneToOneField(Viagem, models.DO_NOTHING, db_column='Viagem_numero_ordem', primary_key=True)
    class Meta:
        managed = False
        db_table = 'abastecimento'

```

Model Abastecimento

Figura 15 – Model Cavalo

```python
from django.db import models

class Cavalo(models.Model):
    placa = models.IntegerField(primary_key=True)
    renavam = models.CharField(max_length=11)
    chassi = models.CharField(max_length=17)
    marca = models.CharField(max_length=1)
    ano_fab = models.CharField(max_length=4)
    ano_mod = models.CharField(max_length=4)

    class Meta:
        managed = False
        db_table = 'cavalo'

    def __str__(self):
        return self.placa
```

Model Cavalo

Figura 16 – Model Conjunto

conjunto > models.py > Conjunto > Meta

```python
from django.db import models
from cavalo.models import Cavalo
from implemento.models import Implemento

class Conjunto(models.Model):
    cavalo_placa = models.OneToOneField(Cavalo, models.DO_NOTHING, db_column='Cavalo_placa', primary_key=True)  # Field name made lowercase.
    implemento_placa = models.ForeignKey(Implemento, models.DO_NOTHING, db_column='Implemento_placa')  # Field name made lowercase.

    class Meta:
        managed = False
        db_table = 'conjunto'
        unique_together = (('cavalo_placa', 'implemento_placa'),)
```

Model Conjunto

Figura 17 – Model Implemento

implemento > models.py > ...

```python
from django.db import models
from implemento.models import Implemento

class Implemento(models.Model):
    placa = models.IntegerField(primary_key=True)
    renavam = models.CharField(max_length=11)
    chassi = models.CharField(max_length=17)

    class Meta:
        managed = False
        db_table = 'implemento'

    def __str__(self):
        return self.placa


class Carreta(models.Model):
    placa = models.IntegerField(primary_key=True)
    renavam = models.CharField(max_length=11)
    cha = models.CharField(max_length=17)
    implemento_placa = models.ForeignKey(Implemento, models.DO_NOTHING, db_column='Implemento_placa')  # Field name made lowercase.

    class Meta:
        managed = False
        db_table = 'carreta'
        unique_together = (('placa', 'implemento_placa'),)

```

Model Implemento

Figura 18 – Model Manutenção

manutencao > models.py > ...

```python
from django.db import models
from implemento.models import Implemento
from conjunto.models import Conjunto

class Manutencao(models.Model):
    idmanutencao = models.IntegerField(db_column='idManutencao', primary_key=True)  # Field name made lowercase.
    data_manutencao = models.DateField()
    valor = models.FloatField()
    tipo = models.CharField(max_length=1)
    implemento_placa = models.ForeignKey(Implemento, models.DO_NOTHING, db_column='Implemento_placa')  # Field name made lowercase.
    conjunto_cavalo_placa = models.ForeignKey(Conjunto, models.DO_NOTHING, db_column='Conjunto_Cavalo_placa')  # Field name made lowercase.
    conjunto_implemento_placa = models.ForeignKey(Conjunto, models.DO_NOTHING, db_column='Conjunto_Implemento_placa')  # Field name made lowercase.

    class Meta:
        managed = False
        db_table = 'manutencao'

    def __str__(self):
        return self.idmanutencao
```

Model Manutenção

Figura 19 – Model Motorista

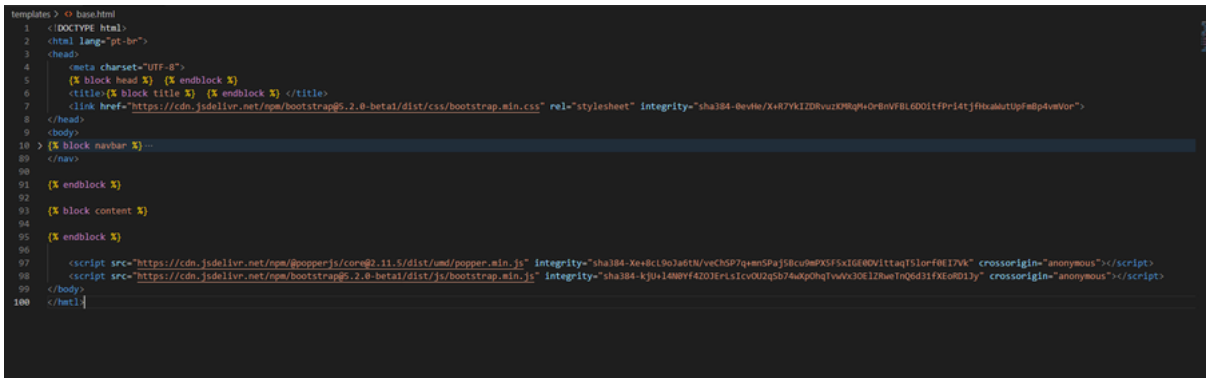
motorista > models.py > Motorista > __str__

```python
from django.db import models

class Motorista(models.Model):
    cnh = models.IntegerField(primary_key=True)
    nome = models.CharField(max_length=100)
    cpf = models.CharField(max_length=11)
    sexo = models.CharField(max_length=1)
    data_nascimento = models.DateField()
    estado_civil = models.CharField(max_length=1)
    telefone = models.CharField(max_length=11)
    admissao = models.DateField()

    def __str__(self):
        return self.nome
```

Model Motorista

Template Base

Figura 20 – Legenda



Template Base

View

Figura 21 – View Abastecimento

```python
abastecimento > views.py > deletarAbastecimento
from django.shortcuts import render, redirect
from abastecimento.forms import AbastecimentoForm
from abastecimento.models import Abastecimento

def dashboardAbastecimento(response):
    return render(response, 'dashboard_abastecimento.html')

def listarAbastecimentos(request):
    abastecimentos = Abastecimento.objects.all()
    return render(request, 'abastecimento.html', {'abastecimentos': abastecimentos})

def cadastrarAbastecimento(request):
    formAbastecimento = AbastecimentoForm(request.POST or None)
    if formAbastecimento.is_valid():
        formAbastecimento.save()
        return redirect('lista_abastecimento')

    return render(request, 'abastecimento_form.html', {'formAbastecimento': formAbastecimento})

def editarAbastecimento(request):
    abastecimento = Abastecimento.objects.get(id=id)
    form = AbastecimentoForm(request.POST or None, instance=abastecimento)

    if form.is_valid():
        form.save()
        return redirect('listar_cavalo')

    return render(request, 'cavalo_form.html', {'form': form})

def deletarAbastecimento(request):
    abastecimento = Abastecimento.objects.get(id=id)
    form = AbastecimentoForm(request.POST or None, instance=abastecimento)

    if form.is_valid():
        form.save()
        return redirect('listar_cavalo')

    return render(request, 'cavalo_form.html', {'form': form})
```

View Abastecimento

Figura 22 – View Cavalo

cavalo > views.py > deletar_cavalo

```python
from django.shortcuts import render, redirect
from urllib import response
from .forms import CavaloForm
from cavalo.models import Cavalo

def dashboard_cavalo(response):
    return render(response, 'dashboard_cavalo.html')

def listar_cavalos(request):
    cavalos = Cavalo.objects.all()
    return render(request, 'cavalo.html', {'cavalos': cavalos})

def cadastrar_cavalo(request):
    formCavalo = CavaloForm(request.POST or None)
    if formCavalo.is_valid():
        formCavalo.save()
        return redirect('lista_cavalo')

    return render(request, 'cavalo_form.html', {'formCavalo': formCavalo})

def editar_cavalo(request):
    cavalo = Cavalo.objects.get(id=id)
    form = CavaloForm(request.POST or None, instance=cavalo)

    if form.is_valid():
        form.save()
        return redirect('listar_cavalo')

    return render(request, 'cavalo_form.html', {'form': form})

def deletar_cavalo(request):
    cavalo = Cavalo.objects.get(id=id)
    form = CavaloForm(request.POST or None, instance=cavalo)

    if form.is_valid():
        form.save()
        return redirect('listar_cavalo')

    return render(request, 'cavalo_form.html', {'form': form})


```

View Cavalo

Figura 23 – View Implemento

implemento > views.py > deletar_implemento

```python
from django.shortcuts import render, redirect
from implemento.forms import ImplementoForm
from implemento.models import Implemento


def dashboard_implemento(response):
    return render(response, 'dashboard_implemento.html')

def listar_implementos(request):
    implementos = Implemento.objects.all()
    return render(request, 'implemento.html', {'implementos': implementos})

def cadastrar_implemento(request):
    formImplemento = ImplementoForm(request.POST or None)
    if formImplemento.is_valid():
        formImplemento.save()
        return redirect('lista_implemento')

    return render(request, 'implemento_form.html', {'formImplemento': formImplemento})

def editar_implemento(request):
    implemento = Implemento.objects.get(id=id)
    form = ImplementoForm(request.POST or None, instance=implemento)

    if form.is_valid():
        form.save()
        return redirect('listar_implemento')

    return render(request, 'implemento_form.html', {'form': form})


def deletar_implemento(request):
    implemento = Implemento.objects.get(id=id)
    form = ImplementoForm(request.POST or None, instance=implemento)

    if form.is_valid():
        form.save()
        return redirect('listar_implemento')

    return render(request, 'implemento_form.html', {'form': form})
```

View Implemento

Figura 24 – View Manutenção

```python
from django.shortcuts import render, redirect
from manutencao.forms import ManutencaoForm
from manutencao.models import Manutencao


def dashboard_manutencao(response):
    return render(response, 'dashboard_manutencao.html')

def listar_manutencao(request):
    manutencoes = Manutencao.objects.all()
    return render(request, 'manutencao.html', {'manutencoes': manutencoes})


def cadastrar_manutencao(request):
    formManutencao = ManutencaoForm(request.POST or None)
    if formManutencao.is_valid():
        formManutencao.save()
        return redirect('lista_manutencao')

    return render(request, 'manutencao_form.html', {'formManutencao': formManutencao})


def editar_manutencao(request):
    manutencao = Manutencao.objects.get(id=id)
    formManutencao = ManutencaoForm(request.POST or None, instance=manutencao)

    if formManutencao.is_valid():
        formManutencao.save()
        return redirect('lista_manutencao')

    return render(request, 'manutencao_form.html', {'form': formManutencao})


def deletar_manutencao(request):
    manutencao = Manutencao.objects.get(id=id)
    formManutencao = ManutencaoForm(request.POST or None, instance=manutencao)

    if formManutencao.is_valid():
        formManutencao.save()
        return redirect('lista_manutencaoo')

    return render(request, 'manutencao_form.html', {'form': formManutencao})

```

View Manutenção

Figura 25 – Motorista

motorista > views.py > deletar_motorista

```python
from urllib import response
from django.shortcuts import redirect, render
from motorista.models import Motorista
from .forms import MotoristaForm

def dashboard_motorista(response):
    return render(response, 'dashboard_motorista.html')

def listar_motorista(request):
    motoristas = Motorista.objects.all()
    return render(request, 'motorista.html', {'motoristas': motoristas})

def cadastrar_motorista(request):
    formMotorista = MotoristaForm(request.POST or None)
    if formMotorista.is_valid():
        formMotorista.save()
        return redirect('lista_motorista')

    return render(request, 'motorista_form.html', {'formMotorista': formMotorista})

def editar_motorista(request, motorista_pk):
    motorista = Motorista.objects.get(pk=motorista_pk)
    motorista = MotoristaForm(request.POST or None, instance=motorista)

    if request.POST():
        if motorista.is_valid():
            motorista.save()
            return redirect('lista_motorista')

    return render(request, 'motorista_form_edit.html', {'motorista': motorista})

def deletar_motorista(request, motorista_pk):
    motorista = Motorista.objects.get(pk=motorista_pk)
    motorista = MotoristaForm(request.POST or None, instance=motorista)

    if request.POST():
        if motorista.is_valid():
            motorista.save()
            return redirect('lista_motorista')

    return render(request, 'motorista_form_edit.html', {'motorista': motorista})

```

View Motorista

Figura 26 – View Viagem

viagem > views.py > deletar_viagem

```python
from django.shortcuts import redirect, render
from viagem.forms import ViagemForm
from viagem.models import Viagem


def dashboard_viagem(response):
    return render(response, 'dashboard_viagem.html')

def listar_viagens(request):
    viagens = Viagem.objects.all()
    return render(request, 'viagem.html', {'viagens': viagens})

def cadastrar_viagem(request):
    formViagem = ViagemForm(request.POST or None)
    if formViagem.is_valid():
        formViagem.save()
        return redirect('lista_viagem')

    return render(request, 'viagem_form.html', {'formViagem': formViagem})

def editar_viagem(request):
    viagem = Viagem.objects.get(id=id)
    formViagem = ViagemForm(request.POST or None, instance=viagem)

    if formViagem.is_valid():
        formViagem.save()
        return redirect('lista_viagem')

    return render(request, 'viagem_form.html', {'form': formViagem})

def deletar_viagem(request):
    viagem = Viagem.objects.get(id=id)
    formViagem = ViagemForm(request.POST or None, instance=viagem)

    if formViagem.is_valid():
        formViagem.save()
        return redirect('lista_viagem')

    return render(request, 'viagem_form.html', {'form': formViagem})


```

View Viagem

URLS

Figura 27 – URLS

```python
from django.contrib import admin
from django.urls import path, include
from .views import home

urlpatterns = [
    path('', home, name="home"),
    path('abastecimento/', include('abastecimento.urls')),
    path('admin/', admin.site.urls),
    path('cavalo/', include('cavalo.urls')),
    path('conjunto/', include('conjunto.urls')),
    path('implemento/', include('implemento.urls')),
    path('manutencao/', include('manutencao.urls')),
    path('motorista/', include('motorista.urls')),
    path('viagem/', include('viagem.urls')),
]
```

URLS

Anexos

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
GABINETE DO REITOR

Av. Universitária, 1069 • Setor Universitário
Caixa Postal 86 • CEP 74605-010
Goiânia • Goiás • Brasil
Fone: (62) 3946.1000
www.pucgoias.edu.br • reitoria@pucgoias.edu.br

RESOLUÇÃO n° 038/2020 – CEPE

ANEXO I

APÊNDICE ao TCC

Termo de autorização de publicação de produção acadêmica

O estudante Renato Queiroz Silva do Curso de Engenharia de Computação matrícula nº 2017.1.0033.0147-2,telefone:(62)98147-2992, e-mail renatoqueiroz65@gmail.com, na qualidade de titular dos direitos autorais, em consonância com a Lei nº 9.610/98 (Lei dos Direitos do Autor), autoriza a Pontifícia Universidade Católica de Goiás (PUC Goiás) a disponibilizar o Trabalho de Conclusão de Curso intitulado Sistema de Informação para Controle de Veículos de uma Transportadora, gratuitamente, sem ressarcimento dos direitos autorais, por 5 (cinco) anos, conforme permissões do documento, em meio eletrônico, na rede mundial de computadores, no formato especificado (Texto(PDF); Imagem (GIF ou JPEG); Som (WAVE, MPEG, AIFF, SND); Vídeo (MPEG, MWV, AVI, QT); outros, específicos da área; para fins de leitura e/ou impressão pela internet, a título de divulgação da produção científica gerada nos cursos de graduação da PUC Goiás.

Goiânia, 17 de Junho de 2022.

gov.br Documento assinado digitalmente
RENATO QUEIROZ SILVA
Data: 17/06/2022 12:32:23-0300
Verifique em https://verificador.iti.br

Assinatura do autor: _____

Nome completo do autor: Renato Queiroz Silva

Assinatura do professor–orientador: _____

Nome completo do professor-orientador: Olegário Correa da Silva Neto