


# Stand Auto Manager

Bases de Dados  
P3G2

Rui Machado 65081   Marco Almeida 103440

1 de junho de 2023

# Introdução




Baseia-se numa pequena aplicação para gerir uma cadeia de Stands Auto.

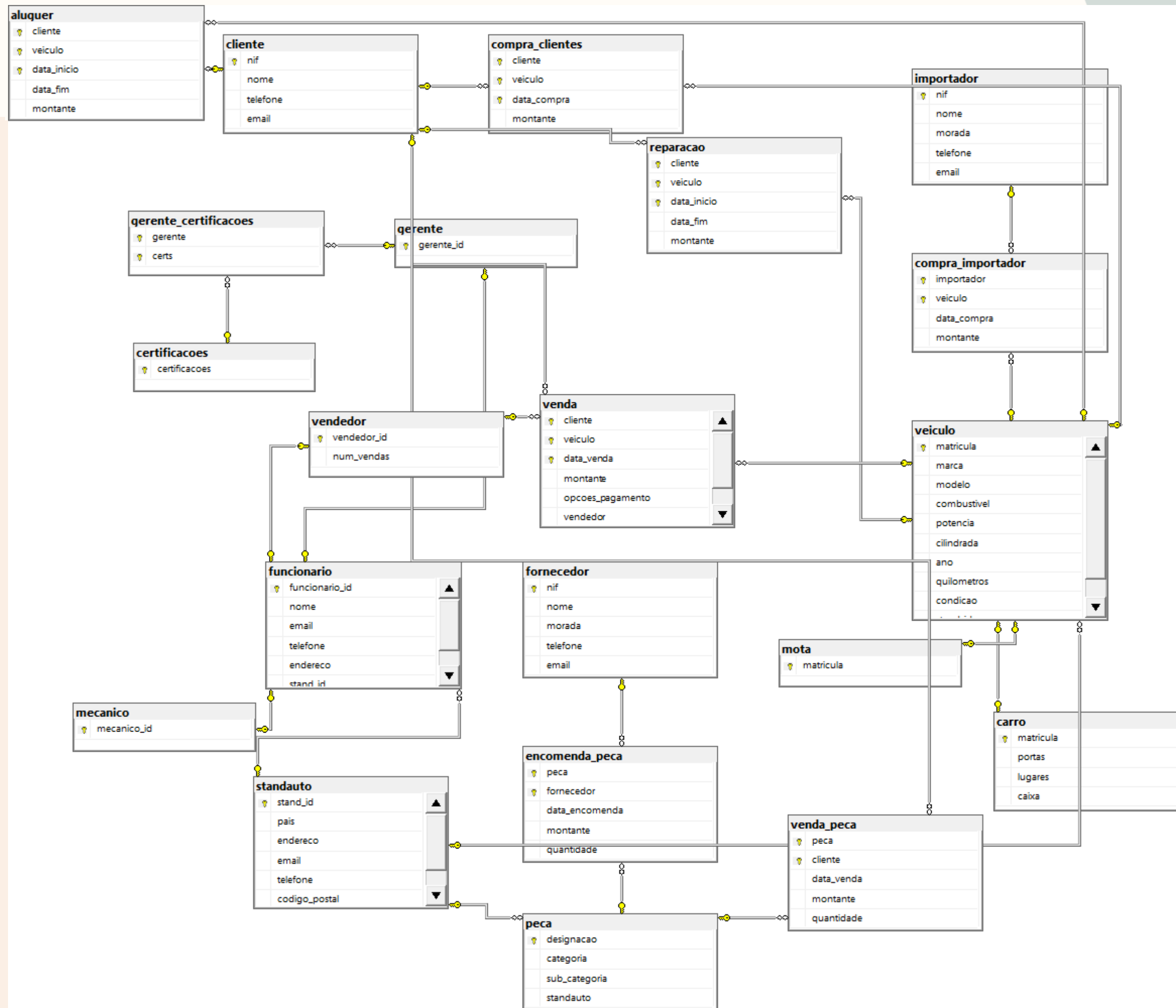
Gere cada Stand, os seus funcionários, veículos e peças.

Mantém um registo de todos os clientes.

Mantém um registo de todas as operações de negócio do Stand:

- 
- Vendas e Compras
  - Alugueres
  - Reparações
  - Importações de Veículos e Peças

# ER



# SQL Scripts

1

DDL

2

DML

3

User Defined Functions

4

Stored Procedures

5

Views

6

Triggers

7

Indexes

# UDFs

- Obter todos os veiculos disponiveis especificando o stand

```
CREATE FUNCTION GetAvailableVehiclesInStand(@standauto_id INT) RETURNS TABLE --
AS --
RETURN (
    SELECT veiculo.matricula,
           veiculo.marca,
           veiculo.modelo,
           veiculo.combustivel,
           veiculo.potencia,
           veiculo.cilindrada,
           veiculo.ano,
           veiculo.quilometros,
           veiculo.condicao
    FROM veiculo
    FULL OUTER JOIN aluguer ON veiculo.matricula = aluguer.veiculo
    WHERE (
        (
            data_inicio IS NULL
            AND data_fim IS NULL
        )
        OR (
            data_inicio IS NOT NULL
            AND data_fim IS NOT NULL
        )
    )
    AND veiculo.stand_id = @standauto_id
);
```

# Stored Procedures

- Registrar a compra de um veículo a um cliente

Assume que o cliente já está previamente registrado na base de dados, mas o seu veículo não.

```
CREATE PROCEDURE BuyClientVeiculo @matricula VARCHAR(20), @marca VARCHAR(30), @modelo VARCHAR(255), @combustivel
VARCHAR(30), @potencia INTEGER, @cilindrada INTEGER, @ano INTEGER, @quilometros INTEGER, @condicao VARCHAR(100),
@stand_id INTEGER, @portas INTEGER, @lugares INTEGER, @caixa VARCHAR(30), @client_nif INTEGER, @data_compra DATE,
@montante MONEY, @tipo VARCHAR(10)
AS --
BEGIN --
UPDATE Cliente
SET telefone = @client_phone
WHERE nif = @client_nif IF NOT EXISTS (
    SELECT *
    FROM cliente
    WHERE nif = @client_nif
) BEGIN RAISERROR('Client does not exist', 16, 1);
RETURN;
END IF EXISTS (
    SELECT *
    FROM veiculo
    WHERE matricula = @matricula
) BEGIN RAISERROR('Vehicle already exists', 16, 1);
RETURN;
END IF @tipo = 'Carro' BEGIN
INSERT INTO veiculo (matricula, marca, modelo, combustivel, potencia, cilindrada, ano, quilometros, condicao, stand_id)
VALUES ( @matricula, @marca, @modelo, @combustivel, @potencia, @cilindrada, @ano, @quilometros, @condicao, @stand_id);
INSERT INTO carro (matricula, portas, lugares, caixa)
VALUES (@matricula, @portas, @lugares, @caixa);
INSERT INTO compra_clientes
VALUES (@client_nif, @matricula, @data_compra, @montante);
END IF @tipo = 'Mota' BEGIN
INSERT INTO veiculo (matricula, marca, modelo, combustivel, potencia, cilindrada, ano, quilometros, condicao, stand_id)
VALUES (@matricula, @marca, @modelo, @combustivel, @potencia, @cilindrada, @ano, @quilometros, @condicao, @stand_id);
INSERT INTO mota (matricula)
VALUES (@matricula);
INSERT INTO compra_clientes
VALUES (@client_nif, @matricula, @data_compra, @montante);
```

# Views

- Obter os melhores vendedores em todos os stands

```
CREATE VIEW GetEmployeeRoles --
AS
SELECT f.*,
       CASE
         WHEN v.vendedor_id IS NOT NULL THEN 'Vendedor'
         WHEN m.mecanico_id IS NOT NULL THEN 'Mecanico'
         WHEN g.gerente_id IS NOT NULL THEN 'Gerente'
         ELSE 'Unknown Role'
       END AS role
FROM funcionario f
LEFT JOIN vendedor v ON f.funcionario_id = v.vendedor_id
LEFT JOIN mecanico m ON f.funcionario_id = m.mecanico_id
LEFT JOIN gerente g ON f.funcionario_id = g.gerente_id;
```

```
CREATE VIEW GetBestSellers --
AS
SELECT vendedor.vendedor_id,
       SUM(venda.montante) AS total_montante
FROM vendedor
       JOIN venda ON vendedor.vendedor_id = venda.vendedor_id
GROUP BY vendedor.vendedor_id
HAVING SUM(venda.montante) > (
       SELECT AVG(montante)
       FROM venda
);
```

# Triggers

- Quando um funcionário é despedido, os seus tuplos relevantes são eliminados em todas as tabelas necessárias.

```
GO --
CREATE TRIGGER del_mecanico ON mecanico
AFTER DELETE --
AS --
BEGIN --
DELETE FROM funcionario
WHERE funcionario.funcionario_id IN (
    SELECT mecanico_id
    FROM deleted
)
```

## Indexes

```
CREATE INDEX aluga_datas ON aluguer (data_fim, data_inicio);
```

- Um novo cliente quer principalmente saber que veículos estão, de momento, disponíveis.





DEMO