



ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

Unidade Curricular de Base de Dados

Ano Letivo de 2021/2022

SGBD Michelin Star - Restaurante

Grupo n.º 17

Diogo Bragança 8200547

Luís Oliveira 8190370

10 de Junho, 2022

Data de Receção	
Responsável	
Avaliação	
Observações	

Diogo Bragança 8200547

Luís Oliveira 8190370

10 de Junho, 2022

Resumo

Este relatório descreve as várias fases da criação e implementação dessa base de dados, tais como o planeamento, a descrição dos objetivos e limites da aplicação, identificação de requisitos, desenho conceptual, desenho lógico para o modelo relacional e a tradução desse modelo lógico global para o SGBD.

Este projeto é relativo à componente prática da Unidade Curricular de Base de Dados.

Índice

Resumo	iii
Índice	iv
Índice de Figuras	vi
Índice de Tabelas	vii
Índice Excertos de SQL.....	viii
1. Introdução.....	1
1.1 Contextualização	1
1.2 Apresentação do Caso de Estudo.....	1
1.3 Motivação e Objetivos	1
1.4 Estrutura do Relatório.....	1
2 Requisitos Gerais e Use Cases	2
2.1 Requisitos Gerais.....	2
3 Desenho Conceptual	3
3.1 Identificação das entidades	4
3.2 Relações entre Entidades	5
3.3 Multiplicidade	6
3.4 Atributos para as Entidades	6
3.5 Documentação de atributos.....	7
3.6. Atribuição de chaves primarias	10
4 Desenho Lógico	12
4.1 Mockups.....	13
4.2 Derivação de relações para o modelo de dados lógico	15
4.2.1 Relação de um para muitos (1: *)	15
4.2.2 Relação de um para um (1 : 0...1).....	16
4.2.3 Relação de muitos para muitos (* : *)	16
4.3 Normalização	18
4.3.1 Primeira Forma Normal (1FN)	18
4.3.2 Segunda Forma Normal (2FN)	18
4.3.3 Terceira Forma Normal (3FN).....	18
4.3.4 Normalização das tabelas.....	19

4.3.5 Desenho Logico	22
5 Restrições de integridade – Regras de negócio	23
6 Desenho Físico	25
6.1 Criação da Tabelas, relacionamentos e restrições aos atributos	27
6.1.1 Criação da Tabela Ingrediente.....	27
6.1.2 Criação da Tabela IngredientePrato	28
6.1.3 Criação da Tabela Prato	29
6.1.4 Criação da Tabela PratoEmenta.....	30
6.1.5 Criação da Tabela Ementa	31
6.1.6 Criação da Tabela PratoPedido.....	32
6.1.7 Criação da Tabela Mesa	33
6.1.8 Criação da Tabela Pedido	34
6.1.9 Criação da Tabela Empregado	35
6.1.10. Criação da Tabela Fatura	36
6.1.11. Criação da Tabela ModoPagamento	37
6.2 T-SQL	38
6.2.1 Triggers	38
6.2.2 Stored Procedures	45
6.3 Views/Vistas	48
7. Conclusões e Trabalho Futuro	52
Bibliografia.....	53
Referências WWW	54
Lista de Siglas e Acrónimos	55
Anexos.....	56

Índice de Figuras

Figura 1 - Diagrama E-R sem atributos	5
Figura 2 – Diagrama concetual só com chaves primárias	10
Figura 3 - Diagrama Conceptual ER	11
Figura 4 - Mockup Ingredientes	13
Figura 5 - Mockup Pratos	13
Figura 6 - Mockup Ementa	14
Figura 7 - Mockup Pedido	14
Figura 8 - Mockup Fatura	15
Figura 9 - Modelo Lógico - Diagrama ER.....	22
Figura 10 Diagrama ER do SQL Management Studio	26

Índice de Tabelas

Tabela 1 - Identificação das entidades	4
Tabela 2 - Multiplicidade	6
Tabela 3 - Atributos para as Entidades	6
Tabela 4 - Relação - Ementa.....	7
Tabela 5 - Relação - Prato.....	7
Tabela 6 - Relação - Ingrediente	7
Tabela 7 - Relação - Pedido	8
Tabela 8 – Relação – Empregado	8
Tabela 9 – Relação – Fatura	9
Tabela 12 – Integridade Referencial	24
Tabela 13 – Tipos de dados	25

Índice Excertos de SQL

Excerto SQL 1 - Criação Tabela Ingrediente	27
Excerto SQL 2 - Criação da Tabela IngredientePrato	28
Excerto SQL 3 - Criação da Tabela Prato.....	29
Excerto SQL 4 - Criação da Tabela PratoEmenta	30
Excerto SQL 5 - Criação da Tabela Ementa.....	31
Excerto SQL 6 - Criação da Tabela PratoPedido	32
Excerto SQL 7 - Criação da Tabela Mesa	33
Excerto SQL 8 - Criação da Tabela Pedido	35
Excerto SQL 9 - Criação da Tabela Empregado.....	35
Excerto SQL 10 - Criação da Tabela Fatura.....	37
Excerto SQL 11 - Criação da Tabela ModoPagamento.....	37
Excerto SQL 12 - Trigger MesaOcupada.....	39
Excerto SQL 13 - Trigger MesaFicaLivre.....	40
Excerto SQL 14 - Trigger VerificaMesaLivreEHorario	42
Excerto SQL 15 - Trigger TotalFaturadoPedido.....	42
Excerto SQL 16 - Trigger VerificaDataHoraFatura	43
Excerto SQL 17 - Trigger VerificaSePratoEstaEmenta	44
Excerto SQL 18 - StoredProcedure PratosCarneEntreDatas	45
Excerto SQL 19 - StoredProcedure PedidoDeClienteComPreco	46
Excerto SQL 20 - StoredProcedure PedidoDeClienteComPreco	46
Excerto SQL 21 - StoredProcedure DescricaoPedidoDeCliente	47
Excerto SQL 22 - View EmentadeHoje	48
Excerto SQL 23 - View ProdutosEmentaAmanha.....	49
Excerto SQL 24 - PedidosDosClientes	49
Excerto SQL 25 – View PedidosComDescritivoEPreco.....	50
Excerto SQL 26 - View PedidosTotalAPagar.....	50
Excerto SQL 27 - View DiasCorrenteMesCarneEPeixe	51

1. Introdução

1.1 Contextualização

O projeto é relativo à componente prática da Unidade Curricular de Base de Dados e tem como objetivo principal o desenvolvimento de uma base de dados com base nos conhecimentos adquiridos ao longo do semestre nesta UC

1.2 Apresentação do Caso de Estudo

O restaurante “Michelin Star” decidiu tomar medidas ao nível da gestão e do tratamento dos processos do restaurante. Após análise feita por uma empresa de consultadoria, decidiram informatizar vários processos.

1.3 Motivação e Objetivos

Após verificação um decréscimo em termos de qualidade dos serviços prestados e consequente análise por empresa de consultadoria, a gerência do restaurante decidiu informatizar os seguintes processos:

- Elaboração e impressão das ementas diárias;
- Pedidos dos clientes;
- Confeção dos pratos das ementas
- Gestão e pagamentos das contas

Com isto, gerência do restaurante também espera conseguir obter rapidamente elementos que lhe permita responder às seguintes questões:

- qual é a ementa de hoje e quais os pratos que nele figuram;
- quais os produtos que são necessários para cumprir a ementa de amanhã;
- quais foram os pratos de carne servidos durante um período de tempo a designar;
- em que dias do corrente mês é que foi servido o prato de peixe “P” juntamente com o

prato de carne “C”.

1.4 Estrutura do Relatório

De forma a facilitar a sua consulta, a estrutura do relatório é dividida em capítulos e subcapítulos, onde se descreve os passos para a realização deste projeto.

É apresentado o tema, a contextualização e o caso de estudo deste projeto na introdução. De seguida, descreve-se o processo de desenvolvimento deste projeto e por fim, uma reflexão sobre o projeto.

2 Requisitos Gerais e Use Cases

2.1 Requisitos Gerais

Foram identificados os seguintes requisitos:

- É possível consultar os ingredientes de cada prato;
- A ementa é diária, não havendo diferença na ementa do almoço e do jantar;
- O preço de cada prato é fixo, não havendo promoções e/ou descontos;
- Todos os preços já incluem o IVA;
- Os produtos utilizados na confeção dos pratos são frescos;
- Não se realizam reservas;
- Não há serviço take-away;
- Considera-se que o pedido está concluído após emissão da fatura.
- Os modos de pagamento disponíveis são dinheiro, multibanco e MBway;
- Um funcionário é responsável por várias mesas;
- Os pedidos devem ser feitos sempre ao funcionário responsável pela mesa;
- O restaurante está aberto todos os dias, feriados incluídos;
- O horário de funcionamento: almoço das 12h às 15h, jantar 19h às 23h;
- Não se aceitam pedidos fora do horário ou 15 min antes do horário de fecho do serviço;
- Mesmo que o pagamento seja dividido pelos clientes, apenas é emitida uma fatura;
- Todos os pedidos estão associados a uma fatura, exceto quando o cliente envia os pratos para trás por falta de qualidade e abandona o restaurante
- O máximo de mesas deste restaurante é 20.

3 Desenho Conceptual

O primeiro passo para o desenho de uma base de dados é definir o modelo conceptual de dados. Fases seguintes deste capítulo:

1. Identificação de tipo de entidades
2. Relações entre entidades:
3. Atributos para as entidades
4. Documentação de atributos
5. Atribuição de chaves primarias
6. Documentação de chaves primarias (PK)
7. Utilização de conceitos de modelação avançada
8. Validação do modelo conceptual de dados de forma a satisfazer as transações dos utilizadores
9. Revisão do modelo conceptual de dados com o utilizador

3.1 Identificação das entidades

Tabela 1 - Identificação das entidades

Entidade	Descrição	Ocorrência
Ementa	Informação geral das entas	Cada ementa tem vários pratos
Prato	Informação sobre os pratos	Um prato está associado a vários ementas e tem vários ingredientes, à exceção dos pratos tipo “Bebida”
Ingrediente	Informação geral sobre os ingredientes	Um ingrediente pode estar associado a vários pratos
Pedido	Informação geral sobre o pedido do cliente	Cada pedido está associado a uma mesa, a um empregado e a uma data e hora do próprio pedido
Empregado	Informação geral sobre o Empregado	O Empregado está associado a vários pedidos
Fatura	Informação sobre um serviço realizado, a data de emissão da fatura e o modo de pagamento	Quando o serviço acontece pode ser feito uma fatura do mesmo

3.2 Relações entre Entidades

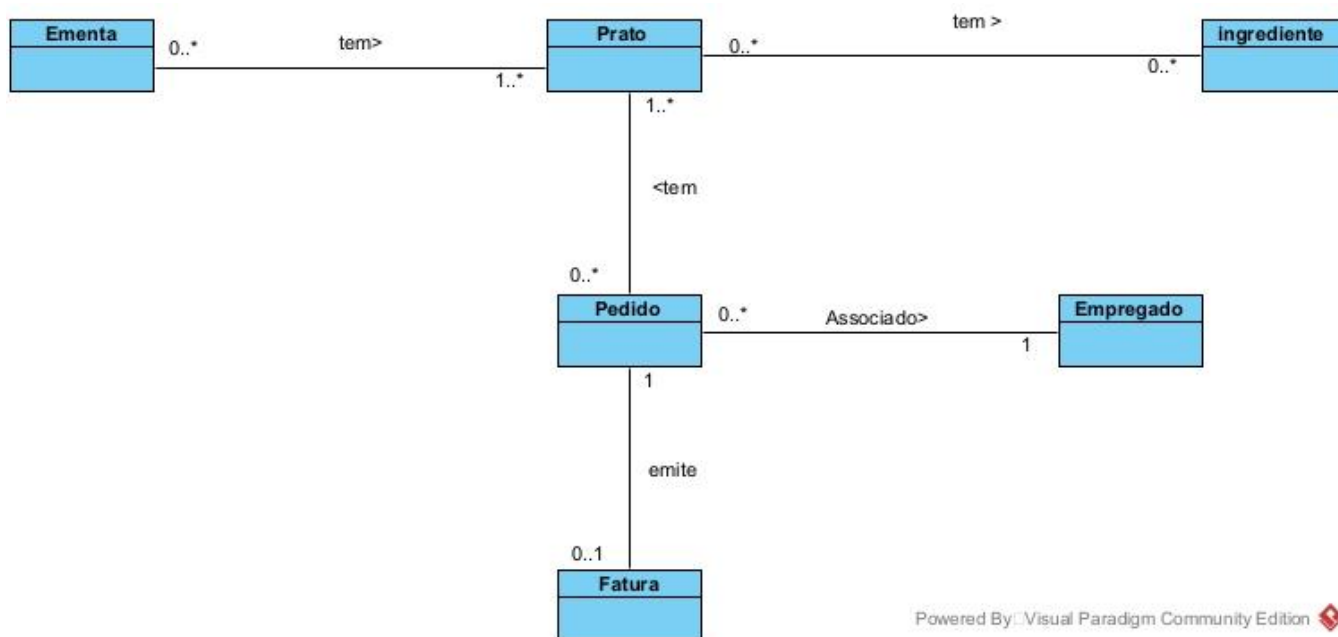


Figura 1 - Diagrama E-R sem atributos

3.3 Multiplicidade

Tabela 2 - Multiplicidade

Entidade	Multiplicidade	Relação	Multiplicidade	Entidade
Ementa	0...*	Tem	1...*	Prato
Prato	0..*	Tem	0...*	Ingrediente
Pedido	1...*	Tem	0...*	Prato
Pedido	0...*	Associado	1	Empregado
Pedido	1	Emite	0..1	Fatura

3.4 Atributos para as Entidades

Tabela 3 - Atributos para as Entidades

Ementa	ementaID, data, descricaoEmenta
Prato	pratoID, tipoPrato, preco descricaoPrato, nomePrato
Ingrediente	ingredienteID, nomeIngrediente, quantidade, unidade
Pedido	pedidoID, mesa, empregadoID, dataPedido, obs, pratoID, horaPedido
Empregado	empregadoID, primeiroNome, ultimoNome, sexo
Fatura	faturaID, modoPagamento, dataEmissao, valorTotal, pedidoID, nifCliente, horaEmissao

3.5 Documentação de atributos

Tabela 4 - **Relação** - Ementa

Atributo	Descrição	Tipo de informação	Nulls	Multi-Valued
ementaID	Identificador único de cada ementa	Até 3 números inteiros	Não	Não
dataEmenta	Data que a ementa da servida	Date	Não	Não
descricaoEmenta	Descrição da ementa	250 carateres variaveis	Sim	Não

Tabela 5 - **Relação** - Prato

Atributo	Descrição	Tipo de informação	Nulls	Multi-Valued
pratoID	Identificador único de cada prato	Até 4 números inteiros	Não	Não
tipoPrato	Identificador do tipo de prato: - E – entrada - B – Bebida - P – Peixe - C – Carne - S - Sopa - D - Sobremesa	1 caracter	Não	Não
Preco	Preço do prato	Smallmoney	Não	Não
descricaoPrato	Descrição do prato	250 carateres disponiveis	Sim	Não
nomePrato	Nome do prato	30 carateres variáveis	Não	Não

Tabela 6 - **Relação** - Ingrediente

Atributo	Descrição	Tipo de informação	Nulls	Multi-Valued
ingredienteID	Identificação único de ingrediente	Até 5 números inteiros	Não	Não
nomeIngrediente	Nome do Ingrediente	30 carateres variáveis	Não	Não
quantidade	Quantidade do ingrediente	Número de 5 dígitos com 2 casas decimais	Não	Não
Unidade	Unidade da quantidade do ingrediente	2 caracteres variaveis	Não	Não

Tabela 7 - **Relação** - Pedido

Atributo	Descrição	Tipo de informação	Nulls	Multi-Valued
pedidoID	Identificação única de cada pedido	Até 9 números inteiros	Não	Não
Mesa	Número da mesa		Não	Não
empregadoID	Identificador único de cada empregado	Até 3 números inteiros	Não	Não
dataPedido	Data do pedido	Date	Não	Não
Obs	Possíveis comentários e observações acerca do pedido	250 caracteres disponíveis	Sim	Não
pratoID	Identificador único de cada prato	Até 4 números inteiros	Não	Não
horaPedido	Hora do pedido	Time	Não	Não

Tabela 8 – **Relação** – Empregado

Atributo	Descrição	Tipo de informação	Nulls	Multi-Valued
empregadoID	Identificador único de cada empregado	Até 3 números inteiros	Não	Não
primeiroNome	Primeiro nome do Empregado	ntext	Não	Não
ultimoNome	Ultimo nome do Empregado	ntext	Não	Não
Sexo	Sexo do Empregado (M/F)	1 caracter	Não	Não

Tabela 9 – **Relação – Fatura**

Atributo	Descrição	Tipo de informação	Nulls	Multi-Valued
faturaID	Identificador único de cada fatura	Até 9 números inteiros	Não	Não
modoPagamento	Descrição do modo de pagamento	25 carateres variáveis	Não	Não
dataEmissao	Data de emissão da fatura	Date	Não	Não
valorTotal	Valor total do documento	Smallmoney	Sim	Não
pedidoID	Identificador único do pedido	9 dígitos numéricos	Não	Não
nifCliente	NIF do cliente	9 dígitos numéricos	Sim	Não
horaEmissao	Hora da emissão da fatura	Time	Não	Não

3.6. Atribuição de chaves primarias

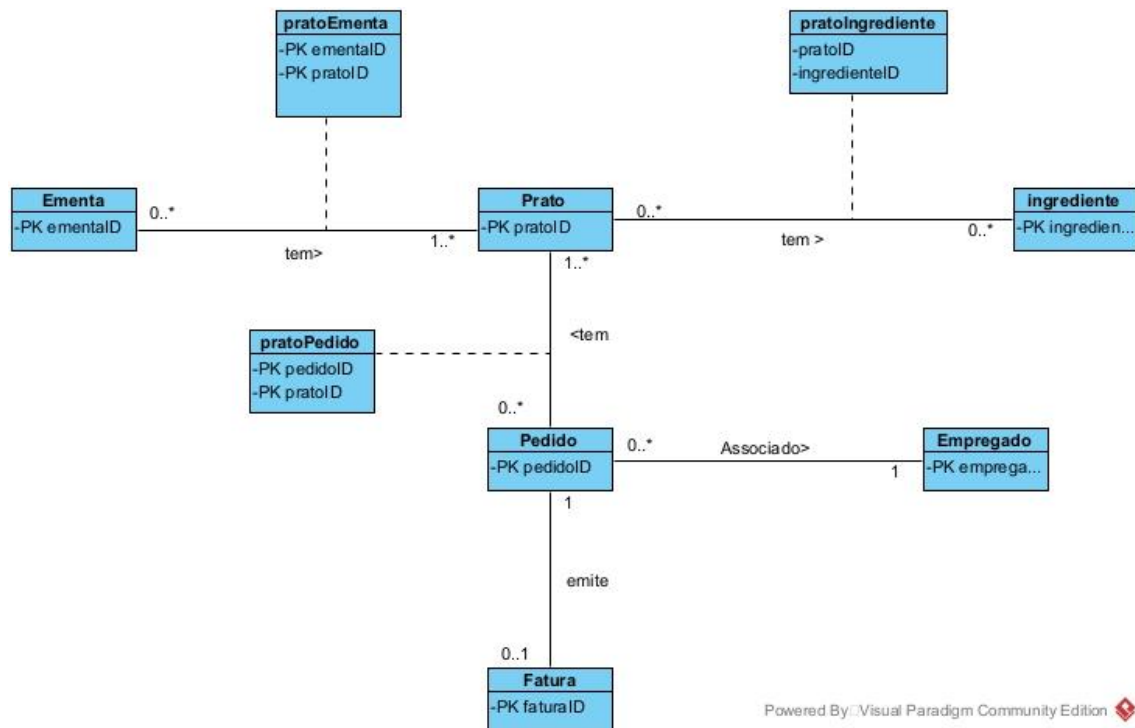
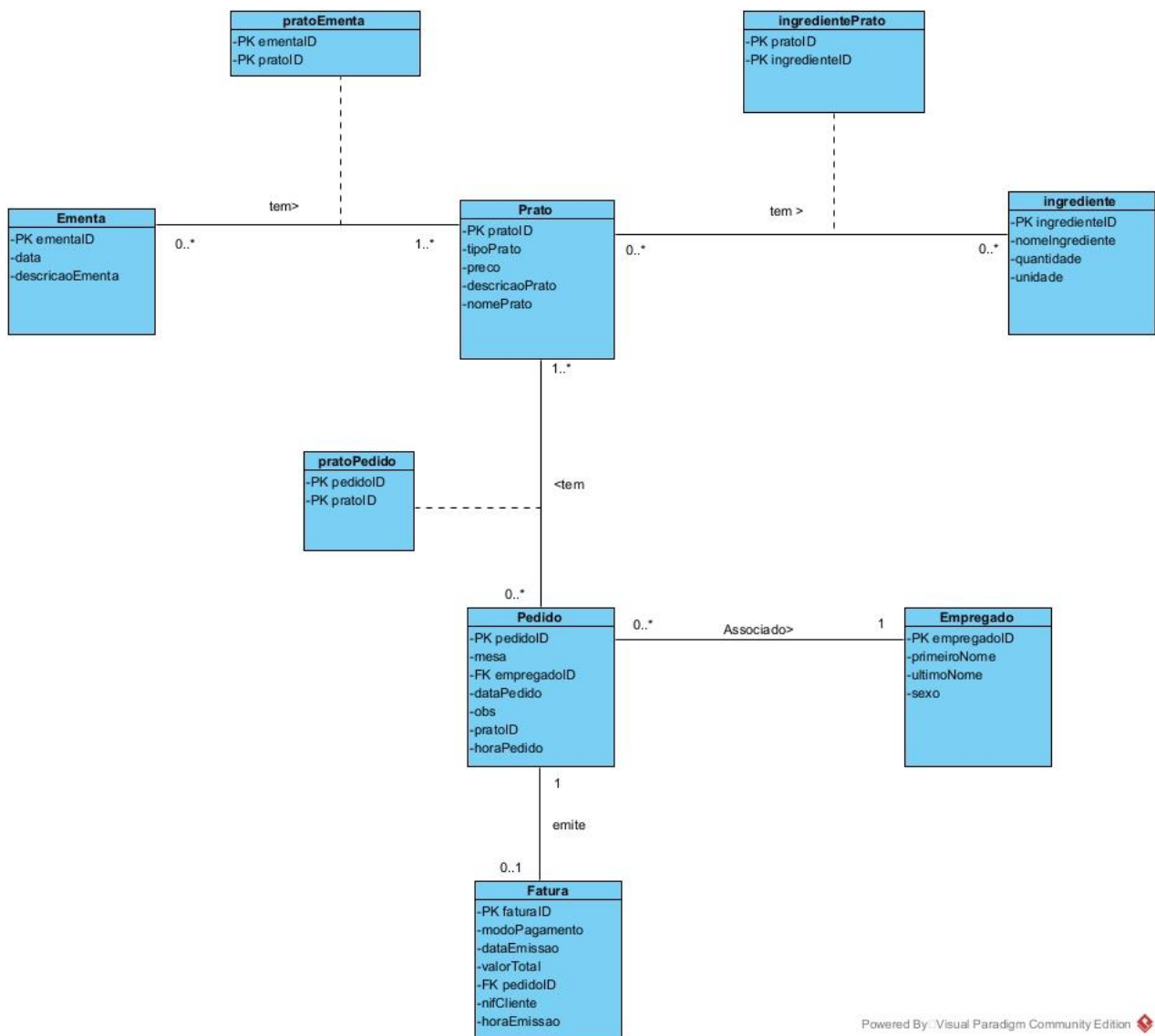


Figura 2 – Diagrama conceitual só com chaves primárias

Documentação de chaves primarias (PK)

Entidade	Chave Primaria	Chaves Candidatas
Ementa	ementaID	—
Prato	pratoID	—
Ingrediente	ingredienteID	—
Pedido	pedidoID	—
Empregado	empregadoID	—
Fatura	faturaID	—
PratoIngrediente	pratoID, ingredientesID	—
PratoEmenta	ementaID, pratoID	—
PratoPedido	pedidoID, pratoID	—



Powered By: Visual Paradigm Community Edition

Figura 3 - Diagrama Conceptual ER

4 Desenho Lógico

Nesta fase o modelo conceptual é traduzido para o modelo lógico, os utilizadores verificam se este se encontra estruturalmente correto. As fases desta etapa são:

1. Derivação de relações para o modelo de dados lógico;
2. Documentação de relações entre entidades;
3. Normalização
4. Restrições de integridade;
5. Rever o modelo lógico de dados com o utilizador.

4.1 Mockups

Ingredientes		
Sal	Azeite	Ovo
Queijo	Pao	Azeitona
Ketchup	Mostarda	Carne de Vaca
Bacalhau	Farinha	Massa

Novo

Voltar

Figura 4 - Mockup Ingredientes

Prato	Descrição	Ingredientes	Tipo	Preço	Actions
Pao com manteiga	pao, mateiga	E	€2	Edit
Cheeseburger	...	carne vaca, pao, queijo	C	€7,50	Edit
Super Bock	...	August 17, 1936	B	€1,50	Edit
Leite Creme	...	leite, açucares	D	€3,50	Edit

Novo

Voltar

Figura 5 - Mockup Pratos

07/06/2022

MENU ESTUDANTE

- Entradas

Pão com Chouriço ===== Fatias de pão no forno com manteiga ===== €2.00

- Sopa

Creme de Cenoura ===== Sopa de cenoura com textura creme ===== €3.00

- Prato Carne

Cheeseburger ===== Hamburguer com queijo e carne de vaca ===== €7.50

- Bebida

Agua ===== Copa de Agua com gelo ===== €0.50

Gerar Necessidades

Novo

Voltar

Figura 6 - Mockup Ementa

06/09/2022

Diogo Bragança

Pedido 33

Mesa 4

Sel.	Prato	Descrição	Tipo	Preço	Quantidade
<input type="radio"/>	Azeitonas Temperadas	E	€2	
<input checked="" type="radio"/>	Cheeseburger	...	C	€7,50	2
<input checked="" type="radio"/>	Super Bock	...	B	€1,50	2
<input type="radio"/>	Leite Creme	...	D	€3,50	

obs

Fatura

Novo

Voltar


Figura 7 - Mockup Pedido

Fatura
120

Data : 07/06/2022

Descritivo	Quantidade	Preço (c/IVA)
Pão com Chouriço	1	€2.00
Creme de Cenoura	1	€3.00
Cheeseburger	1	€7.50
Agua	1	€0.50

Emitido por: Ana
Mesa: 7
Modo Pagamento: MB



TOTAL:
€14,50

Imprimir

Nova

Voltar

Figura 8 - Mockup Fatura

4.2 Derivação de relações para o modelo de dados lógico

4.2.1 Relação de um para muitos (1: *)

Empregado 1 : * Pedido

Entidade pai

Pedido { pedidoID, mesa, empregadoID, dataPedido, obs, horaPedido }

Chave primária: pedidoID

Chave estrangeira: empregadoID

Entidade filho

Empregado { empregadoID, primeiroNome, ultimoNome, sexo }

Chave primária: empregadoID

Põe empregadoID em Pedido para modelar uma relação 1 : *

4.2.2 Relação de um para um (1 : 0...1)

Pedido 1 : 1 Fatura

Entidade pai

Pedido { pedidoID, mesa, empregadoID, dataPedido, obs, horaPedido }

Chave primária: pedidoID

Chave estrangeira: empregadoID

Entidade filho

```
Fatura { faturaID, modoPagamento, dataEmissao, valorTotal, pedidoID, nifCliente,
horaEmissao }
```

Chave primaria faturalID

Chave estrangeira pedidoID **refere** Pedido (pedidoID)

Relação 1 : 0..1 com participação obrigatória do lado da Pedido.

4.2.3 Relação de muitos para muitos (* : *)

Ingrediente * : * Prato

Ingrediente { ingredienteID, nomeIngrediente}

Chave primaria ingredientelD

```
Prato { pratoID, tipoPrato, preco, descricaoPrato, nomePrato }
```

Chave primaria pratolD

IngredientePrato { pratoID , ingredienteID, unidade, quantidade }

Chave primaria pratolD , ingredientelD

Chave estrangeira pratolD **refere** Prato(pratolD)
 ingredienteID **refere** Ingrediente(ingredienteID)

Ementa * : * Prato

Ementa { ementalD, data, descricaoEmenta}

Chave primaria ementalD

```
Prato { pratoID, tipoPrato, preco, descricaoPrato, nomePrato }
```

Chave primaria pratolD

PratoEmenta { pratoID , ementalID, }

Chave primaria pratoID , ementalID

Chave estrangeira pratoID **refere** Prato(pratoID)
ementalID **refere** Ementa(ementalID)

Prato * : * Pedido

Pedido { pedidoID, mesa, empregadoID, dataPedido, obs, horaPedido}

Chave primaria pedidoID

Chave estrangeira empregadoID

Prato { pratoID, tipoPrato, preco, descricaoPrato, nomePrato }

Chave primaria pratoID

pratoPedido { pratoID , pedidoID, }

Chave primaria pratoID , pedidoID

Chave estrangeira pratoID **refere** Prato(pratoID)
pedidoID **refere** Pedido(pedidoID)

4.3 Normalização

O processo de normalização surge como uma atividade associada ao Desenho Lógico que visa validar se as relações obtidas anteriormente são válidas.

Este processo de validação assenta no estudo das dependências funcionais existentes entre atributos de uma relação

4.3.1 Primeira Forma Normal (1FN)

A normalização de uma tabela na 1.^a Forma Normal (1FN) exige que a tabela tenha uma estrutura bidimensional correta, ou seja, cada linha deve corresponder a um só registo e cada coluna a um só campo.

O objetivo é eliminar redundância e introduzir dados apropriados nas colunas vazias das linhas que contêm grupos repetidos

Os autores fizeram uso das dependências funcionais que descrevem o relacionamento entre atributos de uma relação, para ajudar nesta forma de normalização e nas próximas.

4.3.2 Segunda Forma Normal (2FN)

A 2.^a forma normal diz que a tabela tem de estar na 1FN e que cada atributo não chave tem de ser funcionalmente dependente da totalidade da chave primária e não apenas de uma parte dessa chave.

Assim depois de identificada a chave primária de uma tabela, pode dar-se um dos dois casos:

A chave primária é constituída por um só atributo (chave elementar). Neste caso, a tabela está seguramente na 2FN (nenhum atributo depende de uma parte da chave, visto que a chave não é composta por partes);

A chave primária é constituída por mais que um atributo (chave primária composta). Neste caso, se existe algum ou alguns atributos que dependem de uma parte da chave (ou seja, de algum atributo que constitui a chave), então a tabela não está na 2FN.

4.3.3 Terceira Forma Normal (3FN)

A 3.^a Forma Normal (3FN) diz que a tabela tem de estar na 2FN e que nenhum atributo não chave pode depender funcionalmente de algum outro atributo que não seja a chave primária.

É baseada no conceito de dependência transitiva.

Portanto, para normalizar uma tabela de acordo com a 3FN, deve-se analisar todos os atributos não chave e verificar se existem dependências transitivas sobre a chave primária, removê-las e colocá-las numa nova relação

4.3.4 Normalização das tabelas

Ingrediente

UNF: { ingredienteID, nomeIngrediente }

1º Forma

Dependências funcionais

Ingrediente -> ingredienteID, nomeIngrediente

Não há repetição dados então já se encontra na primeira forma de normalização

2º Forma

Só uma chave primaria então não existe dependências parciais o que significa que já está na segunda forma de normalização

3º Forma

Não existem dependências transitivas sobre a chave primária o que significa que já está na terceira forma de normalização

Prato

UNF: { pratoID, tipoPrato, preco, descricaoPrato, nomePrato }

1º Forma

Dependências funcionais

pratoID -> tipoPrato, preco, descricaoPrato, nomePrato

O tipoPrato pode assumir 6 valores diferentes, ou seja, aparecerão dados repetidos, mas decidimos não colocar numa nova relação

2º Forma

Só existe uma chave primaria então não existe dependências parciais o que significa que já está na segunda forma de normalização

3º Forma

Não existem dependências transitivas sobre a chave primária o que significa que já está na terceira forma de normalização

Ementa

UNF: { ementaID, descricaoEmenta, dataEmenta }

1º Forma

Dependências funcionais

ementaID -> descricaoEmenta, dataEmenta

Não há repetição dados então já se encontra na primeira forma de normalização

2º Forma

Só uma chave primaria então não existe dependências parciais o que significa que já está na segunda forma de normalização

3º Forma

Não existem dependências transitivas sobre a chave primária o que significa que já está na terceira forma de normalização

Pedido

UNF: { pedidoID, mesa, empregadoID, dataPedido, obs, horaPedido }

1º Forma

Dependências funcionais

pedidoID -> mesa, empregadoID, dataPedido, obs, horaPedido

Mesa repete-se várias vezes. Decidimos criar a tabela **Mesa**. Nesta tabela o mesaID corresponde ao número da mesa e o estado, indica se a mesa está ocupada (1) ou não (0). Após emissão da fatura do pedido da mesa, o estado passa para 0. Por defeito, o estado está a 0.

pedidoID -> mesaID, empregadoID, dataPedido, obs, horaPedido

mesaID -> estado

2º Forma

Só uma chave primaria então não existe dependências parciais o que significa que já está na segunda forma de normalização

3º Forma

Não existem dependências transitivas sobre a chave primária o que significa que já está na terceira forma de normalização

Empregado

UNF: { empregadoID, primeiroNome, ultimoNome, sexo }

1º Forma

Dependências funcionais

empregadoID -> mesa, primeiroNome, ultimoNome, sexo

“sexo” aparece repetido pois só assume dois valores mas vimos vantagem na criação de uma nova tabela para armazenar estes dados.

2º Forma

Só uma chave primaria então não existe dependências parciais o que significa que já está na segunda forma de normalização

3º Forma

Não existem dependências transitivas sobre a chave primária o que significa que já está na terceira forma de normalização

Fatura

UNF: { faturaID, modoPagamento, dataEmissao, valorTotal, pedidoID, nifCliente, horaEmissao }

1º Forma

Dependências funcionais

faturaID -> modoPagamento, dataEmissao, valorTotal, pedidoID, nifCliente, horaEmissao

“modoPagamento” aparece repetido. Decidimos criar uma nova tabela “ModoPagamento”

UNF: { faturaID, dataEmissao, valorTotal, pedidoID, nifCliente, horaEmissao }

ModoPagamento

UNF: { modoPagamentoID, designacaoPagamento }

1º Forma

Dependências funcionais

faturaID -> dataEmissao, valorTotal, pedidoID, nifCliente, horaEmissao, modoPagamentoID

modoPagamento -> designacaoPagamento

2º Forma

Só uma chave primaria então não existe dependências parciais o que significa que já está na segunda forma de normalização

3º Forma

Não existem dependências transitivas sobre a chave primária o que significa que já está na terceira forma de normalização

4.3.5 Desenho Logico

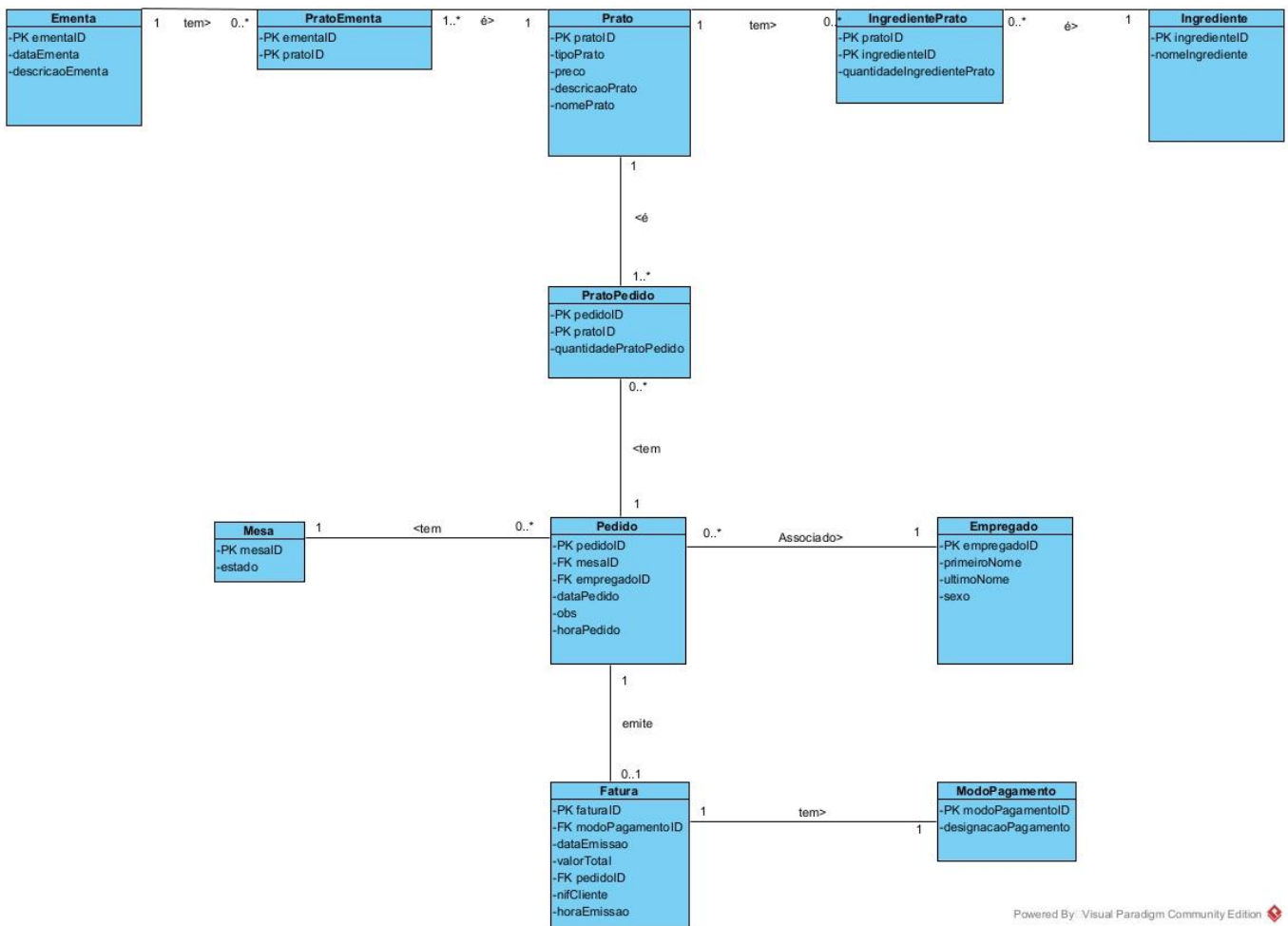


Figura 9 - Modelo Lógico - Diagrama ER

5 Restrições de integridade – Regras de negócio

Ingrediente

- Todos os campos são obrigatórios.

IngredientePrato

- Todos os campos são obrigatórios.
- A quantidadeIngredientePrato é a quantidade de ingrediente para fazer uma dose do prato.

Prato

- Todos os campos são obrigatórios à exceção de descricaoPrato.
- tipoPrato só pode assumir os seguintes valores:
 - E – entrada;
 - B – Bebida;
 - P – Peixe;
 - C – Carne;
 - S – Sopa;
 - D – Sobre mesa.

PratoEmenta

- Todos os campos são obrigatórios.

Ementa

- Todos os campos são obrigatórios à exceção de descricaoEmenta.

pratoPedido

- Todos os campos são obrigatórios.
- quantidadePratoPedido é em doses, não aceitando meias doses.
- Quantidade do prato pedido tem que ser maior que zero e menor que cem.

Preco

- Preco tem que ser maior que zero.

Pedido

- Todos os campos são obrigatórios à exceção de obs.

Empregado

- Todos os campos são obrigatórios.

Mesa

- Todos os campos são obrigatórios.
- estado assume 2 valores, 1 se está ocupada, 0 se não estiver.
- O valor por defeito de estado é 0.
- Após emissão da fatura do pedido, o estado da mesa passa a 0.
- mesaID corresponde ao número de mesas e não pode ser maior que o número de mesas que o restaurante tem.

Fatura

- Todos os campos são obrigatórios à exceção de nif.
- dataEmissao tem que ser posterior a dataPedido.
- horaEmissao tem que ser posterior a horaPedido.
- O nif deve ter exatamente 9 dígitos e serem válidos.

ModoPagamento

- Todos os campos são obrigatórios.
- designacaoPagamento pode assumir os valores:
 - D – Dinheiro Vivo;
 - M – Multibanco;
 - W – MBWay.

Integridade referencial

Tabela 10 – Integridade Referencial

Tabela base da primary key	Tabela base da foreign key	Clausula DELETE	Clausula UPDATE
Ingrediente	IngredientePrato	CASCADE	CASCADE
Prato	IngredientePrato	CASCADE	CASCADE
Prato	PratoEmenta	CASCADE	CASCADE
Ementa	PratoEmenta	CASCADE	CASCADE
Prato	PratoPedido	CASCADE	CASCADE
Pedido	PratoPedido	CASCADE	CASCADE
Empregado	Pedido	No Action	CASCADE
Pedido	Fatura	CASCADE	CASCADE
Mesa	Pedido	No Action	CASCADE
ModoPagamento	Fatura	No Action	CASCADE

6 Desenho Físico

Nesta Etapa, o modelo lógico será traduzido para um modelo físico através de um SGBD, que neste caso foi usado o Microsoft SQL Server e contemplou as seguintes fases:

1. Desenho das relações e restrições
2. Representação de Dados Derivados
3. Desenho das Restrições Gerais
4. Desenho das Vistas de Utilizador
5. Desnormalização
6. Monitorização e Manutenção do Sistema

Criação de tipos de dados

Tabela 11 – Tipos de dados

Nome	Tipo de dado	Tamanho precisão escala
ingredienteID	Numeric	(5)
pratoID	Numeric	(4)
nifPedidoFatura	Numeric	(9)
ementaEEmpregado	Numeric	(3)
modoPagamentoID	Numeric	(1)
quantidade	Numeric	(2)
nome	char	(30)
tipoPratoEPagamento	char	(1)
descricoesEOBs	char	(250)

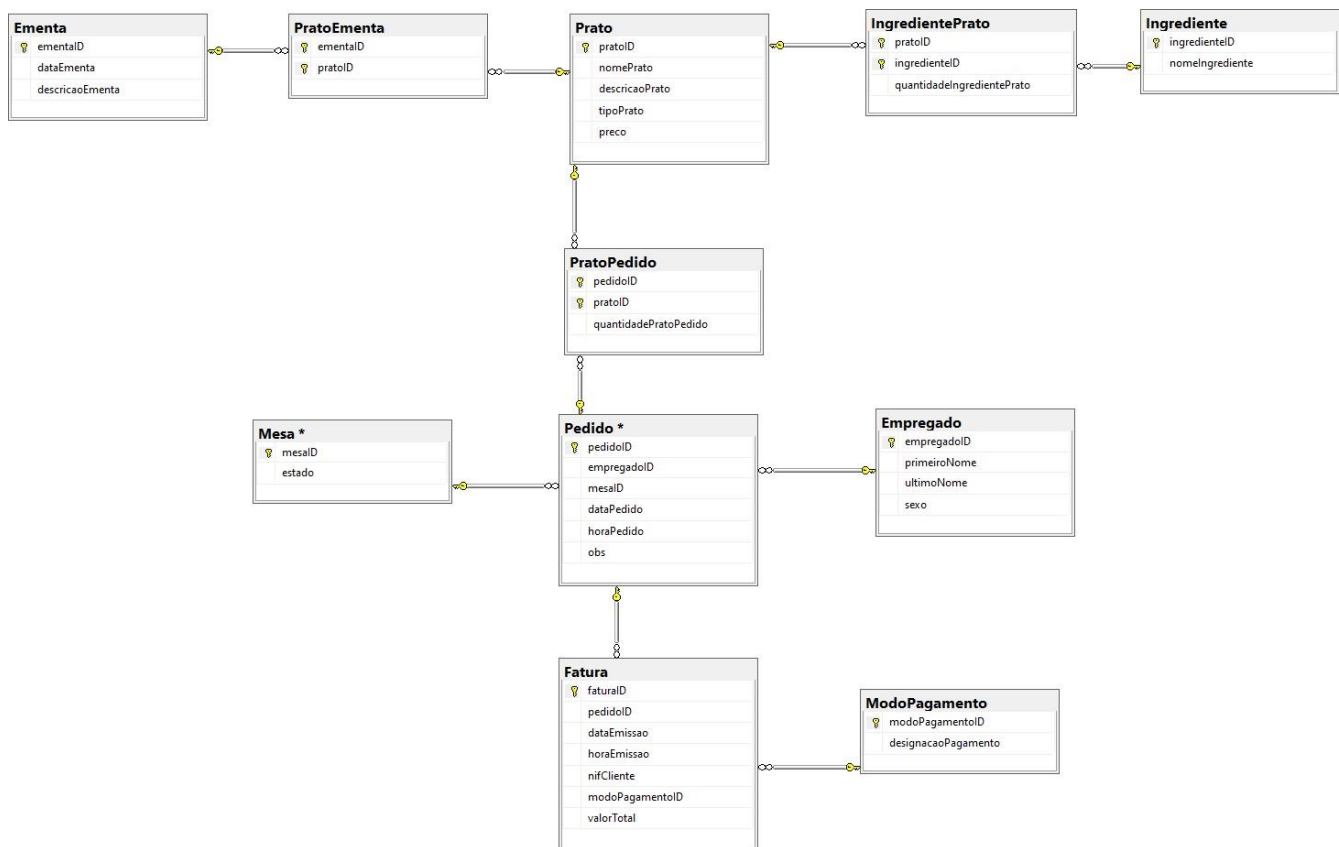


Figura 10 Diagrama ER do SQL Management Studio

6.1 Criação da Tabelas, relacionamentos e restrições aos atributos

6.1.1 Criação da Tabela Ingrediente

```
USE [michelinStar]
GO

/***** Object: Table [dbo].[Ingrediente]    Script Date: 06/06/2022 17:42:28
*****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Ingrediente](
    [ingredienteID] [dbo].[ingredienteID] NOT NULL,
    [nomeIngrediente] [dbo].[nome] NOT NULL,
    CONSTRAINT [PK_Ingrediente] PRIMARY KEY CLUSTERED
(
    [ingredienteID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)
ON [PRIMARY]
) ON [PRIMARY]
GO
```

Excerto SQL 1 - Criação Tabela Ingrediente

6.1.2 Criação da Tabela IngredientePrato

```
USE [michelinStar]
GO

/***** Object: Table [dbo].[IngredientePrato]    Script Date: 06/06/2022
17:46:16 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[IngredientePrato](
    [pratoID] [dbo].[pratoID] NOT NULL,
    [ingredienteID] [dbo].[ingredienteID] NOT NULL,
    [quantidadeIngredientePrato] [dbo].[quantidade] NOT NULL,
    CONSTRAINT [PK_IngredientePrato] PRIMARY KEY CLUSTERED
(
    [pratoID] ASC,
    [ingredienteID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)
ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[IngredientePrato] WITH CHECK ADD CONSTRAINT
[FK_IngredientePrato_Ingrediente] FOREIGN KEY([ingredienteID])
REFERENCES [dbo].[Ingrediente] ([ingredienteID])
ON UPDATE CASCADE
ON DELETE CASCADE
GO

ALTER TABLE [dbo].[IngredientePrato] CHECK CONSTRAINT
[FK_IngredientePrato_Ingrediente]
GO

ALTER TABLE [dbo].[IngredientePrato] WITH CHECK ADD CONSTRAINT
[FK_IngredientePrato_Prato] FOREIGN KEY([pratoID])
REFERENCES [dbo].[Prato] ([pratoID])
ON UPDATE CASCADE
ON DELETE CASCADE
GO

ALTER TABLE [dbo].[IngredientePrato] CHECK CONSTRAINT [FK_IngredientePrato_Prato]
GO
```

Excerto SQL 2 - Criação da Tabela IngredientePrato

6.1.3 Criação da Tabela Prato

```
USE [michelinStar]
GO

/***** Object: Table [dbo].[Prato]    Script Date: 06/06/2022 17:47:47 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Prato](
    [pratoID] [dbo].[pratoID] NOT NULL,
    [nomePrato] [dbo].[nome] NOT NULL,
    [descricaoPrato] [dbo].[descricoesEObs] NULL,
    [tipoPrato] [dbo].[tipoPratoEPagamento] NOT NULL,
    [preco] [smallmoney] NOT NULL,
    CONSTRAINT [PK_Prato] PRIMARY KEY CLUSTERED
(
    [pratoID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)
ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Prato] WITH CHECK ADD CONSTRAINT [CK_precoPositivo] CHECK
(((preco)>(0)))
GO

ALTER TABLE [dbo].[Prato] CHECK CONSTRAINT [CK_precoPositivo]
GO

ALTER TABLE [dbo].[Prato] WITH CHECK ADD CONSTRAINT [CK_tipoPrato] CHECK
(((tipoPrato]='E' OR [tipoPrato]='B' OR [tipoPrato]='P' OR [tipoPrato]='C' OR
[tipoPrato]='S' OR [tipoPrato]='D'))
GO

ALTER TABLE [dbo].[Prato] CHECK CONSTRAINT [CK_tipoPrato]
GO

EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'Preço maior que
zero' , @level0type=N'SHEMA',@level0name=N'dbo',
@level1type=N'TABLE',@level1name=N'Prato',
@level2type=N'CONSTRAINT',@level2name=N'CK_precoPositivo'
GO

EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'Verifica se o
tipo de prato é o correto' , @level0type=N'SHEMA',@level0name=N'dbo',
@level1type=N'TABLE',@level1name=N'Prato',
@level2type=N'CONSTRAINT',@level2name=N'CK_tipoPrato'
GO
```

Excerto SQL 3 - Criação da Tabela Prato

6.1.4 Criação da Tabela PratoEmenta

```
USE [michelinStar]
GO

/***** Object: Table [dbo].[PratoEmenta]    Script Date: 06/06/2022 17:49:28
*****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[PratoEmenta](
    [ementaID] [dbo].[ementaEEmpregado] NOT NULL,
    [pratoID] [dbo].[pratoID] NOT NULL,
    CONSTRAINT [PK_PratoEmenta] PRIMARY KEY CLUSTERED
(
    [ementaID] ASC,
    [pratoID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)
ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[PratoEmenta] WITH CHECK ADD CONSTRAINT
[FK_PratoEmenta_Ementa] FOREIGN KEY([ementaID])
REFERENCES [dbo].[Ementa] ([ementaID])
ON UPDATE CASCADE
ON DELETE CASCADE
GO

ALTER TABLE [dbo].[PratoEmenta] CHECK CONSTRAINT [FK_PratoEmenta_Ementa]
GO

ALTER TABLE [dbo].[PratoEmenta] WITH CHECK ADD CONSTRAINT
[FK_PratoEmenta_Prato] FOREIGN KEY([pratoID])
REFERENCES [dbo].[Prato] ([pratoID])
ON UPDATE CASCADE
ON DELETE CASCADE
GO

ALTER TABLE [dbo].[PratoEmenta] CHECK CONSTRAINT [FK_PratoEmenta_Prato]
GO
```

Excerto SQL 4 - Criação da Tabela PratoEmenta

6.1.5 Criação da Tabela Ementa

```
USE [michelinStar]
GO

/***** Object: Table [dbo].[Ementa]    Script Date: 06/06/2022 17:50:12 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Ementa](
    [ementaID] [dbo].[ementaEEmpregado] NOT NULL,
    [dataEmenta] [date] NOT NULL,
    [descricaoEmenta] [dbo].[descricoesEObs] NULL,
    CONSTRAINT [PK_Ementa] PRIMARY KEY CLUSTERED
(
    [ementaID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)
ON [PRIMARY]
) ON [PRIMARY]
GO
```

Excerto SQL 5 - Criação da Tabela Ementa

6.1.6 Criação da Tabela PratoPedido

```
USE [michelinStar]
GO
/***** Object: Table [dbo].[PratoPedido]    Script Date: 06/06/2022 17:51:11
*****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[PratoPedido](
    [pedidoID] [dbo].[nifPedidoFatura] NOT NULL,
    [pratoID] [dbo].[pratoID] NOT NULL,
    [quantidadePratoPedido] [dbo].[quantidade] NOT NULL,
    CONSTRAINT [PK_PratoPedido] PRIMARY KEY CLUSTERED
(
    [pedidoID] ASC,
    [pratoID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)
ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[PratoPedido] WITH CHECK ADD CONSTRAINT
[FK_PratoPedido_Pedido] FOREIGN KEY([pedidoID])
REFERENCES [dbo].[Pedido] ([pedidoID])
ON UPDATE CASCADE
ON DELETE CASCADE
GO

ALTER TABLE [dbo].[PratoPedido] CHECK CONSTRAINT [FK_PratoPedido_Pedido]
GO

ALTER TABLE [dbo].[PratoPedido] WITH CHECK ADD CONSTRAINT
[FK_PratoPedido_Prato] FOREIGN KEY([pratoID])
REFERENCES [dbo].[Prato] ([pratoID])
ON UPDATE CASCADE
ON DELETE CASCADE
GO

ALTER TABLE [dbo].[PratoPedido] CHECK CONSTRAINT [FK_PratoPedido_Prato]
GO

ALTER TABLE [dbo].[PratoPedido] WITH CHECK ADD CONSTRAINT
[CK_quantidadePratoPedido] CHECK (([quantidadePratoPedido]>(0) AND
[quantidadePratoPedido]<(100)))
GO

ALTER TABLE [dbo].[PratoPedido] CHECK CONSTRAINT [CK_quantidadePratoPedido]
GO

EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'quantidade a
pedir maior que 0 e menor 100', @level0type=N'SHEMA',@level0name=N'dbo',
@level1type=N'TABLE',@level1name=N'PratoPedido',
@level2type=N'CONSTRAINT',@level2name=N'CK_quantidadePratoPedido'
GO
```


6.1.7 Criação da Tabela Mesa

```
USE [michelinStar]
GO

/***** Object: Table [dbo].[Mesa]    Script Date: 06/06/2022 18:24:49 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Mesa](
    [mesaID] [dbo].[quantidade] NOT NULL,
    [estado] [bit] NOT NULL,
    CONSTRAINT [PK_Mesa] PRIMARY KEY CLUSTERED
(
    [mesaID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)
ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Mesa] ADD CONSTRAINT [DF_Mesa_estado] DEFAULT ((0)) FOR
[estado]
GO

ALTER TABLE [dbo].[Mesa] WITH CHECK ADD CONSTRAINT [CK_mesaID] CHECK
((([mesaID]<(21))))
GO

ALTER TABLE [dbo].[Mesa] CHECK CONSTRAINT [CK_mesaID]
GO

EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'mesa ID = nr
mesa menor que 21 (mesas do restaurante)',
@level0type=N'SCHEMA',@level0name=N'dbo',
@level1type=N'TABLE',@level1name=N'Mesa',
@level2type=N'CONSTRAINT',@level2name=N'CK_mesaID'
GO
```

Excerto SQL 7 - Criação da Tabela Mesa

6.1.8 Criação da Tabela Pedido

```
USE [michelinStar]
GO

/***** Object: Table [dbo].[Pedido]    Script Date: 06/06/2022 18:23:41 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Pedido](
    [pedidoID] [dbo].[nifPedidoFatura] NOT NULL,
    [empregadoID] [dbo].[ementaEEmpregado] NOT NULL,
    [mesaID] [dbo].[quantidade] NOT NULL,
    [dataPedido] [date] NOT NULL,
    [horaPedido] [time](7) NOT NULL,
    [obs] [dbo].[descricoesEObs] NULL,
    CONSTRAINT [PK_Pedido] PRIMARY KEY CLUSTERED
(
    [pedidoID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)
ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Pedido] ADD CONSTRAINT [DF_Pedido_dataPedido] DEFAULT
(CONVERT([date],getdate())) FOR [dataPedido]
GO

ALTER TABLE [dbo].[Pedido] ADD CONSTRAINT [DF_Pedido_horaPedido] DEFAULT
(CONVERT([time],getdate())) FOR [horaPedido]
GO

ALTER TABLE [dbo].[Pedido] WITH CHECK ADD CONSTRAINT [FK_Pedido_Empregado]
FOREIGN KEY([empregadoID])
REFERENCES [dbo].[Empregado] ([empregadoID])
ON UPDATE CASCADE
ON DELETE CASCADE
GO

ALTER TABLE [dbo].[Pedido] CHECK CONSTRAINT [FK_Pedido_Empregado]
GO

ALTER TABLE [dbo].[Pedido] WITH CHECK ADD CONSTRAINT [FK_Pedido_Mesa] FOREIGN
KEY([mesaID])
REFERENCES [dbo].[Mesa] ([mesaID])
ON UPDATE CASCADE
GO

ALTER TABLE [dbo].[Pedido] CHECK CONSTRAINT [FK_Pedido_Mesa]
GO

ALTER TABLE [dbo].[Pedido] WITH CHECK ADD CONSTRAINT [CK_mesa] CHECK
(((mesaID)<(21)))
GO

ALTER TABLE [dbo].[Pedido] CHECK CONSTRAINT [CK_mesa]
GO
```

```
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'Só temos 20
mesas no restaurante', @level0type=N'SHEMA',@level0name=N'dbo',
@level1type=N'TABLE',@level1name=N'Pedido',
@level2type=N'CONSTRAINT',@level2name=N'CK_mesa'
GO
```

Excerto SQL 8 - Criação da Tabela Pedido

6.1.9 Criação da Tabela Empregado

```
USE [michelinStar]
GO

/***** Object: Table [dbo].[Empregado]    Script Date: 06/06/2022 17:54:03
*****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Empregado](
    [empregadoID] [dbo].[ementaEEmpregado] NOT NULL,
    [primeiroNome] [ntext] NOT NULL,
    [ultimoNome] [ntext] NOT NULL,
    [sexo] [dbo].[tipoPratoEPagamento] NOT NULL,
    CONSTRAINT [PK_Empregado] PRIMARY KEY CLUSTERED
    (
        [empregadoID] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)
    ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO

ALTER TABLE [dbo].[Empregado] WITH CHECK ADD CONSTRAINT [CK_sexo] CHECK
((([sexo]='M' OR [sexo]='F')))
GO

ALTER TABLE [dbo].[Empregado] CHECK CONSTRAINT [CK_sexo]
GO

EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'Sexo assume M =
male ou F = female', @level0type=N'SHEMA',@level0name=N'dbo',
@level1type=N'TABLE',@level1name=N'Empregado',
@level2type=N'CONSTRAINT',@level2name=N'CK_sexo'
GO
```

Excerto SQL 9 - Criação da Tabela Empregado

6.1.10. Criação da Tabela Fatura

```
USE [michelinStar]
GO

/***** Object: Table [dbo].[Fatura]    Script Date: 06/06/2022 17:55:39 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Fatura](
    [faturaID] [dbo].[nifPedidoFatura] NOT NULL,
    [pedidoID] [dbo].[nifPedidoFatura] NOT NULL,
    [dataEmissao] [date] NOT NULL,
    [horaEmissao] [time](7) NOT NULL,
    [nifCliente] [dbo].[nifPedidoFatura] NULL,
    [modoPagamentoID] [dbo].[modoPagamentoID] NOT NULL,
    [valorTotal] [smallmoney] NULL,
    CONSTRAINT [PK_Fatura] PRIMARY KEY CLUSTERED
(
    [faturaID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)
ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Fatura] ADD CONSTRAINT [DF_Fatura_dataEmissao] DEFAULT
(CONVERT([date],getdate())) FOR [dataEmissao]
GO

ALTER TABLE [dbo].[Fatura] ADD CONSTRAINT [DF_Fatura_horaEmissao] DEFAULT
(CONVERT([time],getdate())) FOR [horaEmissao]
GO

ALTER TABLE [dbo].[Fatura] WITH CHECK ADD CONSTRAINT [FK_Fatura_ModoPagamento]
FOREIGN KEY([modoPagamentoID])
REFERENCES [dbo].[ModoPagamento] ([modoPagamentoID])
ON UPDATE CASCADE
GO

ALTER TABLE [dbo].[Fatura] CHECK CONSTRAINT [FK_Fatura_ModoPagamento]
GO

ALTER TABLE [dbo].[Fatura] WITH CHECK ADD CONSTRAINT [FK_Fatura_Pedido] FOREIGN
KEY([pedidoID])
REFERENCES [dbo].[Pedido] ([pedidoID])
ON UPDATE CASCADE
ON DELETE CASCADE
GO

ALTER TABLE [dbo].[Fatura] CHECK CONSTRAINT [FK_Fatura_Pedido]
GO

ALTER TABLE [dbo].[Fatura] WITH CHECK ADD CONSTRAINT [CK_NIF] CHECK
(((nifCliente)>(99999999) AND [nifCLiente]<(100000000)))
GO

ALTER TABLE [dbo].[Fatura] CHECK CONSTRAINT [CK_NIF]
GO
```

```
EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'Valores validos de NIF', @level0type=N'SHEMA',@level0name=N'dbo',
@level1type=N'TABLE',@level1name=N'Fatura',
@level2type=N'CONSTRAINT',@level2name=N'CK_NIF'
GO
```

Excerto SQL 10 - Criação da Tabela Fatura

6.1.11. Criação da Tabela ModoPagamento

```
USE [michelinStar]
GO

/***** Object: Table [dbo].[ModoPagamento]    Script Date: 06/06/2022 17:56:25
*****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[ModoPagamento](
    [modoPagamentoID] [dbo].[modoPagamentoID] NOT NULL,
    [designacaoPagamento] [dbo].[tipoPratoEPagamento] NOT NULL,
    CONSTRAINT [PK_ModoPagamento] PRIMARY KEY CLUSTERED
(
    [modoPagamentoID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)
ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[ModoPagamento] WITH CHECK ADD CONSTRAINT [CK_tiposPagamento]
CHECK (([designacaoPagamento]='D' OR [designacaoPagamento]='M' OR
[designacaoPagamento]='W'))
GO

ALTER TABLE [dbo].[ModoPagamento] CHECK CONSTRAINT [CK_tiposPagamento]
GO

EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'Valores validos = D ou M ou W', @level0type=N'SHEMA',@level0name=N'dbo',
@level1type=N'TABLE',@level1name=N'ModoPagamento',
@level2type=N'CONSTRAINT',@level2name=N'CK_tiposPagamento'
GO
```

Excerto SQL 11 - Criação da Tabela ModoPagamento

6.2 T-SQL

6.2.1 Triggers

Nesta BD foram criados cinco triggers de forma a validar as restrições de negócio, referidas anteriormente no relatório, e os constraints de cada relação.

Os triggers criados para a base de dados foram:

- MesaOcupada

Este trigger refere-se à inserção de dados na Tabela Pedido. Altera o estado da Tabela Mesa para 1 (ocupado).

- MesaFicaLivre

Este trigger refere-se à inserção de dados na Tabela Fatura. Altera o estado da Tabela Mesa para 0 (livre) após emissão da fatura do pedido atribuído à mesa em questão.

- VerificaMesaLivreEHorario

Este trigger refere-se à inserção de dados na Tabela Pedido. Caso a mesa esteja ocupada ou fora do horário de serviço, não é possível inserir pedido, aparecendo a causa.

- TotalFaturadoPedido

Este trigger refere-se à inserção de dados na Tabela Fatura. Após se adicionar uma fatura, atualiza automaticamente o valor da fatura para o total do pedido associado.

- VerificaDataHoraFatura

Este trigger refere-se à inserção de dados na Tabela Fatura. Impede que se adicione uma fatura com data e hora anteriores à data e hora do pedido correspondente.

- VerificaSePratoEstaEmenta

Este trigger refere-se à inserção de dados na Tabela PratoPedido. Impede que se adicione um prato ao pedido que não conste da ementa do dia.

```

USE [michelinStar]
GO

/***** Object: Trigger [dbo].[MesaOcupada]    Script Date: 08/06/2022 18:21:34
*****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

-- =====
-- Author:          <Author,,Name>
-- Create date:    <Create Date,,>
-- Description:    <Description,,>
-- =====
CREATE TRIGGER [dbo].[MesaOcupada]
ON [dbo].[Pedido]
AFTER INSERT
AS
BEGIN

    SET NOCOUNT ON;
    Declare @mesaID numeric (2,0)

    Select @mesaID = i.mesaID
    From inserted i

    UPDATE [dbo].[Mesa] set estado = 1
    where @mesaID = mesaID

END
GO

```

Excerto SQL 12 - Trigger MesaOcupada

```

USE [michelinStar]
GO
/***** Object:  Trigger [dbo].[MesaFicaLivre]    Script Date: 08/06/2022
18:59:50 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:      <Author,,Name>
-- Create date: <Create Date,,>
-- Description:  <Description,,>
-- =====
ALTER TRIGGER [dbo].[MesaFicaLivre]
ON [dbo].[Fatura]
AFTER INSERT
AS
BEGIN

    SET NOCOUNT ON;
    Declare @pedidoID numeric (9,0)
    Declare @mesaID numeric (2,0)

    Select @pedidoID = i.pedidoID
    From inserted i

    Select @mesaID = p.mesaID
    From Mesa m, Pedido p
    Where p.pedidoID = @pedidoID

    UPDATE [dbo].[Mesa] set estado = 0
    where @mesaID = mesaID

END

```

Excerto SQL 13 - Trigger MesaFicaLivre


```

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:          <Author,,Name>
-- Create date:    <Create Date,,>
-- Description:     <Description,,>
-- =====
CREATE TRIGGER [dbo].[VerificaMesaLivreEHorario]
ON [dbo].[Pedido]
INSTEAD OF INSERT
AS
BEGIN

    SET NOCOUNT ON;
    Declare @mesaID numeric (2,0),
            @pedidoID numeric (9,0),
            @empregadoID numeric (3,0),
            @dataPedido date,
            @horaPedido time,
            @obs char(250),
            @horaAberturaManha time,
            @horaFechoManha time,
            @horaAberturaTarde time,
            @horaFechoTarde time,
            @estado bit

    Set @horaAberturaManha = '12:00:00'
    Set @horaFechoManha = '14:45:00'
    Set @horaAberturaTarde = '19:00:00'
    Set @horaFechoTarde = '22:45:00'

    Select @estado = m.estado, @mesaID = i.mesaID, @dataPedido =
i.dataPedido, @horaPedido = i.horaPedido, @empregadoID = i.empregadoID, @obs =
i.obs

    From inserted i, Mesa m
    Where m.mesaID = i.mesaID
    IF (@estado = 1)
    BEGIN
        RAISERROR ('Mesa já está ocupada, aguarde que cliente
acabe', 10, 11)
    END
    ELSE
    IF ((@horaPedido NOT BETWEEN @horaAberturaManha AND
@horaFechoManha) OR (@horaPedido BETWEEN @horaAberturaTarde AND
@horaFechoTarde))
    BEGIN
        RAISERROR ('Pedido rejeitado por estar fora do horario de
serviço', 10, 11)
    END
    ELSE
        INSERT INTO Pedido (empregadoID, mesaID, dataPedido,
horaPedido, obs) values (@empregadoID, @mesaID, @dataPedido, @horaPedido, @obs)

END

GO

```

```
USE [michelinStar]
GO

/***** Object: Trigger [dbo].[TotalFaturadoPedido]    Script Date: 09/06/2022
12:20:55 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

-- =====
-- Author:      <Author,,Name>
-- Create date: <Create Date,,>
-- Description:  <Description,,>
-- =====
CREATE TRIGGER [dbo].[TotalFaturadoPedido]
ON [dbo].[Fatura]
AFTER INSERT
AS
BEGIN

    SET NOCOUNT ON;

    Declare @pedidoID numeric (9,0)

    Select @pedidoID = i.pedidoID
    From inserted i

    UPDATE Fatura set valorTotal = (Select SUM ( DISTINCT a.preco *
b.quantidadePratoPedido)
From Prato a, PratoPedido b, Pedido c, Fatura f
Where a.pratoID = b.pratoID and b.pedidoID = c.pedidoID and c.pedidoID =
@pedidoID)
where @pedidoID = pedidoID

END
GO

ALTER TABLE [dbo].[Fatura] ENABLE TRIGGER [TotalFaturadoPedido]
GO
```

```

USE [michelinStar]
GO

/***** Object: Trigger [dbo].[VerificaDataHoraFatura]    Script Date:
09/06/2022 19:09:57 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

-- =====
-- Author:          <Author,,Name>
-- Create date:    <Create Date,,>
-- Description:     <Description,,>
-- =====
CREATE TRIGGER [dbo].[VerificaDataHoraFatura]
ON [dbo].[Fatura]
INSTEAD OF INSERT
AS
BEGIN

    SET NOCOUNT ON;
    Declare @pedidoID numeric(9, 0),
    @dataEmissao date,
    @horaEmissao time,
    @nifCliente numeric(9, 0),
    @modoPagamentoID numeric(1, 0),
    @valorTotal smallmoney,
    @dataPedido date,
    @horaPedido time

    Select @pedidoID = i.pedidoID, @dataEmissao = i.dataEmissao, @horaEmissao
    = i.horaEmissao, @nifCliente = i.nifCliente, @modoPagamentoID =
    i.modoPagamentoID, @valorTotal = i.valorTotal, @dataPedido = p.dataPedido,
    @horaPedido = p.horaPedido
    From Pedido p, inserted i
    Where (p.pedidoID = i.pedidoID)
    IF(@dataEmissao < @dataPedido OR (@dataEmissao >= @dataPedido AND
    @horaEmissao < @horaPedido ))
    BEGIN
        RAISERROR ( 'Data e/ou hora da fatura não podem ser inferiores às do pedido
        ', 10, 11)
    END
    ELSE
        INSERT INTO Fatura (pedidoID, dataEmissao, horaEmissao, nifCliente,
        modoPagamentoID, valorTotal)
        values(@pedidoID, @dataEmissao, @horaEmissao, @nifCliente,
        @modoPagamentoID, @valorTotal)

END
GO

```

Excerto SQL 16 - Trigger VerificaDataHoraFatura

```

USE [michelinStar]
GO

/***** Object: Trigger [dbo].[VerificaSePratoEstaEmenta]    Script Date:
11/06/2022 01:45:58 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

-- =====
-- Author:          <Author,,Name>
-- Create date: <Create Date,,>
-- Description:     <Description,,>
-- =====
CREATE TRIGGER [dbo].[VerificaSePratoEstaEmenta]
ON [dbo].[PratoPedido]
INSTEAD OF INSERT
AS
BEGIN

    SET NOCOUNT ON;
    Declare @pedidoID numeric (9,0),
            @pratoID numeric(4, 0),
            @quantidadePratoPedido numeric(2, 0),
            @contador numeric(2,0)

    Set @pedidoID = (Select i.pedidoID
From inserted i)
    Set @pratoID = (Select i.pratoID
From inserted i)
    Set @quantidadePratoPedido = (Select i.quantidadePratoPedido
From inserted i)

    Set @contador = ( Select Count(*)
From inserted i, Prato p, PratoEmenta pre, Ementa e
Where (i.pratoID = p.pratoID and p.pratoID = pre.pratoID and
pre.ementaID = e.ementaID) and e.dataEmenta = (Select ped.dataPedido
From inserted i, Pedido ped
Where i.pedidoID = ped.pedidoID))
    IF(@contador < 1)
        BEGIN
            RAISERROR ('O prato não consta na ementa', 10, 11)
        END
    ELSE
        INSERT INTO PratoPedido( pedidoID, pratoID,
quantidadePratoPedido)
        values(@pedidoID, @pratoID, @quantidadePratoPedido)

END
GO

```

Excerto SQL 17 - Trigger VerificaSePratoEstaEmenta

6.2.2 Stored Procedures

Para cumprir as restrições de negócio foram criados três procedimentos:

PratosCarneEntreDatas - é responsável por apresentar quais foram os pratos de carne servidos durante um período a designar.

PedidoDeClienteComPreco - é responsável por apresentar o número do pedido, a mesa, o empregado responsável, o prato pedido, as doses e o preço da dose, bem como o preço do pedido total, de um pedido a designar (à semelhança de uma fatura).

DescricaoPedidoDeCliente - é responsável por apresentar o número do pedido, a mesa, o empregado responsável, o prato pedido, as doses e as observações de um pedido específico.

```
USE [michelinStar]
GO

/***** Object:  StoredProcedure [dbo].[PratosCarneEntreDatas]    Script Date:
09/06/2022 14:52:57 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

-- =====
-- Author:          <Author,,Name>
-- Create date:    <Create Date,,>
-- Description:     <Description,,>
-- =====
CREATE PROCEDURE [dbo].[PratosCarneEntreDatas] @date1 date, @date2 date
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    SELECT e.dataEmenta, p.nomePrato, p.tipoPrato
    From Ementa e, PratoEmenta pe, Prato p
    Where (e.ementaID = pe.ementaID and pe.pratoID = p.pratoID) and
    (e.dataEmenta between @date1 and @date2) and p.tipoPrato = 'C'

END
GO
```

Excerto SQL 18 - StoredProcedure PratosCarneEntreDatas

```

USE [michelinStar]
GO
/***** Object: StoredProcedure [dbo].[PedidoDeClienteComPreco]    Script Date:
09/06/2022 15:30:27 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

-- =====
-- Author:          <Author,,Name>
-- Create date: <Create Date,,>
-- Description:     <Description,,>
-- =====
CREATE PROCEDURE [dbo].[PedidoDeClienteComPreco] @pedidoID numeric(9,0)
AS
BEGIN

    SET NOCOUNT ON;

    Select ped.pedidoID, ped.mesaID as Mesa, emp.primeiroNome as "Nome do
Empregado", prt.nomePrato, prd.quantidadePratoPedido as Doses, prt.preco,
fat.valorTotal as "Total pedido completo a Pagar"
    From Pedido ped, PratoPedido prd, Prato prt, Empregado emp, Fatura fat
    Where ped.pedidoID = @pedidoID and (ped.pedidoID = fat.pedidoID and
ped.pedidoID = prd.pedidoID and prd.pratoID = prt.pratoID) and ped.empregadoID =
emp.empregadoID
END
GO

```

Excerto SQL 19 - StoredProcedure PedidoDeClienteComPreco

```

USE [michelinStar]
GO
/***** Object: StoredProcedure [dbo].[PedidoDeClienteComPreco]    Script Date:
09/06/2022 15:30:27 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

-- =====
-- Author:          <Author,,Name>
-- Create date: <Create Date,,>
-- Description:     <Description,,>
-- =====
CREATE PROCEDURE [dbo].[DescricaoPedidoDeCliente] @pedidoID numeric(9,0)
AS
BEGIN

    SET NOCOUNT ON;

    Select ped.pedidoID, ped.mesaID as Mesa, emp.primeiroNome as "Nome do
Empregado", prt.nomePrato, prd.quantidadePratoPedido as Doses, ped.obs
    From Pedido ped, PratoPedido prd, Prato prt, Empregado emp
    Where ped.pedidoID = @pedidoID and (ped.pedidoID = prd.pedidoID and
prd.pratoID = prt.pratoID) and ped.empregadoID = emp.empregadoID
END
GO

```

Excerto SQL 20 - StoredProcedure PedidoDeClienteComPreco

```

USE [michelinStar]
GO

/***** Object: StoredProcedure [dbo].[PedidoDeClienteComPreco]    Script Date:
09/06/2022 15:30:27 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

-- =====
-- Author:          <Author,,Name>
-- Create date: <Create Date,,>
-- Description:      <Description,,>
-- =====
CREATE PROCEDURE [dbo].[DescricaoPedidoDeCliente] @pedidoID numeric(9,0)
AS
BEGIN

    SET NOCOUNT ON;

    Select ped.pedidoID, ped.mesaID as Mesa, emp.primeiroNome as "Nome do
Empregado", prt.nomePrato, prd.quantidadePratoPedido as Doses, ped.obs
    From Pedido ped, PratoPedido prd, Prato prt, Empregado emp
    Where ped.pedidoID = @pedidoID and (ped.pedidoID = prd.pedidoID and
prd.pratoID = prt.pratoID) and ped.empregadoID = emp.empregadoID
END
GO

```

Excerto SQL 21 - StoredProcedure DescricaoPedidoDeCliente

6.3 Views/Vistas

As vistas SQL foram utilizadas como auxílio para as consultas da BD. No total foram criadas seis vistas:

EmentadeHoje – permite obter a ementa de hoje e quais os pratos que nele figuram.

ProdutosEmentaAmanha – permite obter quais os produtos que são necessários para cumprir a ementa de amanhã.

PedidosDosClientes – cada pedido do cliente considera: o número da mesa onde está sentado, a lista dos pratos/artigos que deseja consumir, a identificação do funcionário que recebeu o pedido e um conjunto de observações, além de outras informações.

PedidosComDescritivoEPrecio – permite obter os pedidos agrupados, com o empregado responsável pelo pedido, o nome do prato, a quantidade pedida, o preço por dose e o preço total das doses pedidas

PedidosTotalAPagar – permite obter o total a pagar por pedido.

DiasCorrenteMesCarneEPeixe - permite obter em que dias do corrente mês é que foi servido o prato de peixe “P” juntamente com o prato de carne “C”.

```
USE [michelinStar]
GO

/***** Object: View [dbo].[ EmentadeHoje]    Script Date: 07/06/2022 22:35:35
*****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE VIEW [dbo].[Ementadehoje ]
AS
SELECT          e.ementaID, e.dataEmenta, e.descricaoEmenta, p.tipoPrato,
p.nomePrato, p.preco
FROM            dbo.Ementa AS e INNER JOIN
                dbo.PratoEmenta AS f ON e.ementaID = f.ementaID INNER
JOIN
                dbo.Prato AS p ON f.pratoID = p.pratoID
WHERE           (e.dataEmenta = CONVERT([date], GETDATE()))
GO
```

Excerto SQL 22 - View EmentadeHoje


```

USE [michelinStar]
GO

/***** Object: View [dbo].[ProdutosEmentaAmanha]    Script Date: 08/06/2022
00:25:22 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE VIEW [dbo].[ProdutosEmentaAmanha]
AS
SELECT      e.dataEmenta, i.nomeIngrediente
FROM        dbo.Ementa AS e INNER JOIN
              dbo.PratoEmenta AS pe ON e.ementaID = pe.ementaID INNER
JOIN
              dbo.Prato AS p ON pe.pratoID = p.pratoID INNER JOIN
              dbo.IngredientePrato AS ip ON p.pratoID = ip.pratoID
INNER JOIN
              dbo.Ingrediente AS i ON ip.ingredienteID =
i.ingredienteID
WHERE       (e.dataEmenta = CONVERT([date], GETDATE() + 1))
GO

```

Excerto SQL 23 - View ProdutosEmentaAmanha

```

USE [michelinStar]
GO

/***** Object: View [dbo].[PedidosDosClientes]    Script Date: 08/06/2022
15:55:24 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE VIEW [dbo].[PedidosDosClientes]
AS
SELECT      ped.pedidoID AS [Numero do Pedido], ped.empregadoID AS [Numero do
Empregado], e.primeiroNome AS [Nome do Empregado], ped.mesaID AS Mesa, ped.obs,
p.nomePrato AS [Nome do Prato],
              prd.quantidadePratoPedido AS Doses
FROM        dbo.Pedido AS ped INNER JOIN
              dbo.Empregado AS e ON ped.empregadoID = e.empregadoID
INNER JOIN
              dbo.PratoPedido AS prd ON ped.pedidoID = prd.pedidoID
INNER JOIN
              dbo.Prato AS p ON prd.pratoID = p.pratoID
GO

```

Excerto SQL 24 - PedidosDosClientes

```

USE [michelinStar]
GO

/***** Object: View [dbo].[PedidosComDescritivoEPreco]    Script Date:
08/06/2022 15:29:40 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE VIEW [dbo].[PedidosComDescritivoEPreco]
AS
SELECT      ped.pedidoID, ped.empregadoID, p.nomePrato AS [Nome do Prato],
prd.quantidadePratoPedido AS Doses, p.preco AS [Preço a dose], p.preco *
prd.quantidadePratoPedido AS Preço
FROM        dbo.Pedido AS ped INNER JOIN
            dbo.PratoPedido AS prd ON ped.pedidoID = prd.pedidoID
INNER JOIN
            dbo.Prato AS p ON prd.pratoID = p.pratoID
GO

```

Excerto SQL 25 – View PedidosComDescritivoEPreco

```

USE [michelinStar]
GO

/***** Object: View [dbo].[PedidosTotalAPagar]    Script Date: 08/06/2022
15:45:47 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE VIEW [dbo].[PedidosTotalAPagar]
AS
SELECT      ped.pedidoID, SUM(prd.quantidadePratoPedido) AS [Total de Pratos],
SUM(p.preco * prd.quantidadePratoPedido) AS [Preço Total]
FROM        dbo.Pedido AS ped INNER JOIN
            dbo.PratoPedido AS prd ON ped.pedidoID = prd.pedidoID
INNER JOIN
            dbo.Prato AS p ON prd.pratoID = p.pratoID
GROUP BY   ped.pedidoID
GO

```

Excerto SQL 26 - View PedidosTotalAPagar

```

USE [michelinStar]
GO

/***** Object: View [dbo].[DiasCorrenteMesCarneEPeixe]    Script Date:
09/06/2022 14:14:57 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE VIEW [dbo].[DiasCorrenteMesCarneEPeixe]
AS
SELECT DISTINCT day(e.dataEmenta) AS DAYS, MONTH(GETDATE()) AS MONTH
FROM      Ementa e, PratoEmenta f, Prato p
WHERE      (p.tipoPrato = 'P') AND (e.ementaID = f.ementaID AND p.pratoID =
f.pratoID) AND MONTH(e.dataEmenta) = MONTH(GETDATE())
INTERSECT
SELECT DISTINCT day(e.dataEmenta) AS DAYS, MONTH(GETDATE()) AS MONTH
FROM      Ementa e, PratoEmenta f, Prato p
WHERE      (p.tipoPrato = 'C') AND (e.ementaID = f.ementaID AND p.pratoID =
f.pratoID) and MONTH(e.dataEmenta) = MONTH(GETDATE())
GO

```

Excerto SQL 27 - View DiasCorrenteMesCarneEPeixe

7. Conclusões e Trabalho Futuro

Durante a realização deste projeto surgiram várias dúvidas que obrigaram a procura de respostas para os problemas. Permitiu conscientizar para a complexidade na criação e manutenção de uma BD.

O resultado esperado foi obtido, obtendo-se uma BD que o grupo acredita que colmataria as necessidades do restaurante, havendo ainda margem para a melhoria da mesma.

O grupo espera que no futuro consiga aumentar o conhecimento no domínio de BD.

Bibliografia

Connolly, T., & Begg, C. (s.d.). *Database Systems A Pratical Approach to Design, Implementation and Management*. Sixth Edition.

Referências WWW

- [01] <https://docs.microsoft.com/en-us/sql>
Página com documentação oficial da ultima versão do SQL Server
- [02] <https://pt.stackoverflow.com/>
Site de perguntas e respostas sobre programação

Lista de Siglas e Acrónimos

BD	Base de Dados
OLTP	<i>On-Line Analytical Processing</i>
SGBD	Sistema de Gestão de Base de Dados
ER	Entidade-Relação
SQL	Structured Query Language
PK	Primary Key
FK	Foreign Key
NIF	Número de Identificação Fiscal

Anexos

1. michelinStar.bak
2. bd.vpp