**PROJETO**: SISTEMA DE CONTROLE DE SERVIÇOS - OFICINA MECÂNICA

**Nome do Projeto:** MotorMax

**1. Objetivo do Projeto**

Desenvolver um sistema em Python para gerenciar ordem de serviços de uma oficina mecânica, incluindo cadastro de clientes, veículos, serviços, peças, preço,mecânicos e histórico de atendimento. (é uma ideia, porém vcs podem mudarem se necessários)

**2. Contexto**

Os clientes levam seus veículos à oficina para realização de consertos. Cada veículo é atribuído a um mecânico, responsável por avaliar e executar os serviços necessários. Para registrar os procedimentos realizados e o histórico do atendimento, é criada uma Ordem de Serviço (OS). O sistema gerencia o controle do status e os dados relacionados à OS.

**3. Funcionalidades Principais:**

- Cadastro de Clientes (Nome, email, contato)

- Cadastro de Veículos (Modelo/marca, placa)

- Cadastro de Serviços (troca de óleo, troca de pneus, revisão e manutenção preventivo, reparo de freios, serviços de climatização, sistema de direção etc...(verificar os **necessários**)

- Peças: (nome, tipo, descrição)

- Ordem de Serviço (OS) (status)

- Mecânico: (nome, )

- Histórico de atendimento: (atendimento, descrição)

**Obs: verificar se tem outros atributos necessários para acrescentar.**

**4. Estrutura do Banco de Dados:**

O banco de dados é composto por tabelas que representam as principais entidades do sistema:

Cliente: armazena informações dos clientes.

Veículo: vincula os veículos aos clientes.

Ordem de Serviço (OS) : registra as ordens de serviço, status.

Serviço: tabela de referência para serviços disponíveis.

Peça: tabela de referência para peças utilizadas.

Mecânico: registre os mecânicos.

**4.1 Projeto lógico (falta construir)**

**4.2 Projeto estrutural (falta construir)**

**5.Ferramenta Utilizada**

Banco de dados: MySQL

Linguagem de programação: Python

Designer: Qt designer

IDE: Visual Studio Code

Diagrama: Draw.io

**6. Possibilidades de Expansão (tipo ideias para desenvolvimento futuros, verificar com o professor se é necessário)**

**7. Considerações Finais**

Este projeto é ideal para oficinas que desejam informatizar o controle de gerenciamento de ordens de serviços. Ele está sendo projetado para registrar e organizar informações dos clientes, veículos, serviços, peças e mecânicos.

**Requisitos:**

Cliente:

nome do cliente, Telefone, Email

Carro:

Placa, veículo, cor, o cliente relacionado ao veículo

Serviços:

tipo, valor do serviço, quantidade de serviços, valor total dos serviços

Peças:

Quantidade, descrição, valor unitário, valor total

**1. Levantamento de Requisitos (primeira etapa)**

Desenvolver um sistema em Python para gerenciar ordem de serviços de uma oficina mecânica, incluindo cadastro de clientes, veículos, serviços, peças, preço,mecânicos e histórico de atendimento. (é uma ideia, porém vcs podem mudarem se necessários)

Os clientes levam seus veículos à oficina para realização de consertos. Cada veículo é atribuído a um mecânico, responsável por avaliar e executar os serviços necessários. Para registrar os procedimentos realizados o orçamentista deverá realizar a avaliação do veículo, é criada uma Ordem de Serviço (OS). O sistema gerencia o controle do status e os dados relacionados à OS.

* **Identifique os Stakeholders:** Cliente, proprietário, funcionário(mecânico e orçamentista)
  + **Requisitos Funcionais:**
  + Cadastro de Clientes (Nome, CPF, email, Telefone)
  + - Cadastro de Veículos (Tipo veículo, Marca, ano e placa)
  + - Cadastro de Serviços (valor, troca de óleo, troca de pneus, revisão e manutenção preventiva, reparo de freios, serviços de climatização e status)
  + Peças: (nome, tipo, descrição)
  + - Ordem de Serviço (OS) (status)
  + - Mecânico: (nome, )
  + - Histórico de atendimento: (atendimento, descrição)
* **Requisitos Não-Funcionais:** (juliene)Como o sistema deve se comportar?
  + **Usabilidade:** O sistema deve ser fácil de usar para os funcionários, mesmo os menos experientes.
  + **Segurança:** Acesso restrito a informações sensíveis.
  + **Desempenho:** Resposta rápida ao agendamento e consulta de dados.
  + **Manutenibilidade:** O código deve ser fácil de atualizar e corrigir.
* **Crie um Diagrama de Caso de Uso:(Victor)** Uma ótima ferramenta visual para mostrar a interação entre os usuários e o sistema. (criar o diagrama de caso de uso)

**2. Modelagem do Banco de Dados (Segunda etapa)**

Com os requisitos em mãos, o próximo passo é estruturar os dados.

* **Modelo Entidade-Relacionamento (MER):**
  + **Identifique as Entidades:** Quais são os "objetos" principais do sistema? (Ex: Cliente, Pet, Serviço, Produto, Agendamento).
  + **Identifique os Atributos:** Quais são as características de cada entidade? (Ex: para a entidade Cliente, os atributos podem ser nome, telefone, endereco).
  + **Identifique os Relacionamentos:** Como as entidades se conectam? (Ex: um Cliente pode ter vários Pets).
  + **Desenhe o MER:** Use uma ferramenta como o Draw.io ou mesmo uma lousa branca para visualizar as entidades, seus atributos e os relacionamentos entre elas.
* **Modelo de Entidade e Relacionamento (DER):**
  + O **DER** é a representação visual do MER, geralmente desenhado com notação específica (como a de "pé de galinha" ou UML). Ele detalha as cardinalidades dos relacionamentos (um para um, um para muitos, muitos para muitos). Você pode usar o MySQL Workbench ou o Lucidchart para criar esse diagrama.
* **Regras do Negócio**
  + O Mecânico só pode fazer um trabalho por vez

**3. Modelo Lógico e Físico do Banco de Dados**

A partir do DER, você traduz a estrutura para o ambiente do banco de dados.

* **Modelo Lógico:**
  + **Normalização:** Aplique as formas normais (1FN, 2FN, 3FN) para evitar redundância e inconsistência de dados.
  + **Crie as Tabelas:** Defina as tabelas com base nas entidades e seus relacionamentos. Por exemplo, um relacionamento de muitos para muitos entre Cliente e Servico pode se tornar uma nova tabela Agendamento.
  + **Defina Chaves:** Identifique as chaves primárias (PRIMARY KEY) e as chaves estrangeiras (FOREIGN KEY) que farão a ligação entre as tabelas.
* **Modelo Físico:**
  + **Escolha o SGBD:** Qual sistema de gerenciamento de banco de dados você vai usar? (Ex: MySQL, PostgreSQL, SQLite).
  + **Crie o Esquema Físico:** Defina os tipos de dados para cada atributo (ex: VARCHAR(50), INT, DATE).
  + **Gere o Script SQL:** Escreva os comandos CREATE TABLE e ALTER TABLE para criar o banco de dados e suas tabelas.

**4. Documento de Projeto de Software**

Esse documento é a "planta" do seu sistema. Ele deve ser um guia claro para você e para qualquer pessoa que venha a trabalhar no projeto.

* **Introdução:** Descreva o propósito do sistema e o problema que ele resolve.
* **Escopo do Projeto:** O que o sistema vai e, mais importante, **não** vai fazer.
* **Requisitos:** Inclua os requisitos funcionais e não-funcionais detalhados na primeira etapa.
* **Arquitetura do Sistema:** Descreva a arquitetura que você pretende usar (ex: desktop, cliente-servidor). Inclua os diagramas de classe e de sequência, se necessário.
* **Modelagem de Dados:** Anexe os diagramas **MER** e **DER** e a descrição das tabelas do modelo lógico.
* **Plano de Implementação:** Defina as tecnologias que serão usadas (linguagem de programação, framework, SGBD).
* **Plano de Testes:** Como você vai garantir que o sistema funciona corretamente? (Testes de unidade, de integração, de aceitação).

## **5. APROVAÇÃO**

Este documento de projeto de software será revisado e aprovado pelos seguintes stakeholders:

|  |  |  |  |
| --- | --- | --- | --- |
| **Nome Completo** | **Função** | **Assinatura** | **Data** |
| Ruy Ribeiro | Proprietário do Petshop Cãobalhota |  |  |
| Victor Oliveira | Professor de TI / Analista de Negócios |  |  |
| PatricK Oliveira | Líder de Desenvolvimento do Projeto |  |  |
| Juliene de Sousa | Representante da Equipe de Testes |  |  |
| Diogo Miranda  Antônio Silva | Equipe de front-end |  |  |

**Autores:**

Diogo Felipe do Rosario Miranda

Juliene de Sousa Ferreira

Antonio Silva

Rui

Victor Oliveira

Anderson Patric