Desenho de um círculo

Descrição gerada automaticamente com confiança baixa

**CENTRO UNIVERSITÁRIO CATÓLICA DO LESTE DE MINAS GERAIS – UNILESTE**

# Curso de Engenharia de Software

**Projeto Extensionista**

***Alunos:***

Daniel Maciel Machado

Fabricio Correia Silva Santiago

Fernando Elias da Silva Freitas

Guilherme Freitas de Aquino

Julio Oliveira Eler

Raniery Neiva Martins

Coronel Fabriciano – MG

Setembro de 2023

1. Introdução

Inicialmente a ideia do projeto partiu de uma necessidade identificada por um dos integrantes do grupo na empresa Djafer o qual se resumia em um sistema para gerenciamento da frota de caminhões e carros da empresa.

* 1. Objetivo

O projeto tem como objetivo facilitar o processo de gerenciar a frota de veículos da empresa controlando os gastos realizados com os veículos incluindo manutenções e os tickets de abastecimento, sendo capaz de gerar relatórios e auxiliar na definição de rotas otimizadas diminuindo as dificuldades em controlar os gastos e as documentações que são geralmente feitos em papéis e planilhas do Excel podendo ter problemas com perdas e fraudes.

* 1. Público-Alvo

No momento a ideia partiu para suprir uma nescessidade que havia na empresa, mais no desenvolvimento do projeto, percebemos a oportunidade de ampliar o esse projeto para todas as empresas que gerencia suas frotas.

* 1. Benefícios

Nosso software tem como objetivo oferecer aos clientes automátização do gerenciamento de frotas, demonstrado por dados seguros e presciso.

Dentro do sistema os clientes poderam criar acessos para usuário com regras de acesso, cadastro de veículos, cadastro de postos de combústiveis que tem parceirias com as empresas, os motoristas poderam acessar o sistema e receber e enviar informações em tempo real sobre rotas, solicitação de tickets de combústiveis e um Dash Board personalizados que podem ser tranformados em relatórios de gastos.

1. Glossário

**SULCOS DO PNEU -** São canais localizados no sentido tangencial da banda de rodagem e sua função principal é escoar a água para evitar aquaplanagem.

**TICKETS -** são atendimentos ou solicitações abertas pelos usuários (clientes) ou os próprios funcionários da sua empresa.

Banco de dados- Um banco de dados é uma coleção organizada de informações - ou dados - estruturadas, normalmente armazenadas eletronicamente em um sistema de computador.

**CRUD -** (acrónimo do inglês Create, Read, Update and Delete) são as quatro operações básicas (criação, consulta, atualização e destruição de dados) utilizadas em bases de dados relacionais (RDBMS) fornecidas aos utilizadores do sistema.

**PHP** é uma linguagem interpretada livre, usada originalmente apenas para o desenvolvimento de aplicações presentes e atuantes no lado do servidor, capazes de gerar conteúdo dinâmico na World Wide Web.

**PDF -** portable document format (formato de documento portatil) é um formato de arquivo, desenvolvido pela Adobe Systems em 1993, para representar documentos de maneira independente do aplicativo, do hardware e do sistema operacional usados para criá-los.

**MVC -** é uma sigla do termo em inglês Model (modelo) View (visão) e Controller (Controle) que facilita a troca de informações entre a interface do usuário aos dados no banco, fazendo com que as respostas sejam mais rápidas e dinâmicas.

1. Descrição do Minimundo do Projeto

Aparti de uma idéia que surgiu dentro de uma empresa em que o Guilherme Aquino integrante do grupo trabalhava, começamos a levanta uma análise do fluxo de trabalho da empresa, onde descrevemos através de relatos de um outro funcionário cujo o nome é Vitor e responsável pela logística da empresa atráves de uma entrevista.

O controle realizado do gerenciamento das frotas se baseia na nota fiscal enviada pelo posto de gasolina, anotações feitas pelo motorista, gastos com manutenções e pela medida dos sulcos dos pneus e sua respectiva quilometragem. Esse processo ocorre da seguinte forma:

**Avaliação dos Pneus**

O motorista tem a responsabilidade de, diariamente realizar o checklist dos itens básicos do caminhão, necessários para que o mesmo trafegue com segurança.

A cada 15 dias o funcionário responsável pelo controle da frota realiza uma inspeção para acompanhar os sulcos dos pneus. O pneu novo vem com sulcos de 15 milímetros e quando chega a 3 milímetros o pneu já tem que ser trocado ou ir para recauchutagem. Quando o pneu vai para recauchutagem, ele passa por uma avaliação de estrutura para ver se pode ou não realizar a recauchutagem, caso seja possível, a empresa economiza aproximadamente 50% do valor que seria um pneu novo. O mesmo pode ser recauchutado no máximo 3 vezes e todo pneu passado por esse processo não pode ser rodado na dianteira, é proibido por lei.

**Controle de consumo de combustível**

A Djafer tem parceria com dois postos, sendo eles, o posto Shell e o Torque Diesel.

É passado para o motorista um ticket o qual permite ele completar o tanque do caminhão nos postos citados. Nesse ticket ele anota seu nome, a placa do caminhão, a quilometragem, a quantidade de litros abastecidos naquele dia e o valor. No final do mês o funcionário do posto vai até a Djafer portanto uma nota fiscal de todos os tickets utilizados naquele mês, que é repassada para o responsável pelo controle da logística.

Com esses tickets e a nota fiscal em mãos, o funcionário realização a aferição das informações, e alimenta uma planilha onde faz a comparação dos gastos nos meses anteriores com o mês atual fazendo um balanço, podendo chegar à conclusão se o consumo do caminhão foi menor ou maior, visando otimizar rotas e cargas, onde terá um gasto menor com os fretes.

Esse balanço serve para monitorar também a depreciação dos veículos, visto que os gastos com manutenções também entram nessa planilha.

Há um grande risco de haver erros no momento do preenchimento da planilha sabotando assim todo controle, pois, quando o funcionário realiza essa atividade no fim do mês, ele se depara com uma grande quantidade de informações acumuladas necessitando de um longo tempo para aferi-las e passá-las para a planilha.

Pode ocorrer também a perda dos tickets e da própria planilha, que de acordo com o funcionário, ela fica salva apenas em um computador e muitas vezes ele demora para encontrá-la em meio aos outros arquivos salvos no mesmo.

1. Envolvidos no Projeto

* Daniel Maciel Machado
* Fabricio Correia Silva Santiago
* Fernando Elias da Silva Freitas
* Guilherme Freitas de Aquino
* Raniery Neiva Martins

Essa tabela mostra as suposições de recrutamento para o projeto.

|  |  |  |
| --- | --- | --- |
| Recursos Humanos | | |
| Trabalhador | Recursos Mínimos Recomendado | Responsabilidades Específicas ou Comentários |
| Gerente de Teste,  Gerente do Projeto  de Teste | Guilherme | * Fornece supervisionamento gerencial. * Responsabilidades: * provê direcionamento técnico. * adquire recursos apropriados. * Fornece relatórios de gerenciamento. |
| Test Designer | Daniel | * Identifica, prioriza, e implementa os casos de teste. * Responsabilidades: * Gera o plano de teste * Cria o modelo de teste * Avalia a efetividade do esforço de teste |
| Testador | Fernando | * Executa os testes. * Responsabilidades: * Executar os testes. * Registrar os resultados. * Reestabelecer-se dos erros * Documentar solicitações de mudança. |
| Administrador do Sistema de Teste | Raniery | * Garante que o ambiente e os bens de teste sejam gerenciados e mantido * Responsabilidades: * Administrar o sistema de gerenciamento teste * Instalar e gerenciar o acesso do trabalhador ao sistema de testes. |
| Gerente do Banco de  Dados,  Administrador do  Banco de Dados | Fabricio Santiago | * Garante que o ambiente e bens de teste de dados (banco de dados) sejam gerenciados e mantidos. * Responsabilidades: * Administrar os dados de teste. (base de dados) |
| Designer | Fernando | * Identifica e define as operações, atributos, e associações das classes de teste. * Responsabilidades: * Identificar e definir as classes de teste. * identificar e definir os pacotes de teste. |
| Implementador | Fabricio/ Raniery | * Implementa e faz os testes unitários das classes e pacotes de teste. * Responsabilidades: * Cria as classes e pacotes de teste implementados no modelo de teste. |

1. Delimitação do Projeto
2. Levantamento de Requisitos

**6.1 Descrição dos Atores**

|  |  |  |  |
| --- | --- | --- | --- |
| **Num.** | **Nome** | **Descrição** | **Frequência de Uso** |
| 1 | Gerente | Responsável por definir e controlar permições dos usuários no sistema. | Semanal |
| 2 | Funcionário Financeiro | Responsável por controla toda parte financeira. | Diário |
| 3 | Funcionário Administrativo | Responsável por todos os cadastros, e controle das frotas. | Diário |
| 4 | Motorista | Responsável por fazer os checklists e entregas. | Diário |

**6.2 Requisitos funcionais**

|  |  |  |  |
| --- | --- | --- | --- |
| **Ordem** | **Nome da função** | **Descrição** | **Ator(es) relacionado(s)** |
| 1 | Gerenciar Veículos | CRUD de caminhões. É necessário cadastrar os caminhões para realizar o controle de rotas e emitir tickets de abastecimento. | 3 |
| 2 | Gerenciar Motoristas | CRUD de motoristas. É necessário cadastrar motoristas para emitir tickets de abastecimento e identificação nos checklist. | 3,4 |
| 3 | Realizar Checklist | Permite realizar diariamente as verificações básicas do veículo a ser utilizado. | 4 |
| 4 | Gerenciar Tickets de Abastecimento | CRUD de tickets. Permite registrar o gasto com combustível de cada caminhão e fazer solicitação e baixa de tickets. Necessário cadastrar caminhões e motoristas. | 2,3 |
| 5 | Gerenciar Manutenção | CRUD de manutenções. Permite relacionar as manutenções com os caminhões. | 3 |
| 7 | Visualizar Relatório de Gastos | Permite o funcionário visualizar os gastos com manutenções e tickets de abastecimentos sobre o veículo no período desejado. | 1,2,3 |
| 8 | Gerenciar Usuário | Permite criar usuários e definir suas permissões no sistema. | 1 |
| 9 | Login | Permite acesso ao sistema atribuindo à cada usuário as suas permissões. | 1,2,3,4 |

**6.3 Requisitos não funcionais**

**RNF 01 –** O software deve ser codificado na linguagem JavaScript.

**RNF 02 –** Os relatórios do sistema devem estar no formato PDF.

**RNF 03 –** O sistema deverá ser web, compatível com os navegaderes mais utilizados atualmente como o Chrome, Edge, Firefox, Opera e Safari.

**6.4 Detalhamento dos casos de uso**

|  |  |
| --- | --- |
| **Nome do caso de Uso** | Login |
| **Atores relacionados** | 1,2,3,4 |
| **Pré-condições** | O sistema está iniciando e a conexão do banco de dados foi estabelecida. |
| **Funcionalidades** | Validar login, validar senha, criptografar senha |
| **Fluxo Principal** | |
| **Ações do Ator** | **Ações do Sistema** |
|  | Exibir a interface de login |
| Informa os dados de login |  |
| Acionar o botão de "entrar" |  |
|  | Verificar os dados de login |
|  | Informa o status do login |

|  |  |
| --- | --- |
| **Nome do caso de Uso** | Gerenciar Usuários |
| **Atores relacionados** | 1 |
| **Pré-condições** | O usuário está logado no sistema. |
| **Funcionalidades** | Receber dados, salvar dados no banco, salvar permissões. |
| **Fluxo Principal** | |
| **Ações do Ator** | **Ações do Sistema** |
|  | Exibir a interface de gerenciamento de usuários |
| Informa os dados do funcionário a ser cadastrado |  |
| Acionar o botão de "salvar" |  |
|  | Salvar dados recebido |
|  | Exibe tela de permissões do usuário |
| Define as permissões |  |
| Acionar o botão de "salvar" |  |
|  | Salva dados recebido |
|  | Exibe mensagem de "Usuário cadastrado com sucesso" |

|  |  |  |
| --- | --- | --- |
| **Nome do caso de Uso** | Gerar relatório de gastos | |
| **Atores relacionados** | 1,2,3 | |
| **Pré-condições** | O usuário está logado no sistema. | |
| **Funcionalidades** | Pesquisar dados pelo filtro, exibir dados de acordo com a pesquisa, enviar dados para impressão | |
| **Fluxo Principal** | | |
| **Ações do Ator** | **Ações do Sistema** | |
|  | Exibir a interface de Relatório de gastos | |
| Informa os dados do veículo e período desejado |  | |
| Acionar o botão de "Gerar relatório" |  | |
|  | Fazer leitura de dados recebido | |
|  | Exibir dados relacionados de acordo com os dados informados nos filtros. | |
| Acionar o botão de "Imprimir" |  |  |
|  | Enviar dados para impressora conectada | |

|  |  |
| --- | --- |
| **Nome do caso de Uso** | Gerenciar Tickets de abastecimento |
| **Atores relacionados** | 2,3 |
| **Pré-condições** | O usuário está logado no sistema. |
| **Funcionalidades** | Receber solicitações, gerar tickets, validar tickets, salvar dados no banco |
| **Fluxo Principal** | |
| **Ações do Ator** | **Ações do Sistema** |
|  | Receber solicitação de tickets |
|  | Exibir a interface de gerar tickets |
| Informa os dados |  |
| Acionar o botão "Gerar tickets" |  |
|  | Salva dados no banco |
|  | Libera Tickets para uso |
| **Fluxo Secundário** | |
|  | Status dos tickets "Utilizado" |
| Informa número da nota fiscal |  |
| Acionar o botão "Baixar tickets" |  |

|  |  |
| --- | --- |
| **Nome do caso de Uso** | Realizar Checklist |
| **Atores relacionados** | 4 |
| **Pré-condições** | O usuário está logado no sistema. |
| **Funcionalidades** | Informar dados, salvar dados no banco |
| **Fluxo Principal** | |
| **Ações do Ator** | **Ações do Sistema** |
|  | Exibir a interface de checklist |
| Informa os dados |  |
| Acionar o botão de "Salvar" |  |
|  | Salvar dados recebidos |
|  | Exibir uma mensagem de confirmado |

|  |  |
| --- | --- |
| **Nome do caso de Uso** | Gerenciar Motorista |
| **Atores relacionados** | 4 |
| **Pré-condições** | O usuário está logado no sistema. |
| **Funcionalidades** | Informar dados, salvar dados no banco |
| **Fluxo Principal** | |
| **Ações do Ator** | **Ações do Sistema** |
|  | Exibir a interface de Checklist |
| Informa os dados |  |
| Acionar o botão de "Salvar" |  |
|  | Salvar dados recebidos |
|  | Exibir uma mensagem de confirmado |

|  |  |
| --- | --- |
| **Nome do caso de Uso** | Gerenciar Veículos |
| **Atores relacionados** | 3 |
| **Pré-condições** | O usuário está logado no sistema. |
| **Funcionalidades** | Informar dados, salvar dados no banco |
| **Fluxo Principal** | |
| **Ações do Ator** | **Ações do Sistema** |
|  | Exibir a interface de Cadastro de Veículos |
| Informa os dados |  |
| Acionar o botão de "Salvar" |  |
|  | Salvar dados recebidos |
|  | Exibir uma mensagem de confirmado |

|  |  |
| --- | --- |
| **Nome do caso de Uso** | Gerenciar Rotas |
| **Atores relacionados** | 3 |
| **Pré-condições** | O usuário está logado no sistema. |
| **Funcionalidades** | Informar dados, gerar informações, enviar informações |
| **Fluxo Principal** | |
| **Ações do Ator** | **Ações do Sistema** |
|  | Exibir a interface de gerar rotas |
| Informa os endereços |  |
|  | Verificar dados |
| Acionar botão " Gerar rotas" |  |
|  | Traçar melhor trajeto de acordo com os endereços informados |
| Acionar botão " Enviar rotas" |  |
|  | Disponibilizar rota na visão do motorista |

|  |  |
| --- | --- |
| **Nome do caso de Uso** | Gerenciar Postos de combustível |
| **Atores relacionados** | 3 |
| **Pré-condições** | O usuário está logado no sistema. |
| **Funcionalidades** | Informar dados, salvar dados no banco |
| **Fluxo Principal** | |
| **Ações do Ator** | **Ações do Sistema** |
|  | Exibir a interface de Cadastro de Postos de Combustíveis |
| Informa os dados |  |
| Acionar o botão de "Salvar" |  |
|  | Salvar dados recebidos |
|  | Exibir uma mensagem de confirmado |

1. Modelo de Dados
2. Projeto de Interface
   1. Projeto de Interface do Site
3. **Tela Login**

Interface gráfica do usuário

Descrição gerada automaticamente

1. **Dashboard (Tela Inicial)**

Tela principal da aplicação, que apresenta de forma resumida algumas acoes importantes feitas pelos usuarios.

Gráfico, Gráfico de linhas

Descrição gerada automaticamente

1. Cadastro de funcionários

Tela de realização de cadastro de funcionários da empresa, com campos para inserção de dados do novo funcionário na aplicação.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

1. Feedback (cadastrado com sucesso)

Popup de confirmação de sucesso da realização do cadastro de um funcionário

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

1. Cadastro de Posto de Combustível

Tela de realização de cadastro de posto de combustível parceiro da empresa, com campos para inserção de dados do novo posto parceiro na aplicação.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

1. Feedback (cadastrado com sucesso)

Popup de confirmação de sucesso da realização do cadastro de um posto de combustível parceiro

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

1. Cadastro de veículo

Tela de realização de cadastro de um veículo do motorista vniculado a empresa, com campos para inserção de dados do novo veiculo na aplicação.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

1. Feedback (cadastrado com sucesso)

Popup de confirmação de sucesso da realização do cadastro de um posto de combustível parceiro

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

1. Lista de veículos cadastrados

Tela que listar todos os veículos cadastro no sistema. Cada um desses veículos tem ação de ***Visualizar***, ***editar e Excluir***

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

1. Gerar Rotas

Tela de gerenciamento e acompanhamento de rotas. Cada motorista consegue acompanhar suas respectivas rotas planejadas.

Interface gráfica do usuário, Aplicativo, Mapa

Descrição gerada automaticamente

Mapa

Descrição gerada automaticamente

1. Gerar Tickets

Tela de gerenciamento dos tickets.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

1. Listas de tickets gerados

Tela que listar todos os tickets cadastro no sistema. Cada um desses tickets tem ação de ***Visualizar***, ***editar e Excluir***

Interface gráfica do usuário

Descrição gerada automaticamente

1. Pesquisa de relatórios

Tela onde são apresentados gastos de veículos. Permite listar relatórios dos veículos

cadastrado dentro do sistema.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

1. Detalhamento de relatório

Tela com informações detalhado de um veiculo específico que permite filtra por uma determinada data e imprimir relatório.Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

* 1. Projeto de Interface do Dispositivo Mobile

1. Tela de Login

Tela inicial do sistema onde permite usuário logar. para acessa o sistema.

Interface gráfica do usuário

Descrição gerada automaticamente

1. Tela Inicial

Tela onde permite as principais funções dos sistemas como, solicitar tickets e realizar check list, além de visualizar detalhes dos veículos.

Interface gráfica do usuário

Descrição gerada automaticamente

1. Checklist

Tela onde o motorista irá preencher com informações do veículos e realizar a verificação da situação do caminhão que está sobre a sua responsabilidade naquela rota. Checklist deverá ser realizada no momento de saída e chegada do veículo.

Aplicativo

Descrição gerada automaticamente com confiança baixa Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

* 1. Projetos de Interface de outras versões do sistema

1. Diagramas (UML)

9.1 Casos de uso do projeto Museu

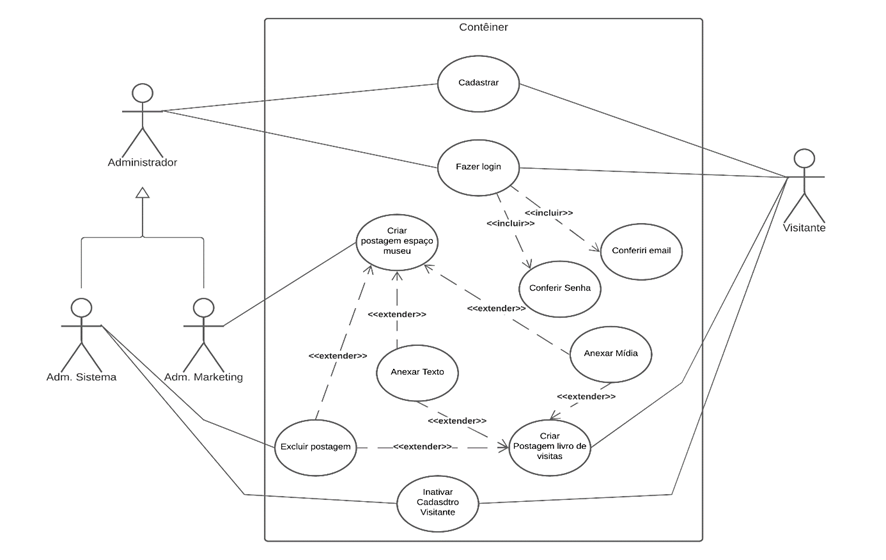


Figura 1 -Diagrama caso de uso Museu

9.2 Diagrama Entidade Relacionamento

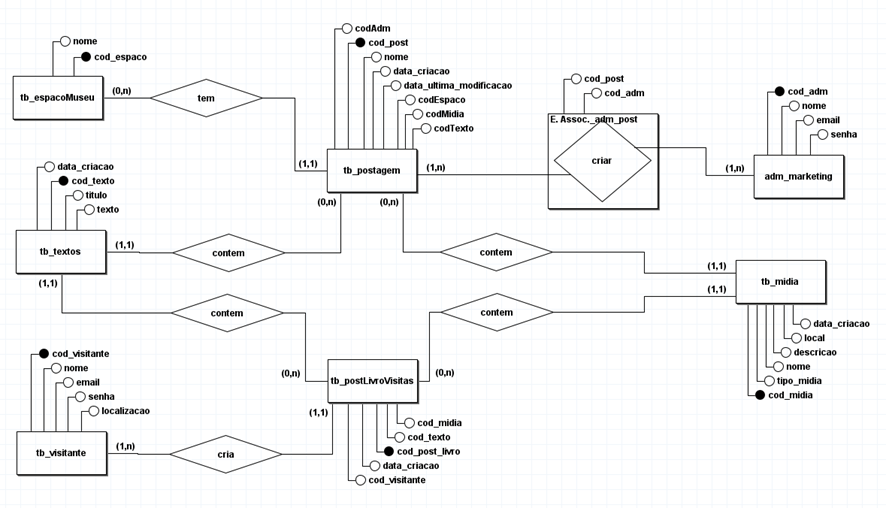


Figura 2 DER do sistema do museu Unileste

9.3 Diagrama de Casos de uso – Rastrear Frotas

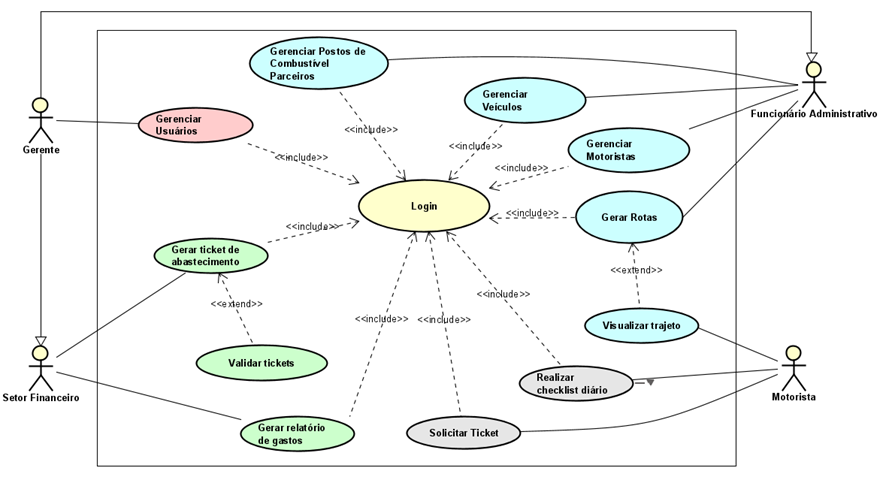


Figura 4 Diagrama caso de uso Rastrear Frotas

9.4 Diagrama de Classes

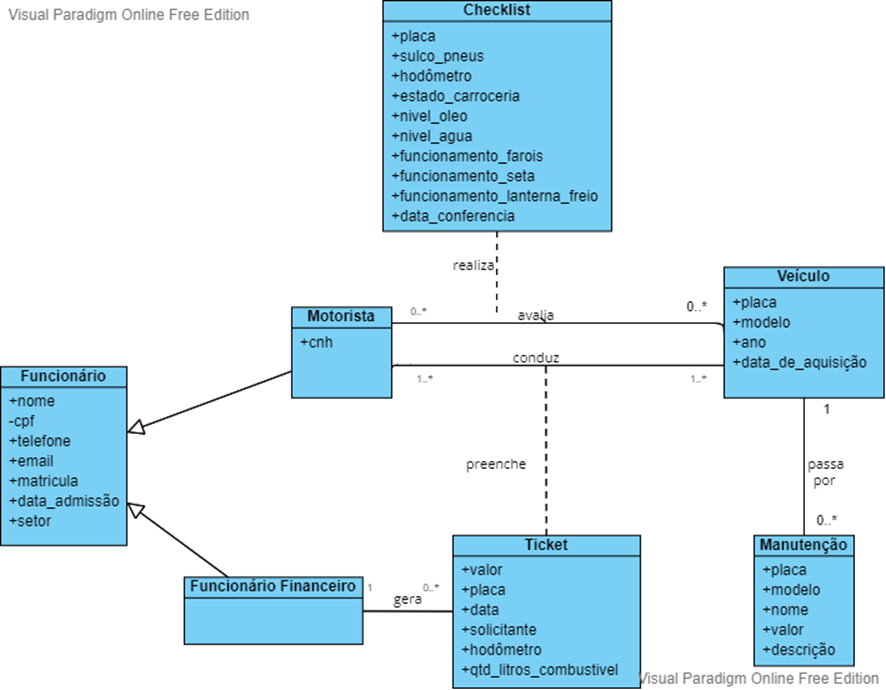


Figura 5 Diagrama de classes

9.5 Diagramas de Sequência

Processo: Gerar Relatórios

O diagrama a seguir ilustra o processo de gerar relatório dos veículos cadastrados no sistema de acordo com os dados informados no filtro, sendo ele o veículo desejado e o período.

O sistema irá relacionar os dados das manutenções cadastradas e os tickets utilizado no abastecimento do veículo informado.

Figura 6 Diagrama Funcionário realizando o relatório.

9.6 Diagrama Processo: Gerar Tickets

O diagrama a seguir ilustra o processo de solicitação e geração de tickets de abastecimento no sistema, processo esse que inclui participação do motorista e do setor financeiro da empresa.

O motorista deve gerar uma solicitação para que o setor financeiro gere um ticket de abastecimento para seu veículo. Uma vez gerado, o motorista utiliza o ticket que estará liberado no sistema. Após utilizar o ticket, o motorista entrega a nota fiscal individual daquele abastecimento no setor financeiro da empresa, com isso o funcionário do setor financeiro informa o número da nota fiscal no sistema para dar baixa no ticket.

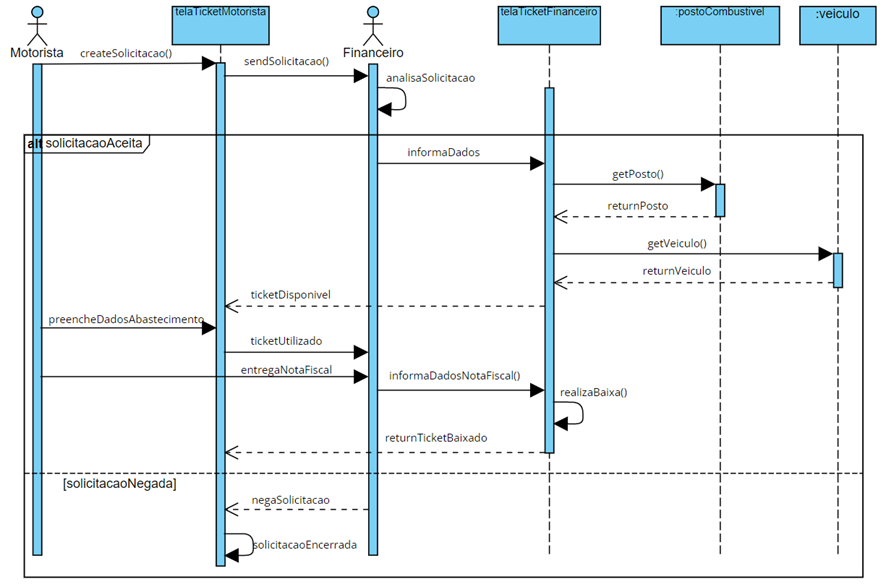


Figura 7 Motorista solicitando Ticket

9.7 Diagrama Processo: Gerar Rotas

O diagrama a seguir, demostra o processo de criação de rotas de entregas, onde é necessário o funcionário informar endereços de entregas das mercadorias para o sistema gerar a melhor trajeto e disponibilizar a rota na interface do motorista.

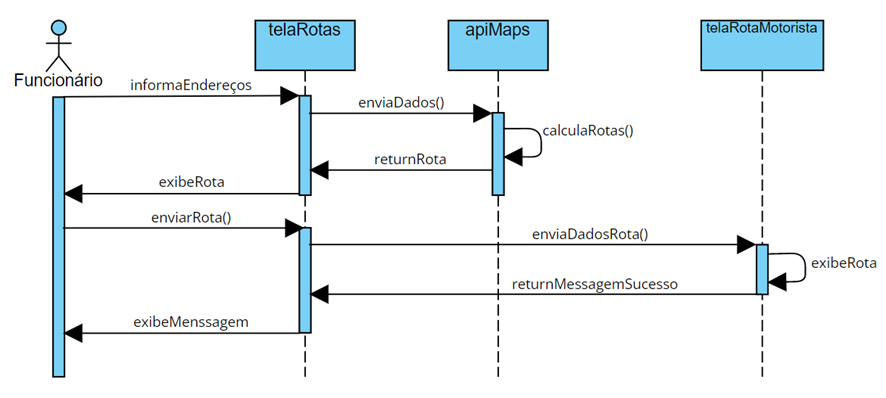
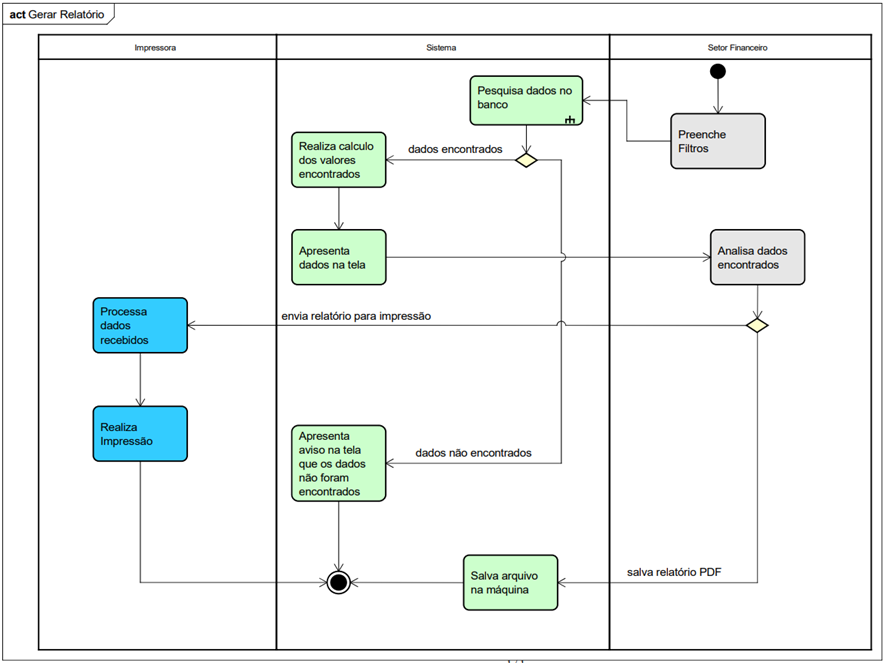


Figura 8 Motorista solicitando rotas

9.8 Diagrama de Atividades: Gerar Relatório Salvar/Imprimir

O diagrama de atividades a seguir, representa o processo de pesquisa e geração dos relatórios do veículo, representa as ações que o usuário faz nesse processo, as

ações do sistema de acordo com o caminho que o fluxo segue e de uma impressora caso o usuário decida imprimir o relatório.

Figura 10 Fluxo de geração de relatório

O diagrama abaixo representa o refinamento da atividade de “Pesquisar dados no banco”

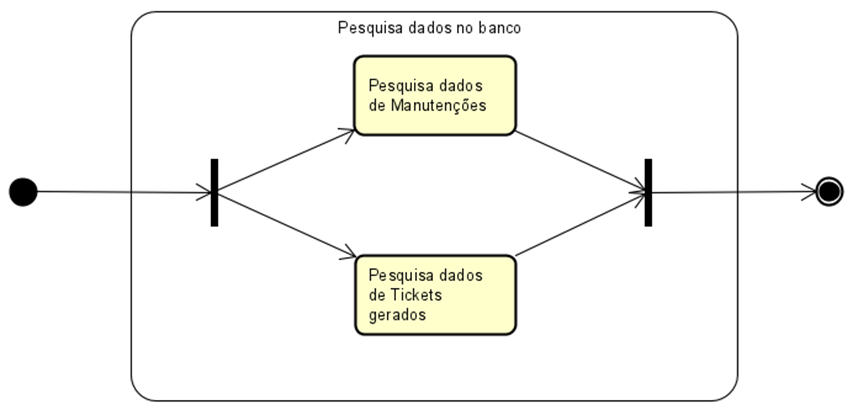


Figura 11 Pesquisa no banco de dados

9.9 Diagrama Processo: Gerenciamento de tickets

O diagrama a seguir modela as ações realizadas pelo motorista, funcionário do setor financeiro e do sistema no processo de solicitação, geração e utilização de um ticket de abastecimento

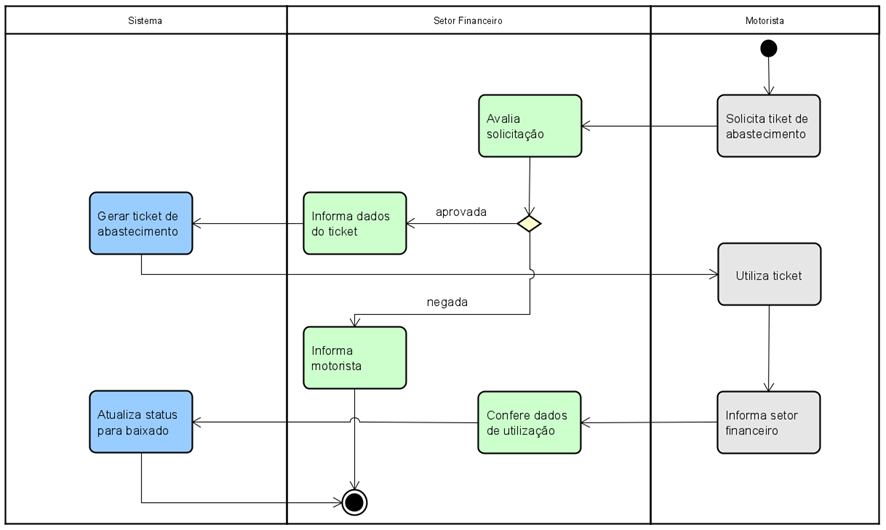


Figura 12 Ações realizadas pelo motorista

9.10 Diagrama Processo: Gerenciamento de rotas

O diagrama a seguir, demostra o processo de geração e utilização de uma rota de entregas, onde é necessário ações em que o funcionário administrativo informa os

endereços de entregas das mercadorias para o sistema gerar a melhor trajeto. Esse trajeto posteriormente é utilizado pelo motorista para realização das entregas

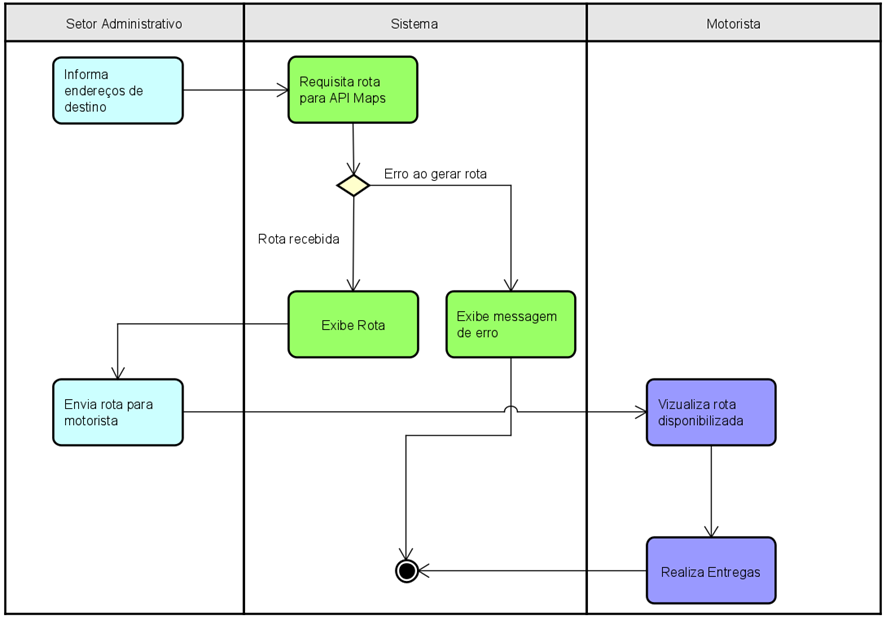


Figura 13 Fluxo de gerenciamento de rotas

9.11 Diagramas de Máquina de Estados

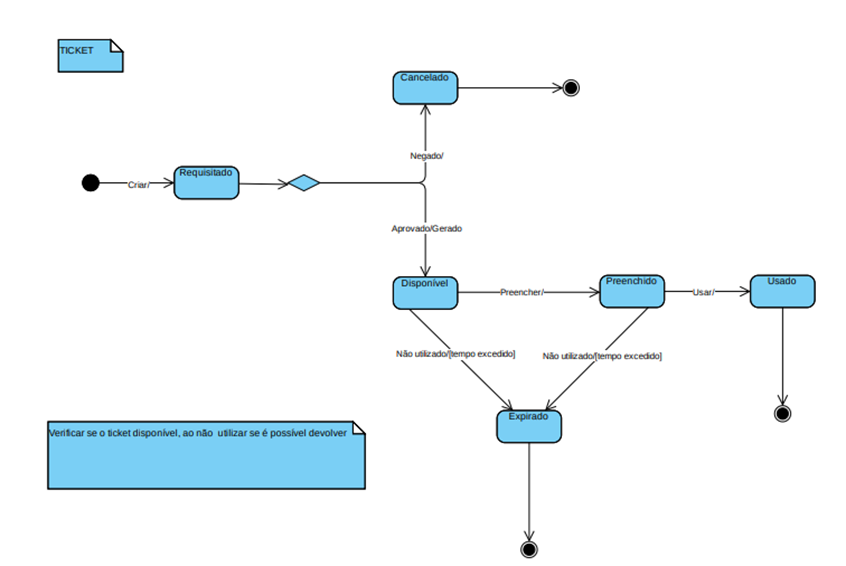


Figura 14 Gerenciamento de tickets

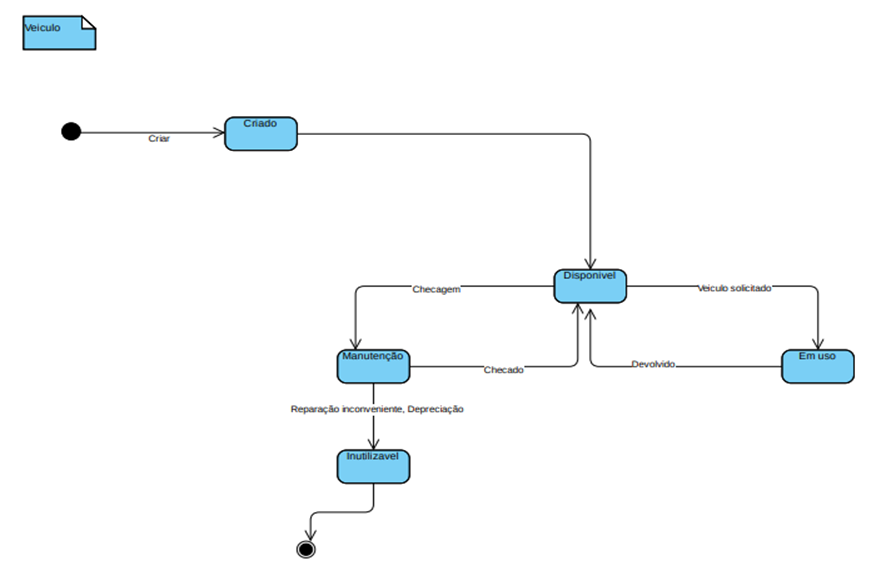


Figura 15 Disponibilização de Veículo

9.12 Diagrama de Componentes

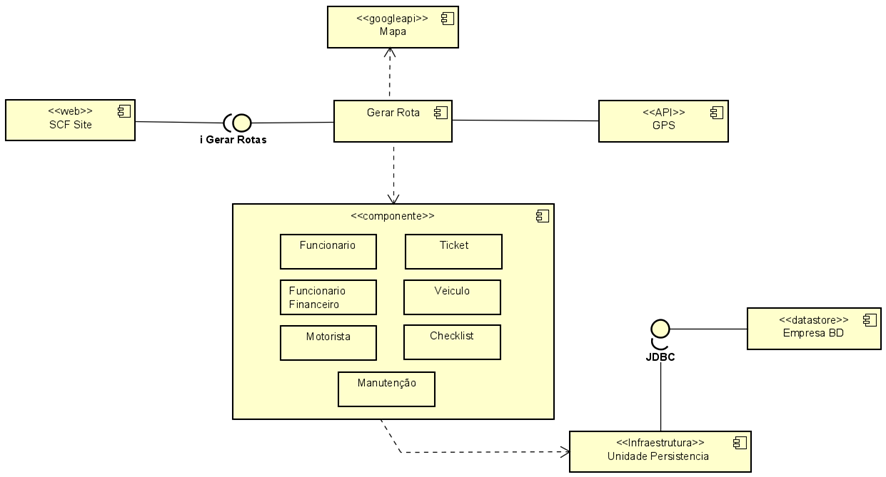


Figura 16 API utilizada para geração de rotas

1. Plano de Testes (caso seja aluno de CES)

Aqui apresentamos o planejamento das atividades de testes do sistema Rastrear – Sistema de Controle de Frotas o qual será utilizado como base para as atividades de acompanhamento, revisão, verificação e validação do projeto, desde seu início até sua conclusão, a fim de garantir a análise comparativa do resultado real versus planejado. Desta forma, ações corretivas e preventivas poderão ser tomadas, sempre que resultados reais desviarem significativamente do planejado.

# 1. Introdução

Este documento apresenta o planejamento das atividades de testes do sistema Rastrear – Sistema de Controle de Frotas o qual será utilizado como base para as atividades de acompanhamento, revisão, verificação e validação do projeto, desde seu início até sua conclusão, a fim de garantir a análise comparativa do resultado real versus planejado. Desta forma, ações corretivas e preventivas poderão ser tomadas, sempre que resultados reais desviarem significativamente do planejado.

## 1.1 Objetivos

O documento do Plano de Testes do software Rastrear – Sistema de Controle de Frotas, tem como objetivo listar os Requisitos que serão testados recomendando e descrevendo as estratégias a serem empregadas nesses testes. Este documento também identifica os recursos necessários e disponibiliza uma estimativa dos esforços de teste.

## 1.2 O sistema “Rastrear”

Tem como objetivo facilitar o processo de gerenciar a frota de veículos da empresa controlando os gastos realizados com a frota de veículos da empresa, controlando as manutenções e os tickets de abastecimento, sendo capaz de gerar relatórios e auxiliar na estimação dos gastos e rastreamento de frotas diminuindo as dificuldades em controlar os gastos e documentação que geralmente são feitos em papeis e planilhas do Excel podendo ter problemas com perdas e fraudes, além do aperfeiçoamento das rotas de entrega. Os benefícios estão relacionados com a agilidade e praticidade no controle da logística dentro da empresa, na economia de papel e na mão de obra do funcionário. Empresas que podem aderir esse projeto são empresas que possui sua própria frota, empresas de transporte, entregas, entre outras.

## 1.3 Escopo

O Gerenciador de Serviços Automotivos deverá ser submetido a testes de unidade, integração, sistema e aceitação.

Os testes de unidade avaliarão isoladamente o banco de dados, a interface gráfica, e todos os outros componentes do projeto.

O teste de integração testa os componentes, previamente testados isoladamente, uma vez que estão acoplados. O objetivo é identificar possíveis falhas nos acoplamentos.

Os testes de sistema avaliarão o funcionamento e o desempenho do sistema como um todo, verificando a eficácia e segurança, além da compatibilidade e integração do software em diferentes ambientes.

Os testes de aceitação apresentarão o produto final para o usuário para validação e últimos ajustes.

Para realizar os testes serão utilizadas máquinas com as configurações mais próximas o possível das máquinas que serão utilizadas pelo usuário final, tentando assim, simular o ambiente final em que o programa será executado.

**1.4 Identificação do Projeto**

|  |  |  |
| --- | --- | --- |
| **Documento** | **Criado ou Disponível** | **Recebido ou Revisado** |
| Especificação de Requisitos | [x] Sim [] Não | [x] Sim  Não |
| Plano de Projeto | [x] Sim [] Não | [x] Sim [] Não |
| Modelo de Análise | [x] Sim [] Não | [x] Sim [] Não |
| Modelo de Projeto | [x] Sim [] Não | [x] Sim [] Não |
| Documento de Arquitetura | [] Sim [x] Não | [] Sim [x] Não |
| Protótipo | [x] Sim [] Não | [x] Sim [] Não |
| Manual do Usuário | [] Sim [x] Não | [] Sim [x] Não |
| Lista de Riscos | [] Sim [x] Não | [] Sim [x] Não |

# 1. REQUISITOS A SEREM TESTADOS

Estabilidade, desempenho, segurança, interface de usuário, controle de acesso, funcionalidades.

## 1.1 Teste do Banco de Dados

Verificar se os dados dos:

* **funcionários,**
* **postos de gasolina,**
* **checklist,**
* **veículos,**
* **manutenções e**
* **rotas**

Podem ser salvos, consultados, modificados e deletados.

Verificar se as transações estão funcionando corretamente.

* **Ex.: Ao realizar a baixa de um ticket de abastecimento, casso ocorra algum erro em um dos passos da rotina, a transação estabelecida com o banco deve realizar um rollback nas modificações realizadas no banco.**

Verificar se os campos de data estão sendo salvos com o fuso horário correto.

## 1.2 Teste Funcional

Descrição dos casos de teste das funções do sistema

1.Realizar o cadastro de funcionário.

Teste 1: Cadastro do funcionário válida.

Entrada 1: Usuário inserir todas as informações corretas ao cadastrar

Saída 1: Operação válida

Entrada 2: Cadastro de funcionário invalido

Usuário inserir dados incorretos:

Inserir CPF inválido.

Saída 2: Operação inválida: “CPF inválido”

Entrada 3: Cadastro de funcionário valido

Usuário inseriu todos os dados obrigatórios

Saída 3: Operação válida

Entrada 4: Cadastro de funcionário inválida

Usuário não inserir dados obrigatórios:

Usuário não inseriu o nome ao cadastrar.

Saída 4: Operação inválida: “Não inseriu o nome ao cadastrar”

Entrada 5: Cadastro de funcionário válido

Usuário inseriu o e-mail válido.

Saída 5: Operação válida.

Entrada 6: Cadastro de Funcionário Válida

Usuário não inseriu o e-mail válido.

Inseriu o e-mail incorreto, exemplo faltando @(provedor)

Saída 6: Operação inválida: “Não inseriu um e-mail válido”

Entrada 7: Busca de funcionário válida

A busca realizada de um funcionário especifico deu certo

Saída 7: Operação Válida

Entrada 8: Busca de funcionário inválida

Ao buscar um funcionário inexistente

Saída 8: Operação inválida: “Funcionário não cadastrado”

Entrada 9: Cadastro de veículo válida

Usuário inseriu corretamente todos os dados ao cadastrar o veículo

Saída 9: Operação válida

Entrada 10: Busca por categoria de veículo válida

Ao realizar a busca por modelo de um veículo cadastrado

Saída 10: Operação válida

Entrada 11: Busca por categoria de veículo inválida

Ao realizar a busca por um modelo de um veículo e não retornar os veículos cadastrados

Saída 11: Operação inválida ao buscar por modelos.

Entrada 12: Cadastro de veículo inválida

Usuário inseriu erroneamente os dados do veículo

Inseriu a placa com mais de 7 dígitos

Saída 12: Operação inválida: “Placa inválida: contém mais de 7 dígitos”

Entrada 13: Cadastro posto de combustível invalida

Não inseriu o nome do posto de combustível

Saída 13: Operação inválida: “Não inseriu o nome do posto de combustível”

Entrada 14: Exclusão de posto de combustível válida

Ao apertar excluir deve aparecer um pop-up de confirmação e excluir ao apertar SIM

Saída 14: operação válida

Entrada 15: Exclusão de posto de combustível válida

Ao apertar excluir e não excluir o selecionado.

Saída 15: operação válida

Entrada 16: Exclusão de posto de combustível invalida

Ao apertar excluir, excluir de imediato sem exibir o pop-up de confirmação

Saída 16: Operação inválida

Entrada 17: preenchimento das informações do ticket válida.

Ao apertar gerar tickets, o usuário preencheu todos os campos da forma devida.

Saída 17: operação válida.

Entrada 18: preenchimento das informações do ticket inválida.

não foi preenchido os campos devidos.

Saída 18: o sistema não reconhece os campos como obrigatórios e adiciona o ticket.

Entrada 19: Busca por ticket válida.

foi feita a busca por um posto, veículo e status válido.

Saída 19: operação válida.

Entrada 20: Busca por ticket inválida.

foi feita a busca por um posto, veículo e status inválido.

Saída 20: ticket não encontrado.

Entrada 21: Remoção de um ticket válida.

foi realizada a remoção de um ticket.

Saída 21: operação válida.

Entrada 22: foi realizada a edição de um ticket inválido.

Usuário ao clicar no ícone de edição não consegui editar.

Saída 23: operaação válida.

Entrada 23: foi realizada a visualização de um ticket inválido.

Usuário ao clicar no ícone de visualização uma tela branca é visualizada.

Saída 23: operação inválida.

Entrada 24: Utilização do dashboard inválida.

Usuário ao tentar utilizar o dashboard visualiza 3 imagens.

Saída 24: operação inválida.

Entrada 25: Utilização dos Mapas inválida.

Usuário ao tentar utilizar os mapas nada acontece.

Saída 25: operação inválida.

Entrada 26: Sair da aplicação válida.

Usuário clicou no ícone para poder sair do site.

Saída 26: operação válida.

Entrada 27: Acesso as configurações inválidas.

Usuário ao clicar no ícone das configurações nada é apresentado na tela.

Saída 27: operação inválida.

Entrada 28: Acesso ao ícone de perfil válida.

Ao clicar no ícone do usuário dois campos são apresentados, configurações e sair.

Saída 28: operação válida.

Entrada 29: Acesso aos relatórios inválida.

Usuária ao tentar acessar os relatórios nada é apresentado na tela.

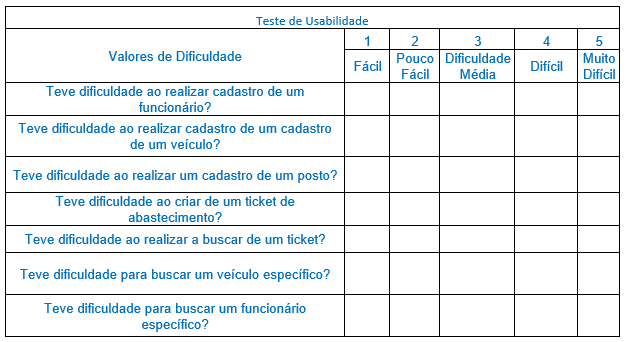
Saída: operação inválida.

## 1.3 Teste do Ciclo de Negócios

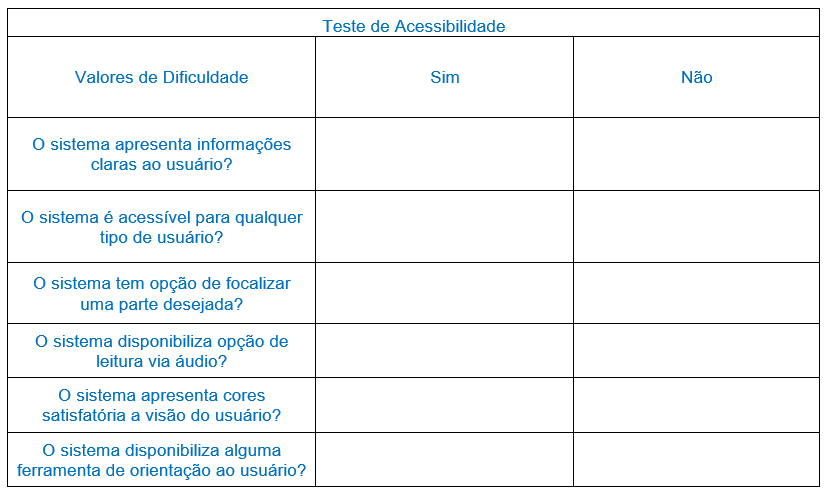
* **Verificar se os relatórios estão sendo gerados corretamente.**
* **Verificar se o tratamento de exceções está correto**
* **Verificar se os campos obrigatórios estão sendo preenchidos em cada formulário e se a validação deles está sendo realizadas**
* **Verificar se os campos estão sendo preenchidos com informações no formato correto em cada formulário**

## 1.1 Teste da Interface do Usuário

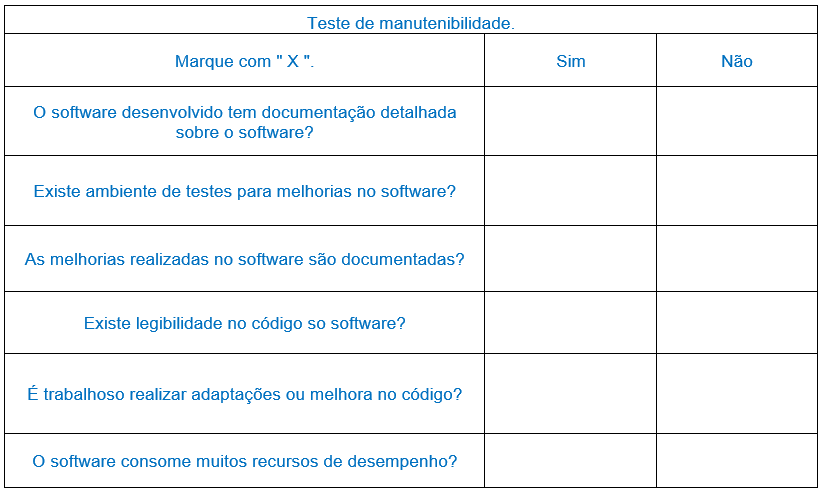
* **Teste de usabilidade**



* **Teste de acessibilidade**



* **Teste de manutenibilidade**



**Teste de preenchimento de formulário (Teste de aceitação)**

Ø **Quem são os testadores e sua função no projeto**

**Testador: Raniery Neiva Martins**

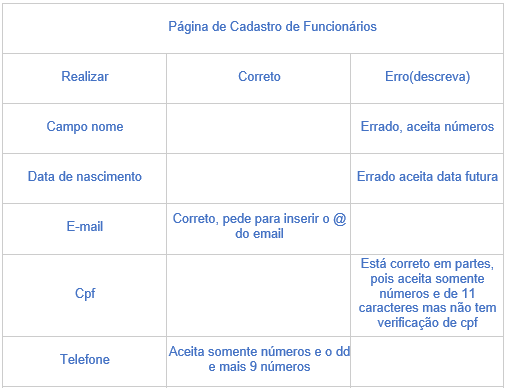
# Ø Funcionalidades a serem testadas:

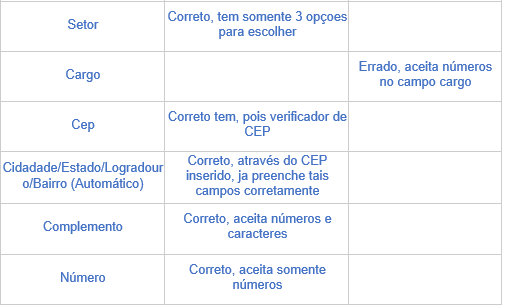
**Preencha o campo referente a função a ser testada, Se houver algum erro descreva-o:**

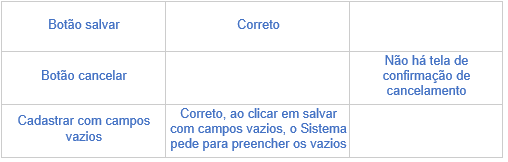
Exemplo.: Campo “Nome” está aceitando letras/números e caracteres especiais.

Marque a caixa Erro, pois esse campo só devia aceitar letras, e não números e caracteres especiais.









# Ø Erros encontrados / Inconsistências e em quais requisitos o sistema não atende

Algumas inconsistências foram encontradas no sistema, como a possibilidade de inserir números e caracteres no campo de nome, além de aceitar datas futuras e muito antigas no campo de data de nascimento.

**Exemplo**: inserir a data de 1760 em data de nascimento.

Outro problema encontrado foi a falta de verificação do CPF inserido, que somente aceita 11 caracteres. Além disso, não há uma tela de confirmação para o botão cancelar, o que pode levar à perda de conteúdo inserido sem querer. Outro problema é que o sistema aceita qualquer coisa no cargo, até mesmo números.

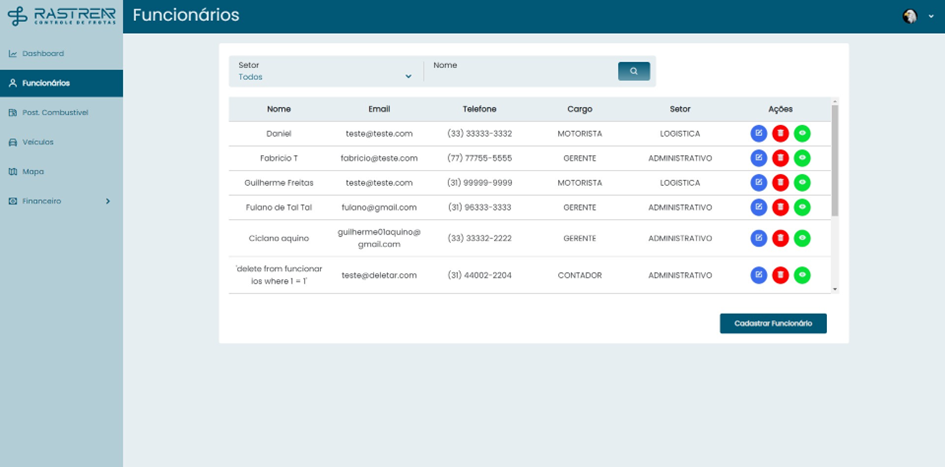
No entanto, o sistema apresenta algumas funcionalidades corretas, como o verificador de CEP, que preenche automaticamente os campos de endereço.

O sistema apresenta algumas inconsistências, mas no quesito organização e o layout escolhido para o sistema, ele atende muito bem.

Como pode ser visto nas imagens a seguir, as ações que podem ser realizadas são diferenciadas pela cor, azul para editar, vermelho para excluir, e verde para visualizar Temos organizados em colunas, os Nomes, E-mail, Telefone, Cargo, Setor Facilitando o fluxo de visualização ao “bater” o olho na página.

Podemos visualizar que o sistema possui, exemplos para a maioria dos campos a serem preenchidos, não deixando dúvida para o que necessita ser inserido

Exemplo telefone: (00) 00000-0000 Email: [joao@gmail.com](mailto:joao@gmail.com)



*Figura 1 - Tela de listagem de funcionários cadastrados*

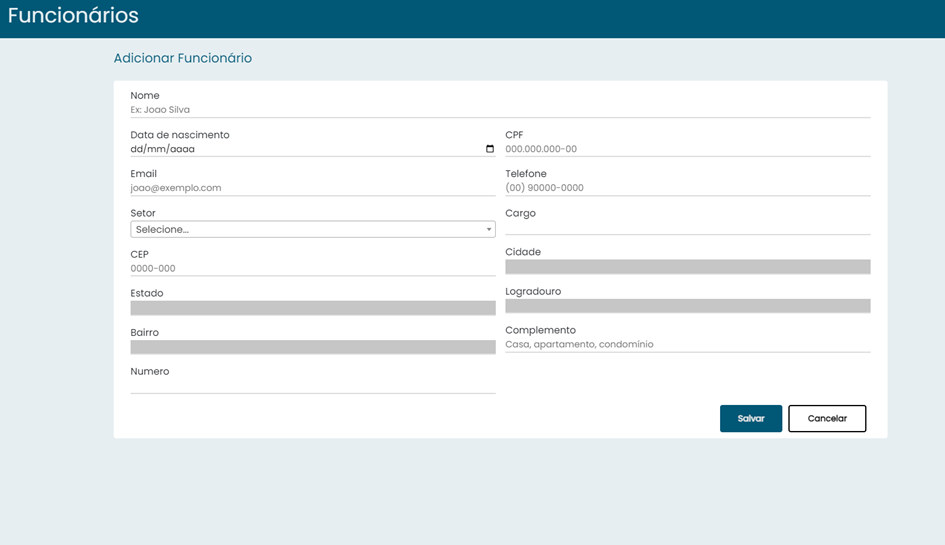


Figura 2 - Tela de cadastro de funcionários

## 1.1 Perfil da Performance

Verificar tempo de resposta no rastreio de veículos em “tempo real”. A localização dos veículos deve atualizar a cada 1 segundo.

## 1.2 Teste de Carga

O teste de carga no presente momento, para controle veículos e acesso administrativo não se aplica a esse projeto, pelo fato do sistema não possuir muitos acessos simultâneos.

Para controle de frotas via GPS, funcionalidade ainda não disponível no sistema web, o número de acessos aumenta significativamente, neste caso será necessário um teste de carga para avaliar até que ponto o sistema tem a capacidade de responder da forma correta a demanda requisitada.

## 1.3 Teste de Stress

O mesmo caso no Teste de Carga também se aplica ao Teste de Stress, poucos usuários irão fazer acessos ao mesmo tempo, facilitando a operação normal da aplicação. Em questão do uso de memória RAM, por ser um sistema web pouca memória interna é requisitada. O sistema não demanda muito mais do que um site no navegador.

## 1.4 Teste de Segurança e de Controle de Acesso

**Caso de teste: emissão de ticket de abastecimento usuário Motorista:**

Usuário tem permissão apenas de solicitar a emissão de tickets de abastecimento.

**Entrada**: Usuário com login do tipo motorista tenta emitir um ticket de abastecimento.

**Saida**: Opção não disponível para usuário logado.

**Caso de teste: solicitar ticket de abastecimento usuário Motorista:**

Usuário tem permissão apenas de solicitar a emissão de tickets de abastecimento.

**Entrada**: Usuário com login do tipo motorista tenta realizar a solicitação de um ticket de abastecimento.

**Saida**: Opção disponível para usuário logado. Tarefa realizada com sucesso.

**Caso de teste: gerar relatório de manutenções mensais usuário Motorista:**

Usuário tem permissão para pré-cadastrar manutenções no sistema.

**Entrada**: Usuário com login do tipo motorista tenta gerar um relatório das manutenções cadastradas

**Saida**: Opção não disponível para usuário logado.

**Caso de teste: realizar cadastro de um novo veículo usuário Motorista:**

Usuário tem permissão apenas realizar checklist dos veículos cadastrados.

**Entrada**: Usuário com login do tipo motorista tenta realizar o cadastro de um veículo no sistema.

**Saida**: Opção não disponível para usuário logado.

**Caso de teste: realizar checklist de algum veículo cadastrado no sistema usuário Motorista:**

Usuário tem permissão apenas realizar checklist dos veículos cadastrados.

**Entrada**: Usuário com login do tipo motorista tenta preencher o checklist de um caminhão já cadastrado no sistema.

**Saida**: Mediante informação da placa do veículo, operação é realizada com sucesso.

## 1.5 Teste de Falha/Recuperação

Nenhum.

## 1.6 Teste de Instalação

O sistema será Web, não sendo necessário instalar diretamente nas máquinas dos usuários

# 2. Estratégia de Teste

Apresenta um conjunto de tipos de testes a serem realizados, respectivas técnicas empregadas e critério de finalização de teste. Além disso, é listado o conjunto de ferramentas utilizadas.

## 2.1 Tipos de Teste

### *1.1.1* *Teste de Integridade de Dados e do Banco de Dados*

### *1.1.1*

|  |  |
| --- | --- |
| Objetivo do Teste: | Garantir que o acesso ao banco de dados funciona adequadamente e sem inconsistência dos dados. |
| Técnica: | * Invocar cada método de acesso ao banco de dados, necessários para a utilização das funcionalidades do sistema alimentando cada um com dados válidos e inválidos. * Inspecionar o banco de dados através de um gerenciador de banco de dados e verificar se os dados nas tabelas estão de acordo com as ações realizadas |
| Critério de Finalização: | Todos os métodos e processos de acesso à base de dados funcionam como projetados e sem nenhuma corrupção de dados.  Os dados salvos na base de dados estão de acordo com o informado pelo usuário. |
| Considerações Especiais: | * Os testes podem necessitar de um ambiente de desenvolvimento ou drivers de SGBD para inserir ou modificar os dados diretamente na base de dados. |

### *1.1.1* *Teste de Função*

### *1.1.1*

|  |  |
| --- | --- |
| Objetivo do Teste: | * Garantir que as funcionalidades do sistema, especificadas nos casos de usos, estão gerando os resultados esperados conforme especificados nos casos de teste. |
| Técnica: | * Executar cada caso de uso funcional através de seu fluxo principal e secundário, usando dados válidos e inválidos, para verificar o seguinte: * Os resultados esperados ocorrem quando dados válidos são usados. * As mensagens de erro ou aviso apropriadas são exibidas quando dados inválidos são usados. * Cada regra de negócio é aplicada apropriadamente. |
| Critério de Finalização: | * Todos os testes planejados foram executados. * Todos os defeitos identificados foram tratados. |
| Considerações Especiais: | Utilizar framework pHpUnit para realização dos testes funcionais no sistema web. |

### 

**1.1.2 Teste da Interface do Usuário**

### *1.1.1*

|  |  |
| --- | --- |
| Objetivo do Teste: | * Verificar se a navegação nas telas do sistema seja clara e práticas para o usuário e ajude a diminuir a curva de aprendizagem para utilização do sistema. * Objetos e características da janela, tais como menus, tamanho dos botões, posição, estado e foco estão padronizados em todo o sistema. |
| Técnica: | * Criar formulários com tarefas para o usuário realizar dentro do sistema. Para cada tarefa, são realizadas perguntas relacionadas à dificuldade para realização da tarefa, se houve algum erro e se as opções estavam claras e fáceis de encontrar. * Criar tabelas de avaliação dos pontos de usabilidade e acessibilidade da interface do sistema. |
| Critério de Finalização: | * É verificado que cada janela permanece consistente com a versão dos protótipos ou dentro de padrões aceitáveis. * É verificado que o usuário consegue usar a interface sem precisar de treinamento e a considera agradável. |
| Considerações Especiais: | Nenhuma. |

### *1.1.1* *Teste de Performance*

### *1.1.1*

|  |  |
| --- | --- |
| Objetivo do Teste: | Verificar os comportamentos do sistema em relação à sua performance sob as seguintes condições:   * Utilização normal do sistema para algumas funções específicas. |
| Técnica: | * Usar Procedimentos de Teste desenvolvidos para Teste da Função e Ciclo de Negócio. * Scripts devem ser rodados em uma máquina e utilizando transações para simular várias requisições. |
| Critério de Finalização: | * Único usuário ou transação: finalização com sucesso sem nenhuma falha e dentro do tempo especificado * Múltiplos usuários ou transações: finalização bem-sucedida sem qualquer falha e dentro do tempo especificado. |
| Considerações Especiais: | * “Direcionar transações” diretamente para o servidor, usualmente na forma de chamadas SQL. * Usar múltiplos clientes físicos, cada um rodando scripts de teste para gerar uma carga no sistema. * As bases de dados usadas para o Teste de Performance devem ser ou do tamanho real ou proporcionalmente iguais. |

### *1.1.2 Teste de Carga*

|  |  |
| --- | --- |
| Objetivo do Teste: | Verificar o funcionamento do sistema sobrecarregado. |
| Técnica: | Usar testes desenvolvidos para o Teste do Ciclo de Negócio ou Função, aumentando o tamanho da carga de dados inseridos e verificados no servidor, ate encontrar o limite de funcionamento do servidor. Verificando a seguir a compatibilidade dos dados e as regras de negócios. |
| Critério de Finalização: | Uma sobrecarga possível para o ambiente para o qual o ambiente está sendo desenvolvido deve ser suportada corretamente e sem comprometer a eficiência do sistema. |

### 

### *1.1.3 Teste de Segurança e Controle de Acesso*

|  |  |
| --- | --- |
| Objetivo do Teste: | Verificar que apenas aqueles usuários com acesso ao sistema e aplicações têm permissão de acessá-los. Este usuário pode acessar apenas aquelas funções ou dados para os quais o seu tipo de usuário tem permissão. |
| Técnica: | * Segurança do Nível de Aplicação: Identifique e liste cada tipo de usuário e as funções ou dados para os quais cada tipo tem permissão. * Crie testes para cada tipo de usuário e verifique cada permissão criando transações específicas para cada tipo de usuário. * Modifique o tipo de usuário e repita os testes para os mesmos usuários. Em cada caso, verifique que funções ou dados adicionais estão corretamente disponíveis ou negados. * Acesso de Nível de Sistema: Ver Considerações Especiais abaixo. |
| Critério de Finalização: | Para cada tipo de ator conhecido as funções ou dados apropriados estão disponíveis, e todas as transações funcionam como esperado e rodam nos Testes de Função anteriores. |
| Considerações Especiais: | O Acesso ao sistema deve ser revisado ou discutido com o administrador de rede ou de sistema apropriado. Esse teste pode não ser necessário já que ele pode ser uma função da administração da rede ou sistema. |

### 

### *1.1.4 Teste de Instalação*

|  |  |
| --- | --- |
| Objetivo do Teste: | Verifique que os alvos de teste instalam apropriadamente em cada configuração de hardware necessária sobre as seguintes condições:   * Uma nova instalação, em uma nova máquina, que nunca fora anteriormente instalada. * Atualização, numa máquina aonde, o software já fora previamente instalado, para a mesma versão. * Atualização, numa máquina que já disponha do software instalado, de uma versão mais velha. |
| Técnica: | Começar ou executar a instalação. |
| Critério de Finalização: | As transações do software executam de forma bem-sucedida, sem falha. |
| Considerações Especiais: | Saber antecipadamente quais transações do software devem ser selecionadas para abranger um teste de confiança de que a aplicação foi instalada de forma bem-sucedida e que nenhum componente importante de software está faltando. |

# 1. Equipe

Contém descrição da equipe e da infraestrutura utilizada para o desenvolvimento das atividades de testes, incluindo: pessoal, equipamentos, software de apoio, materiais, dentre outros. Isto visa garantir uma estrutura adequada para a execução das atividades de testes previstas no plano.

## 1.1 Equipe de teste

Essa tabela mostra as suposições de recrutamento para o projeto.

|  |  |  |
| --- | --- | --- |
| Recursos Humanos | | |
| Trabalhador | Recursos Mínimos Recomendados | Responsabilidades Específicas ou Comentários |
| Gerente de Teste,  Gerente do Projeto de Teste | Guilherme | Fornece supervisionamento gerencial.  Responsabilidades:   * provê direcionamento técnico * adquire recursos apropriados * fornece relatórios de gerenciamento |
| Test Designer | Daniel | Identifica, prioriza, e implementa os casos de teste.  Responsabilidades:   * gera o plano de teste * cria o modelo de teste * avalia a efetividade do esforço de teste |
| Testador | Fernando | Executa os testes.  Responsabilidades:   * executar os testes * registrar os resultados * reestabelecer-se dos erros * documentar solicitações de mudança |
| Administrador do Sistema de Teste | Raniery | Garante que o ambiente e os bens de teste sejam gerenciados e mantidos.  Responsabilidades:   * administrar o sistema de gerenciamento teste * instalar e gerenciar o acesso do trabalhador ao sistema de testes |
| Gerente do Banco de Dados,  Administrador do Banco de Dados | Fabricio | Garante que o ambiente e bens de teste de dados (banco de dados) sejam gerenciados e mantidos.  Responsabilidades:   * administrar os dados de teste (base de dados) |
| Designer | Fernando | Identifica e define as operações, atributos, e associações das classes de teste.  Responsabilidades:   * identificar e definir as classes de teste * identificar e definir os pacotes de teste |
| Implementador | Fabricio / Raniery | Implementa e faz os testes unitários das classes e pacotes de teste.  Responsabilidades:   * cria as classes e pacotes de teste implementados no modelo de teste |

## 1.2 Infraestrutura

A tabela seguinte expõe os recursos do sistema para o projeto de teste.

|  |
| --- |
| Recursos do Sistema |
| Servidor de Banco de Dados |
| — MySQL DataBase Server |
| Terminais Clientes |
| —1 PC conectado com o banco de dados |
| Repositório de Testes |
| —3 PCs de Desenvolvimento de Teste |

## 2. Cronograma

Contém uma descrição de marcos importantes (milestones) das atividades (incluindo as datas de início e fim da atividade). Apenas marcos relevantes devem ser listados, ou seja, aqueles que contribuirão nas atividades de testes. Por exemplo: projeto de testes, execução de testes ou avaliação de testes.

|  |  |  |
| --- | --- | --- |
| ***ATIVIDADE*** | ***Início*** | ***Final*** |
| Planejamento de Testes | 16/02/2023 | 01/03/2023 |
| Projetar Testes | 02/03/2023 | 12/04/2023 |
| Implementar Testes | 13/04/2023 | 10/05/2023 |
| Execução de Testes | 11/05/2023 | 25/05/2023 |
| Avaliação de Testes | 12/06/2023 | 22/06/2023 |

## 

1. Itens de Configuração

* Sistema de Gerenciamento da Frota: Realiza o gerenciamento dos veículos e custos financeiros gerados por cada um. Além disso permite verificar as condições do veículo pelo checklist realizado pelo funcionário.
* Banco de Dados: faz o CRUD dos dados recebidos pelo sistema e os mantém armazenados.
* Documentação do Projeto: Necessário para formalizar o projeto, além de contribuir para a correção de possíveis falhas de comunicação durante a criação do sistema.
* Configurações do Sistema: Todas as configurações do sistema, incluindo configurações de segurança, configurações de banco de dados, configurações de autenticação e autorização, devem ser tratadas como Itens de Configuração.
* Regras de Acesso:As regras de acesso que definem os privilégios de diferentes usuários no sistema também são importantes Itens de Configuração.
* **Arquivos de Log:** Os arquivos de log do sistema, que registram atividades e eventos, podem ser úteis para solucionar problemas e rastrear atividades. Eles devem ser mantidos como Itens de Configuração.

1. Organização de Código-Fonte
   1. Versão Programação para Web

<?php

function uniqueValue($datas, $typeValue)

{

$dataList = array();

foreach ($datas as $indice => $data) {

array\_push($dataList, $data[$typeValue]);

}

return array\_unique($dataList);

}

?>

<div class="panelBody">

<form action="/funcionarios" method="GET">

<div class="filter-wrapper">

<div class="filter">

<div class="filter-select">

<span class="label">Setor</span>

<select name="setor" id="setor">

<option value=''>Todos</option>

<option <?php if ($selectedSetor == 'administrativo') echo 'selected'; ?> value="administrativo"> Administrativo </option>

<option <?php if ($selectedSetor == 'financeiro') echo 'selected'; ?> value="financeiro"> Financeiro </option>

<option <?php if ($selectedSetor == 'logistica') echo 'selected'; ?> value="logistica"> Logística </option>

</select>

<i class="custom-arrow-down-type bx bxs-chevron-down"></i>

<i class="custom-arrow-up-type bx bxs-chevron-up"></i>

</div>

</div>

<div class="line"></div>

<div class="filter">

<div class="filter-select">

<span class="label">Nome</span>

<input type="text" name="nome" id="nome" value="<?= $selectedNome ?>">

</div>

</div>

<button class="search filter-btn" type="submit" name="search" value="search">

<i class="bx bx-search"></i>

</button>

</div>

</form>

<div class="table">

<div class="table-section">

<table>

<thead>

<tr>

<th>Nome</th>

<th>Email</th>

<th>Telefone</th>

<th>Cargo</th>

<th>Setor</th>

<th>Ações</th>

</tr>

</thead>

<tbody>

<?php

foreach ($funcionarios as $indice => $funcionario) {

?>

<tr>

<td><?= $funcionario->nome ?></td>

<td><?= $funcionario->email ?></td>

<td><?= $funcionario->telefone ?></td>

<td><?= strtoupper($funcionario->cargo) ?></td>

<td><?= strtoupper($funcionario->setor) ?></td>

<td>

<a href="/funcionario\_edit?id=<?= $funcionario->id ?>"><i class="bx bxs-edit"></i></a>

<a href="#" onclick="alertDeleteFuncionario(<?= $funcionario->id ?>)"><i class="bx bxs-trash"></i></a>

<a href="#" onclick="viewFuncionario(<?= $funcionario->id ?>)"><i class="bx bxs-show"></i></a>

</td>

</tr>

<?php } ?>

</tbody>

</table>

</div>

</div>

<div id="fade-view" class="hide"></div>

<div id="modal-view" class="hide">

<div id="modal-header">

<h2>Visualizar</h2>

<button id="close-modal-view">

<i class='bx bx-x'></i>

</button>

</div>

<div id="modal-body">

<div class="containerForm">

<div class="fieldForm">

</div>

</div>

</div>

</div>

<div class="footerButton">

<a href="/funcionario\_add" id="sendBtn">Cadastrar Funcionário</a>

</div>

</div>

<script>

$(".select-brand").click(function() {

$(".container .filter-wrapper .filter .filter-select .custom-arrow-up-brand").toggleClass("selected");

$(".container .filter-wrapper .filter .filter-select .custom-arrow-down-brand").toggleClass("selected");

});

$(".select-type").click(function() {

$(".container .filter-wrapper .filter .filter-select .custom-arrow-up-type").toggleClass("selected");

$(".container .filter-wrapper .filter .filter-select .custom-arrow-down-type").toggleClass("selected");

});

function alertDeleteticket(id) {

Swal.fire({

title: "Excluir Funcionario",

text: "Esse processo não pode ser desfeito.",

icon: 'warning',

showCancelButton: true,

confirmButtonColor: "#ff0000",

cancelButtonColor: "#a9c0d4",

confirmButtonText: "Remover",

cancelButtonText: "Cancelar",

})

.then((result) => {

if (result.isConfirmed) {

$.ajax({

type: "GET",

url: '/funcionario\_delete',

data: {

id: id

},

dataType: "json",

success: function(result) {

console.log(result);

if (result.success) {

Swal.fire({

title: "Funcionário Excluído Com Sucesso!",

icon: "success",

confirmButtonColor: "var(--primary-color)",

confirmButtonText: "Ok",

}).then((result) => {

sessionStorage.setItem("isdeleted", 'true');

location.reload();

})

} else {

Swal.fire({

title: "Error!",

text: result.message,

icon: "error",

confirmButtonColor: "var(--primary-color)",

confirmButtonText: "Ok",

});

}

}

});

} else if (

result.dismiss === swal.DismissReason.cancel

) {

// If cancel do nothing (stay on same page).

}

})

}

$("#close-modal-view").click(function() {

$("#modal-view").toggleClass("hide");

$("#fade-view").toggleClass("hide");

});

$("#fade-view").click(function() {

$("#modal-view").toggleClass("hide");

$("#fade-view").toggleClass("hide");

});

async function viewFuncionario(id) {

await $.ajax({

type: "GET",

url: '/funcionario\_view',

data: {

id: id

},

contentType: "application/json",

dataType: "json",

success: function(response) {

console.log(response);

if (response.success) {

let data = response.data;

$('.fieldForm').html('')

$(".fieldForm").append(`

<div class="box-input-view">

<label>Nome</label>

<input class="input input-view" type="text" value="${data.nome}" readonly>

</div>

<div class="box-input-view">

<label>Data nascimento</label>

<input class="input input-view" type="text" value="${dataFormatada(new Date(data.dataNascimento))}" readonly>

</div>

<div class="box-input-view">

<label>Email</label>

<input class="input input-view" type="text" value="${data.email}" readonly>

</div>

<div class="box-input-view">

<label>Telefone</label>

<input class="input input-view" type="text" value="${data.telefone}" readonly>

</div>

<div class="box-input-view">

<label>Setor</label>

<input class="input input-view" type="text" value="${data.setor.toUpperCase()}" readonly>

</div>

<div class="box-input-view">

<label>Cargo</label>

<input class="input input-view" type="text" value="${data.cargo}" readonly>

</div>

<div class="box-input-view">

<label>Endereço completo</label>

<input class="input input-view" type="text" value="${data.logradouro}, bairro ${data.bairro}, ${data.cidade} - ${data.estado}, CEP - ${data.cep}" readonly>

</div>

`)

$("#modal-view").toggleClass("hide");

$("#fade-view").toggleClass("hide");

} else {

Swal.fire({

title: "Error!",

text: `${response.message}`,

icon: "error",

confirmButtonColor: "var(--primary-color)",

confirmButtonText: "Ok",

});

}

console.log("Veiculo retornado com sucesso.");

},

error: function(response) {

console.log("Error no Banco de Dados.");

Swal.fire({

title: "Error!",

text: "Funcionário não retornado com sucesso",

icon: "error",

confirmButtonColor: "var(--primary-color)",

confirmButtonText: "Ok",

});

}

});

}

</script>

<?php

function uniqueValue($datas, $typeValue)

{

$dataList = array();

foreach ($datas as $indice => $data) {

array\_push($dataList, $data[$typeValue]);

}

return array\_unique($dataList);

}

?>

<div class="panelBody">

<form action="/funcionarios" method="GET">

<div class="filter-wrapper">

<div class="filter">

<div class="filter-select">

<span class="label">Setor</span>

<select name="setor" id="setor">

<option value=''>Todos</option>

<option <?php if ($selectedSetor == 'administrativo') echo 'selected'; ?> value="administrativo"> Administrativo </option>

<option <?php if ($selectedSetor == 'financeiro') echo 'selected'; ?> value="financeiro"> Financeiro </option>

<option <?php if ($selectedSetor == 'logistica') echo 'selected'; ?> value="logistica"> Logística </option>

</select>

<i class="custom-arrow-down-type bx bxs-chevron-down"></i>

<i class="custom-arrow-up-type bx bxs-chevron-up"></i>

</div>

</div>

<div class="line"></div>

<div class="filter">

<div class="filter-select">

<span class="label">Nome</span>

<input type="text" name="nome" id="nome" value="<?= $selectedNome ?>">

</div>

</div>

<button class="search filter-btn" type="submit" name="search" value="search">

<i class="bx bx-search"></i>

</button>

</div>

</form>

<div class="table">

<div class="table-section">

<table>

<thead>

<tr>

<th>Nome</th>

<th>Email</th>

<th>Telefone</th>

<th>Cargo</th>

<th>Setor</th>

<th>Ações</th>

</tr>

</thead>

<tbody>

<?php

foreach ($funcionarios as $indice => $funcionario) {

?>

<tr>

<td><?= $funcionario->nome ?></td>

<td><?= $funcionario->email ?></td>

<td><?= $funcionario->telefone ?></td>

<td><?= strtoupper($funcionario->cargo) ?></td>

<td><?= strtoupper($funcionario->setor) ?></td>

<td>

<a href="/funcionario\_edit?id=<?= $funcionario->id ?>"><i class="bx bxs-edit"></i></a>

<a href="#" onclick="alertDeleteFuncionario(<?= $funcionario->id ?>)"><i class="bx bxs-trash"></i></a>

<a href="#" onclick="viewFuncionario(<?= $funcionario->id ?>)"><i class="bx bxs-show"></i></a>

</td>

</tr>

<?php } ?>

</tbody>

</table>

</div>

</div>

<div id="fade-view" class="hide"></div>

<div id="modal-view" class="hide">

<div id="modal-header">

<h2>Visualizar</h2>

<button id="close-modal-view">

<i class='bx bx-x'></i>

</button>

</div>

<div id="modal-body">

<div class="containerForm">

<div class="fieldForm">

</div>

</div>

</div>

</div>

<div class="footerButton">

<a href="/funcionario\_add" id="sendBtn">Cadastrar Funcionário</a>

</div>

</div>

<script>

$(".select-brand").click(function() {

$(".container .filter-wrapper .filter .filter-select .custom-arrow-up-brand").toggleClass("selected");

$(".container .filter-wrapper .filter .filter-select .custom-arrow-down-brand").toggleClass("selected");

});

$(".select-type").click(function() {

$(".container .filter-wrapper .filter .filter-select .custom-arrow-up-type").toggleClass("selected");

$(".container .filter-wrapper .filter .filter-select .custom-arrow-down-type").toggleClass("selected");

});

function alertDeleteticket(id) {

Swal.fire({

title: "Excluir Funcionario",

text: "Esse processo não pode ser desfeito.",

icon: 'warning',

showCancelButton: true,

confirmButtonColor: "#ff0000",

cancelButtonColor: "#a9c0d4",

confirmButtonText: "Remover",

cancelButtonText: "Cancelar",

})

.then((result) => {

if (result.isConfirmed) {

$.ajax({

type: "GET",

url: '/funcionario\_delete',

data: {

id: id

},

dataType: "json",

success: function(result) {

console.log(result);

if (result.success) {

Swal.fire({

title: "Funcionário Excluído Com Sucesso!",

icon: "success",

confirmButtonColor: "var(--primary-color)",

confirmButtonText: "Ok",

}).then((result) => {

sessionStorage.setItem("isdeleted", 'true');

location.reload();

})

} else {

Swal.fire({

title: "Error!",

text: result.message,

icon: "error",

confirmButtonColor: "var(--primary-color)",

confirmButtonText: "Ok",

});

}

}

});

} else if (

result.dismiss === swal.DismissReason.cancel

) {

// If cancel do nothing (stay on same page).

}

})

}

$("#close-modal-view").click(function() {

$("#modal-view").toggleClass("hide");

$("#fade-view").toggleClass("hide");

});

$("#fade-view").click(function() {

$("#modal-view").toggleClass("hide");

$("#fade-view").toggleClass("hide");

});

async function viewFuncionario(id) {

await $.ajax({

type: "GET",

url: '/funcionario\_view',

data: {

id: id

},

contentType: "application/json",

dataType: "json",

success: function(response) {

console.log(response);

if (response.success) {

let data = response.data;

$('.fieldForm').html('')

$(".fieldForm").append(`

<div class="box-input-view">

<label>Nome</label>

<input class="input input-view" type="text" value="${data.nome}" readonly>

</div>

<div class="box-input-view">

<label>Data nascimento</label>

<input class="input input-view" type="text" value="${dataFormatada(new Date(data.dataNascimento))}" readonly>

</div>

<div class="box-input-view">

<label>Email</label>

<input class="input input-view" type="text" value="${data.email}" readonly>

</div>

<div class="box-input-view">

<label>Telefone</label>

<input class="input input-view" type="text" value="${data.telefone}" readonly>

</div>

<div class="box-input-view">

<label>Setor</label>

<input class="input input-view" type="text" value="${data.setor.toUpperCase()}" readonly>

</div>

<div class="box-input-view">

<label>Cargo</label>

<input class="input input-view" type="text" value="${data.cargo}" readonly>

</div>

<div class="box-input-view">

<label>Endereço completo</label>

<input class="input input-view" type="text" value="${data.logradouro}, bairro ${data.bairro}, ${data.cidade} - ${data.estado}, CEP - ${data.cep}" readonly>

</div>

`)

$("#modal-view").toggleClass("hide");

$("#fade-view").toggleClass("hide");

} else {

Swal.fire({

title: "Error!",

text: `${response.message}`,

icon: "error",

confirmButtonColor: "var(--primary-color)",

confirmButtonText: "Ok",

});

}

console.log("Veiculo retornado com sucesso.");

},

error: function(response) {

console.log("Error no Banco de Dados.");

Swal.fire({

title: "Error!",

text: "Funcionário não retornado com sucesso",

icon: "error",

confirmButtonColor: "var(--primary-color)",

confirmButtonText: "Ok",

});

}

});

}

</script>

<?php

function uniqueValue($data, $typeValue)

{

$dataList = array();

foreach ($data as $indice => $d) {

array\_push($dataList, $d->{$typeValue});

}

return array\_unique($dataList);

}

?>

<div class="panelBody">

<form action="/postos\_combustivel" method="GET">

<div class="filter-wrapper">

<div class="filter">

<div class="filter-select">

<span class="label">Cidade</span>

<select name="cidade" id="cidade">

<option value=''>Todas</option>

<?php

$cidades = uniqueValue($postos, "cidade");

$selected = '';

foreach ($cidades as $c) {

if ($c == $selectedCidade) $selected = 'selected';

?>

<option <?= $selected ?> value='<?= $c ?>'><?= $c ?></option>

<?php } ?>

</select>

<i class="custom-arrow-down-type bx bxs-chevron-down"></i>

<i class="custom-arrow-up-type bx bxs-chevron-up"></i>

</div>

</div>

<div class="line"></div>

<div class="filter">

<div class="filter-select">

<span class="label">Nome</span>

<input type="text" name="nome" id="nome" value="<?= $selectedNome ?>">

</div>

</div>

<button class="search filter-btn" type="submit" name="search" value="search">

<i class="bx bx-search"></i>

</button>

</div>

</form>

<div class="table">

<div class="table-section">

<table>

<thead>

<tr>

<th>Nome</th>

<th>Telefone</th>

<th>Email</th>

<th>Cidade</th>

<th>Ações</th>

</tr>

</thead>

<tbody>

<?php

foreach ($postos as $indice => $posto) {

?>

<tr>

<td><?= $posto->nomeFantasia ?></td>

<td><?= $posto->telefone ?></td>

<td><?= $posto->email ?></td>

<td><?= ucfirst($posto->cidade) ?></td>

<td>

<a href="/posto\_edit?id=<?= $posto->id ?>"><i class="bx bxs-edit"></i></a>

<a onclick="alertDeleteposto(<?= $posto->id ?>)"><i class="bx bxs-trash"></i></a>

<a href="/posto\_view?id=<?= $posto->id ?>"><i class="bx bxs-show"></i></a>

</td>

</tr>

<?php } ?>

</tbody>

</table>

</div>

</div>

<div class="footerButton">

<a href="/adicionar\_posto" id="sendBtn">Cadastrar Posto Combustível</a>

</div>

</div>

<script>

$(".select-brand").click(function() {

$(".container .filter-wrapper .filter .filter-select .custom-arrow-up-brand").toggleClass("selected");

$(".container .filter-wrapper .filter .filter-select .custom-arrow-down-brand").toggleClass("selected");

});

$(".select-type").click(function() {

$(".container .filter-wrapper .filter .filter-select .custom-arrow-up-type").toggleClass("selected");

$(".container .filter-wrapper .filter .filter-select .custom-arrow-down-type").toggleClass("selected");

});

function alertDeleteposto(id) {

Swal.fire({

title: "Excluir posto de combustível?",

text: "Esse processo não pode ser desfeito.",

icon: 'warning',

showCancelButton: true,

confirmButtonColor: "#ff0000",

cancelButtonColor: "#a9c0d4",

confirmButtonText: "Remover",

cancelButtonText: "Cancelar",

})

.then((result) => {

if (result.isConfirmed) {

$.ajax({

type: "GET",

url: '/posto\_delete',

data: {

id: id

},

dataType: "json",

success: function(result) {

console.log(result);

if (result.success) {

Swal.fire({

title: "Registro excluido com sucesso!",

icon: "success",

confirmButtonColor: "var(--primary-color)",

confirmButtonText: "Ok",

}).then((result) => {

sessionStorage.setItem("isdeleted", 'true');

location.reload();

})

} else {

Swal.fire({

title: "Error!",

text: result.message,

icon: "error",

confirmButtonColor: "var(--primary-color)",

confirmButtonText: "Ok",

});

}

}

});

} else if (

result.dismiss === swal.DismissReason.cancel

) {

// If cancel do nothing (stay on same page).

}

})

}

</script>

<style>

.filter-wrapper {

width: 100% !important;

}

.pesquisar {

margin-top: 20px;

text-align: end;

}

</style>

<?php

function uniqueValue($datas, $typeValue)

{

$dataList = array();

foreach ($datas as $indice => $data) {

array\_push($dataList, $data[$typeValue]);

}

return array\_unique($dataList);

}

?>

<div class="panelBody">

<form action="/tickets" method="GET">

<div class="filter-wrapper">

<div class="line"></div>

<div class="filter">

<div class="filter-select">

<span class="label">Posto de Combustível</span>

<select name="postoCombustivelId" id="postoCombustivelId">

<option value="all">Todos</option>

<?php

foreach($postos as $p){

$selected2 = '';

if($postoCombustivelSelected == $p->id) $selected2 = 'selected'?>

<option <?= $selected2 ?> value="<?= $p->id ?>"><?= $p->nomeFantasia ?></option>

<?php } ?>

</select>

<i class="custom-arrow-down-type bx bxs-chevron-down"></i>

<i class="custom-arrow-up-type bx bxs-chevron-up"></i>

</div>

</div>

<div class="line"></div>

<div class="filter">

<div class="filter-select">

<span class="label">Veículo</span>

<select name="veiculoId" id="veiculoId">

<option value="all">Todos</option>

<?php

foreach($veiculos as $v){

$selected1 = '';

if($veiculoSelected == $v->id) $selected1 = 'selected'?>

<option <?= $selected1 ?> value="<?= $v->id ?>"><?= $v->modelo ?>-<?= $v->placa ?></option>

<?php } ?>

</select>

<i class="custom-arrow-down-type bx bxs-chevron-down"></i>

<i class="custom-arrow-up-type bx bxs-chevron-up"></i>

</div>

</div>

<div class="line"></div>

<div class="filter">

<div class="filter-select">

<span class="label">Status</span>

<select name="status" id="status">

<option value="all">Todos</option>

<option <?php if ($statusSelected == 'LIBERADO') echo 'selected'; ?> value="LIBERADO">Liberado</option>

<option <?php if ($statusSelected == 'BAIXADO') echo 'selected'; ?> value="BAIXADO">Baixado</option>

<option <?php if ($statusSelected == 'VENCIDO') echo 'selected'; ?> value="VENCIDO">Vencido</option>

</select>

<!-- <span class="label">Motorista</span>

<select name="motoristaId" id="motoristaId">

<option value="all">Todos</option>

<?php

foreach($motoristas as $m){

$selected3 = '';

if($motoristaSelected == $m->id) $selected3 = 'selected'?>

<option <?= $selected3 ?> value="<?= $m->id ?>"><?= $m->nome ?></option>

<?php } ?>

</select> -->

<i class="custom-arrow-down-type bx bxs-chevron-down"></i>

<i class="custom-arrow-up-type bx bxs-chevron-up"></i>

</div>

</div>

</div>

<div class="pesquisar">

<button class="search filter-btn" type="submit" name="search" value="search">

<i class="bx bx-search"></i>

</button>

</div>

</form>

<div class="table">

<div class="table-section">

<table>

<thead>

<tr>

<th>Posto</th>

<th>Combustível</th>

<th>Valor</th>

<th>Veículo</th>

<th>Status</th>

<th>Ações</th>

</tr>

</thead>

<tbody>

<?php

foreach ($tickets as $indice => $ticket) {

?>

<tr>

<td><?= $ticket->nomePosto ?></td>

<td><?= strtoupper(str\_replace('-', ' ',$ticket->tipoCombustivel)) ?></td>

<td><?= $ticket->valor ?></td>

<td><?= $ticket->veiculo . '' . $ticket->veiculoPlaca ?></td>

<td><?= $ticket->status ?></td>

<td>

<?= $ticket->actions ?>

<!-- <a href="/ticket\_edit?id=<?= $ticket->id ?>"><i class="bx bxs-edit"></i></a>

<a onclick="alertDeleteticket(<?= $ticket->id ?>)"><i class="bx bxs-trash"></i></a>

<a href="/baixar\_ticket?id=<?= $ticket->id ?>"><i class="bx bxs-show"></i></a>

<a href="/ticket\_view?id=<?= $ticket->id ?>"><i class="bx bxs-show"></i></a> -->

</td>

</tr>

<?php } ?>

</tbody>

</table>

</div>

</div>

<div class="footerButton">

<a href="/gerar\_ticket" id="sendBtn">Gerar Ticket</a>

</div>

</div>

<script>

$(".select-brand").click(function() {

$(".container .filter-wrapper .filter .filter-select .custom-arrow-up-brand").toggleClass("selected");

$(".container .filter-wrapper .filter .filter-select .custom-arrow-down-brand").toggleClass("selected");

});

$(".select-type").click(function() {

$(".container .filter-wrapper .filter .filter-select .custom-arrow-up-type").toggleClass("selected");

$(".container .filter-wrapper .filter .filter-select .custom-arrow-down-type").toggleClass("selected");

});

function alertDeleteticket(id) {

Swal.fire({

title: "Excluir Funcionario",

text: "Esse processo não pode ser desfeito.",

icon: 'warning',

showCancelButton: true,

confirmButtonColor: "#ff0000",

cancelButtonColor: "#a9c0d4",

confirmButtonText: "Remover",

cancelButtonText: "Cancelar",

})

.then((result) => {

if (result.isConfirmed) {

$.ajax({

type: "GET",

url: '/deletar\_tickets',

data: {

id: id

},

dataType: "json",

success: function(result) {

console.log(result);

if (result.success) {

Swal.fire({

title: "Funcionário Excluído Com Sucesso!",

icon: "success",

confirmButtonColor: "var(--primary-color)",

confirmButtonText: "Ok",

}).then((result) => {

sessionStorage.setItem("isdeleted", 'true');

location.reload();

})

} else {

Swal.fire({

title: "Error!",

text: result.message,

icon: "error",

confirmButtonColor: "var(--primary-color)",

confirmButtonText: "Ok",

});

}

}

});

} else if (

result.dismiss === swal.DismissReason.cancel

) {

// If cancel do nothing (stay on same page).

}

})

}

</script>

* 1. Versão Mobile

Teste CRUD de funcionários

* **Get e Set dos valores nos atributos das classes:**

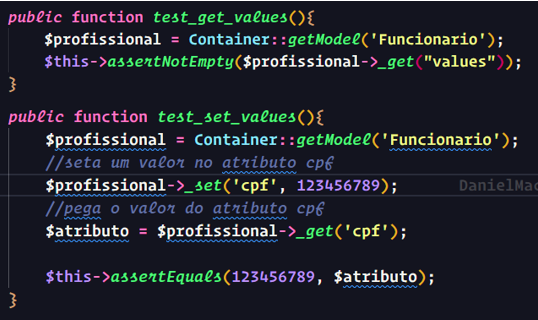
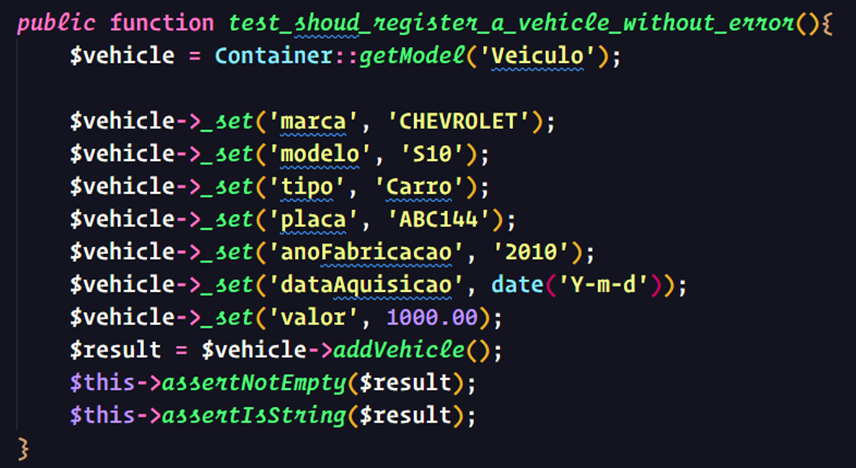


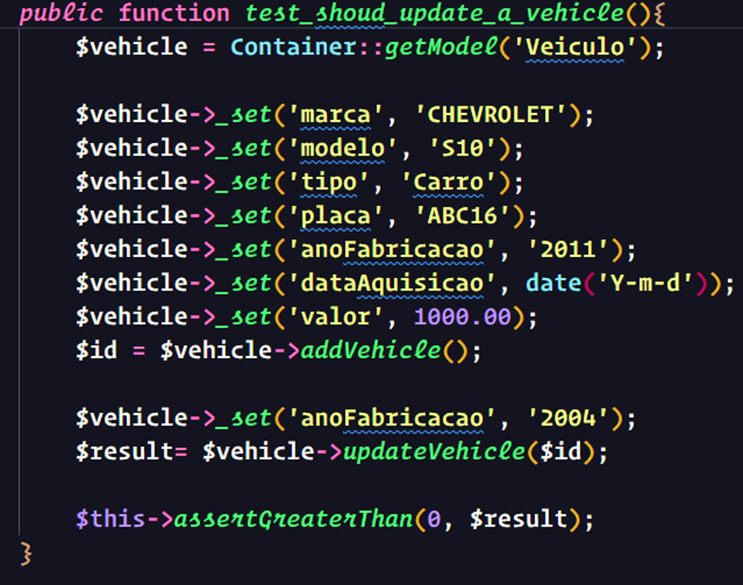
Figura 1 – Teste unitário dos metódos get e set da classe funcionário

* **Criar um veículo:**



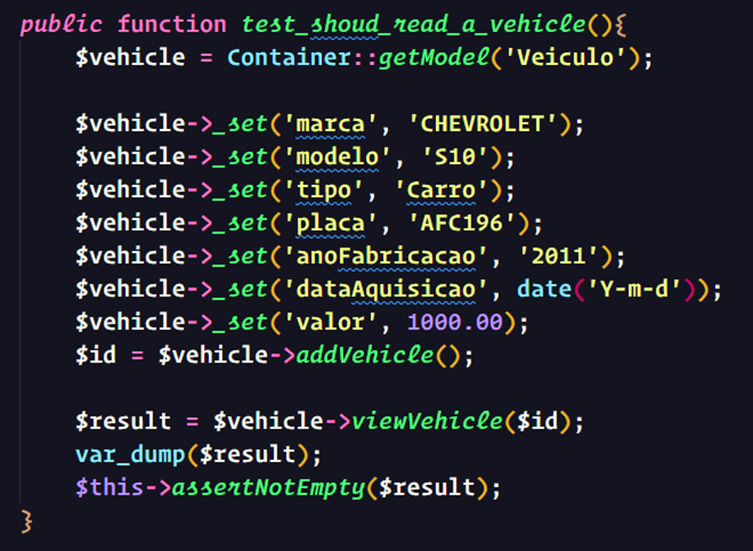
*Figura 2 - Teste unitário para salvar o registro de um veículo*

* **Atualizar um veículo**



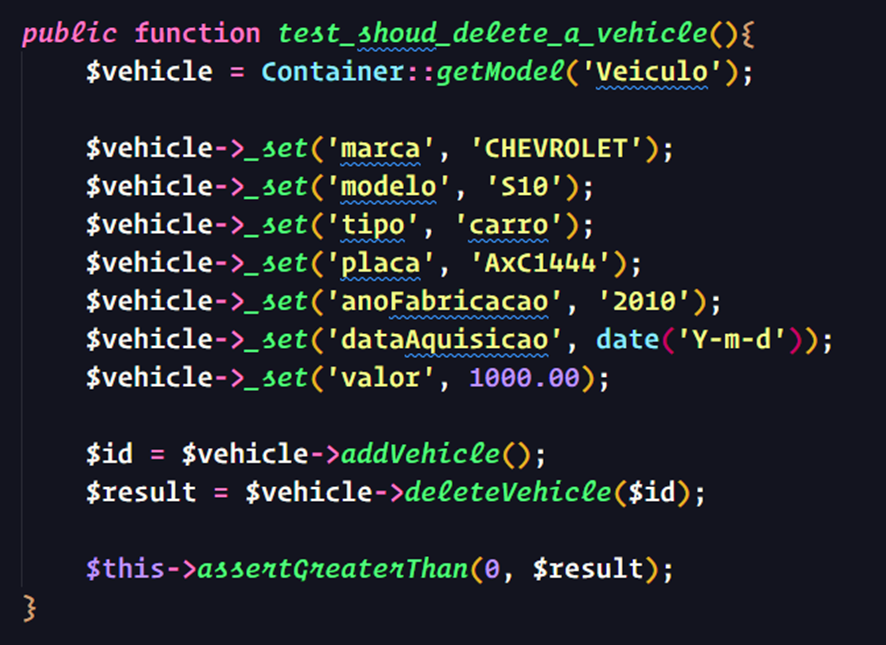
*Figura 3 - Teste unitário para atualizar um veículo*

* **Buscar um veículo**



*Figura 4 - Teste unitário para buscar um veículo*

* **Deletar um veículo**



*Figura 5 - Teste unitário para deletar um veiculo*

TDD Backend mobile

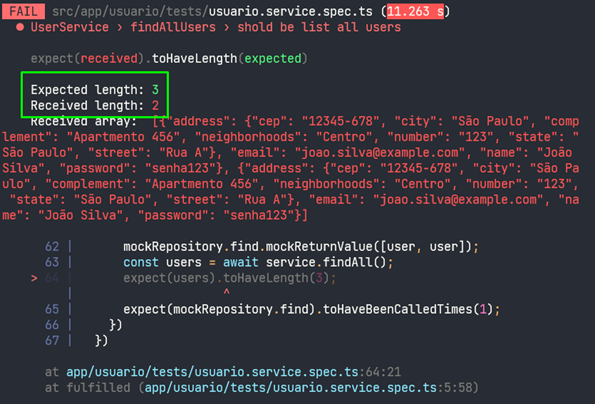
Para a aplicação do TDD no desenvolvimento do backend da aplicação mobile desenvolvida na disciplina de “Sistemas Móveis”, primeiramente foi aplicada em uma função que já estava implementada, a de listar usuários. O intuito era conhecer o framework utilizado.

Para a realização dos testes unitários, foi utilizado o Jest. Um framework de testes para aplicações JavaScript. Para a função de buscar usuários, ao realizar a requisição, a função deve retornar um array contendo os objetos que representam os usuários, segue o código do teste:



*6 - Teste unitário Figura para buscar uma lista de usuários*

Esse teste espera que a função retorne um array com 3 usuários e que a função find seja chamada apenas uma vez.



*Figura 7 – Resultado do teste de buscar usuários*

O teste falhou porque a função retornou um array com 2 usuários, mas foi uma falha no desenvolvimento do próprio teste, pois o mock da função find foi setado para retornar 2 usuários.

Realizando o ajuste, o teste validou a função corretamente.

Após praticar com funções existentes, pensamos nos teste para uma função que ainda não havia sido desenvolvida, que nesse caso foi a função para autenticar um usuário na aplicação.

A função deve receber um email e senha, verificar se o usuário existe pelo email e se a senha é correspondente.

Foi desenvolvido os seguintes testes:

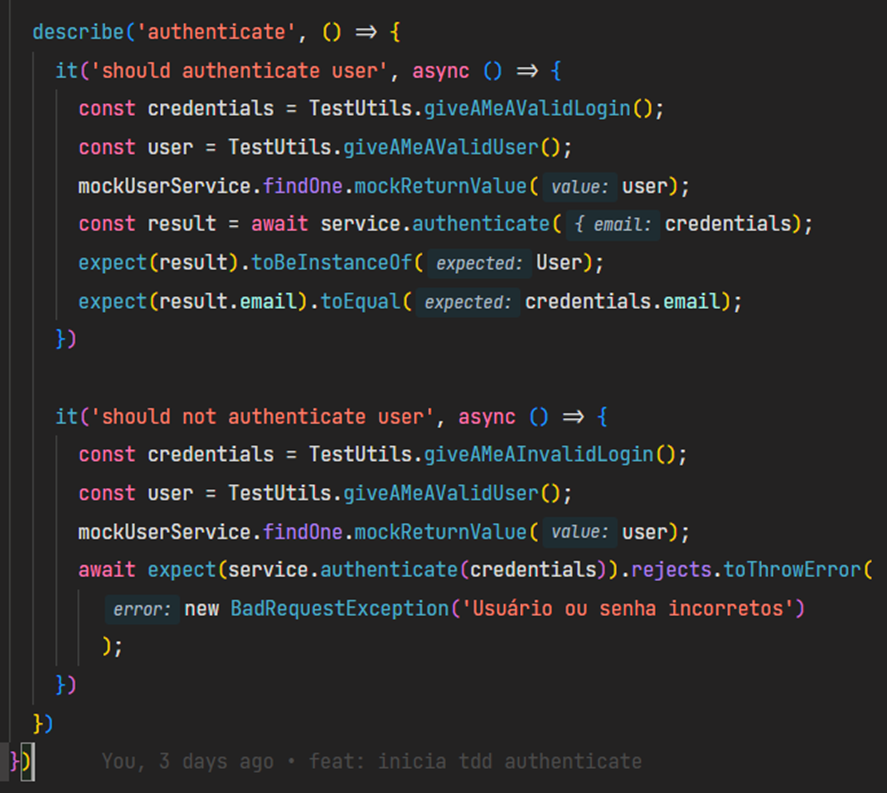
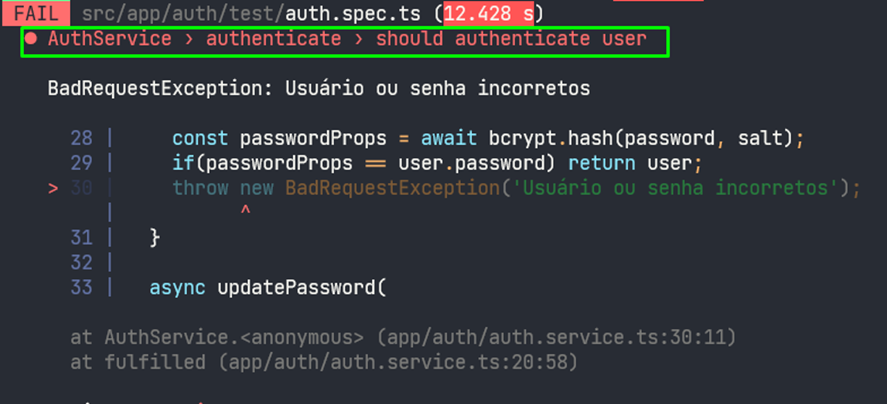


Figura 8 - Testes unitário para autenticar um usuário

O primeiro teste “should authenticate user” passa as credenciais válidas e espera que a função retorne a instancia do usuário que está sendo logado no sistema.

O segundo teste “should not authenticate user”, passa credenciais incorretas e espera que a função lance uma exceção de usuário ou senha incorretos.

Ao realizar os testes, obtivermos o seguinte resultado:



*Figura 9 - Resultado falha do teste para autenticar um usuário válido*

Esse resultado demonstra que o primeiro teste não passou. Verificamos que a funcionalidade de autenticar o usuário estava criptografando a senha para compara com a senha do banco. Porém a função de salvar o usuário não estava criptografando a senha antes de salvar, fazendo com que a comparação não desse certo. Após ajustar as incoerências, rodamos os testes novamente e ele passou, indicando que o erro foi corrigido.



*Figura 10 - Resultado teste de usuário e teste de autenticação*

Como o framework utilizado ainda não era conhecido pela equipe e a prática com TDD é algo novo, houve algumas dificuldades em implementar os testes e pensar em casos de uso racionais que fosse eficaz. No entanto, mesmo com pouca experiencia, foi possível descobrir uma falha de outra função relacionada através dos testes desenvolvidos além de deixar o desenvolvimento da funcionalidade mais decisivo, com a ideia do que a função deve cumprir mesmo antes de começar a desenvolvê-la. Isso mostra que o TDD quando bem aplicado, auxilia na obtenção de produtos com mais qualidade e mais assertividade no desenvolvimento.

* 1. Script do Banco de Dados

-- MySQL dump 10.13 Distrib 8.0.19, for Win64 (x86\_64)

--

-- Host: localhost Database: app\_scf

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

-- Server version 5.5.5-10.10.2-MariaDB

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!50503 SET NAMES utf8mb4 \*/;

/\*!40103 SET @OLD\_TIME\_ZONE=@@TIME\_ZONE \*/;

/\*!40103 SET TIME\_ZONE='+00:00' \*/;

/\*!40014 SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0 \*/;

/\*!40014 SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0 \*/;

/\*!40101 SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='NO\_AUTO\_VALUE\_ON\_ZERO' \*/;

/\*!40111 SET @OLD\_SQL\_NOTES=@@SQL\_NOTES, SQL\_NOTES=0 \*/;

--

-- Table structure for table `checklist`

--

DROP TABLE IF EXISTS `checklist`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `checklist` (

`createdAt` datetime(6) NOT NULL DEFAULT current\_timestamp(6),

`updatedAt` datetime(6) NOT NULL DEFAULT current\_timestamp(6) ON UPDATE current\_timestamp(6),

`deletedAt` datetime(6) DEFAULT NULL,

`id` int(11) NOT NULL AUTO\_INCREMENT,

`typeVehicle` enum('passeio','caminhonete','caminhao','carreta','van') NOT NULL,

`options` longtext DEFAULT NULL,

`commonOptions` longtext DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1\_swedish\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `checklist`

--

LOCK TABLES `checklist` WRITE;

/\*!40000 ALTER TABLE `checklist` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `checklist` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `checklist-vehicle`

--

DROP TABLE IF EXISTS `checklist-vehicle`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `checklist-vehicle` (

`createdAt` datetime(6) NOT NULL DEFAULT current\_timestamp(6),

`updatedAt` datetime(6) NOT NULL DEFAULT current\_timestamp(6) ON UPDATE current\_timestamp(6),

`deletedAt` datetime(6) DEFAULT NULL,

`id` int(11) NOT NULL AUTO\_INCREMENT,

`kmAtual` int(11) NOT NULL,

`type` enum('entrada','saida') NOT NULL,

`options` longtext DEFAULT NULL,

`vehicleId` int(11) DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `FK\_ecc3514abc910b0c0653e69822f` (`vehicleId`),

CONSTRAINT `FK\_ecc3514abc910b0c0653e69822f` FOREIGN KEY (`vehicleId`) REFERENCES `vehicles` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION

) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1\_swedish\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `checklist-vehicle`

--

LOCK TABLES `checklist-vehicle` WRITE;

/\*!40000 ALTER TABLE `checklist-vehicle` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `checklist-vehicle` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `roles`

--

DROP TABLE IF EXISTS `roles`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `roles` (

`createdAt` datetime(6) NOT NULL DEFAULT current\_timestamp(6),

`updatedAt` datetime(6) NOT NULL DEFAULT current\_timestamp(6) ON UPDATE current\_timestamp(6),

`deletedAt` datetime(6) DEFAULT NULL,

`id` int(11) NOT NULL AUTO\_INCREMENT,

`name` varchar(255) NOT NULL,

`permissions` text DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1\_swedish\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `roles`

--

LOCK TABLES `roles` WRITE;

/\*!40000 ALTER TABLE `roles` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `roles` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `users`

--

DROP TABLE IF EXISTS `users`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `users` (

`createdAt` datetime(6) NOT NULL DEFAULT current\_timestamp(6),

`updatedAt` datetime(6) NOT NULL DEFAULT current\_timestamp(6) ON UPDATE current\_timestamp(6),

`deletedAt` datetime(6) DEFAULT NULL,

`id` int(11) NOT NULL AUTO\_INCREMENT,

`name` varchar(255) NOT NULL,

`email` varchar(255) NOT NULL,

`password` varchar(255) NOT NULL,

`status` enum('ativo','inativo') NOT NULL DEFAULT 'ativo',

`roleId` int(11) DEFAULT NULL,

`street` varchar(255) DEFAULT NULL,

`cep` varchar(20) DEFAULT NULL,

`state` varchar(4) DEFAULT NULL,

`city` varchar(255) DEFAULT NULL,

`neighborhoods` varchar(255) DEFAULT NULL,

`number` varchar(255) DEFAULT NULL,

`complement` varchar(255) DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `FK\_368e146b785b574f42ae9e53d5e` (`roleId`),

CONSTRAINT `FK\_368e146b785b574f42ae9e53d5e` FOREIGN KEY (`roleId`) REFERENCES `roles` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION

) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1\_swedish\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `users`

--

LOCK TABLES `users` WRITE;

/\*!40000 ALTER TABLE `users` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `users` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `vehicles`

--

DROP TABLE IF EXISTS `vehicles`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `vehicles` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`model` varchar(255) NOT NULL,

`yearManufacture` varchar(255) NOT NULL,

`plate` varchar(255) NOT NULL,

`dateAcquisition` datetime NOT NULL,

`type` enum('passeio','caminhonete','caminhao','carreta','van') NOT NULL,

`status` enum('ativo','inativo') NOT NULL DEFAULT 'ativo',

`image` varchar(255) DEFAULT NULL,

`createdAt` datetime(6) NOT NULL DEFAULT current\_timestamp(6),

`updatedAt` datetime(6) NOT NULL DEFAULT current\_timestamp(6) ON UPDATE current\_timestamp(6),

`deletedAt` datetime(6) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1\_swedish\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `vehicles`

--

LOCK TABLES `vehicles` WRITE;

/\*!40000 ALTER TABLE `vehicles` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `vehicles` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Dumping routines for database 'app\_scf'

--

/\*!40103 SET TIME\_ZONE=@OLD\_TIME\_ZONE \*/;

/\*!40101 SET SQL\_MODE=@OLD\_SQL\_MODE \*/;

/\*!40014 SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS \*/;

/\*!40014 SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS \*/;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

/\*!40111 SET SQL\_NOTES=@OLD\_SQL\_NOTES \*/;

-- Dump completed on 2023-08-31 20:36:46