

# ***Entrega de Comida ao Domicílio***

## **Bases de Dados**

(12 dezembro 2021)

### **Trabalho Realizado por:**

2LEIC13 – Grupo 1302

Ana Beatriz Fontão – up202003574

Ana Rita Oliveira – up202004155

Matilde Sequeira – up202007623

# Índice

Contexto.....	3
Diagrama UML.....	4
Esquema Relacional.....	5
Análise de dependências funcionais e formas normais.....	6
Restrições.....	9

# Contexto

Este projeto baseia-se na gestão de uma aplicação de entrega de comida ao domicílio, semelhante à aplicação *Uber Eats*. Esta aplicação permite a um cliente fazer um pedido a um ou mais restaurantes e recebê-lo no conforto da sua casa.

Para começar, consideremos o **Utilizador**, a base da nossa aplicação. Os utilizadores podem ser **Clientes** e realizar pedidos, **Estafetas** e fazer entregas ou ambos. Estes inscrevem-se na aplicação com os seguintes dados: nome, email, NIF, telefone e **Morada** e é-lhes atribuído um id, sendo que os estafetas também têm de fornecer o seu número da carta de condução.

De modo a entregar os pedidos, os **Estafetas** utilizam **Veículos** disponibilizados pela empresa coordenadora da aplicação e caracterizados pelo seu tipo e matrícula. Os **Veículos** podem ser partilhados entre vários funcionários e um **Estafeta** apenas pode utilizar um veículo por entrega.

Cada **Cliente** pode navegar na aplicação e ir adicionando ao seu **Pedido** vários **Produtos** e/ou **Menus** de diferentes **Restaurantes**.

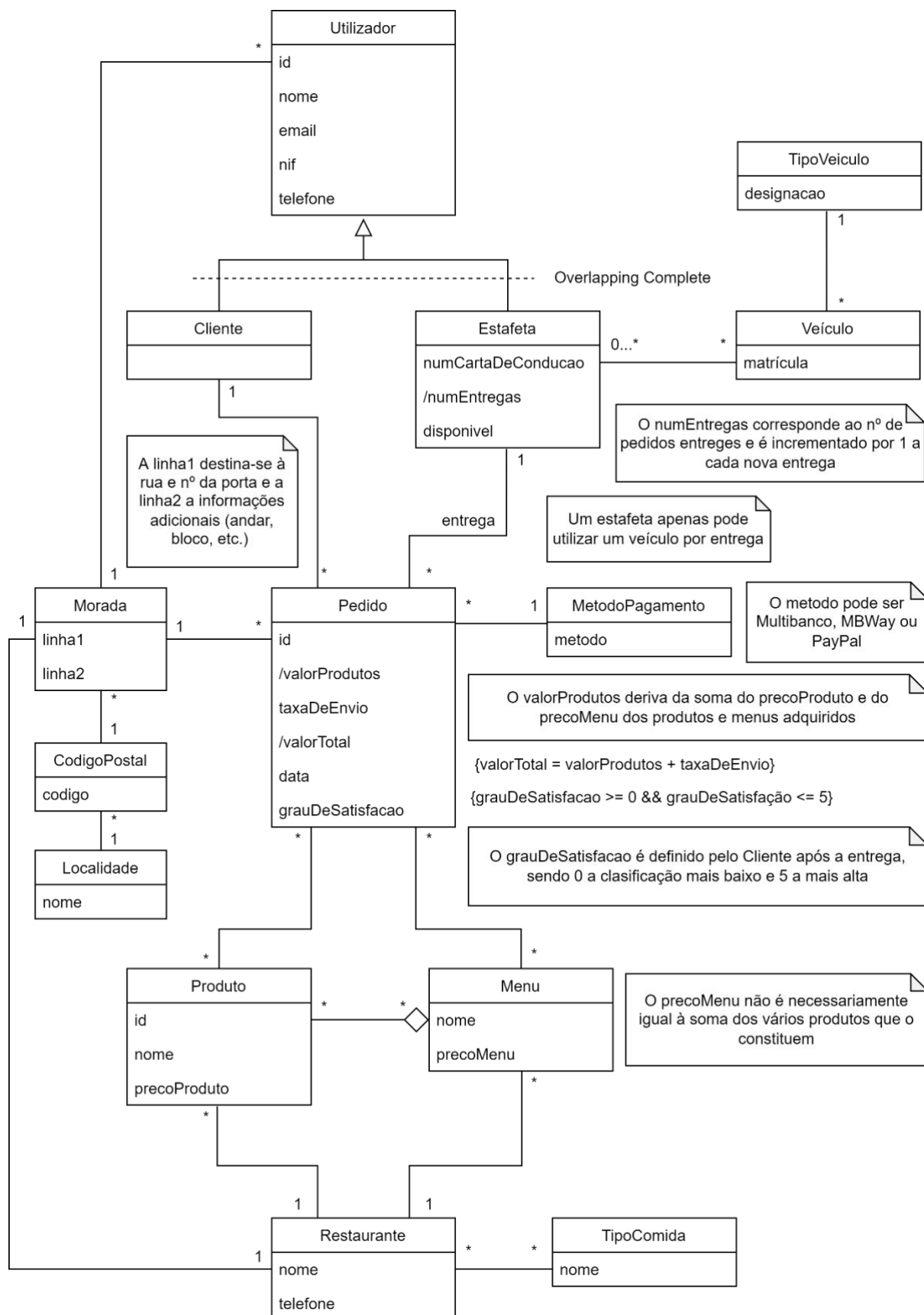
Um **Restaurante**, identificado por nome, telefone e **Morada**, pode vender diferentes **Tipos de Comida** e disponibiliza vários **Produtos** e **Menus**. Os **Menus** são compostos por diversos **Produtos**, podendo um **Produto** estar também em diferentes **Menus**. Apesar dos **Produtos** terem um valor individual, o valor dos **Menus** não tem que ser necessariamente o valor da soma dos **Produtos**, uma vez que são os **Restaurantes** que arbitram o seu preço.

O **Pedido** está sujeito a uma taxa de envio, dependente da morada escolhida para a entrega. Esta taxa irá ser adicionada ao valor dos **Produtos** e **Menus**, sendo que o valor total terá de ser pago pelo cliente através do método de **Pagamento** selecionado. O **Pedido** é então atribuído a um **Estafeta** disponível.

Ao concluir a entrega, o **Estafeta** aumenta o seu número total de entregas, número este importante para a empresa.

No final é pedido ao **Cliente** exprimir o seu grau de satisfação, de 1 a 5, quanto ao **Pedido**.

# Diagrama UML



# Esquema Relacional

**Pedido** (idPedido, valorProdutos, taxaDeEnvio, valorTotal, data, grauDeSatisfacao, idMorada -> Morada, idMetodo -> Pagamento, idEstafeta-> Estafeta, idCliente -> Cliente)

**Morada** (idMorada, linha1, linha2, idCodigo -> CodigoPostal)

**CodigoPostal** (idCodigo, codigo, idLocalidade -> Localidade)

**Localidade** (idLocalidade, localidade)

**MetodoPagamento** (idMetodo, metodo)

**TipoComida** (idTipoComida, nome)

**Restaurante** (idRestaurante, nome, telefone, idMorada -> Morada)

**TipoComidaRestaurante** (idTipoComida -> TipoComida, idRestaurante -> Restaurante)

**Produto** (idProduto, nome, precoProduto, idRestaurante -> Restaurante)

**Menu** (idMenu, nome, precoMenu, idRestaurante -> Restaurante)

**ProdutosMenu** (idProduto -> Produto, idMenu -> Menu)

**MenuPedido** (idMenu -> Menu, idPedido -> Pedido)

**ProdutoPedido** (idProduto -> Produto, idPedido -> Pedido)

**Veiculo** (matricula, idTipoVeiculo -> Tipo)

**TipoVeiculo** (idTipoVeiculo, designacao)

**Cliente** (idCliente, nome, email, nif, telefone, idMorada -> Morada)

**Estafeta** (idEstafeta, nome, email, nif, telefone, numCartaCanducacao, numEntregas, disponivel, idMorada -> Morada)

**VeiculoEstafeta** (idEstafeta -> UtilizadorEstafeta, matricula -> Veiculo)

# Análise de dependências funcionais e formas normais

**Pedido** (idPedido, valorProdutos, taxaDeEnvio, valorTotal, data, grauDeSatisfacao, idMorada -> Morada, idMetodo -> Pagamento, idEstafeta -> Estafeta, idCliente -> Cliente)

idPedido	valorProdutos	taxaDeEnvio	valorTotal	data	grau	idMorada	idMetodo	idEstafeta	idCliente
1234	7	3.15	10.15	1/1	3	666	1	789	567
4321	13	4.76	17.76	2/2	1	553	2	879	567
3241	7	3.15	10.15	1/1	5	666	1	897	657
1111	42	4.76	48.76	3/4	5	553	2	789	765

Tabela de Exemplo para a relação Pedido.

**FDs:** idPedido -> {valorProdutos, taxaDeEnvio, valorTotal, data, grauDeSatisfacao, idMorada, idMetodo, idEstafeta, idCliente}  
{valorProdutos, taxaDeEnvio} -> valorTotal  
idMorada -> taxaDeEnvio  
{idCliente, idPedido} -> grauDeSatisfacao

**Chave:** {idPedido}

**Formas Normais:** BCNF: Não / 3NF: Não

## Algoritmo de Decomposição de BCNF

1) {valorProdutos, taxaDeEnvio} -> valorTotal

2) P1 (valorProdutos, taxaDeEnvio, valorTotal)

P2 (valorProdutos, taxaDeEnvio, data, grauDeSatisfacao, idMorada, idMetodo, idEstafeta, idCliente)

3) {valorProdutos, taxaDeEnvio} -> valorTotal

Key: {valorProdutos, taxaDeEnvio}

→ Respeita BCNF

4) idPedido -> {valorProdutos, taxaDeEnvio, data, grauDeSatisfacao, idMorada, idMetodo, idEstafeta, idCliente}

idMorada -> taxaDeEnvio

{idCliente, idPedido} -> grauDeSatisfacao

Key: {idPedido}

→ Não respeita BCNF

...

\*Repetindo estes 4 passos até encontrar a uma relação que respeite BCNF, chegamos a estas 4 relações:

9) [P1] **PedidoPrecos** (valorProdutos, taxaDeEnvio, valorTotal) -> **Informação sobre preços**

[P3] **PedidoEnvio** (idMorada, taxaDeEnvio) -> **Informação sobre envio**

[P5] **PedidoSatisfacao** (idCliente, idPedido, grauDeSatisfacao) -> **Informação sobre grau de satisfação**

[P6] **Pedido** (idPedido, valorProdutos, data, idMorada, idMetodo, idEstafeta, idCliente)

-> **Informação sobre o pedido e pagamento**

*Novas Relações:*

**PedidoPreco** (valorProdutos, taxaDeEnvio, valorTotal), **PedidoEnvio** (idMorada -> Morada, taxaDeEnvio), **PedidoSatisfacao** (idCliente -> Cliente, idPedido -> Pedido, grauDeSatisfacao), **Pedido** (idPedido, valorProdutos -> PedidoPreco, data, idMorada -> PedidoEnvio, idMetodo -> MetodoPagamento, idEstafeta -> Estafeta, idCliente -> Cliente)

**Morada** (idMorada, linha1, linha2, idcodigo -> CodigoPostal)

**FDs:** idMorada -> {linha1, linha2, idCodigo}

**Chave:** {idMorada}

**Formas Normais:** BCNF: Sim / 3NF: Sim

**CodigoPostal** (idCodigo, codigo, idLocalidade -> Localidade)

**FDs:** idCodigo -> {codigo, idLocalidade}

**Chave:** {idCodigo}

**Formas Normais:** BCNF: Sim / 3NF: Sim

**Localidade** (idLocalidade, localidade)

**FDs:** idLocalidade -> localidade

**Chave:** {idLocalidade}

**Formas Normais:** BCNF: Sim / 3NF: Sim

**MetodoPagamento** (idMetodo, metodo)

**FDs:** idMetodo -> método

**Chave:** {idMetodo}

**Formas Normais:** BCNF: Sim / 3NF: Sim

**TipoComida** (idTipoComida, nome)

**FDs:** idTipoComida -> nome

**Chave:** {idTipoComida}

**Formas Normais:** BCNF: Sim / 3NF: Sim

**Restaurante** (idRestaurante, nome, telefone, idMorada -> Morada)

**FDs:** {idRestaurante} -> {nome, telefone, idMorada}

{idMorada} -> {idRestaurante}

**Chave:** {idRestaurante}, {idMorada}

**Formas Normais:** BCNF: Sim / 3NF: Sim

**TipoComidaRestaurante** (idTipoComida -> TipoComida, idRestaurante -> Restaurante)

**FDs:** -

**Chave:** {idTipoComida, idRestaurante}

**Formas Normais:** BCNF: Sim / 3NF: Sim

**Produto** (idProduto, nome, precoProduto, idRestaurante -> Restaurante)

**FDs:** {idProduto, idRestaurante} -> {nome, precoProduto}

**Chave:** {idProduto, idRestaurante}

**Formas Normais:** BCNF: Sim / 3NF: Sim

\*Assumindo que cada restaurante tem os seus próprios ids para o produto. Restaurantes diferentes podem ter produtos com o mesmo id.

**Menu** (idMenu, nome, precoMenu, idRestaurante -> Restaurante)

**FDs:** {idMenu, idRestaurante} -> {nome, precoMenu}

**Chave:** {idMenu, idRestaurante}

**Formas Normais:** BCNF: Sim / 3NF: Sim

**ProdutosMenu** (idProduto -> Produto, idMenu -> Menu)

**FDs:** -

**Chave:** {idProduto, idMenu}

**Formas Normais:** BCNF: Sim / 3NF: Sim

**MenuPedido** (idMenu -> Menu, idPedido -> Pedido)

**FDs:** -

**Chave:** {idMenu, idPedido}

**Formas Normais:** BCNF: Sim / 3NF: Sim

**ProdutoPedido** (idProduto -> Produto, idPedido -> Pedido)

**FDs:** -

**Chave:** {idProduto, idPedido}

**Formas Normais:** BCNF: Sim / 3NF: Sim

**Veiculo** (matricula, idTipoVeiculo -> Tipo)

**FDs:** matricula -> idTipoVeiculo

**Chave:** {matricula}

**Formas Normais:** BCNF: Sim / 3NF: Sim

**TipoVeiculo** (idTipoVeiculo, designacao)

**FDs:** idTipoVeiculo -> designacao

**Chave:** {idTipoVeiculo}

**Formas Normais:** BCNF: Sim / 3NF: Sim

**Cliente** (idCliente, nome, email, nif, telefone, idMorada -> Morada)

**FDs:** idCliente -> {nome, email, nif, telefone, idMorada}

nif -> {nome, idMorada, telefone, email}

**Chaves:** {idCliente}

**Formas Normais:** BCNF: Sim / 3NF: Sim

\*Assumindo que o Cliente não muda de morada, telefone e email

**Estafeta** (idEstafeta, nome, email, nif, telefone, numCartaCanducacao, numEntregas, disponivel, idMorada -> Morada)

**FDs:** idEstafeta -> {nome, email, nif, telefone, idMorada, numCartaCanducacao, numEntregas}

nif -> {nome, idUtilizador, idMorada}

**Chaves:** {idUtilizador}; {nif}

**Formas Normais:** BCNF: Sim / 3NF: Sim

**VeiculoEstafeta** (idEstafeta -> Estafeta, matricula -> Veiculo)

**FDs:** -

**Chaves:** {idUtilizador, matricula}

**Formas Normais:** BCNF: Sim / 3NF: Sim

A 3ª Forma Normal (3NF) e a Forma Normal Boyce-Codd (BCNF) têm como objetivo prevenir anomalias nos dados.

De forma a que uma relação respeite a BCNF, para cada uma das suas dependências funcionais  $\bar{A} \rightarrow \bar{B}$ , ou  $\bar{A} \rightarrow \bar{B}$  é trivial ou  $\bar{A}$  é uma (super)chave.

De forma a que uma relação respeite a 3NF, para cada uma das suas dependências funcionais não triviais  $\bar{A} \rightarrow \bar{B}$ , ou  $\bar{A}$  é uma (super)chave ou  $\bar{B}$  é apenas constituído por atributos primos (um atributo é primo se pertencer a uma chave). Uma relação sem dependências não triviais encontra-se também na 3NF.

Como podemos observar, inicialmente a relação Pedido não respeitava nenhuma das formas normais (BCNF ou 3NF). Para resolver este problema decomposemos esta relação, usando o algoritmo de decomposição de BCNF, de modo a obter novas relações que seguissem esta forma. Uma vez que as novas relações respeitam BCNF, podemos concluir que também respeitam a 3NF. De acordo com as definições apresentadas acima, todas as outras relações respeitam tanto a BCNF como a 3NF. Uma vez que omitimos as dependências funcionais triviais, as relações que não apresentam FDs também respeitam estas formas.



# Restrições

## Cliente:

- Dois Clientes diferentes não podem ter o mesmo id: **idCliente PRIMARY KEY**
- A cada Cliente tem de estar associado um nome, um email, um nif, um telefone e uma morada: **nome NOT NULL, email NOT NULL ...**
- Não podem existir dois Clientes com o mesmo email, nif ou telefone: **email UNIQUE, nif UNIQUE, telefone UNIQUE.**
- O idMorada deverá corresponder a uma morada, referenciada em Morada: **idMorada REFERENCES Morada (idMorada).**

## CodigoPostal:

- Não pode haver dois códigos postais diferentes com o mesmo id: **idCodigo PRIMARY KEY.**
- Cada código postal tem de ter associado um codigo (único), e uma localidade: **codigo NOT NULL UNIQUE, idLocalidade NOT NULL.**
- O idLocalidade deverá corresponder a uma localidade, referenciada em Localidade: **idLocalidade REFERENCES Localidade (idLocalidade).**

## Estafeta:

- Dois Estafetas diferentes não podem ter o mesmo id: **idEstafeta PRIMARY KEY**
- A cada Estafeta tem de estar associado um nome, um email, um nif, um telefone, uma morada, um numero de carta de condução, um numero de entregas e se este estafeta está disponível: **nome NOT NULL, email NOT NULL ...**
- Não podem existir dois estafetas com o mesmo email, nif, telefone ou numero de carta de condução: **email UNIQUE, nif UNIQUE, telefone UNIQUE.**
- O idMorada deverá corresponder a uma morada, referenciada em Morada: **idMorada REFERENCES Morada (idMorada).**
- Caso não tenham sido especificados os atributos numEntregas e disponivel, estes são inicializados com um valor de default: **numEntregas DEFAULT (0), disponivel DEFAULT (false).**

## Localidade:

- Não pode haver duas Localidades diferentes com o mesmo id: **idLocalidade PRIMARY KEY.**
- Cada localidade tem de ter um nome[localidade] único: **localidade NOT NULL UNIQUE.**

## Menu:

- Não pode haver dois menus diferentes com o mesmo id: **idMenu PRIMARY KEY.**
- Cada menu tem de ter associado um nome, um preço e um restaurante: **nome NOT NULL, preco NOT NULL, idRestaurante NOT NULL.**
- O preço do menu tem de ser maior do que 0: **precoMenu CHECK (precoMenu > 0)**
- O idRestaurante deverá corresponder a um restaurante, referenciado em Restaurante: **idRestaurante REFERENCES Restaurante (idRestaurante).**

## MenuPedido:

- Não pode haver dois pedidos de menus diferentes com os mesmos idMenu e idPedido: **PRIMARY KEY (idMenu, idPedido).**

- O idMenu deverá corresponder a um menu, referenciado em Menu e o idPedido deverá corresponder a um pedido, referenciado em Pedido: **idMenu REFERENCES Menu (idMenu), idPedido REFERENCES Pedido (idPedido).**

#### **Morada:**

- Duas moradas diferentes não podem ter o mesmo id: **idMorada PRIMARY KEY.**
- Cada morada tem de ter pelo menos uma linha e um código postal: **linha1 NOT NULL, idCodigo NOT NULL.**
- O idCodigo deverá corresponder a um código postal, referenciado em CodigoPostal: **idCodigo REFERENCES CodigoPostal (idCodigo).**

#### **MetodoPagamento:**

- Não pode haver dois métodos de pagamento diferentes com o mesmo id: **idMetodo PRIMARY KEY.**
- Cada método de pagamento tem de ter um único dos 3 nomes(metodo) disponíveis: **metodo NOT NULL UNIQUE CHECK (metodo = "Multibanco" OR metodo = "MBWay" OR metodo = "PayPal").**

#### **Pedido:**

- Não pode haver dois pedidos diferentes com o mesmo id: **idPedido PRIMARY KEY.**
- A cada pedido tem de estar associado um valor dos produtos (maior que 0), uma data, uma morada, um método de pagamento, um estafeta e um cliente: **valorProdutos REAL NOT NULL CHECK (valorProdutos > 0), data DATE NOT NULL, idMorada INTEGER NOT NULL...**
- Cada valorProduto, idMorada, idMetodo, idEstafeta e idCliente estão associados a um PedidoPreco, um PedidoEnvio, um MetodoPagamento, um Estafeta e a um Cliente, respetivamente. **valorProdutos REFERENCES PedidoPreco (valorProdutos), idMorada REFERENCES PedidoEnvio (idMorada)...**

#### **PedidoEnvio:**

- Cada envio de pedido (PedidoEnvio) tem de ter uma morada e uma taxa de envio (maior do que 0) associadas: **idMorada NOT NULL, taxaDeEnvio NOT NULL CHECK (taxaDeEnvio > 0).**
- O idMorada deverá corresponder a uma morada, referenciada em Morada: **idMorada REFERENCES Morada (idMorada).**

#### **PedidoPreco:**

- Cada preço de pedido (PedidoPreco) tem de ter associado um valor dos produtos, uma taxa de envio e um valor total, todos positivos, sendo que o valor total é gerado pela soma entre o valor dos produtos e a taxa de envio: **valorTotal CHECK (valorTotal > 0) NOT NULL GENERATED ALWAYS AS (valorProdutos + taxaDeEnvio).**

#### **PedidoSatisfacao:**

- Cada satisfação do pedido (PedidoSatisfacao) tem de ter associado um único pedido e um cliente: **idCliente NOT NULL, idPedido INTEGER NOT NULL UNIQUE.**
- O grau de satisfação tem de ser um número inteiro entre 1 e 5: **grauDeSatisfacao INTEGER CHECK (grauDeSatisfacao = 1 OR grauDeSatisfacao = 2 OR grauDeSatisfacao = 3 OR grauDeSatisfacao = 4 OR grauDeSatisfacao = 5).**

- Cada idCliente e idPedido deverão corresponder a um Cliente e a um Pedido, respetivamente: **idCliente REFERENCES Cliente (idCliente), idPedido REFERENCES Pedido (idPedido).**

#### **Produto:**

- Não pode haver dois produtos diferentes com o mesmo id: **idProdutos PRIMARY KEY.**
- Cada produto tem de ter associado um nome, um preço e um restaurante: **nome NOT NULL, precoProduto NOT NULL, idRestaurante NOT NULL.**
- O preço do produto tem de ser maior do que 0: **precoProduto CHECK (precoProduto > 0)**
- O idRestaurante deverá corresponder a um restaurante, referenciado em Restaurante: **idRestaurante REFERENCES Restaurante (idRestaurante).**

#### **ProdutoPedido:**

- Não pode haver dois pedidos de produtos (ProdutoPedido) diferentes com os mesmos idProduto e idPedido: **PRIMARY KEY (idProduto, idPedido)**
- O idProduto deverá corresponder a um produto, referenciado em Produto e o idPedido deverá corresponder a um pedido, referenciado em Pedido: **idPedido REFERENCES Pedido (idPedido), idProduto REFERENCES Produto (idProduto).**

#### **ProdutosMenu:**

- Não pode haver dois produtos de menu (ProdutosMenu) diferentes com o mesmo idProduto e idMenu: **PRIMARY KEY (idProduto, idMenu)**
- O idProduto deverá corresponder a um produto, referenciado em Produto e o idMenu deverá corresponder a um menu, referenciado em Menu: **idProduto REFERENCES Produto (idProduto), idMenu REFERENCES Menu (idMenu).**

#### **Restaurante:**

- Não pode haver dois restaurantes diferentes com o mesmo id: **idRestaurante PRIMARY KEY.**
- Cada restaurante tem de ter um nome, um telefone e uma morada, sendo que o telefone e a morada são únicos: **nome NOT NULL, telefone NOT NULL UNIQUE, idMorada NOT NULL UNIQUE.**
- O idMorada deverá corresponder a uma morada, referenciada em Morada: **idMorada REFERENCES Morada(idMorada).**

#### **TipoComida:**

- Não pode haver dois tipos de comida diferentes com o mesmo id: **idTipoComida PRIMARY KEY.**
- Cada tipo de comida tem de ter um nome único: **nome NOT NULL UNIQUE.**

#### **TipoComidaRestaurante:**

- Não pode haver dois tipos diferentes de comida num restaurante (TipoComidaRestaurante) com os mesmos idTipoComida e idRestaurante: **PRIMARY KEY (idTipoComida, idRestaurante)**
- O idTipoComida deverá corresponder a um tipo de comida, referenciado em TipoComida e o idRestaurante deverá corresponder a um restaurante, referenciado em Restaurante: **idTipoComida REFERENCES TipoComida (idTipoComida), idRestaurante REFERENCES Restaurante (idRestaurante)**

**TipoVeiculo:**

- Não pode haver dois tipos de veículo diferentes com o mesmo id: ***idTipoVeiculo PRIMARY KEY.***
- Um tipo de veículo tem de ter uma designação única que respeite um dos dois valores: ***designacao NOT NULL CHECK (designacao = "Carro" OR designacao = "Mota") UNIQUE.***

**Veiculo:**

- Não pode haver dois veículos diferentes com a mesma matrícula: ***matricula PRIMARY KEY.***
- Cada veículo tem de ter associado um tipo de veículo, referenciado em TipoVeiculo: ***idTipoVeiculo NOT NULL, idTipoVeiculo REFERENCES TipoVeiculo (idTipoVeiculo).***

**VeiculoEstafeta:**

- Não pode haver dois veículos a ser usados por um estafeta (VeiculoEstafeta) diferentes com a mesma matrícula e idEstafeta: ***PRIMARY KEY (matricula, idEstafeta)***
- O idEstafeta deverá corresponder a um estafeta, referenciado em Estafeta e a matrícula deverá corresponder a um veículo, referenciado em Veiculo: ***idEstafeta REFERENCES Estafeta (idEstafeta), matricula REFERENCES Veiculo (matricula).***