**GNA**

**AutoMach**

Guilherme Francisco – RM557648

Nicolas Dorizoti - RM556868

Anderson de Sousa - RM557002

**Sumário**

1. Introdução
2. Objetivo e Escopo do Projeto
3. Funcionalidades da Solução
4. Protótipo do Sistema
5. Modelo do Banco de Dados
6. Diagrama de Classes
7. Procedimentos para Rodar a Aplicação
8. Tabela dos endpoints
9. **Introdução**

A gestão eficaz de veículos e suas respectivas manutenções é um desafio comum em muitos setores, especialmente para empresas que dependem de uma frota para suas operações diárias. Seja para empresas de transporte, serviços de entrega, ou até mesmo para indivíduos que possuem múltiplos veículos, manter o controle sobre o estado de cada veículo e suas necessidades de manutenção é crucial para garantir a eficiência, a segurança e a longevidade dos ativos.

A motivação para o desenvolvimento desta solução surgiu da necessidade de simplificar e automatizar o processo de gestão de veículos e manutenções, que tradicionalmente é realizado de forma manual, muitas vezes utilizando planilhas ou documentos físicos. Esses métodos são suscetíveis a erros humanos, perda de informações e falta de acessibilidade, o que pode resultar em atrasos nas manutenções, falhas operacionais e aumento dos custos.

A solução proposta é um sistema de gerenciamento de veículos e manutenções desenvolvido em Python, com uma interface de linha de comando simples e intuitiva. Este sistema permite aos usuários adicionar e remover veículos, registrar manutenções específicas para cada veículo, e consultar facilmente o histórico de manutenções. A proposta é fornecer uma ferramenta acessível, que não exige infraestrutura complexa, e que possa ser utilizada por qualquer pessoa, independentemente de seu nível de conhecimento técnico.

Este sistema aborda as principais necessidades de gestão de frotas de forma eficiente, garantindo que os veículos estejam sempre operacionais e seguros, e que as manutenções sejam realizadas em tempo hábil, evitando custos elevados com reparos emergenciais e garantindo a continuidade das operações

**2. Objetivo e Escopo do Projeto**

**Objetivo:**  
O principal objetivo deste projeto é desenvolver um sistema simples e eficaz para gerenciar veículos e suas manutenções, permitindo o controle eficiente de uma pequena frota. A solução visa atender as necessidades básicas de gerenciamento, facilitando o registro, a organização e a consulta de informações sobre os veículos e suas respectivas manutenções de forma acessível e prática.

**Escopo:**  
O escopo do projeto abrange as seguintes funcionalidades principais:

* **Adicionar Veículos:** Permitir que o usuário cadastre novos veículos no sistema, assegurando que cada veículo possua um identificador único e evitando duplicações.
* **Remover Veículos:** Oferecer a funcionalidade de remover veículos cadastrados, caso não sejam mais necessários ou estejam fora de uso.
* **Listar Veículos:** Fornecer uma listagem clara e organizada de todos os veículos cadastrados no sistema, permitindo a visualização rápida das informações disponíveis.
* **Registrar Manutenções:** Permitir o registro de manutenções específicas para cada veículo, incluindo descrições e detalhes relevantes para o controle e histórico das intervenções realizadas.
* **Listar Manutenções:** Exibir todas as manutenções registradas para os veículos, facilitando o acompanhamento das ações realizadas e o planejamento de futuras intervenções.

**Limitações do Projeto:**  
Este sistema foca em operações básicas de gerenciamento e não oferece suporte a funcionalidades complexas, como:

* **Relatórios Avançados:** O projeto não inclui a geração de relatórios detalhados ou customizados sobre a frota ou suas manutenções.
* **Persistência de Dados:** Atualmente, o sistema não integra uma base de dados para a persistência de informações; todos os dados são mantidos em memória durante a execução.
* **Integração com Outros Sistemas:** Não há suporte para integração com sistemas externos ou APIs, limitando o uso a operações autônomas.

Com essas funcionalidades e limitações definidas, o projeto busca proporcionar uma solução direta e funcional para o gerenciamento de frotas de pequeno porte, focando na eficiência e simplicidade de uso para o usuário final.

3.Funcionalidades da Solução

1. **Camada Model:**
   * Define as classes de domínio que representam os veículos e as manutenções, seguindo padrões de design apropriados. Essas classes contêm atributos como identificação do veículo, detalhes da manutenção, datas, e outros dados relevantes.
2. **Métodos de Lógica e Regras de Negócio:**
   * Implementa métodos que adicionam lógica e regras de negócio, como adicionar veículos à frota, remover veículos, registrar manutenções específicas para um veículo, e listar todos os veículos ou manutenções cadastradas.
3. **Classe de Teste:**
   * Inclui uma classe de teste para instanciar objetos e testar os métodos principais implementados, garantindo que as funcionalidades do sistema estejam corretas e funcionando conforme o esperado.
4. **Classe de Conexão:**
   * Fornece uma classe para gerenciar a conexão com o banco de dados, incluindo a configuração do usuário e senha diretamente no código, permitindo a persistência de dados e a interação com um banco de dados relacional.
5. **Camada DAO:**
   * Implementa uma classe DAO (Data Access Object) que facilita o acesso aos dados do banco, com métodos básicos como insert para adicionar novos registros ao banco de dados, entre outras operações necessárias para o funcionamento do sistema.

Essas funcionalidades tornam o sistema útil para gerenciar de forma simples e eficaz uma pequena frota de veículos e suas manutenções, oferecendo uma interface programática para manipulação de dados relacionados a veículos e suas manutenções.

Carta, Linha do tempo

Descrição gerada automaticamente

**DESCRIÇÃO**

Relações entre as Classes

Veiculo e Manutencao:

Manutencao tem uma relação de associação com Veiculo, onde um veículo pode ter múltiplas manutenções, mas uma manutenção está ligada a um único veículo.

VeiculoService e VeiculoDAO:

VeiculoService utiliza VeiculoDAO para realizar operações de CRUD (Create, Read, Update, Delete) sobre os veículos.

ManutencaoService e ManutencaoDAO:

ManutencaoService utiliza ManutencaoDAO para realizar operações de CRUD sobre as manutenções.

Instruções para Criar o Diagrama

Ferramenta: Utilize ferramentas como Lucidchart, Draw.io ou Visio.

Classes: Crie caixas representando cada uma das classes mencionadas.

Propriedades e Métodos: Liste as propriedades e métodos dentro das caixas de cada classe.

Relações: Utilize linhas para conectar as classes de acordo com as descrições das relações:

Setas de associação para indicar dependências (ex.: Veiculo -> Manutencao).

Setas de utilização entre serviços e DAOs para indicar o uso de métodos DAO pelos serviços.

9.Tabela dos endpoints

|  |  |  |  |
| --- | --- | --- | --- |
| EndPoint | Verbo HTTP | Descrição | Codigo Status |
| /veiculos | POST | Cadastrar um novo veículo | 201 Created, 400 Bad Request |
| /veiculos | GET | Listar todos os veículos | 200 OK, 500 Internal server error, 400 Bad Request |
| /veiculos /{placa} | GET | Visualizar detalhes de um veículo | 200 OK, 404 Not Found, 400 Bad Request |
| /veiculos /{placa} | GET | Editar informações de um veículo | 200 OK, 400 Bad Request, 500 Internal server error, 400 Bad Request |
| /veiculos /{placa} | DELETE | Excluir um veículo | 200 OK, 204 No Content, 500 ternal server error, 400 Bad Request |
| /usuario | POST | Cadastrar um novo usuário | 201 Created, 400 Bad Request |

**Prototipos**

Tela login

Interface gráfica do usuário

Descrição gerada automaticamente

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário

Descrição gerada automaticamente com confiança média

Diagrama

Descrição gerada automaticamente

