GRUPO :
Brena dos Santos Freitas - RA : 2465710
Maria Gabriella Victor - RA : 2143364
Micael Ribeiro Rocha - RA : 2454424

1. Objetivo

O objetivo deste sistema é auxiliar na gestão das atividades de uma oficina mecânica, tem como funcionalidades o gerenciamento de clientes, veículos, produtos e serviços e visualização de relatórios .

2. Funcionalidades

User: postgres

- Inserir dados nas tabelas - cliente, funcionário, produtos, serviços, veículos
- Consultar dados nas tabelas e views - cliente, funcionário, produtos, serviços, view cliente e produto , view cliente e veículo e backup (clientes excluídos).
- Apagar dados da tabela - - cliente, funcionário, produtos, serviços

User : usuário

- Consultar dados na view cliente e veículo
- Inserir dados na view - cliente e veículo

3. Ferramentas

PostgreSQL: utilizado para a criação e execução do Código SQL, incluindo a inserção de tabelas, criação de trigger/função, views e para criação do próprio Banco de Dados da Oficina.

NetBeans: utilizado para a criação da aplicação Desktop simples usando a linguagem de programação Java. Essa aplicação está conectada com um banco de dados “ Mechanical Register” que está hospedado na Nuvem do servidor Rail, possibilitando por sua vez as operações de INSERT, DELETE e SELECT.

BrModelo: utilizado para a criação do Diagrama Entidade Relacionamento(DER. No Diagrama, é possível ter uma visualização geral melhor do que é preciso fazer.

GitHub: utilizado para o versionamento dos códigos do projeto, dessa forma, os integrantes do projeto conseguem compartilhar as mudanças do código mais facilmente.

Figma: utilizado para a criação inicial do Design da aplicação.

Rail: utilizado para hospedar o banco de dados na nuvem.

4. Script SQL

- Geração e execução do código SQL para criação das tabelas e inserção de pelo menos 5 linhas de dados por tabela.

```
CREATE TABLE servico(
        id_servico SERIAL,
        tipo varchar(100),
        valor numeric(9, 2),
        primary key(id_servico)
);

CREATE TABLE produto(
        id_produto SERIAL,
        nome_produto varchar(100),
        valor numeric(9, 2),
   quantidade int,
        primary key(id_produto)
)

CREATE TABLE funcionario(
        id_funcionario SERIAL,
        id_servico int,
        idade INTEGER,
        nome VARCHAR(100),
        cpf INTEGER,
        rg INTEGER,
        funcao CHAR(100),
        primary key(id_funcionario),
        FOREIGN KEY (id_servico) REFERENCES servico (id_servico)
);

CREATE TYPE tipo_motor AS ENUM('E', 'C', 'H');
CREATE TABLE veiculo (
        id_veiculo SERIAL,
        ano             INTEGER,
        placa           VARCHAR(100),
        marca           VARCHAR(100),
        modelo          VARCHAR(100),
        cor     VARCHAR(100),
        motor           tipo_motor,
        id_cliente INTEGER,
        primary key (id_veiculo),
        FOREIGN KEY (id_cliente) REFERENCES cliente (id_cliente)
);
```

```
CREATE TABLE cliente(
	id_cliente SERIAL,
	nome_cliente varchar(100),
	idade INTEGER,
	id_servico INTEGER,
	id_produto integer,
	cpf INTEGER,
	rg INTEGER,
	telefone varchar(100),
	data_de_registro TIMESTAMP,
	primary key(id_cliente),
	FOREIGN KEY (id_servico) REFERENCES servico (id_servico),
	FOREIGN KEY (id_produto) REFERENCES produto (id_produto)
);

insert into servico (tipo, valor)
values ('Nenhum', 0);

insert into servico (tipo, valor)
values ('Troca de óleo', 200.00);

insert into servico (tipo, valor)
values ('Revisão detalhada do sistema de arrefecimento', 120.00);

insert into servico (tipo, valor)
values ('Troca do filtro do ar', 230.00);

insert into servico (tipo, valor)
values ('Manutenção de embreagem', 300.00);

insert into servico (tipo, valor)
values ('Revisão dos componentes do freio', 300.00);

insert into produto (nome_produto,valor,quantidade)
values ('nenhum', 0, 0);

insert into produto (nome_produto,valor,quantidade)
values ('Óleo de carro semissintético', 30.00, 10);

insert into produto (nome_produto,valor,quantidade)
values ('Roda  Aro 15 4 x100', 590.50, 10);

insert into produto (nome_produto,valor,quantidade)
values ('Motor', 2439.95, 6);

insert into produto (nome_produto,valor,quantidade)
values ('Disco de freio dianteiro', 381, 12);
```

```
insert into produto (nome_produto,valor,quantidade)
values ('Lubrificante 0W20 honda sintético', 90.25, 15);

insert into funcionario (id_servico, idade, nome, cpf, rg, funcao)
values ('3', 42, 'Jorse Santos Pratos', 234242423, 34555, 'Mecanico');

insert into funcionario (id_servico, idade, nome, cpf, rg, funcao)
values ('3', 26, 'Jessica Passos Braga', 234242423, 34555, 'Mecanico');
```

- Criação de trigger/função de backup de dados excluídos de uma tabela, registrando os dados da linha apagada (sem restrições de chave), com usuário e data.

```
CREATE TABLE backup_cliente(
   id_backup SERIAL,
   id_cliente SERIAL,
   nome_cliente varchar(100),
   idade INTEGER,
   id_servico INTEGER,
   id_produto integer,
   cpf INTEGER,
   rg INTEGER,
   telefone varchar(100),
   data_exclusa TIMESTAMP,
     usuario varchar(100), -- Novo atributo para informar o usuário que apagou o cliente
   primary key(id_backup)
);

CREATE OR REPLACE FUNCTION backup()
RETURNS TRIGGER
AS
$$
BEGIN
        INSERT INTO backup_cliente(id_cliente, nome_cliente, idade, id_servico, id_produto, cpf, rg, telefone, data_exclusao, usuario)
         VALUES(OLD.id_cliente, OLD.nome_cliente, OLD.idade, OLD.id_servico, OLD.id_produto, OLD.cpf, OLD.rg, OLD.telefone, current_TIMESTAMP , current_user); -- Utiliza a função current_user para obter o nome do usuário atual
   RETURN OLD;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER backup AFTER DELETE ON cliente
FOR EACH ROW EXECUTE PROCEDURE backup();
```

```
DELETE FROM veiculo WHERE id_cliente = 24;
DELETE FROM cliente WHERE id_cliente = 24;
```

- Criação de pelo menos 4 índices (1 índice por tabela no máximo)

```
CREATE INDEX indice_veiculo ON veiculo(id_veiculo);
CREATE INDEX indice_produto ON produto(id_produto);
CREATE INDEX indice_servico ON servico(id_servico);
CREATE INDEX indice_cliente ON cliente(id_cliente);
CREATE INDEX indice_funcionario ON funcionario(id_funcionario);
```

- Criação de usuário e atribuição de privilégio somente de leitura das tabelas para o usuário criado

```
CREATE ROLE usuario WITH
LOGIN
PASSWORD 'usuario';

GRANT SELECT, INSERT ON carro_cliente TO
usuario;
```

- Criação de no mínimo uma visão (view) a partir de duas ou mais tabelas

```
CREATE VIEW carro_cliente as
SELECT nome_cliente,placa, marca, modelo FROM cliente cli, veiculo veic
WHERE cli.id_cliente = veic.id_cliente;
```

- Criação de trigger que permita inserção de dados na view, e atribuição de privilégio ao usuário criado anteriormente para ver e inserir usando a view

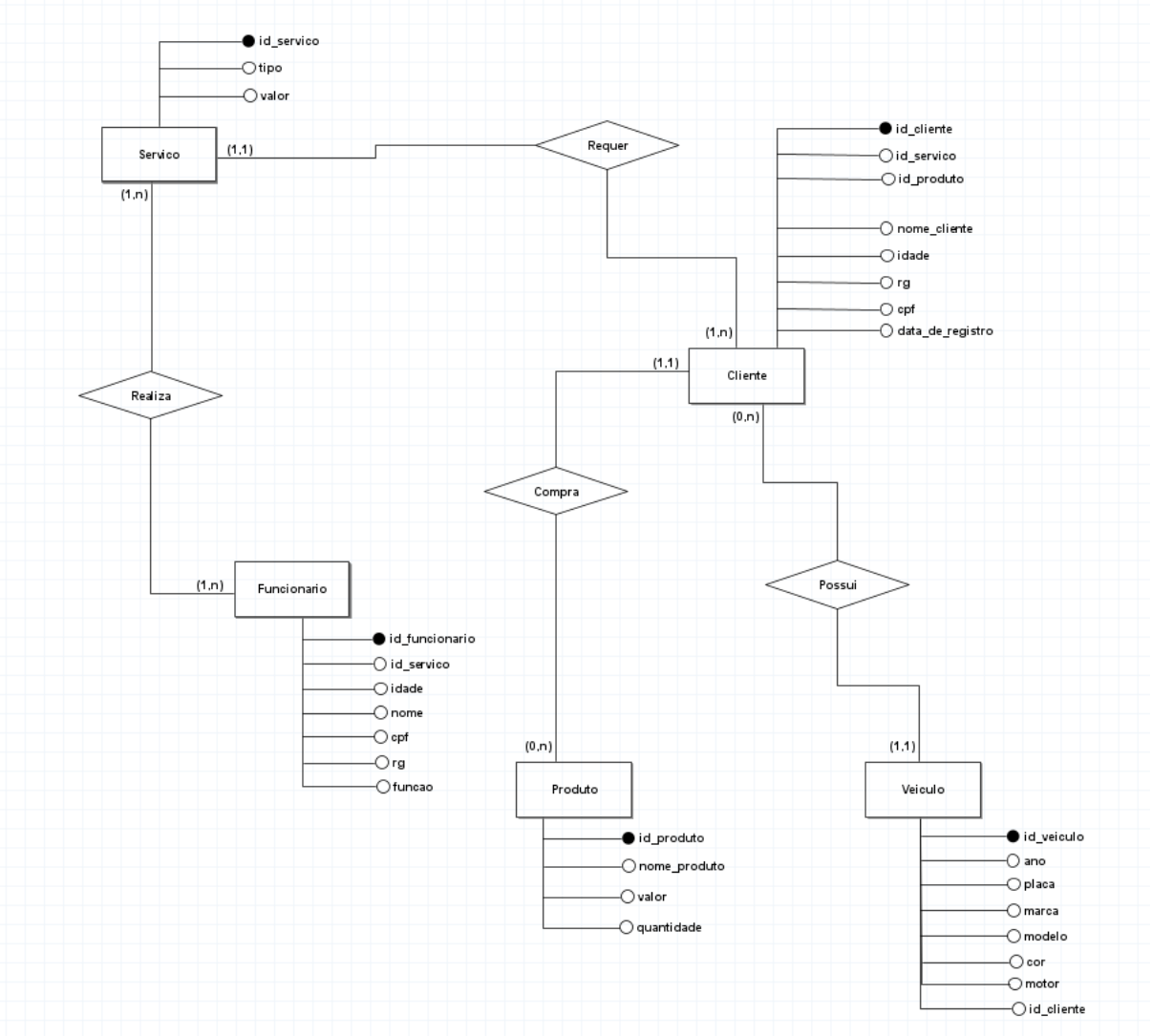
```
CREATE OR REPLACE FUNCTION carro_cliente() RETURNS TRIGGER
AS $$
DECLARE
id_cli integer;
BEGIN
        insert into cliente (nome_cliente, data_de_registro)
        values (NEW.nome_cliente, CURRENT_TIMESTAMP);
        id_cli := (SELECT MAX(id_cliente) FROM cliente);
        insert into veiculo (placa,marca, modelo, id_cliente)
        values (NEW.placa,NEW.marca, NEW.modelo, id_cli);
        RETURN NEW;
END;
$$ LANGUAGE plpgsql SECURITY DEFINER;

CREATE TRIGGER carro_cliente INSTEAD OF
INSERT ON carro_cliente FOR EACH ROW EXECUTE
PROCEDURE carro_cliente();
```

- permita exibição de pelo menos um relatório utilizando uma consulta SQL que envolva no mínimo uma operação JOIN

select id_cliente, nome_cliente from cliente cli NATURAL JOIN servico ser where ser.id_servico != 1; -- Clientes que compraram algum produto

- permita exibição de pelo menos um relatório utilizando no mínimo uma visão previamente criada em uma consulta
SELECT nome_cliente,placa,marca,modelo FROM carro_cliente

5. Diagrama Entidade Relacionamento - DER



6. Conexão do banco de dados na Aplicação Desktop

I. Para conexão com o banco de dados - import java.sql.Connection
II. Para receber retorno de uma consulta - import java. sql. ResultSet