

**Base de Dados**

EIC0023

Ano Letivo 2016/2017

**Mestrado Integrado em Engenharia Informática e Computação**

2º Ano

**CADEIA DE RESTAURANTES *GOURMET***

**Turma 5 - Grupo 3**

André Cruz - up201503776@fe.up.pt

Edgar Carneiro - up201503784@fe.up.pt

Mateus Pedroza - up201601876@fe.up.pt

# Índice

[Índice 2](#_Toc483500244)

[Contexto 4](#_Toc483500245)

[Principais conceitos 4](#_Toc483500246)

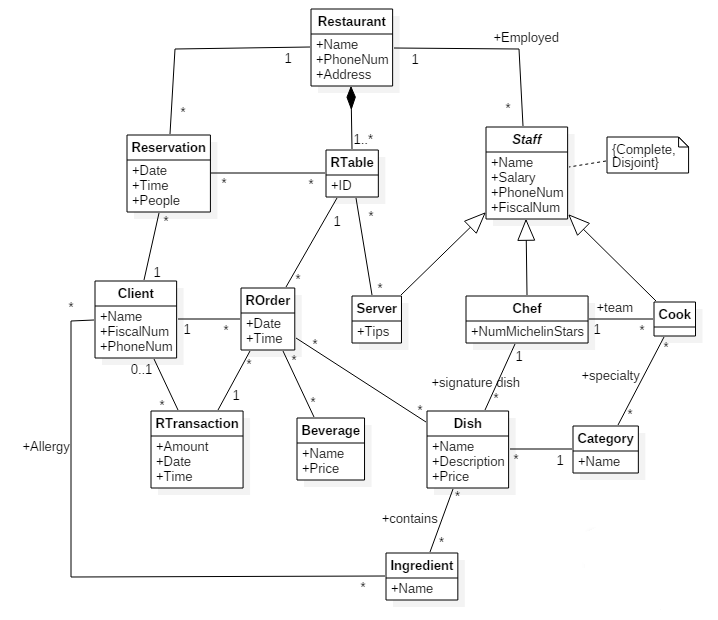
[Modelo Relacional e Dependências Funcionais 6](#_Toc483500247)

[Restrições 11](#_Toc483500248)

[Interrogações 16](#_Toc483500249)

[Gatilhos 17](#_Toc483500250)

[UML 18](#_Toc483500251)

[ 18](#_Toc483500252)

# 

# Contexto

O nosso projeto é uma solução para armazenar informações relativas a uma cadeia de restaurantes *gourmet*. Assim sendo, é necessário ter em consideração os funcionários, as transações comerciais, os pedidos e os pratos e bebidas disponíveis, bem como os clientes e suas particularidades (nomeadamente, alergias). É também essencial guardar informação relativa a reservas, de modo a que seja possível averiguar a disponibilidade de cada restaurante em qualquer data.

# Principais conceitos

Cada restaurante tem um nome distinto, bem como número de telefone e, logicamente, morada. Os funcionários (*Staff*) do restaurante dividem-se em várias categorias (*Chef*, *Cook*, *Server*), cada qual com funções distintas. Todos os pratos servidos no restaurante são de autor (*signature dish*), sendo essencial para os clientes saber qual o *Chef* responsável pelo seu prato. Cada restaurante pode ter mais de um *Chef*, sendo cada um auxiliado pela sua equipa de cozinheiros (*Cook*) altamente qualificados.

Relativamente aos funcionários, apesar de terem funções variadas e, portanto, se relacionarem de maneira diferente com os restantes elementos do restaurante, é necessário armazenar o nome e contacto de cada um, bem como o salário e o número de identificação fiscal. Cada empregado de mesa (*Server*) recebe também gorjetas (*Tips*) e fica responsável pelo atendimento de determinadas mesas.

Adicionalmente, para refletir a reputação do *Cheff*, é necessário saber o número de Estrelas *Michelin* obtidas por cada um. Cada cozinheiro (*Cook*) especializa-se também numa determinada categoria de pratos (entradas, sobremesas, etc.).

No que diz respeito a um cliente (*Client*) é necessário saber o seu nome, contacto e número fiscal (NIF), bem como os ingredientes a que é alérgico para que seja possível detectar pratos perigosos para consumo.

Relativamente às reservas, é necessário saber a data da reserva, a hora e o número de pessoas. Assim, uma reserva ficará associada a uma ou várias mesas, dependendo do número de pessoas, tal como ao cliente que a realiza. A obrigação de uma reserva estar associada a pelo menos uma mesa garante que o restaurante tem vagas à hora requerida.

Quando um grupo de clientes chega a um restaurante, e caso haja disponibilidade, é-lhes indicada uma mesa e um empregado de mesa responsável recolhe os seus pedidos (*Order*). Quando um cliente faz o seu pedido é registado o(s) prato(s) (*Dish*) e bebida(s) (*Beverage*), bem como momento em que o pedido aconteceu. No final, é também registado o pagamento, o valor e o momento em que aconteceu. Mais tarde, os dados acerca do tempo passado entre o registo do pedido e a saída de um cliente, bem como a sua possível relação com o prato pedido, podem ser usados para efeitos estatísticos. Se o cliente quiser pode ainda associar o seu número fiscal à transação.

Em relação às refeições, cada bebida tem um nome e um preço. Cada prato tem um nome, descrição, preço e categoria (*Category*) associados, bem como ingredientes usados na sua confecção, para a segurança de alergénios. Cada categoria, por sua vez, tem um nome e, como referido anteriormente, está associada a diferentes cozinheiros, que se especializam nesta mesma.

# 

# 

# Modelo Relacional e Dependências Funcionais

* Restaurant (ID, PhoneNum, Name, Adress)

Dependências Funcionais:

ID -> PhoneNum, Name, Adress;

Name -> PhoneNum, Adress, ID;

Adress -> PhoneNum, Name, ID;

PhoneNum -> Name, Adress, ID;

Nenhuma das dependências funcionais relativas a esta relação viola a Forma Normal Boyce-Codd, pois todas as dependências funcionais têm uma super key do seu lado esquerdo.

* Reservation(ID, Date, Time, People, restaurant -> Restaurant, client-> Client)

Dependências Funcionais:

ID -> Date, Time, People, restaurant, client;

client, Date, Time, restaurant -> People, ID;

Nenhuma das dependências funcionais relativas a esta relação viola a Forma Normal Boyce-Codd, pois todas as dependências funcionais têm uma super key do seu lado esquerdo.

* RTable (ID, restaurant-> Restaurant)
* Client (FiscalNum, Name, PhoneNum)

Dependências Funcionais:

FiscalNum -> Name, PhoneNum;

Nenhuma das dependências funcionais relativas a esta relação viola a Forma Normal Boyce-Codd, pois todas as dependências funcionais têm uma super key do seu lado esquerdo.

* ROrder (ID, Date, Time, TableID -> RTable.ID, TableRes -> RTable.restaurant, client -> Client, transaction-> RTransaction)

Dependências Funcionais:

ID -> Date, Time, TableID, TableRes, client, transaction;

Date, Time, client -> transaction, TableID, TableRes, ID;

Nenhuma das dependências funcionais relativas a esta relação viola a Forma Normal Boyce-Codd, pois todas as dependências funcionais têm uma super key do seu lado esquerdo.

* Staff (FiscalNum, Name, Salary, PhoneNum, restaurant -> Restaurant)

Dependências Funcionais:

FiscalNum -> Name, Salary, PhoneNum , restaurant;

PhoneNum -> FiscalNum, Name, Salary, restaurant;

Nenhuma das dependências funcionais relativas a esta relação viola a Forma Normal Boyce-Codd, pois todas as dependências funcionais têm uma super key do seu lado esquerdo.

* Server (FiscalNum -> Staff, Tips)

Dependências Funcionais:

FiscalNum -> Tips;

Nenhuma das dependências funcionais relativas a esta relação viola a Forma Normal Boyce-Codd, pois todas as dependências funcionais têm uma super key do seu lado esquerdo.

* Chef (FiscalNum -> Staff, numMichellinStars)

Dependências Funcionais:

FiscalNum -> numMichellinStars;

Nenhuma das dependências funcionais relativas a esta relação viola a Forma Normal Boyce-Codd, pois todas as dependências funcionais têm uma super key do seu lado esquerdo.

* Cook (FiscalNum -> Staff, teamChef -> Chef)

Dependências Funcionais:

FiscalNum -> teamChef;

Nenhuma das dependências funcionais relativas a esta relação viola a Forma Normal Boyce-Codd, pois todas as dependências funcionais têm uma super key do seu lado esquerdo.

* RTransaction (ID, Amount, FiscalNum -> Client, Date, Time)

Dependências Funcionais:

ID -> Amount, FiscalNum, Date, Time;

FiscalNum , Date, Time -> Amount, ID;

Nenhuma das dependências funcionais relativas a esta relação viola a Forma Normal Boyce-Codd, pois todas as dependências funcionais têm uma super key do seu lado esquerdo.

* Beverage (ID, Name, Price)

Dependências Funcionais:

ID -> Name, Price;

Name -> ID, Price;

Nenhuma das dependências funcionais relativas a esta relação viola a Forma Normal Boyce-Codd, pois todas as dependências funcionais têm uma super key do seu lado esquerdo.

* Dish (ID, Name, Description, Price, category -> Category, author -> Chef)

Dependências Funcionais:

ID -> Name, Description, Price, category, author;

Name, author -> ID, description, price, category;

Nenhuma das dependências funcionais relativas a esta relação viola a Forma Normal Boyce-Codd, pois todas as dependências funcionais têm uma super key do seu lado esquerdo.

* Category (ID, Name)

Dependências Funcionais:

ID -> Name;

Name -> ID;

Nenhuma das dependências funcionais relativas a esta relação viola a Forma Normal Boyce-Codd, pois todas as dependências funcionais têm uma super key do seu lado esquerdo.

* Ingredient (ID, Name)

Dependências Funcionais:

ID -> Name;

Name -> ID;

Nenhuma das dependências funcionais relativas a esta relação viola a Forma Normal Boyce-Codd, pois todas as dependências funcionais têm uma super key do seu lado esquerdo.

* Contains (dish -> Dish, ingredient -> Ingredient)
* ReservedTable (reservation -> Reservation , TableID -> RTable.ID, TableRes -> RTable.restaurant,)
* Responsible (TableID -> RTable.ID, TableRes -> RTable.restaurant, server -> Server)
* OrderBeverage (order -> ROrder, beverage -> Beverage)
* OrderDish (order -> ROrder, dish -> Dish)
* Allergy ( client -> Client, ingredient -> Ingredient)

* Specialty (cook -> Cook, category-> Dish)

Por motivos de legibilidade do relatório, as dependências funcionais explicitadas são apenas as não triviais.

# Restrições

* Os números fiscais presentes nas várias classes devem todos ter exatamente 9 dígitos (restrição *CHECK*).
* Os números de telemóvel presentes nas várias classes devem todos ter exatamente 9 dígitos (restrição *CHECK*).
* Todos os restaurantes têm de ter um ID, um número de telemóvel, um endereço e um nome (restrição *NOT NULL*).
* Não podem existir dois restaurantes com o mesmo ID (restrição PRIMARY KEY).
* Não podem existir dois restaurantes com o mesmo nome (restrição *UNIQUE*).
* Não podem existir dois restaurantes com o mesmo número de telefone (restrição *UNIQUE*).
* Não podem existir dois restaurantes com o mesmo endereço (restrição *UNIQUE*).
* Todas as reservas têm de ter um ID, uma data, um momento (hora e minuto), um número de pessoas associadas, o restaurante ao qual a reserva se refere e o cliente que fez essa reserva (restrição *NOT NULL*).
* Não podem existir duas reservas com o mesmo ID (restrição *PRIMARY KEY*).
* Não podem existir duas reservas do mesmo cliente, para o mesmo restaurante, na mesma data, no mesmo momento (restrição *UNIQUE).*
* O ID de Restaurante associado a uma Reserva tem como função representar o Restaurante ao qual esta se destina (restrição de integridade referencial).
* O número fiscal associado a uma Reserva tem como função representar o cliente detentor desse número (restrição de integridade referencial).
* O número mínimo de pessoas para a realização de uma Reserva é uma (restrição *CHECK*).
* Não podem existir duas mesas com igual ID, no mesmo Restaurante associado (restrição *PRIMARY K*E*Y)*.
* O ID de Restaurante associado a uma Mesa tem como função representar o Restaurante ao qual esta pertence (restrição de integridade referencial).
* Todas as mesas têm de ter um ID e um restaurante associado, ao qual pertencem (restrição *NOT NULL*).
* Não podem existir dois clientes com um número fiscal igual (restrição *PRIMARY KEY*).
* Todos os clientes têm de ter um nome, um número fiscal e um número de telefone (restrição *NOT NULL*).
* Não podem existir dois pedidos com o mesmo ID (restrição *PRIMARY KEY*).
* Todos os pedidos têm de ter um ID, uma data, um momento (hora e minuto), uma ID de mesa associado, um Restaurante associado, um cliente associado e um ID de Transação associado (restrição *NOT NULL*).
* Não podem existir dois pedidos do mesmo cliente, na mesma data, no mesmo momento (restrição *UNIQUE*).
* O ID de Mesa associado a um Pedido, bem como o Restaurante a este associado, têm como função representar a Mesa ao qual esta se destina (restrição de integridade referencial).
* O número fiscal associado a um Pedido tem como função representar o cliente detentor desse número (restrição de integridade referencial).
* O ID de Transação associado a um Pedido tem como função representar a Transação que se lhe encontra associada (restrição de integridade referencial).
* Todos os elementos do staff têm de ter um número fiscal, um número de telefone, um nome, um salário e um restaurante associado (restrição *NOT NULL*).
* Não podem existir dois elementos do staff com números fiscais iguais (restrição *PRIMARY KEY*).
* Não podem existir dois elementos do staff com números de telemóvel iguais (restrição UNIQUE).
* O ID de Restaurante associado a um elemento do Staff tem como função representar o Restaurante ao qual este pertence (restrição de integridade referencial).
* Todos os empregados de mesa têm de ter um número fiscal associado, que os identifica com um elemento do staff, e gorjetas, cujo valor por defeito é 0 (restrição *NOT NULL*).
* O número fiscal associado a um Empregado de Mesa tem como função identificar a informação do *Staff* que o representa (restrição de integridade referencial).
* Todos os chefes têm de ter um número fiscal associado, que os identifica com um elemento do staff, e um número de Estrelas *Michellin* ganhas, cujo valor por defeito é 0 (restrição *NOT NULL*).
* O número fiscal associado a um Chefe tem como função representar a informação do staff que o representa (restrição de integridade referencial).
* Todos os cozinheiros têm de ter um número fiscal associado, que os identifica com um elemento do staff, e um chefe de equipa (restrição *NOT NULL*).
* O número fiscal associado a um Cozinheiro tem como função representar a informação do staff que o representa (restrição de integridade referencial).
* O número fiscal do Chefe associado a um Cozinheiro tem como função representar o Chefe que lidera a sua equipa (restrição de integridade referencial).
* Não podem existir duas transações com igual ID (restrição PRIMARY KEY).
* Não podem existir duas transações com a mesma data, o mesmo momento (hora e minuto) e o mesmo cliente associado (restrição *UNIQUE*).
* O número fiscal associado a uma Transação tem como função representar o cliente detentor que realizou a transação, se este assim o quiser (restrição de integridade referencial).
* Todas as Transações têm de ter uma data, um momento (hora e minuto), um montante e um ID (restrição *NOT NULL*).
* Todas as bebidas têm de ter um nome, um ID e um preço (restrição *NOT NULL*).
* Não podem existir duas bebidas com igual ID (restrição *PRIMARY KEY*).
* Não podem existir duas bebidas com nome igual (restrição *UNIQUE*).
* Todos os pratos têm de ter um ID, um nome, uma descrição, um preço, uma categoria associada e um autor associado (restrição *NOT NULL*).
* Não podem existir dois pratos com igual ID (restrição *PRIMARY KEY*).
* Não podem existir dois pratos com igual nome e igual autor (restrição *UNIQUE*).
* O ID de Categoria associado a um Prato tem como função representar a categoria ao qual esse prato pertence (restrição de integridade referencial).
* O número fiscal associado a um Prato tem como função representar o Chef que o confeccionou (restrição de integridade referencial).
* Todas as categorias têm de ter um ID e um nome (restrição *NOT NULL*).
* Não podem existir duas categorias com igual ID (restrição *PRIMARY KEY*).
* Não podem existir duas categorias com igual nome (restrição *UNIQUE*).
* Todos os ingredientes têm de ter um ID e um nome (restrição *NOT NULL*).
* Não podem existir dois ingredientes com igual ID (restrição *PRIMARY KEY*).
* Não podem existir dois ingredientes com igual nome (restrição *UNIQUE*).
* Não podem existir duas relações de Pertença cujo prato e ingrediente em causa sejam iguais (restrição *PRIMARY KEY*).
* O ID de Prato associado a uma relação de Pertença tem como função representar o prato ao qual esta se refere (restrição de integridade referencial).
* O ID de Ingrediente associado a uma relação de Pertença tem como função representar o ingrediente ao qual esta se refere (restrição de integridade referencial).
* Não podem existir duas relações de Reserva de Mesa cuja reserva e mesa associada em causa sejam iguais (restrição *PRIMARY KEY*).
* O ID de Reserva associado a uma relação de Reserva de Mesa tem como função representar a reserva ao qual esta se refere (restrição de integridade referencial).
* O ID de Mesa associado a uma relação de Reserva de Mesa, bem como o Restaurante a esta associada, têm como função representar a mesa que foi reservada (restrição de integridade referencial).
* Não podem existir duas relações de Responsabilidade cujo empregado de mesa e mesa em causa sejam iguais (restrição *PRIMARY KEY*).
* O ID de Mesa associado a uma relação de Responsabilidade, bem como o Restaurante a esta associada, tem como função representar a mesa pela qual o empregado de mesa é responsável (restrição de integridade referencial).
* O número fiscal associado a uma relação de Responsabilidade tem como função representar o empregado de mesa responsável pela mesa (restrição de integridade referencial).
* Não podem existir duas relações de Pedido de Bebida cujo pedido e bebida em causa sejam iguais (restrição *PRIMARY KEY*).
* O ID de Pedido associado a uma relação de Pedido de Bebida tem como função representar o pedido ao qual esta se refere (restrição de integridade referencial).
* O ID de Bebida associado a uma relação de Pedido de Bebida tem como função representar a bebida que terá sido pedida (restrição de integridade referencial).
* Não podem existir duas relações de Pedido de Prato cujo pedido e prato em causa sejam iguais (restrição *PRIMARY KEY*).
* O ID de Pedido associado a uma relação de Pedido de Prato tem como função representar o pedido ao qual esta se refere (restrição de integridade referencial).
* O ID de Prato associado a uma relação de Pedido de Prato tem como função representar prato que terá sido pedido (restrição de integridade referencial).
* Não podem existir duas relações de Alergia cujo cliente e ingrediente em causa sejam iguais (restrição *PRIMARY KEY*).
* O número fiscal associado a uma relação de Alergia tem como função representar o cliente que é alérgico (restrição de integridade referencial).
* O ID de Ingrediente associado a uma relação de Alergia tem como função representar o ingrediente ao qual o cliente é alérgico (restrição de integridade referencial).
* Não podem existir duas relações de Especialização cujo cozinheiro e categoria em causa sejam iguais (restrição *PRIMARY KEY*).
* O número fiscal associado a uma relação de Especialização tem como função representar o cozinheiro ao qual esta se refere (restrição de integridade referencial).
* O ID de Categoria associado a uma relação de Especialização tem como função representar a categoria em que o cozinheiro é especialista (restrição de integridade referencial).

# Interrogações

* *Average Stay Time*: mostra o tempo médio de estadia em cada Restaurante, em minutos.
* *Beverage Most Ordered by Restaurant*: mostra a bebida com mais pedidos em cada Restaurante.
* *Client Allergic to Dishes*: mostra o os pratos aos quais os clientes são alérgicos.
* *Clients Total Transaction*: mostra o montante total transacionado por cada cliente.
* *Cooks specialties*: mostra os pratos que são especialidade de cada Cozinheiro.
* *Dish Most Ordered by Restaurant*: mostra a bebida com mais pedidos em cada Restaurante.
* *Most common table for client*: mostra a mesa mais usada por cada cliente, de entre todos os restaurantes, tal como a sua frequência.
* *Restaurant Expense*: mostra a quantidade monetária gasta em salários, por Restaurante.
* *Restaurant Transaction Total*: mostra o valor total de transações de cada Restaurante.
* *Servers tips Rank*: mostra os empregados de mesa ordenados pelo valor recebido em gorjetas.
* *Clients Allergenic Chef*: mostra os chefes que confecionam pratos contendo ingredientes ao qual o cliente é alérgico, por cliente.
* *Server Average Tip for Client*: mostra o valor médio que cada empregado de mesa recebe por gorjeta, por Cliente que atende.
* *Client Average Amount Spent*: mostra o valor médio que cada Cliente gasta em cada Restaurante.
* *Chef Average Description Size*: mostra o tamanho médio das Descrições dos pratos de cada Chef.
* *ChefView & CookView & ServerView*: interrogações auxiliares no uso dos Triggers.
* *Restaurant most ordered Beverage & Dish*: mostra a bebida e o prato com mais pedidos em cada Restaurante.

# Gatilhos

1. Gatilho que verifica que sempre que se acrescenta um elemento do Staff, este apenas faz parte de uma das especializações: Chefe, Cozinheiro ou Empregado de Mesa (Complete Generalization). Este Gatilho garante também que não podem existir elementos do Staff que não são cozinheiros, chefes ou empregados de mesa (Disjoint Generalization).
2. Gatilho que garante que para se associar um prato a um pedido, o Cliente que fez o pedido, não é a alérgico a nenhum ingrediente constituinte do prato.
3. Gatilho que garante que não se pode associar uma Transição a um Pedido se a Transição tiver ocorrido num momento anterior ao Pedido.

# UML

# ../../Desktop/Screen%20Shot%202017-05-25%20at%2022.41.50.pn