



NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

BASE DE DADOS

LEI

Tema – Stand de Carros

Autores:

Joana Simões Neves (65441)

Rodrigo Rafael C. S. Santos (63263)

Rui Alberto C. Xavier (65815)

js.neves@campus.fct.unl.pt

rrc.santos@campus.fct.unl.pt

ra.xavier@campus.fct.unl.pt

Grupo 44

2023/2024 – 2nd Semestre, P4

Contents

1	Introdução	3
2	Exposição do Tema e Diagrama ER	3
2.1	Objetivos	4
3	Modelo Relacional	6
4	Dependências Funcionais e Normalização	7
5	Consultas Interessantes	10
5.1	Qual é a marca de Carros mais vendida?	10
5.2	Qual é o nome do funcionário com mais vendas da marca Mercedes?	10
5.3	Qual é o tipo de motor mais comum nos carros vendidos?	11
5.4	Qual é o carro que possui o maior numero de TestDrives?	11
5.5	Quem é o cliente que efetuou mais compras?	11
6	Triggers e Vistas	12
6.1	Triggers	12
6.1.1	idMarca	12
6.1.2	idC	12
6.1.3	idV	13
6.1.4	idT	13
6.1.5	incrementDisponibilidade	13
6.1.6	insertVenda	14
6.1.7	insertTestDrive	15
6.1.8	disponibilidade	16
6.1.9	removeCarro	16
6.1.10	removePessoa	17
6.1.11	removeMotor	17
6.1.12	removeTipoMotor	18
6.1.13	removeModelo	18
6.1.14	removeMarca	19
6.2	Vistas	19
6.2.1	percentagemVendasFuncionario	19
6.2.2	FuncionariosInfo	19
6.2.3	ClientesInfo	20
6.2.4	CarrosInfo	20
6.2.5	VendasInfo	20
6.2.6	TestDrivesInfo	20
6.2.7	CombustoesInfo	21
6.2.8	EletricosInfo	21
6.2.9	HibridosInfo	21
6.2.10	ModelosInfo	21

6.3	Triggers nas Vistas	22
6.3.1	insertFuncionariosInfo	22
6.3.2	deleteFuncionarioInfo	22
6.3.3	updateFuncionarioInfo	23
6.3.4	insertClienteInfo	23
6.3.5	deleteClienteInfo	23
6.3.6	updateClienteInfo	24
6.3.7	insertVendasInfo	24
6.3.8	deleteVendasInfo	25
6.3.9	updateVendasInfo	25
6.3.10	insertCarroInfo	25
6.3.11	removeCarroInfo	26
6.3.12	updateCarroInfo	26
6.3.13	insertModelosInfo	26
6.3.14	deleteModelosInfo	27
6.3.15	updateModelosInfo	27
7	Interface	28
7.1	Carros	29
7.2	Vendas	31
7.3	Clientes	32
7.4	Funcionários	33
7.5	Test-Drives Detalhes	35
7.6	Estatísticas Interessantes	36
7.7	Motores Detalhes	37
7.8	Marcas	39
7.9	Modelos	40
7.10	TipoMotor	41
7.11	Test-Drives	41
7.12	Motores	42
8	Limitações da Base de Dados	44
	Appendices	45
A	Mudanças Realizadas	45
B	Uso de IA	46

1 Introdução

O nosso projeto almeja replicar uma base de dados de um stand de carros. Para tal começamos pela criação de um [diagrama ER](#), que nos permitiu visualizar as entidades, os atributos e as relações entre as mesmas. Este diagrama é essencial para a criação do [modelo relacional](#), que será apresentado na secção 2. Após alguma reflexão sobre a forma normal procedemos a alguns [ajustes](#) no diagrama ER, de forma a garantir que o modelo relacional se encontra na **BCNF** e **4NF**. Por fim, apresentamos algumas [consultas interessantes](#) que podem ser feitas à base de dados, e os [triggers, funções e vistas](#) que utilizamos para garantir a integridade dos dados.

2 Exposição do Tema e Diagrama ER

O objetivo geral do projeto é criar uma base de dados para um stand de carros. O stand tem várias marcas de carros, cada uma com vários modelos. Cada modelo tem um tipo de motor, que pode ser elétrico ou a combustão. Cada carro tem um motor, e pode ser novo ou usado. Os carros podem ser vendidos a clientes, que podem fazer test drives. Os funcionários do stand podem ser vendedores ou gerentes. Mais detalhadamente:

- Cada **marca** tem um identificador unico no sistema, um nome (que é único), uma gama e uma disponibilidade.
- Cada **modelo** tem um nome, um tipo de modelo e pertence a uma marca. Não há possibilidade de existirem dois modelos com o mesmo nome.
- Cada **motor** tem um nome que o identifica no sistema, uma autonomia, uma potência e consumos. O motor pode ser:
 - Um motor a **combustão** que tem um tipo de combustível e um tamanho de depósito.
 - Um motor **elétrico** que tem uma capacidade de bateria.
 - Um motor "Hibrido" que é ao mesmo tempo a **combustão** e **elétrico**.
- Assumimos que um **Modelo** não pode ser equipado com um **Motor** de uma **Marca** que não a sua.
- Cada **carro** contem um identificador no sistema, um modelo, um motor, um ano e uma cor. Um carro pode ser **usado** e nesse caso tem um número de quilómetros.
- Cada **pessoa** tem um NIF, um nome, um apelido, uma morada, um email e um número de telefone. Um pessoa no sistema pode ser um funcionário ou um cliente, mas também pode ser ambos:
 - Um **funcionário** que tem um cargo.
 - Um **cliente**.

- Cada **venda** tem a identificação do carro vendido, o funcionário responsável, o comprador, uma data e um montante. Como existe a possibilidade de um funcionario ser um cliente, o sistema tem de garantir que um funcionário não pode vender um carro a si próprio. Além disso, um carro só pode ser vendido uma vez.
- Cada **test drive** tem um carro, um funcionário responsável, um cliente, uma data, uma hora, um ID e uma classificação (de 0 a 5). Um carro pode ter vários test drives, mas um test drive só pode ser feito por um cliente de cada vez (ou seja não podem existir teste drives com o mesmo carro a mesma hora na mesma data).

2.1 Objetivos

Os nossos objetivos com este projeto são os seguintes:

- Criar uma base de dados funcional para um stand de carros que gere eficientemente informações sobre marcas, modelos, motores, carros, pessoas (funcionários e clientes), vendas e test drives.
- Implementar triggers, funções e vistas para manter a integridade referencial e consistência dos dados dentro do sistema.
- Prover consultas que possibilitem extrair informações relevantes e úteis da base de dados.
- Assegurar que o modelo relacional esteja adequadamente normalizado, minimizando redundâncias e garantindo a eficiência no armazenamento e na recuperação de dados.

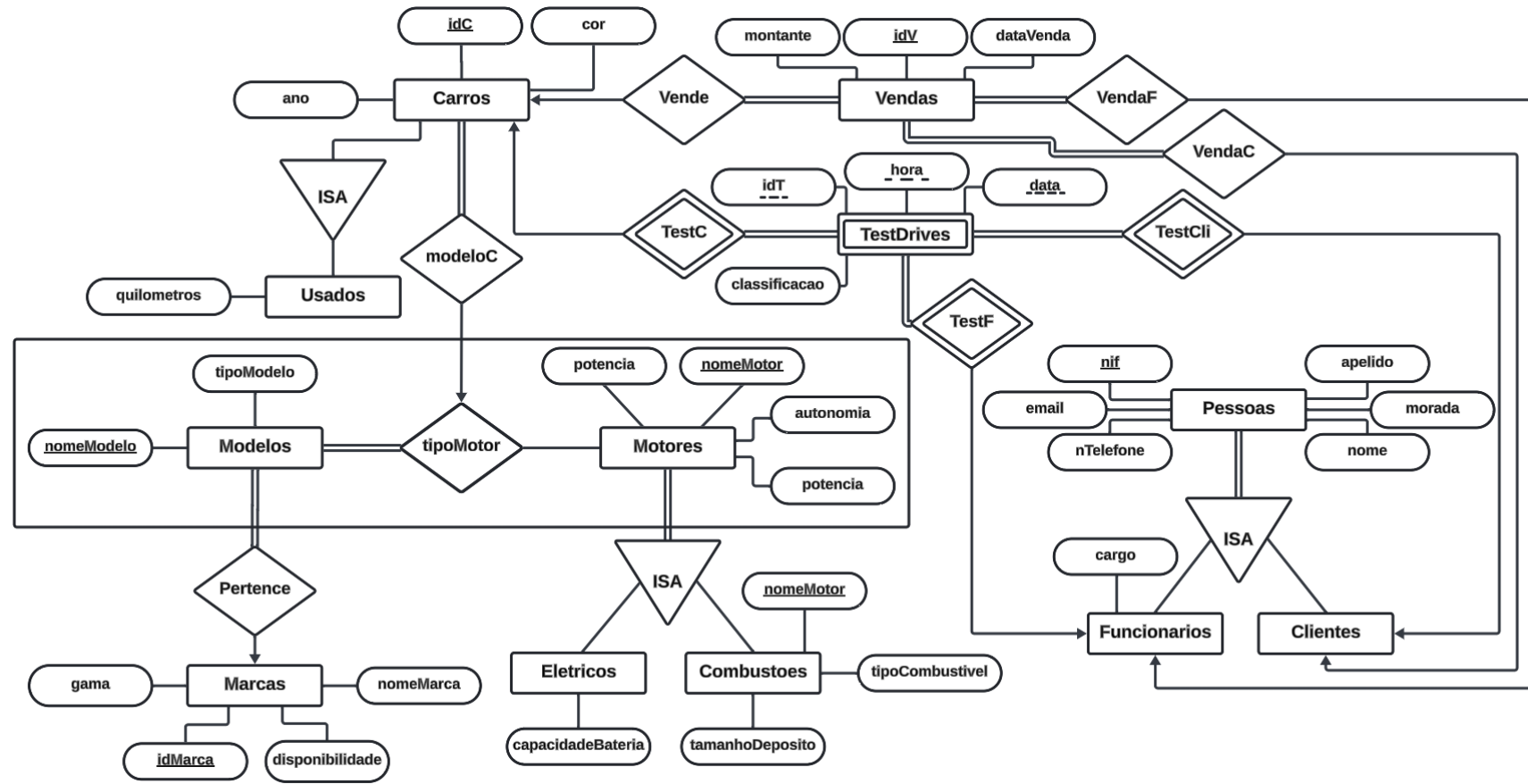


Figure 1: Diagrama ER

3 Modelo Relacional

- **Marcas**(idMarca, nomeMarca, disponibilidade, gama)
- **Modelos**(nomeModelo, idMarca, tipoModelo)
 - idMarca é chave estrangeira de **Marcas**.
- **Motores**(nomeMotor, autonomia, potencia, consumos)
- **Combustoes**(nomeMotor, tipoCombustivel, tamanhoDeposito)
 - nomeMotor é chave estrangeira de **Motores**.
- **Eletricos**(nomeMotor, capacidadeBateria)
 - nomeMotor é chave estrangeira de **Motores**.
- **TipoMotor**(nomeMotor, nomeModelo)
 - nomeMotor é chave estrangeira de **Motores**.
 - nomeModelo é chave estrangeira de **Modelos**.
- **Carros**(idC, nomeMotor, nomeModelo, ano, cor)
 - (nomeModelo, nomeMotor) é chave estrangeira de **TipoMotor**.
- **Usados**(idC, quilometros)
 - idC é chave estrangeira de **Carros**.
- **Pessoas**(nif, nome, apelido, morada, email, nTelefone)
- **Funcionarios**(nif, cargo)
 - nif é chave estrangeira de **Pessoas**.
- **Clientes**(nif)
 - nif é chave estrangeira de **Pessoas**.
- **Vendas**(idV, idC, nifF, nifC, dataVenda, montante)
 - idC é chave estrangeira de **Carros**.
 - nifF é chave estrangeira de **Funcionarios** que referencia o atributo nif.
 - nifC é chave estrangeira de **Clientes** que referencia o atributo nif.
- **TestDrives**(idC, nifF, nifC, data, hora, idT, classificacao)
 - idC é chave estrangeira de **Carros**.
 - nifF é chave estrangeira de **Funcionarios** que referencia o atributo nif.
 - nifC é chave estrangeira de **Clientes** que referencia o atributo nif.

4 Dependências Funcionais e Normalização

Nesta secção vamos analisar as dependências funcionais das nossas relações e verificar se as mesmas se encontram normalizadas. Para tal, vamos seguir os seguintes passos:

1. Identificar as dependências funcionais e chaves candidatas de cada relação.
2. Verificar se as relações se encontram na forma normal de Boyce-Codd (**BCNF**). Para tal, em todas as dependências não triviais temos de ter: $X \rightarrow Y$, onde X é super chave. Caso contrario não está na **BCNF**, e teremos de normalizar a relação.
3. Se existirem dependências multi-valor funcionais na relação (requer que a relação esteja na **BCNF**), vamos verificar se a relação se encontra na quarta forma normal (**4NF**). Para tal, temos de verificar se todas as dependências funcionais multi-valor são triviais. Caso não sejam, a relação não está na **4NF** e temos de proceder à normalização da relação.

- **Marcas**(idMarca, nomeMarca, disponibilidade, gama)

- **Dependências Funcionais:**

- * $\{\text{idMarca}\} \rightarrow \{\text{disponibilidade, gama, nomeMarca}\}.$
 - * $\{\text{nomeMarca}\} \rightarrow \{\text{disponibilidade, gama, idMarca}\}.$

- **Marcas** encontra-se na **BCNF**. Isto porque ambas as dependências funcionais, apesar de não serem triviais, do lado esquerdo apresentam uma *super-chave*.

- **Modelos**(nomeModelo, idMarca, tipoModelo)

- **Dependências Funcionais:**

- * $\{\text{nomeModelo}\} \rightarrow \{\text{idMarca, tipoModelo}\}.$

- **Modelos** encontra-se na **BCNF**. Isto porque a dependência funcional, apesar de não ser trivial, do lado esquerdo apresenta uma *super-chave*.

- **Motores**(nomeMotor, autonomia, potencia, consumos)

- **Dependências Funcionais:**

- * $\{\text{nomeMotor}\} \rightarrow \{\text{autonomia, potenciam, consumos}\}.$

- **Motores** encontra-se na **BCNF**. Isto deve-se ao facto de a dependência funcional, apesar de não ser trivial, do lado esquerdo apresenta uma *super-chave*.

- **Combustoes**(nomeMotor, tipoCombustivel, tamanhoDeposito)

- **Dependências Funcionais:**

- * $\{\text{nomeMotor}\} \rightarrow \{\text{tipoCombustivel, tamanhoDeposito}\}.$

- **Combustoes** encontra-se na **BCNF**. Isto deve-se ao facto de a dependência funcional, apesar de não ser trivial, do lado esquerdo apresenta uma *super-chave*.

- **Eletricos**(nomeMotor, capacidadeBateria)
 - **Dependências Funcionais:**
 - * $\{\text{nomeMotor}\} \rightarrow \{\text{capacidadeBateria}\}$.
 - **Eletricos** encontra-se na **BCNF**. Isto deve-se ao facto de a dependência funcional, apesar de não ser trivial, do lado esquerdo apresenta uma *super-chave*.
- **TipoMotor**(nomeMotor, nomeModelo)
 - **Dependências Funcionais:**
 - * $\{\text{nomeMotor}, \text{nomeModelo}\} \rightarrow \{\text{nomeMotor}, \text{nomeModelo}\}$.
 - **TipoMotor** encontra-se na **BCNF**. Isto deve-se ao facto de só existirem dependências funcionais triviais.
 - Como nomeMotor e nomeModelo são atributos multi-valor, e só existem dependências multi-valor funcionais triviais, a relação está na **4NF**.
 - * Um exemplo de uma dependência funcional multi-valor para **TipoMotor** é: $\text{nomeMotor} \twoheadrightarrow \text{nomeModelo}$; pois a sua união é igual a todos os elementos da relação.
- **Carros**(idC, nomeMotor, nomeModelo, ano, cor)
 - **Dependências Funcionais:**
 - * $\{\text{idC}\} \rightarrow \{\text{nomeMotor}, \text{nomeModelo}, \text{ano}, \text{cor}\}$.
 - **Carros** encontra-se na **BCNF**. Isto deve-se ao facto de a dependência funcional, apesar de não ser trivial, do lado esquerdo apresenta uma *super-chave*.
- **Usados**(idC, quilometros)
 - **Dependências Funcionais:**
 - * $\{\text{idC}\} \rightarrow \{\text{quilometros}\}$.
 - **Usados** encontra-se na **BCNF**. Isto deve-se ao facto de a dependência funcional, apesar de não ser trivial, do lado esquerdo apresenta uma *super-chave*.
- **Pessoas**(nif, nome, apelido, morada, email, nTelefone)
 - **Dependências Funcionais:**
 - * $\{\text{nif}\} \rightarrow \{\text{nome}, \text{apelido}, \text{morada}, \text{email}, \text{nTelefone}\}$.
 - * $\{\text{email}\} \rightarrow \{\text{nif}, \text{nome}, \text{apelido}, \text{morada}, \text{nTelefone}\}$.
 - * $\{\text{nTelefone}\} \rightarrow \{\text{nif}, \text{nome}, \text{apelido}, \text{morada}, \text{email}\}$.
 - **Pessoas** encontra-se na **BCNF**. Isto deve-se ao facto de todas as dependências funcionais, apesar de não serem triviais, do lado esquerdo apresentam uma *super-chave*. (relembramos que o email e o nTelefone são únicos, pelo que identificam também a pessoa).

- **Funcionarios**(nif, cargo)
 - **Dependências Funcionais:**
 - * $\{\text{nif}\} \rightarrow \{\text{cargo}\}$.
 - **Funcionarios** encontra-se na **BCNF**. Isto deve-se ao facto de a dependência funcional, apesar de não ser trivial, do lado esquerdo apresenta uma *super-chave*.
- **Clientes**(nif)
 - **Dependências Funcionais:**
 - * $\{\text{nif}\} \rightarrow \{\text{nif}\}$ (*trivial*).
 - **Clientes** encontra-se na **BCNF**. Isto deve-se ao facto de só existirem dependências funcionais triviais.
- **Vendas**(idV, idC, nifF, nifC, dataVenda, montante)
 - **Dependências Funcionais:**
 - * $\{\text{idV}\} \rightarrow \{\text{idC}, \text{nifF}, \text{nifC}, \text{dataVenda}, \text{montante}\}$.
 - **Vendas** encontra-se na **BCNF**. Isto deve-se ao facto de a dependência funcional, apesar de não ser trivial, do lado esquerdo apresenta uma *super-chave*.
- **TestDrives**(idC, nifF, nifC, data, hora, idT, classificacao)
 - **Dependências Funcionais:**
 - * $\{\text{idT}\} \rightarrow \{\text{idC}, \text{nifF}, \text{nifC}, \text{data}, \text{hora}, \text{classificacao}\}$.
 - * $\{\text{idC}, \text{nifF}, \text{nifC}, \text{data}, \text{hora}\} \rightarrow \{\text{idC}, \text{nifF}, \text{nifC}, \text{data}, \text{hora}, \text{classificacao}, \text{idT}\}$
 - **TestDrives** encontra-se na **BCNF**. Isto deve-se ao facto de todas as dependências funcionais, apesar de não serem triviais, do lado esquerdo apresentam uma *super-chave*.

5 Consultas Interessantes

Nesta secção vamos apresentar algumas consultas que podem ser feitas à nossa base de dados, que nós achamos pertinentes. Para tal, vamos apresentar as consultas e a respetiva implementação em SQL.

5.1 Qual é a marca de Carros mais vendida?

```
1  WITH VendasPorMarca(idMarca, total) AS (  
2      SELECT idMarca, count(idV) AS total  
3      FROM vendas INNER JOIN carros USING(idC)  
4              INNER JOIN modelos USING(nomemodelo)  
5              INNER JOIN marcas USING(idmarca)  
6      GROUP BY idMarca  
7  ),  
8  MaxVendas(total) AS (  
9      SELECT MAX(total) AS total  
10     FROM VendasPorMarca  
11 )  
12 SELECT nomeMarca, total AS Vendas  
13 FROM VendasPorMarca INNER JOIN MaxVendas USING(total)  
14              INNER JOIN marcas USING(idMarca);
```

Listing 1: Qual é a marca de Carros mais vendida?

5.2 Qual é o nome do funcionário com mais vendas da marca Mercedes?

```
1  WITH VendasPorFuncionario(nifF, total) AS (  
2      SELECT nifF, COUNT(idV) AS total  
3      FROM vendas INNER JOIN carros USING(idC)  
4              INNER JOIN modelos USING(nomemodelo)  
5              INNER JOIN marcas USING(idmarca)  
6      WHERE nomeMarca = 'Mercedes'  
7      GROUP BY nifF  
8  ),  
9  MaxVendas(total) AS (  
10     SELECT MAX(total) AS total  
11     FROM VendasPorFuncionario  
12 )  
13 SELECT CONCAT(CONCAT(nome, ' '), apelido) AS Funcionario, total AS Vendas  
14 FROM VendasPorFuncionario INNER JOIN MaxVendas USING(total)  
15              INNER JOIN Pessoas ON nifF = Pessoas.Nif;
```

Listing 2: Qual é o nome do funcionário com mais vendas da marca Mercedes?

5.3 Qual é o tipo de motor mais comum nos carros vendidos?

```
1  WITH VendasPorMotor(nomeMotor, total) AS (  
2      SELECT nomeMotor, COUNT(idV) AS total  
3      FROM vendas INNER JOIN carros USING(idC)  
4      GROUP BY nomeMotor  
5  ),  
6  MaxVendas(total) AS (  
7      SELECT MAX(total) AS total  
8      FROM VendasPorMotor  
9  )  
10 SELECT nomeMotor, total AS Vendas  
11 FROM VendasPorMotor INNER JOIN MaxVendas USING(total);
```

Listing 3: Qual é o tipo de motor mais comum nos carros vendidos?

5.4 Qual é o carro que possui o maior numero de TestDrives?

```
1  WITH TestDrivesPorCarro(idC, total) AS (  
2      SELECT idC, count(idT) AS total  
3      FROM testdrives  
4      GROUP BY idC  
5  ),  
6  MaxTestDrives(total) AS (  
7      SELECT MAX(total) AS total  
8      FROM TestDrivesPorCarro  
9  )  
10 SELECT idC, nomeModelo, nomeMarca, ano, cor, total as TestDrives  
11 FROM TestDrivesPorCarro INNER JOIN MaxTestDrives USING(total)  
12                          INNER JOIN Carros USING(idC)  
13                          INNER JOIN Modelos USING(nomemodelo)  
14                          INNER JOIN marcas USING(idMarca);
```

Listing 4: Qual é o carro que possui o maior numero de TestDrives?

5.5 Quem é o cliente que efetuou mais compras?

```
1  WITH ComprasPorCliente(nifC, total) AS (  
2      SELECT nifC, COUNT(idV) AS total  
3      FROM vendas  
4      GROUP BY nifC  
5  ),  
6  MaxCompras(total) AS (  
7      SELECT MAX(total) AS total  
8      FROM ComprasPorCliente  
9  )  
10 SELECT CONCAT(CONCAT(nome, ' '), apelido) AS Cliente, total AS Compras  
11 FROM ComprasPorCliente INNER JOIN MaxCompras USING(total)  
12                          INNER JOIN Pessoas ON nifC = Pessoas.Nif;
```

Listing 5: Quem é o cliente que efetuou mais compras?

6 Triggers e Vistas

Nesta secção vamos apresentar os **triggers**, e **vistas** que utilizamos na nossa base de dados.

6.1 Triggers

6.1.1 idMarca

Este trigger é executado antes de uma inserção na tabela **Marcas**. É utilizado para atribuir um **idMarca** a uma nova entrada na tabela **Marcas**. O **idMarca** é incrementado automaticamente, sendo que o valor é obtido através de uma sequence.

```
1 CREATE OR REPLACE TRIGGER idm_trigger
2 BEFORE INSERT ON Marcas
3 FOR EACH ROW
4 BEGIN
5     SELECT idm_sequence.NEXTVAL
6     INTO :new.idMarca
7     FROM dual;
8 END;
9 /
```

Listing 6: idMarca Trigger

6.1.2 idC

Este trigger é executado antes de uma inserção na tabela **Carros**. É utilizado para atribuir um **idC** a uma nova entrada na tabela **Carros**. O **idC** é incrementado automaticamente, sendo que o valor é obtido através de uma sequence.

```
1 CREATE OR REPLACE TRIGGER idc_trigger
2 BEFORE INSERT ON Carros
3 FOR EACH ROW
4 BEGIN
5     SELECT idc_sequence.NEXTVAL
6     INTO :new.idC
7     FROM dual;
8 END;
9 /
```

Listing 7: idC Trigger

6.1.3 idV

Este trigger é executado antes de uma inserção na tabela **Vendas**. É utilizado para atribuir um idV a uma nova entrada na tabela **Vendas**. O idV é incrementado automaticamente, sendo que o valor é obtido através de uma sequence.

```
1 CREATE OR REPLACE TRIGGER idv_trigger
2 BEFORE INSERT ON Vendas
3 FOR EACH ROW
4 BEGIN
5     SELECT idv_sequence.NEXTVAL
6     INTO :new.idV
7     FROM dual;
8 END;
9 /
```

Listing 8: idV Trigger

6.1.4 idT

Este trigger é executado antes de uma inserção na tabela **TestDrives**. É utilizado para atribuir um idT a uma nova entrada na tabela **TestDrives**. O idT é incrementado automaticamente, sendo que o valor é obtido através de uma sequence.

```
1 CREATE OR REPLACE TRIGGER idt_trigger
2 BEFORE INSERT ON TestDrives
3 FOR EACH ROW
4 BEGIN
5     SELECT idt_sequence.NEXTVAL
6     INTO :new.idT
7     FROM dual;
8 END;
9 /
```

Listing 9: idT Trigger

6.1.5 incrementDisponibilidade

Este trigger é executado após uma inserção na tabela **Carros**. É utilizado para incrementar a disponibilidade de uma marca de carros sempre que um novo carro é adicionado à tabela **Carros**. Isto é feito através de uma atualização na tabela **Marcas**, onde a disponibilidade é incrementada em 1 para a marca do modelo do carro adicionado.

```
1 CREATE OR REPLACE TRIGGER disponibilidade_increment
2 AFTER INSERT ON Carros
3 FOR EACH ROW
4 BEGIN
5     UPDATE Marcas
6     SET disponibilidade = disponibilidade + 1
7     WHERE idMarca = (SELECT idMarca FROM Modelos WHERE nomeModelo = :
8     new.nomeModelo);
9 END;
10 /
```

Listing 10: incrementDisponibilidade Trigger

6.1.6 insertVenda

Este trigger impõe diversas regras para garantir a consistência e integridade da tabela Vendas, verificando a existência de entidades relacionadas (**Carro**, **Funcionário**, **Cliente**), garantindo dados válidos (valor não negativo, data de venda não futura, vendedor e comprador diferentes, se o carro já foi vendido) e mantendo a contagem de disponibilidade dos modelos de automóveis. Se alguma dessas verificações falhar, o **Trigger** gerará os erros apropriados, impedindo a inserção.

```

1  CREATE OR REPLACE TRIGGER insertVenda
2  BEFORE INSERT ON Vendas
3  FOR EACH ROW
4  DECLARE
5      v_idC NUMBER;
6      v_nifF NUMBER;
7      v_nifC NUMBER;
8      isCarSold NUMBER := 1;
9  BEGIN
10     BEGIN
11         SELECT IDC into v_idC FROM carros WHERE IDC = :new.idC;
12     EXCEPTION
13         WHEN NO_DATA_FOUND THEN
14             RAISE_APPLICATION_ERROR(-20005, 'Carro nao existe');
15     END;
16
17     BEGIN
18         SELECT nif into v_nifF FROM FUNCIONARIOS WHERE nif = :new.nifF;
19     EXCEPTION
20         WHEN NO_DATA_FOUND THEN
21             RAISE_APPLICATION_ERROR(-20003, 'NIF de Funcionario nao existe
22     ');
23     END;
24
25     BEGIN
26         SELECT nif into v_nifC FROM CLIENTES WHERE nif = :new.nifC;
27     EXCEPTION
28         WHEN NO_DATA_FOUND THEN
29             RAISE_APPLICATION_ERROR(-20004, 'NIF de Cliente nao existe');
30     END;
31
32     BEGIN
33         SELECT idc into isCarSold from vendas where idc = :new.idC;
34     EXCEPTION
35         WHEN NO_DATA_FOUND THEN
36             isCarSold := 0;
37     END;
38
39     IF :new.montante < 0 THEN
40         RAISE_APPLICATION_ERROR(-20000, 'Montante nao pode ser
41     negativo');
42     END IF;
43
44     IF :new.dataVenda > SYSDATE THEN
45         RAISE_APPLICATION_ERROR(-20001, 'Data de venda nao pode ser no
46     futuro');

```

```

44         END IF;
45
46         IF :new.nifF = :new.nifC THEN
47             RAISE_APPLICATION_ERROR(-20002, 'Vendedor e Cliente nao podem
48 ser a mesma pessoa');
49         END IF;
50
51         IF isCarSold = 1 THEN
52             RAISE_APPLICATION_ERROR(-20006, 'Carro ja foi vendido');
53         END IF;
54
55         UPDATE Marcas
56         SET disponibilidade = disponibilidade - 1
57         WHERE idMarca = (SELECT idMarca FROM Modelos WHERE nomeModelo = (
58 SELECT nomeModelo FROM Carros WHERE idC = :new.idC));
59     END;
60 /

```

Listing 11: insertVenda Trigger

6.1.7 insertTestDrive

Este **Trigger** impõe diversas regras para garantir a consistência e integridade da tabela TestDrives, verificando a existência de entidades relacionadas (**Carro**, **Funcionário**, **Cliente**), garantindo dados válidos (**Cliente** e **Funcionário** diferentes, **Carro** não pode estar em **Test-drive** na mesma data e hora). Se alguma dessas verificações falhar, o **Trigger** gerará os erros apropriados, impedindo a inserção.

```

1  CREATE OR REPLACE TRIGGER insertTestDrive
2  BEFORE INSERT ON TestDrives
3  FOR EACH ROW
4  DECLARE
5      t_idC NUMBER;
6      t_nifF NUMBER;
7      t_nifC NUMBER;
8  BEGIN
9      BEGIN
10         SELECT IDC into t_idC FROM carros WHERE IDC = :new.idC;
11     EXCEPTION
12         WHEN NO_DATA_FOUND THEN
13             RAISE_APPLICATION_ERROR(-20005, 'Carro nao existe');
14     END;
15
16     BEGIN
17         SELECT nif into t_nifF FROM FUNCIONARIOS WHERE nif = :new.nifF;
18     EXCEPTION
19         WHEN NO_DATA_FOUND THEN
20             RAISE_APPLICATION_ERROR(-20003, 'NIF de Funcionario nao existe
21 ');
22     END;
23
24     BEGIN
25         SELECT nif into t_nifC FROM CLIENTES WHERE nif = :new.nifC;
26     EXCEPTION

```



```

26         WHEN NO_DATA_FOUND THEN
27             RAISE_APPLICATION_ERROR(-20004, 'NIF de Cliente nao existe');
28         END;
29
30         IF :new.dataTest > SYSDATE THEN
31             RAISE_APPLICATION_ERROR(-20001, 'Data de TestDrive nao pode ser no
32             futuro');
33         END IF;
34
35         IF :new.nifF = :new.nifC THEN
36             RAISE_APPLICATION_ERROR(-20002, 'Funcionario e Cliente nao podem
37             ser a mesma pessoa');
38         END IF;
39     END IF;
40 END;
41 /

```

Listing 12: insertTestDrive Trigger

6.1.8 disponibilidade

Este **trigger** é executado antes de uma inserção na tabela **Marcas**. É utilizado para garantir que o valor de **disponibilidade** é sempre 0. Isto é feito atribuindo o valor 0 a **disponibilidade** antes de uma inserção.

```

1 CREATE OR REPLACE TRIGGER disponibilidade_null
2 BEFORE INSERT ON Marcas
3 FOR EACH ROW
4 BEGIN
5     :new.disponibilidade := 0;
6 END;
7 /

```

Listing 13: disponibilidade Trigger

6.1.9 removeCarro

Este **Trigger** garante que quando um carro for removido da tabela **Carros** (removido do sistema) todos os registos relacionados nas tabelas **Usados**, **Vendas** e **TestDrives** também são removidos, mantendo a integridade dos dados e limpando os dados relacionados. Se o **carro** não foi vendido (ou seja, não foi listado na tabela de vendas), a **disponibilidade da Marca** do **Modelo** do **Carro** correspondente é diminuída em 1 na tabela de **Marcas**. Ao realizar essas operações, o **Trigger** garante que todos os dados dependentes sejam tratados adequadamente e que a contagem de **disponibilidade** da marca seja mantida com precisão quando um **Carro** for removido do sistema.

```

1 CREATE OR REPLACE TRIGGER removeCarro
2 AFTER DELETE ON Carros
3 FOR EACH ROW
4 DECLARE isSold NUMBER := 1;
5 BEGIN
6
7     BEGIN
8         SELECT idc into isSold from vendas where idc = :old.idC;

```

```
9      EXCEPTION
10      WHEN NO_DATA_FOUND THEN
11          isSold := 0;
12      END;
13
14      DELETE FROM Usados
15      WHERE idC = :old.idC;
16
17      DELETE FROM Vendas
18      WHERE idC = :old.idC;
19
20      DELETE FROM TestDrives
21      WHERE idC = :old.idC;
22
23      IF isSold = 0 THEN
24          UPDATE Marcas
25          SET disponibilidade = disponibilidade - 1
26          WHERE idMarca = (SELECT idMarca FROM Modelos WHERE nomeModelo
= :old.nomeModelo);
27      END IF;
28      END;
29      /
```

Listing 14: removeCarro Trigger

6.1.10 removePessoa

Este **Trigger** garante que quando uma pessoa for removida da tabela **Pessoas** (removida do sistema) todos os registos relacionados nas tabelas **TestDrives**, **Vendas**, **Funcionarios** e **Clientes** também são removidos, mantendo a integridade dos dados e limpando os dados relacionados. Ao realizar essas operações, o **Trigger** garante que todos os dados dependentes sejam tratados adequadamente quando uma **Pessoa** for removida do sistema.

```
1  CREATE OR REPLACE TRIGGER removePessoa
2  BEFORE DELETE ON Pessoas
3  FOR EACH ROW
4  BEGIN
5      DELETE FROM TESTDRIVES WHERE nifF = :old.nif or nifC = :old.nif;
6      DELETE FROM VENDAS WHERE nifF = :old.nif or nifC = :old.nif;
7      DELETE FROM FUNCIONARIOS WHERE nif = :old.nif;
8      DELETE FROM CLIENTES WHERE nif = :old.nif;
9  END;
10 /
```

Listing 15: removePessoa Trigger

6.1.11 removeMotor

Este **Trigger** garante que quando um motor for removido da tabela **Motores** (removido do sistema) todos os registos relacionados nas tabelas **Combustoes**, **Eletricos**, **Carros** e **TipoMotor** também são removidos, mantendo a integridade dos dados e limpando os dados relacionados. Ao realizar essas operações, o **Trigger** garante que todos os dados dependentes sejam tratados adequadamente quando um **Motor** for removido do sistema.

```
1 CREATE OR REPLACE TRIGGER removeMotor
2 BEFORE DELETE ON Motores
3 FOR EACH ROW
4 BEGIN
5     DELETE FROM Combustoes WHERE nomeMotor = :old.nomeMotor;
6     DELETE FROM Eletricos WHERE nomeMotor = :old.nomeMotor;
7     DELETE FROM CARROS WHERE nomeMotor = :old.nomeMotor;
8     DELETE FROM TipoMotor WHERE nomeMotor = :old.nomeMotor;
9 END;
10 /
```

Listing 16: removeMotor Trigger

6.1.12 removeTipoMotor

Este **Trigger** garante que quando um tipo de motor for removido da tabela **TipoMotor** (removido do sistema) todos os registos relacionados na tabela **Carros** também são removidos, mantendo a integridade dos dados e limpando os dados relacionados. Ao realizar essas operações, o **Trigger** garante que todos os dados dependentes sejam tratados adequadamente quando um **TipoMotor** for removido do sistema.

```
1 CREATE OR REPLACE TRIGGER removeTipoMotor
2 BEFORE DELETE ON TipoMotor
3 FOR EACH ROW
4 BEGIN
5     DELETE FROM CARROS WHERE nomeMotor = :old.nomeMotor and nomeModelo
6     = :old.nomeModelo;
7 END;
8 /
```

Listing 17: removeTipoMotor Trigger

6.1.13 removeModelo

Este **Trigger** garante que quando um modelo for removido da tabela **Modelos** (removido do sistema) todos os registos relacionados nas tabelas **TipoMotor** e **Carros** também são removidos, mantendo a integridade dos dados e limpando os dados relacionados. Ao realizar essas operações, o **Trigger** garante que todos os dados dependentes sejam tratados adequadamente quando um **Modelo** for removido do sistema.

```
1 CREATE OR REPLACE TRIGGER removeModelo
2 BEFORE DELETE ON Modelos
3 FOR EACH ROW
4 BEGIN
5     DELETE FROM TipoMotor WHERE nomeModelo = :old.nomeModelo;
6     DELETE FROM CARROS WHERE nomeModelo = :old.nomeModelo;
7 END;
8 /
```

Listing 18: removeModelo Trigger

6.1.14 removeMarca

Este **Trigger** garante que quando uma marca for removida da tabela **Marcas** (removida do sistema) todos os registos relacionados nas tabelas **Modelos** e **Carros** também são removidos, mantendo a integridade dos dados e limpando os dados relacionados. Ao realizar essas operações, o **Trigger** garante que todos os dados dependentes sejam tratados adequadamente quando uma **Marca** for removida do sistema.

```
1 CREATE OR REPLACE TRIGGER removeMarca
2 BEFORE DELETE ON Marcas
3 FOR EACH ROW
4 BEGIN
5     DELETE FROM Modelos WHERE idMarca = :old.idMarca;
6 END;
7 /
```

Listing 19: removeMarca Trigger

6.2 Vistas

6.2.1 percentagemVendasFuncionario

Esta vista foi criada para facilitar a visualização da percentagem de vendas de cada funcionário.

```
1 CREATE OR REPLACE VIEW percentagemVendasFuncionario AS
2 WITH VendasPorFuncionario(nifF, total) AS (
3     SELECT nifF, COUNT(idV) AS total
4     FROM vendas
5     GROUP BY nifF
6 ),
7 TotalVendas(totalVendas) AS (
8     SELECT COUNT(idV) AS totalVendas
9     FROM vendas
10 )
11 SELECT distinct nifF, CONCAT(CONCAT(nome, ' '), apelido) AS Funcionario,
12     total AS Vendas, (total/totalVendas)*100 AS Percentagem
13 FROM (VendasPorFuncionario INNER JOIN Pessoas ON nifF = Pessoas.Nif),
14     TotalVendas;
```

Listing 20: percentagemVendasFuncionario

6.2.2 FuncionariosInfo

Esta vista junta as tabelas **Pessoas** e **Funcionarios**, para facilitar a visualização de informação sobre os **Funcionários**.

```
1 CREATE OR REPLACE VIEW FuncionariosInfo AS
2 SELECT NIF, NOME, APELIDO, MORADA, EMAIL, NTELEFONE, CARGO
3 FROM PESSOAS INNER JOIN FUNCIONARIOS USING (NIF);
```

Listing 21: FuncionariosInfo

6.2.3 ClientesInfo

Esta vista junta as tabelas **Pessoas** e **Clientes**, para facilitar a visualização de informação sobre os **Clientes**.

```
1 CREATE OR REPLACE VIEW ClientesInfo AS
2 SELECT NIF, NOME, APELIDO, MORADA, EMAIL, NTELEFONE
3 FROM PESSOAS INNER JOIN CLIENTES USING (NIF);
```

Listing 22: ClientesInfo

6.2.4 CarrosInfo

Esta vista junta as tabelas **Carros**, **Modelos** e **Marcas**, para facilitar a visualização de informação sobre os **Carros**.

```
1 CREATE OR REPLACE VIEW CarrosInfo AS
2 SELECT IDC, NOMEMODELO, NOMEMARCA, ANO, COR
3 FROM CARROS INNER JOIN MODELOS USING (NOMEMODELO)
4          INNER JOIN MARCAS USING (IDMARCA);
```

Listing 23: CarrosInfo

6.2.5 VendasInfo

Esta vista junta as tabelas **Vendas**, **FuncionariosInfo**, **ClientesInfo** e **CarrosInfo**, para facilitar a visualização de informação sobre as **Vendas**.

```
1 CREATE OR REPLACE VIEW VendasInfo AS
2 SELECT idV, F.nif AS nifV, CONCAT(CONCAT(F.nome, ' '), F.apelido) Vendedor,
3        C.nif AS nifC, CONCAT(CONCAT(C.nome, ' '), C.apelido) Cliente, idc,
4        dataVenda, nomeModelo, nomeMarca, montante
5 FROM vendas INNER JOIN funcionariosInfo AS F ON (nifF = F.nif)
6          INNER JOIN clientesInfo AS C ON (nifC = C.nif)
7          INNER JOIN carrosInfo USING(idc);
```

Listing 24: VendasInfo

6.2.6 TestDrivesInfo

Esta vista junta as tabelas **TestDrives**, **CarrosInfo**, **FuncionariosInfo** e **ClientesInfo**, para facilitar a visualização de informação sobre os **TestDrives**.

```
1 CREATE OR REPLACE VIEW TestDrivesInfo AS
2 SELECT idt, IDC, nomeModelo, nomeMarca, idmarca, dataTest, HoraTest,
3        classificacao, p1.nif AS nifF,
4        CONCAT(CONCAT(p1.nome, ' '), p1.apelido) AS Funcionario,
5        p2.nif AS nifC, CONCAT(CONCAT(p2.nome, ' '), p2.apelido) AS Cliente
6 FROM TESTDRIVES INNER JOIN carrosInfo USING(idc)
7          INNER JOIN marcas USING(nomeMarca)
8          INNER JOIN pessoas AS p1 ON testdrives.nifc = p1.nif
9          INNER JOIN pessoas AS p2 ON testdrives.niff = p2.nif;
```

Listing 25: TestDrivesInfo

6.2.7 CombustoesInfo

Esta vista junta as tabelas **Motores** e **Combustoes**, para facilitar a visualização de informação sobre os **Motores** que utilizam combustão.

```
1 CREATE OR REPLACE VIEW CombustoesInfo AS
2 SELECT nomeMotor, autonomia, potencia, consumos, tipoCombustao,
   tamanhoDeposito
3 FROM (
4     (SELECT nomeMotor from Motores NATURAL JOIN Combustoes)
5     MINUS
6     (SELECT nomeMotor from Motores NATURAL JOIN Eletricos)
7 ) NATURAL JOIN MOTORES NATURAL JOIN COMBUSTOES;
```

Listing 26: CombustoesInfo

6.2.8 EletricosInfo

Esta vista junta as tabelas **Motores** e **Eletricos**, para facilitar a visualização de informação sobre os **Motores** que utilizam eletricidade.

```
1 CREATE OR REPLACE VIEW EletricosInfo AS
2 SELECT nomeMotor, autonomia, potencia, consumos, capacidadeBateria
3 FROM (
4     (SELECT nomeMotor from Motores NATURAL JOIN Eletricos)
5     minus
6     (SELECT nomeMotor from Motores NATURAL JOIN Combustoes)
7 ) NATURAL JOIN MOTORES NATURAL JOIN ELETRICOS;
```

Listing 27: EletricosInfo

6.2.9 HibridosInfo

Esta vista junta as tabelas **Motores**, **Combustoes**, **Eletricos**, contemplando apenas os motores que são **Combustoes** e **Eletricos** isto para permitir a consulta sobre os motores híbridos.

```
1 CREATE OR REPLACE VIEW HibridosInfo AS
2 SELECT nomeMotor, autonomia, potencia, consumos, tipoCombustao,
   tamanhoDeposito, capacidadeBateria
3 FROM (
4     (SELECT nomeMotor from Motores NATURAL JOIN Combustoes)
5     intersect
6     (SELECT nomeMotor from Motores NATURAL JOIN Eletricos)
7 ) NATURAL JOIN MOTORES
8     NATURAL JOIN COMBUSTOES
9     NATURAL JOIN ELETRICOS;
```

Listing 28: HibridosInfo

6.2.10 ModelosInfo

Esta vista junta as tabelas **Modelos** e **Marcas**, para facilitar a visualização de informação sobre os **Modelos**.

```
1 CREATE OR REPLACE VIEW ModelosInfo AS
2 SELECT nomeModelo, nomeMarca, tipoModelo, idMarca
3 FROM Modelos INNER JOIN Marcas USING (idMarca);
```

Listing 29: ModelosInfo

6.3 Triggers nas Vistas

Os **Triggers** desta secção são utilizados para garantir a integridade dos dados nas **Vistas** criadas anteriormente. Estes **Triggers** são executados em vez de uma inserção, atualização ou remoção numa **Vista**, operando sobre as tabelas subjacentes à **Vista**.

6.3.1 insertFuncionariosInfo

Este **Trigger** é executado quando queremos inserir um tuplo na **Vista** **FuncionariosInfo**. É utilizado para garantir que a inserção na **Vista** é válida, ou seja, que a inserção é executada nas tabelas subjacentes **Pessoas** e **Funcionarios**, em vez de na vista, inserindo o tuplo nas respetivas tabelas (pois não podemos alterar a vista, apenas as tabelas que compõem a vista).

```
1 CREATE OR REPLACE TRIGGER insertFuncionarioInfo
2 INSTEAD OF INSERT ON FuncionariosInfo
3 FOR EACH ROW
4 BEGIN
5     INSERT INTO Pessoas (NIF, NOME, APELIDO, MORADA, EMAIL, NTELEFONE)
6     VALUES (:new.NIF, :new.NOME, :new.APELIDO, :new.MORADA, :new.EMAIL,
7             :new.NTELEFONE);
8
9     INSERT INTO Funcionarios (NIF, CARGO)
10    VALUES (:new.NIF, :new.CARGO);
11 END;
12 /
```

Listing 30: insertFuncionariosInfo

6.3.2 deleteFuncionarioInfo

Este **Trigger** é executado quando queremos remover um tuplo na **Vista** **FuncionariosInfo**. É utilizado para garantir que a remoção na **Vista** é válida, ou seja, que a remoção é executada nas tabelas subjacentes **Pessoas**, em vez de na vista, removendo o tuplo da tabela **Pessoas** (pois não podemos alterar a vista, apenas as tabelas que compõem a vista).

```
1 CREATE OR REPLACE TRIGGER deleteFuncionarioInfo
2 INSTEAD OF DELETE ON FuncionariosInfo
3 FOR EACH ROW
4 BEGIN
5     DELETE FROM Pessoas
6     WHERE NIF = :old.NIF;
7 END;
8 /
```

Listing 31: deleteFuncionarioInfo

6.3.3 updateFuncionarioInfo

Este **Trigger** é acionado quando queremos atualizar um tuplo na **Vista FuncionariosInfo**. É utilizado para garantir que a atualização na **Vista** é válida, ou seja, que a atualização é executada nas tabelas subjacentes **Pessoas** e **Funcionarios**, em vez de na vista, atualizando o tuplo nas respetivas tabelas (pois não podemos alterar a vista, apenas as tabelas que compõem a vista).

```
1 CREATE OR REPLACE TRIGGER updateFuncionarioInfo
2 INSTEAD OF UPDATE ON FuncionariosInfo
3 FOR EACH ROW
4 BEGIN
5     UPDATE Pessoas
6     SET NOME = :new.NOME, APELIDO = :new.APELIDO, MORADA = :new.MORADA,
7     EMAIL = :new.EMAIL, NTELEFONE = :new.NTELEFONE
8     WHERE NIF = :old.NIF;
9
10    UPDATE Funcionarios
11    SET CARGO = :new.CARGO
12    WHERE NIF = :old.NIF;
13
14 END;
15 /
```

Listing 32: updateFuncionarioInfo

6.3.4 insertClienteInfo

Este **Trigger** é executado quando queremos inserir um tuplo na **Vista ClientesInfo**. É utilizado para garantir que a inserção na **Vista** é válida, ou seja, que a inserção é executada nas tabelas subjacentes **Pessoas** e **Clientes**, em vez de na vista, inserindo o tuplo nas respetivas tabelas (pois não podemos alterar a vista, apenas as tabelas que compõem a vista).

```
1 CREATE OR REPLACE TRIGGER insertClienteInfo
2 INSTEAD OF INSERT ON ClientesInfo
3 FOR EACH ROW
4 BEGIN
5     INSERT INTO Pessoas (NIF, NOME, APELIDO, MORADA, EMAIL, NTELEFONE)
6     VALUES (:new.NIF, :new.NOME, :new.APELIDO, :new.MORADA, :new.EMAIL,
7             :new.NTELEFONE);
8
9     INSERT INTO Clientes (NIF)
10    VALUES (:new.NIF);
11 END;
12 /
```

Listing 33: insertClienteInfo

6.3.5 deleteClienteInfo

Este **Trigger** é executado quando queremos remover um tuplo na **Vista ClientesInfo**. É utilizado para garantir que a remoção na **Vista** é válida, ou seja, que a remoção é executada

nas tabelas subjacentes **Pessoas** em vez de na vista, removendo o tuplo da tabela **Pessoas** (pois não podemos alterar a vista, apenas as tabelas que compõem a vista).

```
1 CREATE OR REPLACE TRIGGER deleteClienteInfo
2 INSTEAD OF DELETE ON ClientesInfo
3 FOR EACH ROW
4 BEGIN
5     DELETE FROM Pessoas
6     WHERE NIF = :old.NIF;
7 END;
8 /
```

Listing 34: deleteClienteInfo

6.3.6 updateClienteInfo

Este **Trigger** é acionado quando queremos atualizar um tuplo na **Vista ClientesInfo**. É utilizado para garantir que a atualização na **Vista** é válida, ou seja, que a atualização é executada nas tabelas subjacentes **Pessoas** e **Clientes**, em vez de na vista, atualizando o tuplo nas respetivas tabelas (pois não podemos alterar a vista, apenas as tabelas que compõem a vista).

```
1 CREATE OR REPLACE TRIGGER updateClienteInfo
2 INSTEAD OF UPDATE ON ClientesInfo
3 FOR EACH ROW
4 BEGIN
5     UPDATE Pessoas
6     SET NOME = :new.NOME, APELIDO = :new.APELIDO, MORADA = :new.MORADA,
7     EMAIL = :new.EMAIL, NTELEFONE = :new.NTELEFONE
8     WHERE NIF = :old.NIF;
9 END;
10 /
```

Listing 35: updateClienteInfo

6.3.7 insertVendasInfo

Este **Trigger** é acionado quando queremos introduzir um tuplo na **Vista VendasInfo**. É utilizado para garantir que a atualização na **Vista** é válida, ou seja, que a Introdução é executada na tabela de **vendas**, em vez de na vista (pois não podemos alterar a vista, apenas as tabelas que compõem a vista).

```
1 CREATE OR REPLACE TRIGGER insertVendasInfo
2 INSTEAD OF INSERT ON VendasInfo
3 FOR EACH ROW
4 BEGIN
5     INSERT INTO Vendas (idV, nifF, nifC, idC, dataVenda, montante)
6     VALUES (:new.idV, :new.nifV, :new.nifC, :new.idC, :new.dataVenda,
7             :new.montante);
8 END;
9 /
```

Listing 36: insertVendasInfo

6.3.8 deleteVendasInfo

Este **Trigger** é acionado quando queremos remover um tuplo na **Vista VendasInfo**. É utilizado para garantir que a remoção na **Vista** é válida, ou seja, que a remoção é executada na tabela de **Vendas**, em vez de na vista (pois não podemos alterar a vista, apenas as tabelas que compõem a vista).

```
1 CREATE OR REPLACE TRIGGER deleteVendasInfo
2 INSTEAD OF DELETE ON VendasInfo
3 FOR EACH ROW
4 BEGIN
5     DELETE FROM Vendas
6     WHERE idV = :old.idV;
7 END;
8 /
```

Listing 37: deleteVendasInfo

6.3.9 updateVendasInfo

Este **Trigger** é acionado quando queremos atualizar um tuplo na **Vista VendasInfo**. É utilizado para garantir que a atualização na **Vista** é válida, ou seja, que a atualização é executada na tabela de **vendas**, em vez de na vista (pois não podemos alterar a vista, apenas as tabelas que compõem a vista).

```
1 CREATE OR REPLACE TRIGGER updateVendasInfo
2 INSTEAD OF UPDATE ON VendasInfo
3 FOR EACH ROW
4 BEGIN
5     UPDATE Vendas
6     SET nifF = :new.nifV, nifC = :new.nifC, idC = :new.idC,
7         dataVenda = :new.dataVenda, montante = :new.montante
8     WHERE idV = :old.idV;
9 END;
10 /
```

Listing 38: updateVendasInfo

6.3.10 insertCarroInfo

Este **Trigger** é acionado quando queremos introduzir um tuplo na **Vista CarrosInfo**. É utilizado para garantir que a introdução na **Vista** é válida, ou seja, que a Introdução é executada na tabela de **Carros**, em vez de na vista (pois não podemos alterar a vista, apenas as tabelas que compõem a vista).

```
1 CREATE OR REPLACE TRIGGER insertCarroInfo INSTEAD OF INSERT ON CarrosInfo
2 FOR EACH ROW
3 BEGIN
4     INSERT INTO Carros (IDC, NOMEMODELO, ANO, COR, NOMEMARCA, NOMEMOTOR)
5     VALUES (:new.IDC, :new.NOMEMODELO, :new.ANO, :new.COR, :new.NOMEMARCA,
6             :new.NOMEMOTOR);
7 END;
```

Listing 39: insertCarroInfo

6.3.11 removeCarroInfo

Este **Trigger** é acionado quando queremos remover um tuplo na **Vista CarrosInfo**. É utilizado para garantir que a remoção na **Vista** é válida, ou seja, que a remoção é executada na tabela de **Carros**, em vez de na vista (pois não podemos alterar a vista, apenas as tabelas que compõem a vista).

```
1 CREATE OR REPLACE TRIGGER removeCarroInfo
2 INSTEAD OF DELETE ON CarrosInfo
3 FOR EACH ROW
4 BEGIN
5     DELETE FROM Carros
6     WHERE IDC = :old.IDC;
7 END;
8 /
```

Listing 40: removeCarroInfo

6.3.12 updateCarroInfo

Este **Trigger** é acionado quando queremos atualizar um tuplo na **Vista CarrosInfo**. É utilizado para garantir que a atualização na **Vista** é válida, ou seja, que a atualização é executada na tabela de **Carros**, em vez de na vista (pois não podemos alterar a vista, apenas as tabelas que compõem a vista).

```
1 CREATE OR REPLACE TRIGGER updateCarroInfo
2 INSTEAD OF UPDATE ON CarrosInfo
3 FOR EACH ROW
4 BEGIN
5     UPDATE Carros
6     SET NOMEMODELO = :new.NOMEMODELO, ANO = :new.ANO, COR = :new.COR,
7       NOMEMARCA = :new.NOMEMARCA, NOMEMOTOR = :new.NOMEMOTOR
8     WHERE IDC = :old.IDC;
9 END;
10 /
```

Listing 41: updateCarroInfo

6.3.13 insertModelosInfo

Este **Trigger** é acionado quando queremos introduzir um tuplo na **Vista ModelosInfo**. É utilizado para garantir que a introdução na **Vista** é válida, ou seja, que a introdução é executada na tabela de **Modelos**, em vez de na vista (pois não podemos alterar a vista, apenas as tabelas que compõem a vista).

```
1 CREATE OR REPLACE TRIGGER insertModelosInfo INSTEAD OF INSERT ON
2 ModelosInfo FOR EACH ROW
3 BEGIN
4     INSERT INTO Modelos (NOMEMODELO, NOMEMARCA, TIPOMODELO, IDMARCA)
5     VALUES (:new.NOMEMODELO, :new.NOMEMARCA, :new.TIPOMODELO, :new.IDMARCA
6     );
7 END;
```

Listing 42: insertModelosInfo

6.3.14 deleteModelosInfo

Este **Trigger** é acionado quando queremos remover um tuplo na **Vista ModelosInfo**. É utilizado para garantir que a remoção na **Vista** é válida, ou seja, que a remoção é executada na tabela de **Modelos**, em vez de na vista (pois não podemos alterar a vista, apenas as tabelas que compõem a vista).

```
1 CREATE OR REPLACE TRIGGER deleteModelosInfo
2   INSTEAD OF DELETE ON ModelosInfo
3   FOR EACH ROW
4   BEGIN
5       DELETE FROM Modelos
6       WHERE NOMEMODELO = :old.NOMEMODELO;
7   END;
8   /
```

Listing 43: deleteModelosInfo

6.3.15 updateModelosInfo

Este **Trigger** é acionado quando queremos atualizar um tuplo na **Vista ModelosInfo**. É utilizado para garantir que a atualização na **Vista** é válida, ou seja, que a atualização é executada na tabela de **Modelos**, em vez de na vista (pois não podemos alterar a vista, apenas as tabelas que compõem a vista).

```
1 CREATE OR REPLACE TRIGGER updateModelosInfo
2   INSTEAD OF UPDATE ON ModelosInfo
3   FOR EACH ROW
4   BEGIN
5       UPDATE Modelos
6       SET NOMEMARCA = :new.NOMEMARCA, TIPOMODELO = :new.TIPOMODELO, IDMARCA
7       = :new.IDMARCA
8       WHERE NOMEMODELO = :old.NOMEMODELO;
9   END;
10  /
```

Listing 44: updateModelosInfo

7 Interface

Neste segmento vamos discutir a implementação da interface. Iremos fazer um descrição detalhada de cada uma das funcionalidades implementadas, e apresentar exemplos de utilização para demonstrar como utilizar a nossa implementação. Aqui esperamos colmatar os ultimos três pontos relativos aos objetivos do relatório do projeto. **Nota:** A interface foi desenvolvida utilizando o **Oracle APEX** no **Servidor Local**.

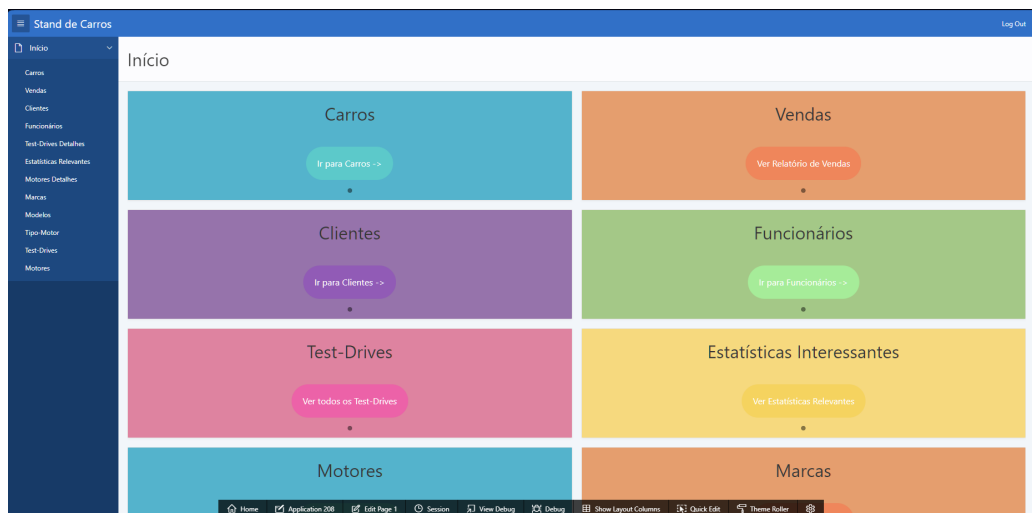


Figure 2: Pagina Inicial do APEX

A interface encontra-se dividida em 12 secções das quais as primeiras 10 são acessíveis através da pagina inicial do **APEX**:

- **Carros**
- **Vendas**
- **Clientes**
- **Funcionários**
- **Test-Drives Detalhes**
- **Estatísticas Interessantes**
- **Motores Detalhes**
- **Marcas**
- **Modelos**
- **TipoMotor**
- **Test-Drives**
- **Motores**

7.1 Carros

Nesta secção é possível visualizar todos os carros disponíveis, bem como adicionar, remover e editar carros. Na tabela estão listados os atributos do carro: **ano**, **cor**, **nomeModelo**, **nomeMarca**, **gama** e **nomeMotor**.

Idc	NomeModelo	NomeMarca	NomeMotor	Ano	Cor
11	CLA	Mercedes	Motor6	2020	Verde
18	G-Class	Mercedes	Motor14	2011	Amarelo
25	Kadjar	Renault	Motor4	2018	Preto
27	Koleos	Renault	Motor6	2010	Azul
28	Zoe	Renault	Motor7	2020	Vermelho
29	Twingo	Renault	Motor8	2019	Verde
30	Kangoo	Renault	Motor9	2018	Amarelo
31	Trafic	Renault	Motor10	2017	Preto
33	Focus	Ford	Motor12	2015	Azul
34	Kuga	Ford	Motor13	2014	Vermelho
35	Mondeo	Ford	Motor14	2013	Verde
36	Puma	Ford	Motor15	2012	Amarelo
38	Ranger	Ford	Motor2	2010	Branco
39	Ranger	Ford	Motor2	2010	Azul

Figure 3: Pagina Inicial de Carros

Sendo possível aceder aos detalhes do motor que equipa o carro clicando no nome do motor em questão (**drill-down**).

NomeMotor	Potência	Consumo	Tamanho depósito	Autonomia	Combustível	Capacidade da bateria
Motor6	250	10	-	600	-	80

Figure 4: Exemplo de Drill-Down do Motor 6

Para adicionar um carro basta clicar no botão **Create** e preencher os campos necessários. Em seguida pressionar o botão **Create**, outra vez.

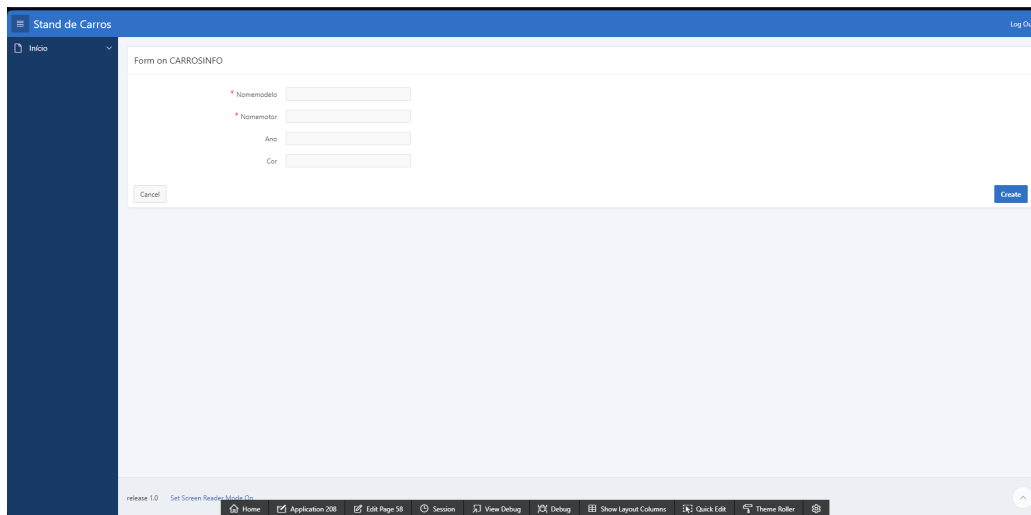
The screenshot shows a web application interface titled 'Stand de Carros'. On the left is a dark blue sidebar with a menu icon and the text 'Inicio'. The main content area is titled 'Form on CARROSINFO'. It contains four input fields: 'Nome modelo', 'Nome motor', 'Ano', and 'Cor'. Each field has a red asterisk to its left, indicating it is required. Below the fields are two buttons: 'Cancel' on the left and 'Create' on the right. At the bottom of the application, there is a status bar with various icons and text including 'release 1.0', 'Set Screen Reading Mode On', and a list of application tools like 'Home', 'Application Tools', 'Edit Page 10', 'Session', 'View Debug', 'Debug', 'Show Layout Columns', 'Quick Edit', and 'Theme Roller'.

Figure 5: Criação de um novo carro

Para remover um carro basta clicar no "Lápis", da linha do carro que se pretende remover, e de seguida no botão **Delete**. Para editar um carro basta clicar no "Lápis" e de seguida no botão **Apply Changes** após alterar os dados pretendidos.

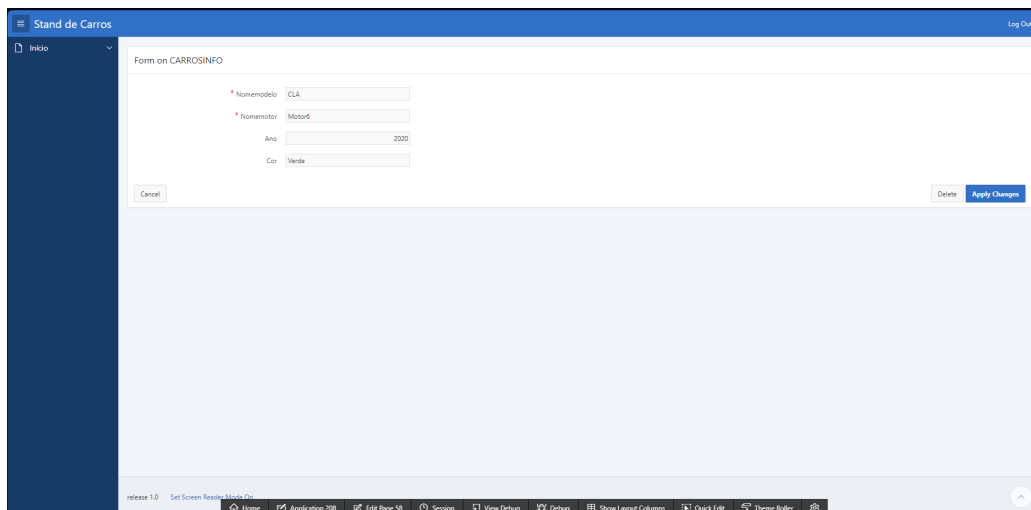
This screenshot shows the same 'Stand de Carros' application, but in edit mode. The 'Form on CARROSINFO' now contains pre-filled data: 'Nome modelo' is 'CLA', 'Nome motor' is 'Motor6', 'Ano' is '2020', and 'Cor' is 'Verde'. The 'Cancel' button remains on the left, but the 'Create' button has been replaced by 'Delete' and 'Apply Changes' buttons on the right. The bottom status bar is identical to the previous figure.

Figure 6: Edição/Remoção de um carro

Na parte superior da página é possível filtrar os carros por **Marca** e por **Ano**. Para tal basta selecionar a *select list* pretendida e clicar no valor esperado. Para limpar os filtros basta voltar ao valor default (primeiro na lista de valores possíveis).

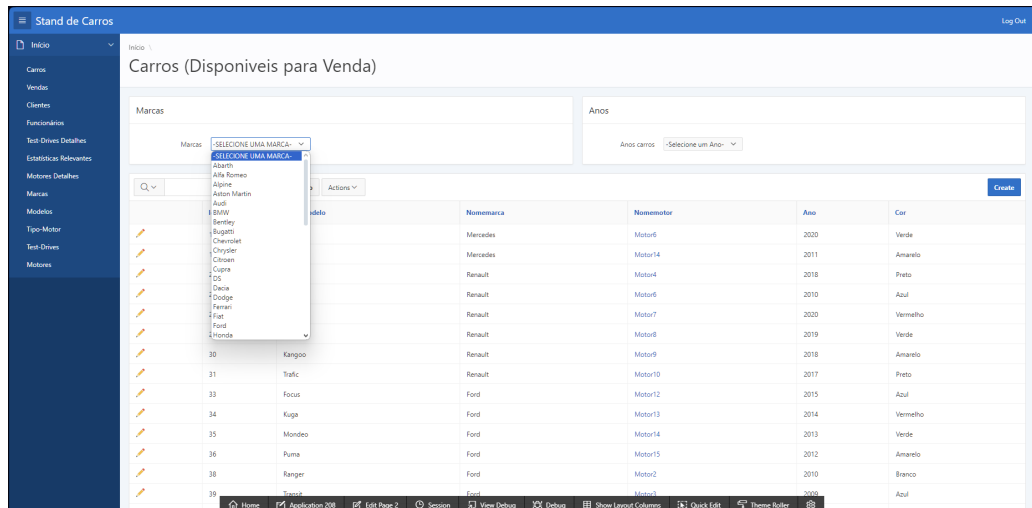


Figure 7: Exemplo do Filtro de Carros por Marca

7.2 Vendas

Nesta partição é possível consultar os dados relativos às vendas realizadas, bem como adicionar, remover e editar vendas. Na tabela estão listados os atributos da venda: *idV*, *nomeVendedor*, *nomeCliente*, *dataVenda*, *nomeModelo*, *nomeMarca* e *montante*.

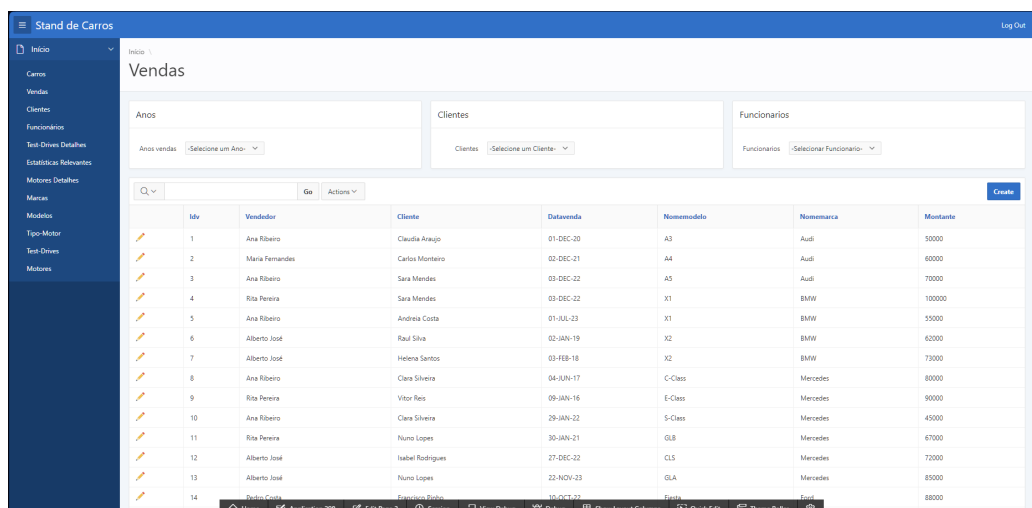


Figure 8: Página Inicial de Vendas

Para adicionar uma venda basta clicar no botão **Create** e preencher os campos necessários. Logo após o preenchimento dos dados, pressionamos o botão **Create**. Para remover uma venda basta clicar no "Lápis", da linha da venda que se pretende remover, e de seguida no botão **Delete**. Para editar uma venda basta clicar no "Lápis", da linha da venda que se pretende editar, e de seguida no botão **Apply Changes**.

Dispomos de três filtros na parte superior da página que representam a possibilidade de apresentar as vendas de um determinado ano, as vendas executadas a um determinado cliente e as vendas realizadas por um determinado funcionário. Para tal basta seleccionar o ano, o cliente ou o funcionário pretendido, respetivamente, em uma das *select lists* que se situam na parte superior da página. Para limpar os filtros basta clicar seleccionar o valor default (O primeiro valor que aparece na *select list*).

Todas estas ações são similares em operação às descritas na secção **Carros** onde podemos visualizar o funcionamento das mesmas.

7.3 Clientes

Nesta secção apresentamos apenas uma tabela com os detalhes de cada cliente no sistema, nomeadamente: NIF, Nome, Apelido, Morada, Email e nTelefone.

NIF	Nome	Apelido	Morada	Email	nTelefone
678912345	Pedro	Costa	Rua do Sol, 678, Viseu	pedrocar@gmail.com	967890123
67332345	Alberto	Jose	Rua das Flores, 765, Carvalhal	albertojose@gmail.com	105206207
890234567	Sara	Mendes	Avenida da Liberdade, 567, Cascais	sara.mendes@example.com	923456780
890234568	Paulo	Nunes	Rua da Paz, 890, Sintra	paulo.nunes@example.com	934567891
123456780	Ines	Lima	Rua da Alegria, 012, Leiria	ines.lima@example.com	945678912
234567891	Tiago	Martins	Rua do Arco, 345, Funchal	tiago.martins@example.com	956789013
345678902	Helena	Carvalho	Avenida da República, 678, Vila Real	helena.carvalho@example.com	967890124
456789013	Bruno	Almeida	Rua das Flores, 901, Évora	bruno.almeida@example.com	912345680
567890124	Marta	Barbosa	Rua do Carmo, 234, Setúbal	marta.barbosa@example.com	923456781
678901236	Ricardo	Rodrigues	Avenida dos Combatentes, 567, Guarda	ricardo.rodrigues@example.com	934567892
789012346	Clara	Silveira	Rua da Estrela, 890, Beja	clara.silveira@example.com	945678913
890123456	Joana	Fonseca	Avenida do Brasil, 123, Lagos	joana.fonseca@example.com	956789014
901234567	Miguel	Sousa	Rua de Santa Catarina, 456, Torres Vedras	miguel.sousa@example.com	967890125
234567890	Leonor	Monteiro	Rua das Pedras, 789, Odivelas	leonor.monteiro@example.com	912345681
345678901	Guilherme	Teixeira	Avenida da Igreja, 012, Maia	guilherme.teixeira@example.com	923456782
456789012	Beatriz	Magalhães	Rua do Limoeiro, 345, Peniche	beatriz.magalhaes@example.com	934567893
567890123	Vitor	Reis	Avenida da Boavista, 678, Chaves	vitor.reis@example.com	945678914
678901234					956789015

Figure 9: Página Inicial de Clientes

Aqui é também possível consultar os dados relativos aos clientes, bem como adicionar, remover e editar os mesmos. Para adicionar um Cliente basta clicar no botão **Create** e preencher os campos necessários. Para remover um cliente basta clicar no "Lápis", da linha do cliente que se pretende remover, e de seguida no botão **Delete**. Para editar uma venda basta clicar no "Lápis", editar os valores, e de seguida no botão **Apply Changes**.

Todas estas ações são similares em operação às descritas na secção **Carros** onde podemos visualizar o funcionamento das mesmas.

7.4 Funcionários

Nesta secção é possível visualizar todos os funcionários, bem como adicionar, remover e editar funcionários. Na tabela estão listados os atributos do funcionário: Nome, Apelido, Morada, Email, nTelefone e Cargo. Importante referir que na tabela de cima (visível na [imagem](#)) consta o nosso **Master-Detail**. Em que o **Master** são os funcionários e as vendas que estes realizaram são o **Detail**.

Nome	Apelido	Morada	Email	Telefone	Cargo
João	Silva	Rua das Flores, 123, Lisbon	joao.silva@example.com	912345678	Vendedor
Maria	Fernandes	Avenida dos Aliados, 456, Porto	maria.fernandes@example.com	923456789	Vendedor
Carlos	Santos	Rua do Carmo, 789, Coimbra	carlos.santos@example.com	934567890	Gerente
Ana	Ribeiro	Rua do Ouro, 012, Faro	ana.ribeiro@gmail.com	945678901	Vendedor
Rita	Pereira	Rua do Comercio, 345, Braga	rita.pereira@example.com	956789012	Vendedor
Pedro	Costa	Rua do Sol, 678, Viseu	pedro.costa@gmail.com	967890123	Vendedor
Alberto	José	Rua das Freiras, 765, Cavalhal	albertosjose@gmail.com	205096207	Vendedor
Luís	Gomes	Rua do Mercado, 234, Aveiro	luis.gomes@example.com	912345679	Gerente

NIF	Nome	Percentagem de vendas	Vendas
673332345	Alberto José	34.62	9
456789123	Ana Ribeiro	30.77	8
670965678	Rita Pereira	19.23	5

Figure 10: Página Inicial de Funcionários

Tal como nas outras secções acima descritas (podemos consultar com mais detalhe na secção [Carros](#)), é possível remover, adicionar e editar funcionários. Para adicionar um funcionário basta clicar no botão **Create** e preencher os campos necessários. Para remover um funcionário basta clicar no "Lápis", da linha do funcionário que se pretende remover, e de seguida no botão **Delete**. Para editar um funcionário basta clicar no "Lápis", editar os valores, e de seguida no botão **Apply Changes**.

No **Detail** é possível visualizar os detalhes de cada venda realizada tal como adicionar, remover ou editar vendas **realizadas por esse funcionário**.

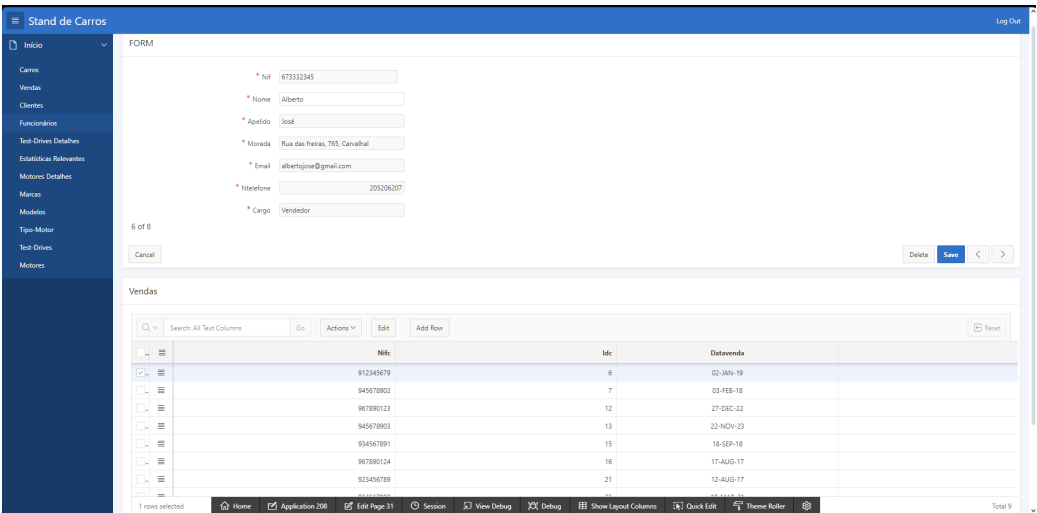


Figure 11: Detail de Funcionários

Na parte de baixo da secção, encontra-se uma tabela que mostra a percentagem de vendas realizadas por cada um dos funcionarios do stand, desde que tenham realizado pelo menos uma venda. Esta tabela é atualizada automaticamente após a realização de uma venda. Realçamos também que a percentagem de vendas representa um **Atributo Derivado**.

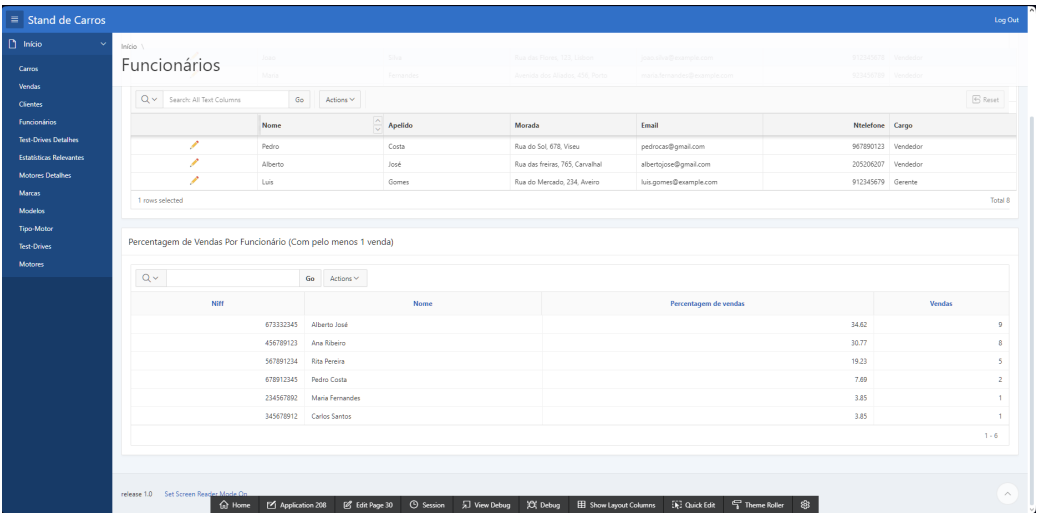


Figure 12: Percentagem de Vendas por Funcionário

7.5 Test-Drives Detalhes

Aqui encontra-se a página que permite visualizar os detalhes dos test-drives realizados. Na tabela estão listados os atributos do test-drive: `idC`, `nomeMarca`, `Data`, `Hora`, `Cliente`, `Funcionário`, `Classificação` e `nomeModelo`.

The screenshot shows the 'Test-Drives Detalhes' page. At the top, there are three filter sections: 'Marca' (with a dropdown menu), 'Ano' (with a dropdown menu), and 'Mês' (with a dropdown menu). Below these filters is a table with 13 rows of test-drive data. The table has columns for 'IdC', 'NomeMarca', 'Data', 'Hora', 'Cliente', 'Funcionario', 'Classificacao', and 'NomeModelo'. The data is as follows:

IdC	NomeMarca	Data	Hora	Cliente	Funcionario	Classificacao	NomeModelo
1	Audi	01-DEC-20	10:30:00	Joao Silva	Leonor Monteiro	5	A3
1	Audi	01-DEC-20	10:45:00	Joao Silva	Miguel Moura	5	A3
2	Audi	02-DEC-20	11:30:00	Rita Pereira	Miguel Moura	4	A4
3	Audi	03-DEC-20	12:30:00	Joao Silva	Antonio Neves	3	A5
2	Audi	01-JAN-21	09:00:00	Rita Pereira	Andreia Costa	4	A4
3	Audi	02-JAN-21	10:00:00	Joao Silva	Diogo Vieira	5	A5
4	BMW	03-JAN-21	11:00:00	Carlos Santos	Joana Fonseca	5	X1
5	BMW	04-JAN-21	12:00:00	Ana Ribeiro	Leonor Monteiro	4	X1
6	BMW	05-JAN-21	13:00:00	Rita Pereira	Fernando Oliveira	3	X2
7	BMW	06-JAN-21	14:00:00	Rita Pereira	Antonio Neves	4	X2
8	Mercedes	07-JAN-21	15:00:00	Rita Pereira	Pedro Ferreira	5	C-Class
9	Mercedes	08-JAN-21	16:00:00	Rita Pereira	Antonio Neves	5	E-Class
10	Mercedes	09-JAN-21	17:00:00	Carlos Santos	Mariana Campos	3	S-Class
11	Mercedes	10-JAN-21	09:30:00	Rita Pereira	Beatriz Magalhães	4	CLA
12	Mercedes	11-JAN-21	10:30:00	Rita Pereira	Hugo Castro	5	CLS
13	Mercedes					4	GLA

Figure 13: Página Inicial de Test-Drives

Na parte superior da página é possível filtrar os test-drives por **Marca**, **Ano** e **Mês**. Para tal basta selecionar a *select list* pretendida e clicar no valor esperado. Para limpar os filtros basta voltar ao valor default (primeiro na lista de valores possíveis).

Salientar que nesta secção não é possível adicionar, remover ou editar test-drives. Para tal temos um secção dedicada a isso [Test-Drives](#).

No fundo da página é possível consultar três **Detalhes Condicionais**, nomeadamente: test-drives da marca selecionada no mês selecionado, test-drives da marca selecionada no ano selecionado e a média mensal de test-drives da marca selecionada no ano selecionado. Para selecionar os valores mencionados podem ser utilizadas as *select lists* que se encontram no topo da página.

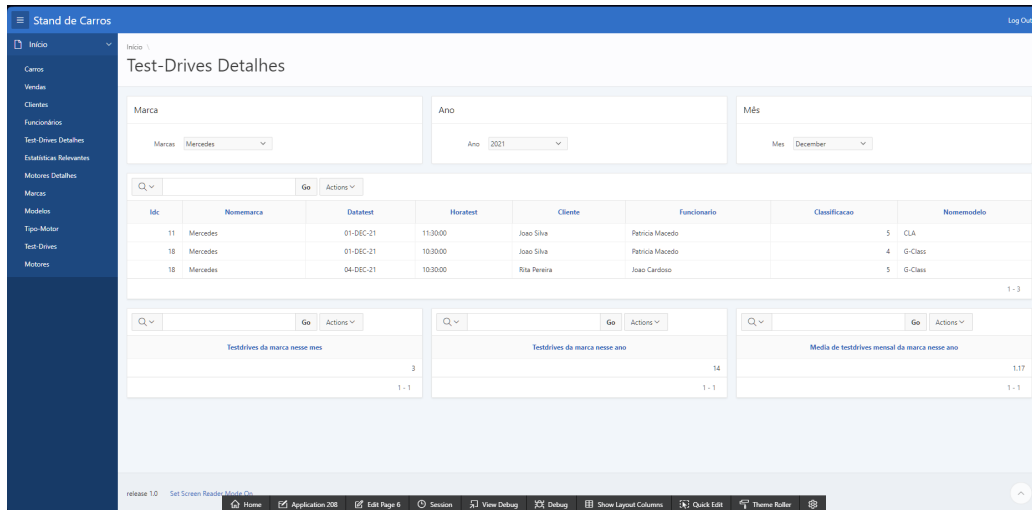


Figure 14: Detalhes Condicionais de Test-Drives Detalhes

7.6 Estatísticas Interessantes

Nesta parte da interfaces encontram se várias estatísticas interessantes sobre o stand. Estas estatísticas são atualizadas automaticamente. Correspondem à representação das consultas da secção [Consultas Importantes](#).

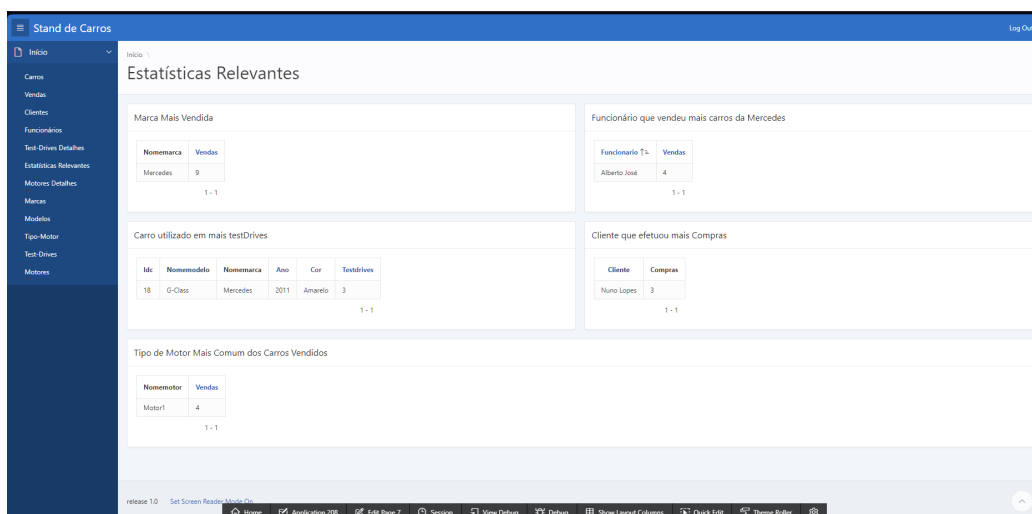


Figure 15: Estatísticas Interessantes

7.7 Motores Detalhes

Nesta página podemos encontrar as informações relevantes acerca dos motores. Na tabela estão listados os atributos do motor: **nomeMotor**, **autonomia**, **potencia**, **consumos**. De acordo com o tipo de combustão encontram-se divididos em três categorias: **Combustão**, **Elétricos** e **Híbridos**. Nas quais podemos consultar os detalhes dos motores que utilizam combustão, eletricidade e ambos, respetivamente.

NomeMotor	Autonomia	Potencia	Consumos	TipoCombustao	TamanhoDeposito
Motor1	500	200	5	Gasolina	50
Motor11	678	150	7	Gasolina	60
Motor12	660	175	8	Gasolina	58
Motor13	450	656	16	Gasolina	55
Motor14	890	120	6	Gasolina	85
Motor15	900	100	5	Gasolina	70
Motor2	600	250	6	Gasolina	60
Motor3	700	300	7	Gasolina	70
Motor7	350	450	10	Diesel	80
Motor8	553	755	13	Diesel	90

Figure 16: Página Inicial de Motores Detalhes

No topo da página é possível filtrar os motores por **Tipo de Motor** ou por um **Motor** em concreto. Para tal basta selecionar a *select list* pretendida e clicar no valor esperado. Para limpar os filtros basta voltar ao valor default (primeiro na lista de valores possíveis). Aqui apresentamos um pequeno detalhe, quando escolhemos o **Tipo de Motor**, as tabelas relativas aos restantes tipos de motores são escondidas.

NomeMotor	Autonomia	Potencia	Consumos	TipoCombustao	TamanhoDeposito	CapacidadeBateria
Motor9	1083	93	4	Diesel	100	150
Motor10	700	550	12	Gasolina	70	130

Figure 17: Exemplos de Filtro de Motores por Híbrido

Quando seleccionamos o nome de motor em concreto, é possível visualizar os detalhes do motor em questão, ignorando o outro filtro, caso esteja seleccionado. Aparece uma tabela nova apenas com os detalhes do motor seleccionado. Todas as outras tabelas são escondidas. (Para executarmos estas operações procedemos ao uso de *Dynamic Actions*).

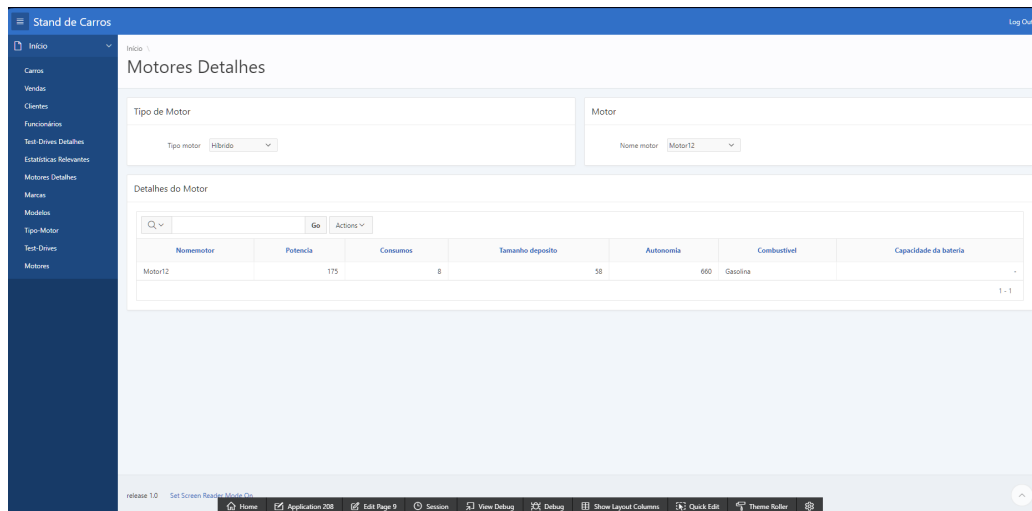
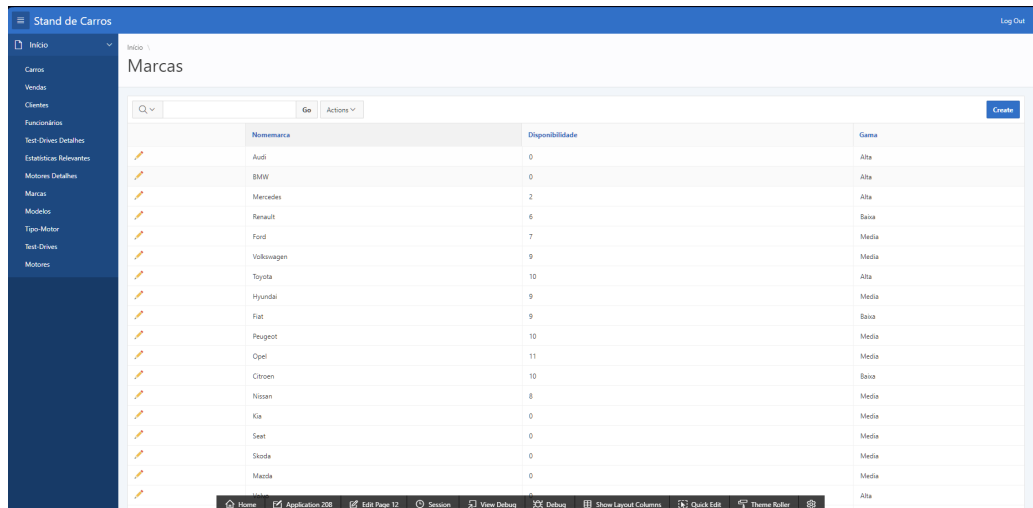


Figure 18: Exemplos de Filtro de Motores por Eletrico

Salientar que nesta secção não é possível adicionar, remover ou editar Motores. Para tal temos um secção dedicada a isso [Motores](#).

7.8 Marcas

Nesta secção é possível visualizar todos as marcas, bem como adicionar, remover e editar marcas. Na tabela estão listados os atributos da marca: **nomeMarca**, **disponibilidade**, **Gama**. Sendo a disponibilidade um atributo que atualiza automaticamente quando um **Carro** é inserido na **Marca**.



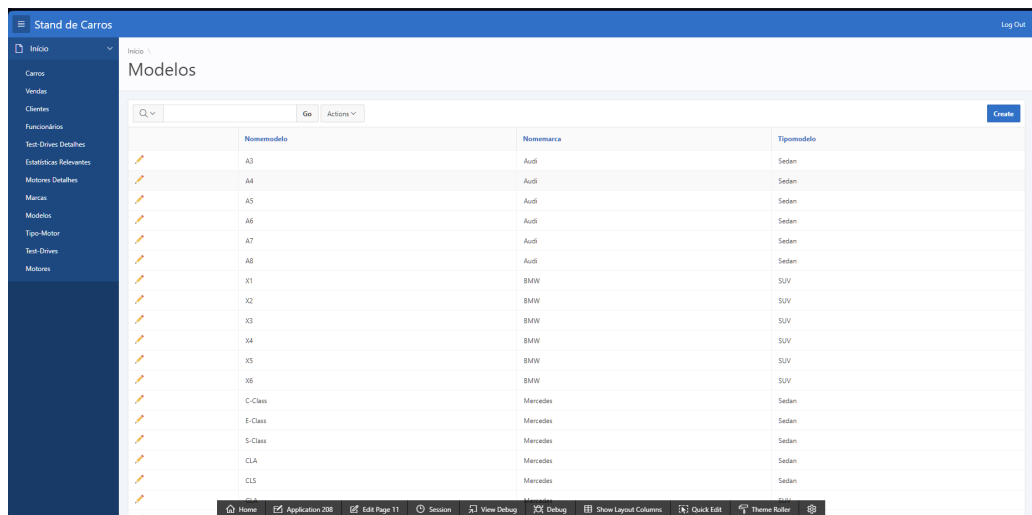
	NomeMarca	Disponibilidade	Gama
	Audi	0	Alta
	BMW	0	Alta
	Mercedes	2	Alta
	Renault	6	Baixa
	Ford	7	Media
	Volkswagen	9	Media
	Toyota	10	Alta
	Hyundai	9	Media
	Fiat	9	Baixa
	Peugeot	10	Media
	Opel	11	Media
	Citroen	10	Baixa
	Nissan	8	Media
	Kia	0	Media
	Seat	0	Media
	Skoda	0	Media
	Mazda	0	Alta

Figure 19: Página Inicial de Marcas

Tal como em outras secções acima descritas (podemos consultar com mais detalhe na secção [Carros](#)), é possível remover, adicionar e editar marcas. Para adicionar uma marca basta clicar no botão **Create** e preencher os campos necessários. Para remover uma marca basta clicar no "Lápis", da linha da marca que se pretende remover, e de seguida no botão **Delete**. Para editar uma marca basta clicar no "Lápis", editar os valores, e de seguida no botão **Apply Changes**.

7.9 Modelos

Nesta secção é possível visualizar todos os modelos, bem como adicionar, remover e editar modelos. Na tabela estão listados os atributos do modelo: `nomeModelo`, `nomeMarca`, `tipoModelo`, `idMarca`.



	NomeModelo	NomeMarca	TipoModelo
	A3	Audi	Sedan
	A4	Audi	Sedan
	A5	Audi	Sedan
	A6	Audi	Sedan
	A7	Audi	Sedan
	A8	Audi	Sedan
	X1	BMW	SUV
	X2	BMW	SUV
	X3	BMW	SUV
	X4	BMW	SUV
	X5	BMW	SUV
	X6	BMW	SUV
	C-Class	Mercedes	Sedan
	E-Class	Mercedes	Sedan
	S-Class	Mercedes	Sedan
	CLA	Mercedes	Sedan
	CLS	Mercedes	Sedan

Figure 20: Página Inicial de Modelos

Tal como em outras secções acima descritas (podemos consultar com mais detalhe na secção [Carros](#)), é possível remover, adicionar e editar modelos. Para adicionar um modelo basta clicar no botão **Create** e preencher os campos necessários. Para remover um modelo basta clicar no "Lápis", da linha do modelo que se pretende remover, e de seguida no botão **Delete**. Para editar um modelo basta clicar no "Lápis", editar os valores, e de seguida no botão **Apply Changes**.

7.10 TipoMotor

Esta secção é um bocadinho diferente das outras pois encontra-se implementada com uma *Grid*. Nesta secção é possível visualizar todas as combinações de motores com modelos existentes no sistema, bem como adicionar, remover e editar combinações possíveis. Na tabela estão listados os atributos do tipo de motor: **nomeMotor** e **nomeModelo**.

NomeMotor	NomeModelo
Motor1	5008
Motor1	718 GT4
Motor1	812 GT5
Motor1	A3
Motor1	Clio
Motor1	EcoSport
Motor1	F8 Competizione GT3
Motor1	Idriq
Motor1	Leaf
Motor1	Mosano
Motor1	Panamera GT3
Motor1	X1
Motor1	Yaris
Motor10	570S
Motor10	911 Turbo S Cabriolet
Motor10	Astra
Motor10	C4 Picasso

Figure 21: Página Inicial de TipoMotor

Esta secção como já foi referido é um bocadinho diferente das outras. Para adicionar um tipo de motor basta clicar no botão **Add Row** e preencher os campos necessários, em seguida clicar no botão **Save**. Para remover um tipo de motor basta clicar nas "Quatro Linhas Horizontais", da linha do tipo de motor que se pretende remover, e de seguida no botão **Delete Row** seguido do botão **Save**. Para editar um tipo de motor basta clicar duas vezes no campo que se pretende editar, editar os valores, e de seguida no botão **Save**.

7.11 Test-Drives

Tal como a secção anterior esta secção encontra-se implementada com uma *Grid*. Nesta secção é possível visualizar todos os test-drives, bem como adicionar, remover e editar test-drives. Na tabela estão listados os atributos do test-drive: **idC**, **Data**, **Hora**, **Nif Funcionário**, **nif Cliente** e **Classificação**.

O funcionamento desta secção é semelhante à secção [TipoMotor](#). Para adicionar um test-drive basta clicar no botão **Add Row** e preencher os campos necessários, em seguida clicar no botão **Save**. Para remover um test-drive basta clicar nas "Quatro Linhas Horizontais", da linha do test-drive que se pretende remover, e de seguida no botão **Delete Row** seguido do botão **Save**. Para editar um test-drive basta clicar duas vezes no campo que se pretende editar, editar os valores, e de seguida no botão **Save**.

	Mk	DataTest	HoraTest	NHT	NIK
1	1	01-DEC-20	1030:00	123456789	234567890
2	1	01-DEC-20	1045:00	123456789	956789013
3	2	02-DEC-20	1130:00	567891234	956789013
4	3	03-DEC-20	1230:00	123456789	956789012
5	2	01-JAN-21	0900:00	567891234	678901234
6	3	02-JAN-21	1000:00	123456789	789012345
7	4	03-JAN-21	1100:00	345678912	890123456
8	5	04-JAN-21	1200:00	456789123	234567890
9	6	05-JAN-21	1300:00	567891234	912345678
10	7	06-JAN-21	1400:00	567891234	956789012
11	8	07-JAN-21	1500:00	567891234	934567892
12	9	08-JAN-21	1600:00	567891234	956789012
13	10	09-JAN-21	1700:00	345678912	956789014
14	11	10-JAN-21	0930:00	567891234	456789012
15	12	11-JAN-21	1030:00	567891234	978901234
16	13	12-JAN-21	1130:00	567891234	989012345
17	14	13-JAN-21	1230:00	789123456	990123456
18	15	14-JAN-21	1230:00	245678913	91745680

Figure 22: Página Inicial de Test-Drives

7.12 Motores

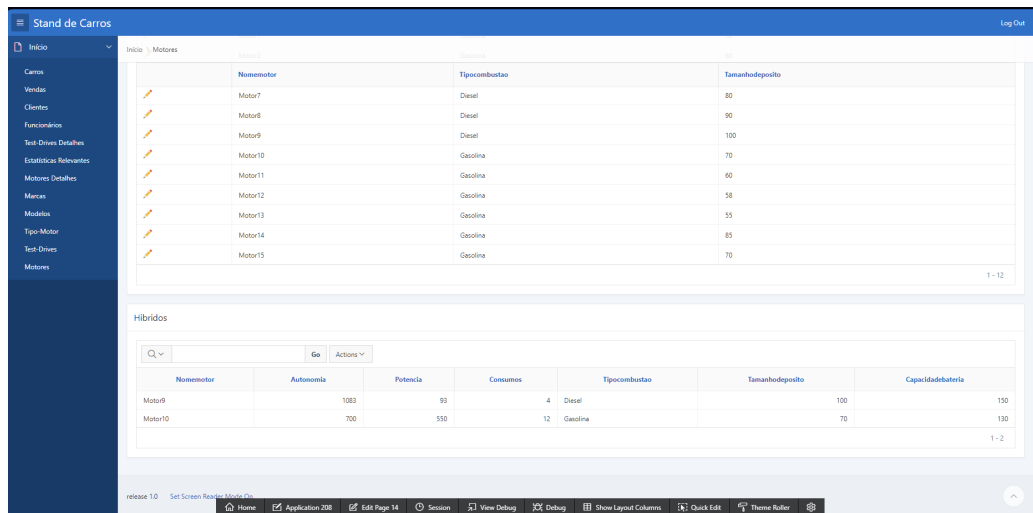
Nesta secção é possível visualizar todos os motores, bem como adicionar, remover e editar motores. Na tabela de cima estão listados os atributos do motor: **nomeMotor**, **autonomia**, **potencia**, **consumos**. Na tabela de baixo estão listados os atributos dos motores Elétricos: **nomeMotor**, **capacidadeBateria**. Em baixo da anterior encontra-se a tabela dos motores a combustão: **nomeMotor**, **tipoCombustao**, **tamanhoDeposito**. E por fim a tabela dos motores híbridos: **nomeMotor**, **autonomia**, **potencia**, **consumos**, **tipoCombustao**, **tamanhoDeposito** e **capacidadeBateria**.

	NomeMotor	Autonomia	Potencia	Consumos
1	Motor1	500	200	5
2	Motor2	600	250	6
3	Motor3	700	300	7
4	Motor4	500	612	15
5	Motor5	420	320	14
6	Motor6	600	250	10
7	Motor7	350	450	10
8	Motor8	553	755	13
9	Motor9	1083	93	4
10	Motor10	700	550	12
11	Motor11	678	150	7
12	Motor12	660	175	8
13	Motor13	450	856	16
14	Motor14	890	120	6
15	Motor15	900	100	5

Figure 23: Página Inicial de Motores

Tal como em outras secções acima descritas (podemos consultar com mais detalhe na secção [Carros](#)), é possível remover, adicionar e editar motores. Para adicionar um motor basta clicar no botão **Create** e preencher os campos necessários. Para remover um motor basta clicar no

”Lápis”, da linha do motor que se pretende remover, e de seguida no botão **Delete**. Para editar um motor basta clicar no ”Lápis”, editar os valores, e de seguida no botão **Apply Changes**. Mas o mesmo não se aplica à ultima tabela descrita de motores híbridos não pode ser editada, visto que apenas serve para visualização.



The screenshot shows a web application titled "Stand de Carros" with a sidebar menu. The main content area displays two tables. The first table, "Motores", lists various car models with their engine types and tank capacities. The second table, "Híbridos", lists hybrid car models with their autonomy, power, consumption, engine type, tank capacity, and battery capacity.

Nomemotor	Tipocombustao	Tamanhodeposito
Motor7	Diesel	80
Motor8	Diesel	90
Motor9	Diesel	100
Motor10	Gasolina	70
Motor11	Gasolina	60
Motor12	Gasolina	58
Motor13	Gasolina	55
Motor14	Gasolina	85
Motor15	Gasolina	70

Nomemotor	Autonomia	Potencia	Consumos	Tipocombustao	Tamanhodeposito	Capacidadebateria
Motor9	1083	93	4	Diesel	100	130
Motor10	700	550	12	Gasolina	70	130

Figure 24: Tabela de Motores Híbridos

8 Limitações da Base de Dados

Nesta secção vamos discutir acerca das limitações, bem como certas decisões tomadas na realização do projeto.

- Duas **Marcas** não podem ter **Modelos** com o mesmo nome.
- Não temos um sistema de pagamentos implementado, logo não existe flexibilidade na forma de pagamento.
- Um **Funcionário** apenas pode ter um cargo, pelo que excluimos a hipótese de um funcionário ter mais do que uma função no Stand.
- Não conseguimos diferenciar a **Marca** do **Modelo** da marca do **Motor**, isto leva a que não possamos ter um modelo equipado com um motor de outra marca que não a dele.

Appendices

A Mudanças Realizadas

Mudanças realizadas no projeto:

- Modificamos o [Diagrama ER](#):
 - Removemos o conjunto de entidades **Novos**, pois não era necessário, visto que podemos obter os carros novos vendo quais são os carros que não constam na tabela de usados.
 - Removemos o sistemas de pagamentos, pois para além de não ser muito relevante, não estava devidamente implementado. Além disso, por motivos de simplicidade e de tempo, decidimos remover esta entidade.
 - Alteramos o Conjunto de entidade **Modelos**, passamos de uma entidade fraca para uma entidade forte. Isto porque, decidimos que o modelo de um carro é uma entidade forte, visto que é uma entidade que tem um identificador próprio (nomeModelo).
 - Em relação aos **Test-Drives** fizemos algumas alterações pois a implementação proposta na primeira fase não estava correta. Para alcançarmos uma implementação correta, decidimos que o **Test-Drive** é uma entidade fraca, que depende de um carro, um funcionário e um cliente, uma data e uma hora. Assim, o **Test-Drive** é uma entidade fraca que depende de 3 entidades fortes e dois discriminates, permitindo identifica-los inequivocamente.
- Modificamos o [Modelo Relacional](#) para refletir as mudanças realizadas no [Diagrama ER](#).
- Alteramos também as consultas interessantes para se adequarem ao novo contexto do projeto.
- Simplificamos também a [Exposição do Tema](#), pois a apresentada na primeira fase era um pouco confusa.

B Uso de IA

Para elaboramos o projeto recorreremos ao uso de ferramentas de inteligência artificial para o preenchimento de tuplos nas tabelas, ou seja, recorreremos ao uso de **IA** para preencher as tabelas com dados fictícios. Para tal utilizamos a ferramenta **ChatGPT**. Isto permitiu-nos preencher as tabelas com dados fictícios de forma mais rápida e eficiente.