



ESCOLA  
SUPERIOR  
DE TECNOLOGIA  
E GESTÃO

## Unidade Curricular de Base de Dados

Ano Letivo de 2021/2022

### Base de Dados Restaurante Michelin Star

Grupo 9

Rui Neto, 8200321

Simão Santos, 8200322

Junho, 2022

Data de Receção	
Responsável	
Avaliação	
Observações	

# Agradecimentos

Gostaríamos de agradecer ao Professor Vasco Santos pela paciência, competência científica e por todas as correções, críticas, avaliações e sugestões levantadas durante a execução do Trabalho Prático.

## Resumo

Este Trabalho Prático foi realizado no âmbito da Unidade Curricular de Base de Dados, com o intuito de colocar em prática o desenho e implementação de uma base dados direcionada à gestão de um restaurante.

Esta base de dados, de suporte ao restaurante, veio proporcionar uma maior eficácia na gestão e no tratamento de processos. De um modo geral, a base de dados deveria suportar a elaboração e impressão das ementas diárias, pedidos dos clientes, confeção dos pratos das ementas, e gestão e pagamento das contas.

O desenho foi desenvolvido com o auxílio da metodologia proposta pelo professor, apresentada no Livro “Database Systems - A Practical Approach to Design, Implementation and Management”. A metodologia segue uma abordagem estruturada que usa procedimentos, técnicas, ferramentas e documentação de forma a facilitar o processo de desenho, e é subdividida em três grandes tópicos: Desenho Conceptual, Desenho Lógico e Desenho Físico.

Relativamente à implementação, foi usado o “Microsoft SQL Server Management Studio 18”, ferramenta esta também proposta pelo professor.

Todos os objetivos propostos foram atingidos não existindo inconsistência de dados.

# Índice

1. Introdução.....	7
1.1. Contextualização e Caso de Estudo .....	7
1.2. Motivação e Objetivos .....	8
1.2.1. <i>Mission statement</i> .....	8
1.2.2. <i>Mission objectives</i> .....	8
1.3. Estrutura do Relatório.....	9
2. Desenvolvimento.....	10
2.1. Desenho Conceptual .....	10
2.1.1. Identificação dos tipos de entidades.....	10
2.1.2. Identificação das relações entre entidades .....	10
2.1.3. Identificação e associação de atributos com entidades e relações.....	11
2.1.4. Determinar domínio dos atributos .....	14
2.1.5. Determinar chaves primárias .....	18
2.1.6. Validar modelo conceptual em relação a transações do utilizador .....	18
2.2. Desenho Lógico .....	19
2.2.1. Derivar relações para o Modelo Lógico .....	19
2.2.2. Validar relações com recurso à normalização .....	22
2.2.3. Diagrama do Modelo Lógico .....	38
2.2.4. Validar relações em relação a transações do utilizador .....	39
2.2.5. Verificar restrições de integridade .....	39
2.3. Modelo Físico .....	41
2.3.1. Diagrama do Modelo Físico .....	41
2.3.1. Traduzir o modelo Lógico de dados para o DBMS destino .....	43
2.3.1.1. Desenho de Relações Base e <i>General Constraints</i> .....	43
2.3.1.2. Desenho da representação de dados derivados .....	68
2.3.2. Análise de Transações.....	76
2.3.2.1. <i>Stored Procedures</i> e Funções que satisfazem as transações da <i>View Pedido</i> ...	77
2.3.2.2. Outras Transações de Utilizador relevantes para o negócio .....	88
2.3.3. Desenho de <i>User Views</i> .....	97
3. Conclusões e Trabalho Futuro.....	101

4. Bibliografia.....	102
5. Referências WWW .....	103
6. Lista de Siglas e Acrónimos .....	104

# Índice de Figuras

Figura 1 - Ilustração do diagrama E/R .....	11
Figura 2 - Diagrama E/R melhorado .....	18
Figura 3 - Derivação da relação Funcionario Regista Pedido .....	19
Figura 4 - Derivação da relação Funcionario Supervisiona Funcionario .....	19
Figura 5 - Derivação da relação Pedido inclui Produto.....	19
Figura 6 - Derivação da relação Ementa Contem Produtos .....	20
Figura 7 - Derivação da relação Confeccionado Contem Ingredientes .....	20
Figura 8 – Derivação da relação Produto pode ser Artigo ou Confeccionado .....	20
Figura 9 - Resultado das relações para o Modelo Lógico .....	21
Figura 10 - Documento 'Produtos da Ementa' .....	23
Figura 11 - Diagrama de dependências 'Produtos da ementa'.....	24
Figura 12 - Documento 'Receita do Confeccionado'.....	27
Figura 13 - Diagrama de dependências 'Receita do Confeccionado' .....	28
Figura 14 - Documento 'Recursos Humanos' .....	31
Figura 15 - Diagrama de dependências 'Recursos Humanos'.....	32
Figura 16 - Documento 'Pedido' .....	34
Figura 17 - Diagrama de dependências 'Pedido' .....	35
Figura 18 – Resultado das relações para o modelo Lógico.....	38
Figura 19 - Restrições de Integridade Referencial.....	39
Figura 20 - Diagrama do Modelo Físico .....	41
Figura 21 – DBDL para Artigo pt. 1 .....	43
Figura 22 – DBDL para Artigo pt. 2 .....	44
Figura 23 - DBDL para Artigo pt. 3.....	44
Figura 24 - DBDL para Cargo .....	45
Figura 25 - DBDL para Confeccionado pt.1 .....	46
Figura 26 - DBDL para Confeccionado pt.2 .....	47
Figura 27 - DBDL para ConfeccionadoIngredient pt.1.....	48
Figura 28 - DBDL para ConfeccionadoIngredient pt.2.....	49
Figura 29 - DBDL para Ementa .....	50
Figura 30 - DBDL para EmentaArtigo .....	51
Figura 31 - DBDL para EmentaConfeccionado .....	52
Figura 32 - DBDL para EstadoCivil .....	53
Figura 33 - DBDL para Funcionario pt.1 .....	54
Figura 34 - DBDL para Funcionario pt.2 .....	55
Figura 35 - DBDL para Funcionario pt.3 .....	56
Figura 36 - DBDL para GrupoAlimentar .....	57
Figura 37 - DBDL para Ingrediente .....	58
Figura 38 - DBDL para Localizacao .....	59

Figura 39 - DBDL para MetodoPagamento.....	60
Figura 40 - DBDL para Pedido pt.1 .....	61
Figura 41 - DBDL para Pedido pt.2 .....	62
Figura 42 - DBDL para PedidoArtigo pt.1 .....	63
Figura 43 - DBDL para PedidoArtigo pt.2 .....	64
Figura 44 - DBDL para PedidoConfeccionado pt.1.....	65
Figura 45 – DBDL para PedidoConfeccionado pt.2.....	66
Figura 46 - DBDL para TipoProduto.....	66
Figura 47 - DBDL para TipoTaxa .....	67
Figura 48 - Trigger PedidoArtigo - HandleRowDelete.....	68
Figura 49 - Trigger PedidoArtigo - HandleRowInsert.....	69
Figura 50 – Trigger PedidoArtigo - HandleRowUpdate pt.1 .....	70
Figura 51 - Trigger PedidoArtigo - HandleRowUpdate pt.2 .....	71
Figura 52 - Trigger PedidoConfeccionado - HandleRowDelete.....	72
Figura 53 - Trigger PedidoConfeccionado - HandleRowInsert.....	73
Figura 54 - Trigger PedidoConfeccionado – HandleRowUpdate pt.1.....	74
Figura 55 - Trigger PedidoConfeccionado - HandleRowUpdate pt.2 .....	75
Figura 56 – Transações de Utilizador associadas à View Pedido .....	76
Figura 57 – Stored Procedure InsertPedido pt.1.....	77
Figura 58 - Stored Procedure InsertPedido pt.2 .....	78
Figura 59 - Stored Procedure ClosePedido pt.1 .....	79
Figura 60 - Stored Procedure ClosePedido pt.2 .....	80
Figura 61 - Stored Procedure DeletePedido pt.1 .....	81
Figura 62 – Stored Procedure DeletePedido pt.2 .....	82
Figura 63 - Stored Procedure AddConfeccionadoToPedido pt.1 .....	83
Figura 64 - Stored Procedure AddConfeccionadoToPedido pt.2 .....	84
Figura 65 - Stored Procedure AddArtigoToPedido pt.1 .....	85
Figura 66 - Stored Procedure AddArtigoToPedido pt.2 .....	86
Figura 67 – Função getFaturaFromPedido pt.1 .....	87
Figura 68 - Função getEmentaAmanha .....	88
Figura 69 - Output da Função getEmentaAmanha .....	89
Figura 70 - Função getEmentaHoje .....	90
Figura 71 - Output da Função getEmentaHoje .....	91
Figura 72 - Função getFaturaFromPedido .....	92
Figura 73 - Output da Função getFaturaFromPedido .....	93
Figura 74 - Função getIngredientesAmanha.....	93
Figura 75 - Output da Função getIngredientesAmanha.....	93
Figura 76 – Função whatWasSoldBetweenTwoDates.....	94
Figura 77 - Output da Função whatWasSoldBetweenTwoDates.....	94
Figura 78 – Função onWhichDayWereSoldTogether .....	95

Figura 79 - Select da Função onWichDayWereSoldTogether .....	96
Figura 80 - Output da Função onWichDayWereSoldTogether .....	96
Figura 81 - User View Artigos .....	98
Figura 82 - Output User View Artigos .....	98
Figura 83 - User View Funcionarios .....	99
Figura 84 - Output User View Funcionarios .....	99
Figura 85 - User View Pedidos Fechados.....	99
Figura 86 - Output User View Pedidos Fechados.....	100
Figura 87 - User View Faturado Mensalmente .....	100
Figura 88 - Output User View Faturado Mensalmente.....	100
Figura 89 - Diagrama Cronológico e Dificuldades do Grupo .....	101

# Índice de Tabelas

Tabela 1 - Identificação dos tipos de entidades .....	10
Tabela 2 - Identificação das relações entre entidades .....	10
Tabela 3 - Identificação e associação de atributos com entidades ou relações.....	14
Tabela 4 - Forma não normalizada 'Produtos da ementa' .....	23
Tabela 5 - Primeira Forma normal 'Produtos da ementa' .....	24
Tabela 6 - Segunda Forma normal 'Produtos da ementa' - Ementa .....	24
Tabela 7 - Segunda Forma normal 'Produtos da ementa' - Produto .....	25
Tabela 8 - Segunda Forma normal 'Produtos da ementa' - EmentaProduto.....	25
Tabela 9 - Terceira Forma normal 'Produtos da ementa' - Ementa.....	25
Tabela 10 - Terceira Forma normal 'Produtos da ementa' - Produto .....	26
Tabela 11 - Terceira Forma normal 'Produtos da ementa' - EmentaProduto .....	26
Tabela 12 - Terceira Forma normal 'Produtos da ementa' - TipoProduto .....	26
Tabela 13 - Forma não normalizada 'Receita do Confeccionado' .....	27
Tabela 14 - Primeira Forma normal 'Receita do Confeccionado' .....	28
Tabela 15 - Segunda Forma normal 'Receita do Confeccionado' - Confeccionado .....	28
Tabela 16 - Segunda Forma normal 'Receita do Confeccionado' - Ingrediente.....	29
Tabela 17 - Segunda Forma normal 'Receita do Confeccionado' - ConfeccionadoIngrediente ....	29
Tabela 18 - Terceira Forma normal 'Receita do Confeccionado' - Confeccionado.....	29
Tabela 19 - Terceira Forma normal 'Receita do Confeccionado' - Ingrediente .....	30
Tabela 20 - Terceira Forma normal 'Receita do Confeccionado' - ConfeccionadoIngrediente.....	30
Tabela 21 - Terceira Forma normal 'Receita do Confeccionado' – GrupoAlimentar .....	30
Tabela 22 - Forma não normalizada 'Recursos Humanos' .....	31
Tabela 23 - Primeira Forma normal 'Recursos Humanos' .....	32
Tabela 24 - Terceira Forma normal 'Recursos Humanos' – Funcionario .....	33
Tabela 25 - Terceira Forma normal 'Recursos Humanos' – Localizacao .....	33
Tabela 26 - Terceira Forma normal 'Recursos Humanos' – Cargo .....	33
Tabela 27 - Terceira Forma normal 'Recursos Humanos' – EstadoCivil .....	33
Tabela 28 - Forma não normalizada 'Pedido' .....	34
Tabela 29 - Primeira Forma normal 'Pedido'.....	35
Tabela 30 - Segunda Forma normal 'Pedido' - Pedido .....	35
Tabela 31 - Segunda Forma normal 'Pedido' - Produto.....	36
Tabela 32 - Segunda Forma normal 'Pedido' - PedidoProduto .....	36
Tabela 33 - Terceira Forma normal 'Pedido' - Pedido .....	36
Tabela 34 - Terceira Forma normal 'Pedido' - Produto.....	36
Tabela 35 - Terceira Forma normal 'Pedido' - PedidoProduto.....	37
Tabela 36 - Terceira Forma normal 'Pedido' – TipoTaxa.....	37
Tabela 37 - Terceira Forma normal 'Pedido' – MetodoPagamento .....	37
Tabela 38 - Transações e relações de referência cruzada.....	76
Tabela 39 - User Views principais da Base de dados Michelin Star.....	97

# 1. Introdução

## 1.1. Contextualização e Caso de Estudo

A gerência do restaurante Michelin Star verificou uma decadência, em termos de qualidade:

- Das ementas: Já não eram tão bem preparadas devido à desorganização na gestão dos ingredientes e confeccionados. Por causa disso, também não estavam a ter a melhor apresentação;
- Dos pedidos dos clientes: Demoravam a ser processados e isso levava ao atraso na entrega dos mesmos;
- Das contas: Demoravam a ser calculadas e, em algumas situações, apresentavam erros que deixavam os clientes pouco satisfeitos e a ter que verificar a conta mais que uma vez.

Desta forma, e com base num processo de análise executado por uma empresa de consultoria, a gerência tomou medidas e decidiram criar uma base de dados de suporte ao restaurante. Esta deveria ser capaz de:

- Elaborar e imprimir ementas diárias – todos os dias são elaboradas ementas diferentes de acordo com produtos frescos encomendados no mercado local;
- Registar os pedidos dos clientes;
- Confeccionar os pratos das ementas – a cozinha para satisfazer os pratos das ementas precisa de elaborar previamente uma lista de necessidades pormenorizada com os produtos necessários à confeção da ementa do dia seguinte, e com base na definição das ementas essa lista deverá ser gerada automaticamente;
- Gerir as contas – com base nos pedidos efetuados pelos clientes as contas devem ser automaticamente calculadas o que permite elaborar o talão de cobrança e com o pagamento do cliente deve ser emitido o respetivo recibo.

Este projeto passa pelo desenho e implementação dessa base de dados.

## **1.2. Motivação e Objetivos**

### **1.2.1. *Mission statement***

O Objetivo da Base de Dados do restaurante Michelin Star é armazenar os dados relativos à gestão e processos do restaurante, facilitando assim o trabalho às diversas entidades inseridas no mesmo.

### **1.2.2. *Mission objectives***

Gerir (inserir, editar e eliminar) dados das Ementas.

Gerir (inserir, editar e eliminar) dados dos Artigos.

Gerir (inserir, editar e eliminar) dados dos Confeccionados.

Gerir (inserir, editar e eliminar) dados dos Pedidos.

Gerir (inserir, editar e eliminar) dados dos Funcionários.

Gerir (inserir, editar e eliminar) dados dos Ingredientes.

Possibilitar a pesquisa de Ementas.

Possibilitar a pesquisa de Artigos.

Possibilitar a pesquisa de Confeccionados.

Possibilitar a pesquisa de Pedidos.

Possibilitar a pesquisa de Funcionários.

Possibilitar a pesquisa de Ingredientes.

Visualizar Ingredientes de um Confeccionado.

Visualizar Confeccionados de uma Ementa.

Visualizar Artigos de uma Ementa.

Visualizar Confeccionados de um Pedido.

Visualizar Artigos de um Pedido.

Visualizar Funcionário de um Pedido.

## 1.3. Estrutura do Relatório

### Desenvolvimento

Tópico onde será encontrado o desenvolvimento do Projeto incluindo:

- Desenho Conceptual
  - Tópico destinado à identificação dos tipos de entidades, identificação das relações entre entidades, identificação e associação de atributos com entidades e relações, determinação do domínio dos atributos, determinação das chaves primárias, e validação do modelo conceptual em relação a transações do utilizador.
- Desenho Lógico
  - Tópico destinado à derivação das relações para o Modelo Lógico, validação das relações com recurso à normalização, diagrama do Modelo Lógico, validação das relações em relação a transações do utilizador, e verificação de restrições de integridade.
- Modelo Físico
  - Tópico destinado ao diagrama do Modelo Físico, tradução do modelo Lógico de dados para o DBMS destino, desenho de Relações Base e *General Constraints*, desenho da representação de dados derivados, análise de Transações, *Stored Procedures* e Funções que satisfazem as transações de uma View, e desenho de User Views.

### Conclusões e Trabalho Futuro

Tópico onde são abordadas as dificuldades, facilidades, conclusões e futuros trabalhos relativos ao projeto.

### Bibliografia

Tópico destinado à identificação das fontes que auxiliaram no desenvolvimento do projeto.

### Referências WWW

Tópico destinado à identificação das páginas Web que auxiliaram no desenvolvimento do projeto.

### Lista de Siglas e Acrónimos

Tópico destinado à listagem de siglas e acrónimos presentes ao longo de todo o documento.

## 2. Desenvolvimento

### 2.1. Desenho Conceptual

#### 2.1.1. Identificação dos tipos de entidades

Nome da entidade	Descrição	Outros nomes	Ocorrências
Ementa	Termo que descreve uma opção de ementa para um determinado dia da semana.	N.A.	Em cada dia da semana é apresentada uma ementa por parte do restaurante.
Confeccionado	Termo que descreve uma opção da ementa confeccionada no restaurante.	OpçõesDaCasa	Cada ementa é constituída por opções confeccionadas no restaurante.
Artigo	Termo que descreve uma opção da ementa importada (ex. Coca-cola, gelado Olá).	Importados	Cada ementa é constituída por opções importadas.
Ingrediente	Termo que descreve um ingrediente integrante de um confeccionado.	N.A.	Cada opção confeccionada no restaurante é constituída por vários ingredientes.
Pedido	Termo que descreve os pedidos de uma determinada mesa.	N.A.	Os pedidos mantêm um histórico daquilo que é consumido numa determinada mesa.
Funcionario	Termo que descreve um funcionário que opera nas instalações do restaurante.	Empregado, Chefe de cozinha, Cozinheiro	A cada pedido é atribuído um funcionário.

Tabela 1 - Identificação dos tipos de entidades

#### 2.1.2. Identificação das relações entre entidades

Nome da Entidade	Multiplicidade	Relação	Multiplicidade	Nome de Entidade
Funcionario	1..1	Regista	0..*	Pedido
Funcionario	1..1	Supervisiona	0..*	Funcionario
Confeccionado	0..*	Contem	1..*	Ingrediente
Pedido	0..*	Inclui	1..*	Artigo
Pedido	0..*	Inclui	1..*	Confeccionado
Ementa	0..*	Contem	1..*	Artigo
Ementa	0..*	Contem	1..*	Confeccionado

Tabela 2 - Identificação das relações entre entidades

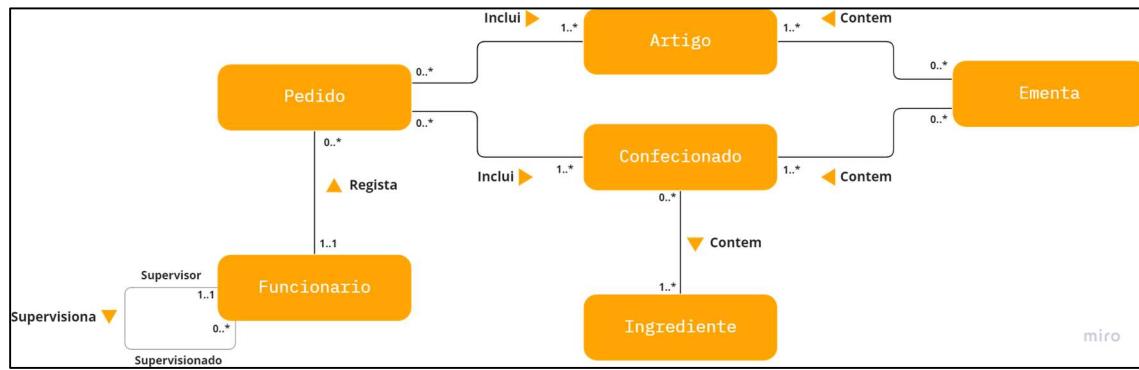


Figura 1 - Ilustração do diagrama E/R

### 2.1.3. Identificação e associação de atributos com entidades e relações

Nome da Entidade	Atributos	Descrição	Tipo de dados e tamanho	Nulls	Multi-valued
Ementa	diaSemana	Identificador único de uma Ementa.	1 tinyint	Não	Não
	eObs	Campo de observações relativas à Ementa.	1000 caracteres	Sim	Não
Confeccionado	produtoNo	Identificador único de um Confeccionado.	1 int	Não	Não
	prodNome	Nome do Confeccionado.	30 caracteres	Não	Não
	prodPreco	Preço em euros do Confeccionado.	Decimal com 5 de precisão e 2 de escala	Não	Não
	prodTipoTaxa	Tipo de taxa a ser atribuída ao Confeccionado.	1 int	Não	Não
	prodDescricao	Descrição do Confeccionado.	150 caracteres	Sim	Não
	prodCalorias	Quantidade calórica em Kcal do Confeccionado.	1 smallint	Não	Não
	prodTipo	Secção da ementa à qual o Confeccionado se insere.	1 int	Não	Não

	prodComGluten	Indica se o Confeccionado contém glúten ou não.	1 bit	Não	Não
	confProcedimento	Indica o procedimento para a confeção do confeccionado.	1000 caracteres	Sim	Não
	ativo	Campo que indica se o Confeccionado está ativo ou eliminado.	1 bit	Não	Não
Artigo	produtoNo	Identificador único de um Artigo.	1 int	Não	Não
	prodNome	Nome do Artigo.	30 caracteres	Não	Não
	prodPreco	Preço em euros do Artigo.	Decimal com 5 de precisão e 2 de escala	Não	Não
	prodTipoTaxa	Tipo de taxa a ser atribuída ao Artigo.	1 int	Não	Não
	prodDescricao	Descrição do Artigo.	150 caracteres	Sim	Não
	prodCalorias	Quantidade calórica em Kcal do Artigo.	1 smallint	Não	Não
	prodTipo	Secção da ementa à qual o Artigo se insere.	1 int	Não	Não
	prodComGluten	Indica se o Artigo contém glúten ou não.	1 bit	Não	Não
	ativo	Campo que indica se o Artigo está ativo ou eliminado	1 bit	Não	Não
Ingrediente	ingredienteNo	Identificador único de um Ingrediente.	1 int	Não	Não
	ingNome	Nome do Ingrediente	50 caracteres	Não	Não
	ingGruAlim	Grupo alimentar ao qual o Ingrediente pertence.	1 int	Não	Não
	ativo	Campo que indica se o Ingrediente está ativo ou eliminado.	1 bit	Não	Não
Pedido	pedidoNo	Identificador único de um Pedido.	1 int	Não	Não
	pedNumeroMesa	Identifica a mesa à qual o Pedido está associado.	1 int	Não	Não

	pedObs	Campo de observações relativas ao Pedido.	1000 caracteres	Sim	Não
	pedPreco	Preço total em euros do Pedido.	Decimal com 8 de precisão e 2 de escala	Não	Não
	pedDataCriacao	Campo que indica a data de criação do Pedido.	Date	Não	Não
	pedEstado	Campo que indica o estado atual do Pedido.	1 caracter	Não	Não
	pedNumeroContribuinte	Campo que indica o número de contribuinte associado ao Pedido.	9 dígitos	Sim	Não
	pedMetodoPagamento	Campo que indica o método de pagamento.	1 int	Sim	Não
	ativo	Campo que indica se o Pedido está ativo ou eliminado.	1 bit	Não	Não
Funcionario	fPrimeiroNome	Primeiro nome de um Funcionário.	20 caracteres	Não	Não
	fUltimoNome	Último nome de um Funcionário.	20 caracteres	Não	Não
	fDob	Data de nascimento de um Funcionário.	Date	Não	Não
	fRua	Rua de um Funcionário.	60 caracteres	Não	Não
	fNumPorta	Número da porta de um Funcionário.	1 int	Não	Não
	fFreguesia	Freguesia de um Funcionário.	60 caracteres	Não	Não
	fCidade	Cidade de um Funcionário.	50 caracteres	Não	Não
	fCodPostal	Código postal de um Funcionário.	8 dígitos	Não	Não
	fEmail	Email de um Funcionário.	40 caracteres	Não	Não
	fNumTelemovel	Número de telemóvel de um Funcionário.	9 dígitos	Não	Não
	fSalarioHora	Salário por hora de um Funcionário.	Decimal com 4 de precisão e 2 de escala	Não	Não

	fNif	Número de identificação fiscal de um Funcionário.	9 dígitos	Não	Não
	fCargo	Cargo de um Funcionário.	1 int	Não	Não
	fGenero	Género de um Funcionário.	1 caracter (M, F ou O)	Não	Não
	fEstadoCivil	Estado civil de um Funcionário.	1 int	Não	Não
	ativo	Campo que indica se o Funcionário está ativo ou eliminado.	1 bit	Não	Não
<hr/>					
PedidoProduto	quantidade	Quantidade de um produto num Pedido.	1 smallint	Não	Não
<hr/>					
ConfeccionadoIngrediente	ingQuantidade	Quantidade de um Ingrediente num Confeccionado.	Decimal com 6 de precisão e 3 de escala	Não	Não
	ingUnidadeMedida	Unidade de medida de um Ingrediente num Confeccionado.	5 caracteres	Não	Não

Tabela 3 - Identificação e associação de atributos com entidades ou relações

## 2.1.4. Determinar domínio dos atributos

### Ementa:

- i. O atributo “diaSemana” deve ser um número inteiro  $\geq 1$  e  $\leq 7$  sendo que 1 deverá ser interpretado como “Domingo”.
- ii. O atributo “eObs” deve conter no máximo 1000 caracteres.

**Confeccionado:**

- i. O atributo “prodNo” deve ser um número inteiro com auto incremento de 1 (com seed 5000).
- ii. O atributo “prodNome” deve conter no máximo 30 caracteres.
- iii. O atributo “prodPreco” deve ser um número  $\geq 0,00$  e  $\leq 999,99$ .
- iv. O atributo “prodTipoTaxa” deve ser um número inteiro  $> 0$  sendo que 1 poderá ser equivalente a “Normal”, 2 a “Intermédia”, 3 a “Reduzida”...
- v. O atributo “prodDescricao” deve conter no máximo 150 caracteres.
- vi. O atributo “prodCalorias” deve ser um número inteiro positivo  $> 0$  e  $\leq 9999$ .
- vii. O atributo “prodTipo” deve ser um número inteiro  $> 0$  sendo que 1 poderá ser equivalente a “Entrada”, 2 a “Sobremesa”, 3 a “Bebida”...
- viii. O atributo “prodComGluten” deve ser um bit.
- ix. O atributo “confProcedimento” deve conter no máximo 1000 caracteres.
- x. O atributo “ativo” deve ser um bit sendo que o valor *default* é 1 (ativo).

**Artigo:**

- i. O atributo “prodNo” deve ser um número inteiro com auto incremento de 1 (com seed 1) e  $< 5000$ .
- ii. O atributo “prodNome” deve conter no máximo 30 caracteres.
- iii. O atributo “prodPreco” deve ser um número  $\geq 0,00$  e  $\leq 999,99$ .
- iv. O atributo “prodTipoTaxa” deve ser um número inteiro  $> 0$  sendo que 1 poderá ser equivalente a “Normal”, 2 a “Intermédia”, 3 a “Reduzida”...
- v. O atributo “prodDescricao” deve conter no máximo 150 caracteres.
- vi. O atributo “prodCalorias” deve ser um número inteiro positivo  $\geq 0$  e  $\leq 9999$ .
- vii. O atributo “prodTipo” deve ser um número inteiro  $> 0$  sendo que 1 poderá ser equivalente a “Entrada”, 2 a “Sobremesa”, 3 a “Bebida”...
- viii. O atributo “prodComGluten” deve ser um bit.
- ix. O atributo “ativo” deve ser um bit sendo que o valor *default* é 1 (ativo).

**Ingrediente:**

- i. O atributo “ingredienteNo” deve ser um número inteiro com auto incremento de 1 (com seed 1).
- ii. O atributo “ingNome” deve conter no máximo 50 caracteres.
- iii. O atributo “ingGruAlim” deve ser um número inteiro  $> 0$  sendo que 1 poderá ser equivalente a “Carboidratos”, 2 a “Verduras e Legumes”, 3 a “Frutas”, 4 a “Leite e Derivados”, 5 a “Carnes e Ovos”, 6 a “Leguminosas e Oleaginosas”, 7 a “Óleos e Gorduras”, 8 a “Açucares e Doces”...
- iv. O atributo “ativo” deve ser um bit sendo que o valor *default* é 1 (ativo).

**Pedido:**

- i. O atributo “pedidoNo” deve ser um número inteiro com auto incremento de 1 (com seed 1).
- ii. O atributo “pedNumeroMesa” deve ser um número inteiro  $> 0$ .
- iii. O atributo “pedObs” deve conter no máximo 1000 caracteres.
- iv. O atributo “pedPreco” deve ser um número  $\geq 0,00$  e  $\leq 100.000,00$ , sendo que o valor por defeito é 0.
- v. O atributo “pedDataCriacao” deve ser do tipo *Date*.
- i. O atributo “pedEstado” deve ser um caracter cujos valores permitidos são: “A” ou “F”, sendo que o valor por defeito é “A”.
- vi. O atributo “pedNumeroContribuinte” deve ser um número inteiro positivo com 9 dígitos.
- vii. O atributo “pedMetodoPagamento” deve ser um número inteiro  $> 0$  sendo que 1 poderá ser equivalente a “Dinheiro”, 2 a “Multibanco”, 3 a “MBWay”...
- viii. O atributo “ativo” deve ser um bit sendo que o valor *default* é 1 (ativo).

**Funcionario:**

- ii. O atributo “fPrimeiroNome” deve conter no máximo 20 caracteres.
- iii. O atributo “fUltimoNome” deve conter no máximo 20 caracteres.
- iv. O atributo “fDob” deve ser do tipo *Date*.
- v. O atributo “fRua” deve conter no máximo 60 caracteres.
- vi. O atributo “fNumPorta” deve ser um número inteiro positivo  $> 0$  e  $\leq 9999$ .
- vii. O atributo “fFreguesia” deve conter no máximo 60 caracteres.
- viii. O atributo “fCidade” deve conter no máximo 50 caracteres.
- ix. O atributo “fCodPostal” deve ser um número inteiro positivo com 4 dígitos, seguido de um hífen e um número inteiro positivo com 3 dígitos.
- x. O atributo “fEmail” deve conter x caracteres, um “@”, y caracteres, um “.” e z caracteres pela ordem apresentada. Sendo que  $x + y + z + 2 \leq 40$ .
- xi. O atributo “fNumTelemovel” deve ser um inteiro positivo com 9 dígitos.
- xii. O atributo “fSalarioHora” deve ser um número  $\geq 0,00$  e  $\leq 99,99$ .
- xiii. O atributo “fNif” deve ser um inteiro positivo com 9 dígitos.
- xiv. O atributo “fCargo” deve ser um número inteiro  $> 0$  sendo que 1 poderá ser equivalente a “Chefe de Cozinha”, 2 a “Cozinheiro”, 3 a “Empregado de Mesa”...
- xv. O atributo “fGenero” deve ser um caracter cujos valores permitidos são: “M”, “F” ou “O”.
- xvi. O atributo “fEstadoCivil” deve ser um número inteiro  $> 0$  sendo que 1 poderá ser equivalente a “Solteiro”, 2 a “Casado”, 3 a “Divorciado”, 4 a “Viúvo”, 5 a “Separado”...
- xvii. O atributo “ativo” deve ser um bit sendo que o valor *default* é 1 (ativo).

**PedidoProduto:**

- i. O atributo “quantidade” deve ser um número inteiro positivo.

**ConfeccionadoIngrediente:**

- i. O atributo “ingQuantidade” deve ser um número  $> 0,000$ .
- ii. O atributo “ingUnidadeMedida” deve ser igual a “Kg”, “L” ou “Un”.

## 2.1.5. Determinar chaves primárias

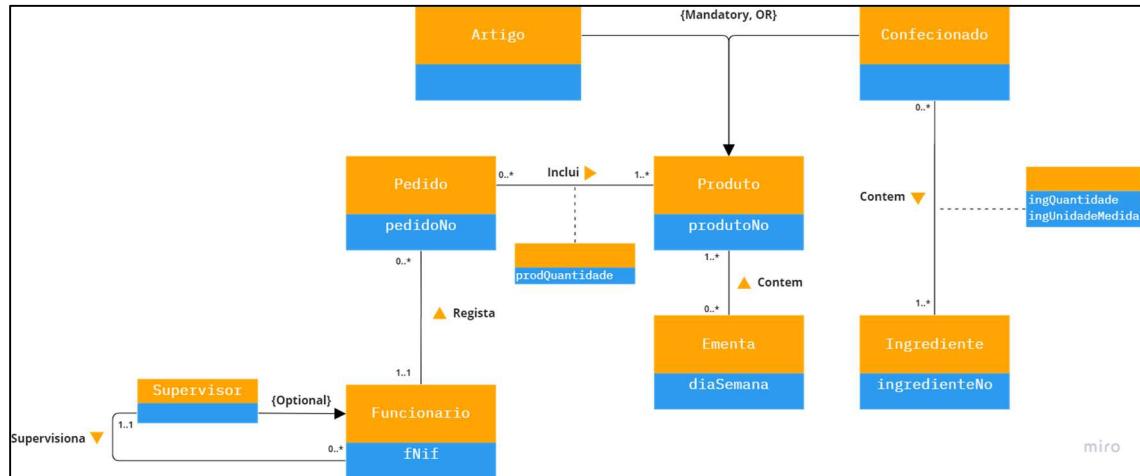


Figura 2 - Diagrama E/R melhorado

## 2.1.6. Validar modelo conceptual em relação a transações do utilizador

### a) Listar os detalhes de todos os pedidos de um determinado funcionário.

Os detalhes dos pedidos são assegurados pela entidade ‘Pedido’ e os detalhes de um funcionário são assegurados pela entidade ‘Funcionario’. Neste caso podemos usar a relação ‘Funcionario’ *Regista* ‘Pedido’ para produzir a lista requerida.

### b) Listar os detalhes dos produtos, por ordem decrescente de frequência em pedidos.

Os detalhes dos produtos são assegurados pela entidade ‘Produto’ e os detalhes dos pedidos são assegurados pela entidade ‘Pedido’. Neste caso podemos usar a relação ‘Pedido’ *Inclui* ‘Produto’ para produzir a lista requerida.

### c) Listar os detalhes dos produtos do tipo ‘Sobremesa’ de uma determinada ementa, ordenados do mais calórico para o menos calórico.

Os detalhes dos produtos são assegurados pela entidade ‘Produto’ e os detalhes das ementas são assegurados pela entidade ‘Ementa’. Neste caso podemos usar a relação ‘Ementa’ *Contem* ‘Produto’ para produzir a lista requerida.

### d) Listar os detalhes dos ingredientes de um determinado confeccionado.

Os detalhes dos ingredientes são assegurados pela entidade ‘Ingrediente’ e os detalhes dos confeccionados são assegurados pela entidade ‘Confeccionado’. Neste caso podemos usar a relação ‘Confeccionado’ *Contem* ‘Ingrediente’ para produzir a lista requerida.

**e) Listar os funcionários supervisionados por um determinado supervisor.**

Os detalhes dos funcionários são assegurados pela entidade ‘Funcionario’ e os detalhes dos supervisores são assegurados pela entidade ‘Funcionario’. Neste caso podemos usar a relação ‘Supervisor’ *Supervisiona* ‘Funcionario’.

## 2.2. Desenho Lógico

### 2.2.1. Derivar relações para o Modelo Lógico

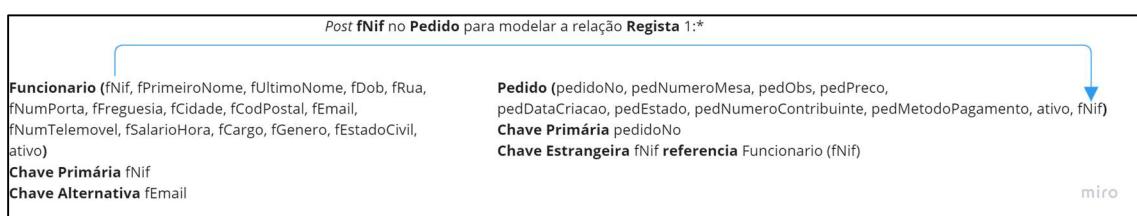


Figura 3 - Derivação da relação Funcionario Regista Pedido

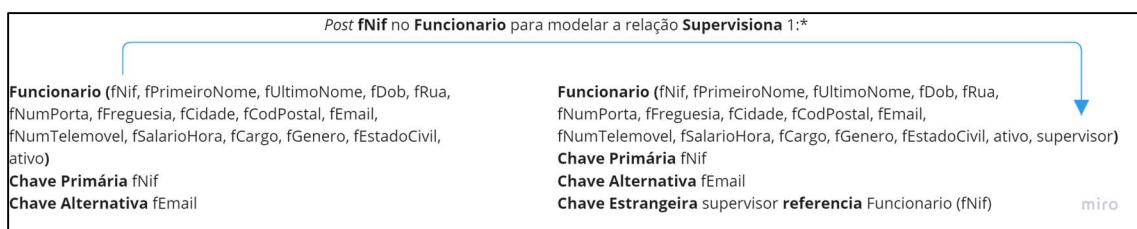


Figura 4 - Derivação da relação Funcionario Supervisiona Funcionario

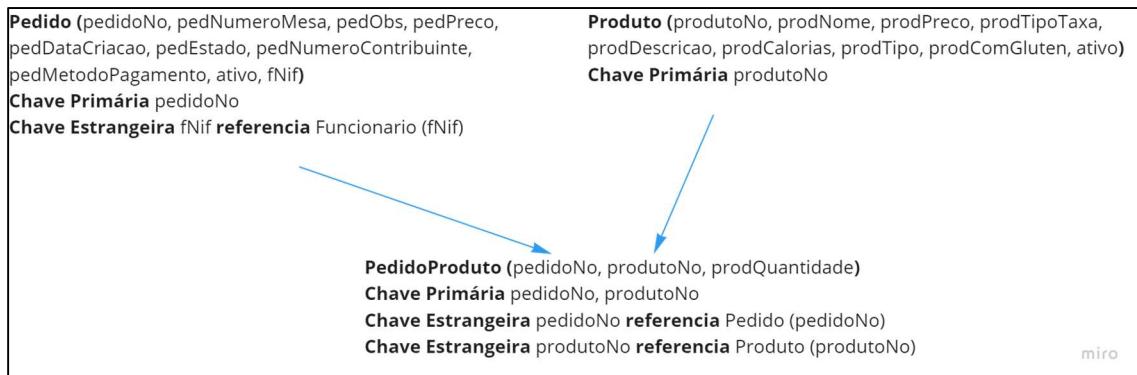


Figura 5 - Derivação da relação Pedido inclui Produto

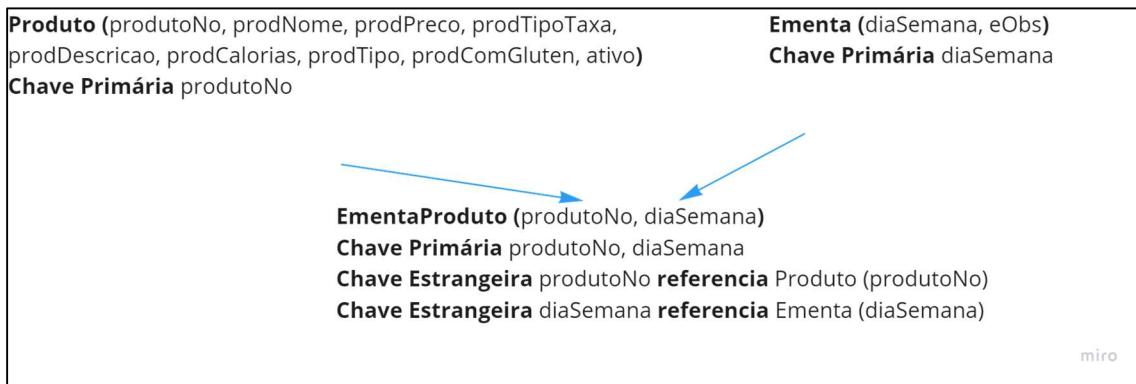


Figura 6 - Derivação da relação Ementa Contem Produtos

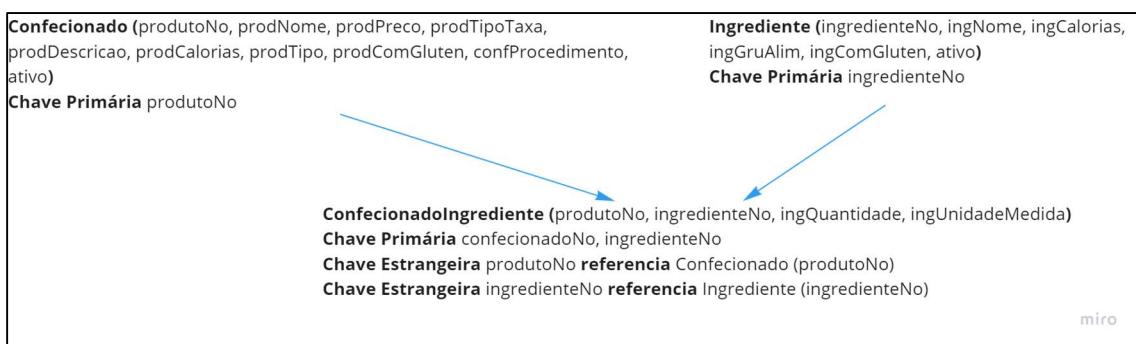


Figura 7 - Derivação da relação Confeccionado Contem Ingredientes

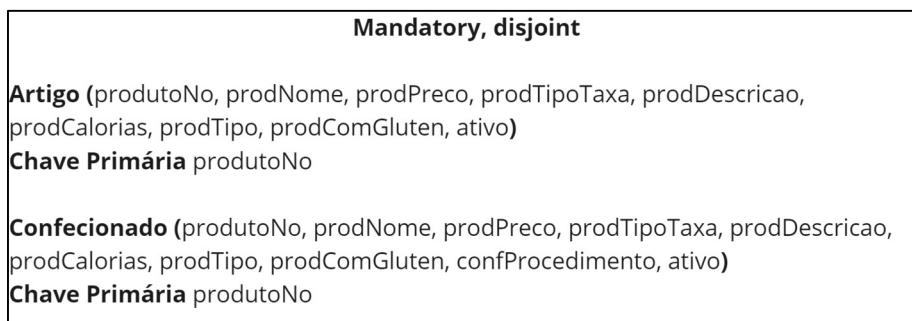


Figura 8 – Derivação da relação Produto pode ser Artigo ou Confeccionado

<p><b>Funcionario</b> (fNif, fPrimeiroNome, fUltimoNome, fDob, fRua, fNumPorta, fFreguesia, fCidade, fCodPostal, fEmail, fNumTelemovel, fSalarioHora, fCargo, fGenero, fEstadoCivil, ativo, supervisor)</p> <p><b>Chave Primária</b> fNif</p> <p><b>Chave Alternativa</b> fEmail</p> <p><b>Chave Estrangeira</b> supervisor <b>referencia</b> Funcionario (fNif)</p>	<p><b>PedidoProduto</b> (pedidoNo, produtoNo, prodQuantidade)</p> <p><b>Chave Primária</b> pedidoNo, produtoNo</p> <p><b>Chave Estrangeira</b> pedidoNo <b>referencia</b> Pedido (pedidoNo)</p> <p><b>Chave Estrangeira</b> produtoNo <b>referencia</b> Produto (produtoNo)</p>
<p><b>Pedido</b> (pedidoNo, pedNumeroMesa, pedObs, pedPreco, pedDataCriacao, pedEstado, pedNumeroContribuinte, pedMetodoPagamento, ativo, fNif)</p> <p><b>Chave Primária</b> pedidoNo</p> <p><b>Chave Estrangeira</b> fNif <b>referencia</b> Funcionario (fNif)</p>	<p><b>Ementa</b> (ementaNo, eObs)</p> <p><b>Chave Primária</b> ementaNo</p>
<p><b>Artigo</b> (produtoNo, prodNome, prodPreco, prodTipoTaxa, prodDescricao, prodCalorias, prodTipo, prodComGluten, ativo)</p> <p><b>Chave Primária</b> produtoNo</p>	<p><b>EmentaProduto</b> (produtoNo, diaSemana)</p> <p><b>Chave Primária</b> produtoNo, diaSemana</p> <p><b>Chave Estrangeira</b> produtoNo <b>referencia</b> Produto (produtoNo)</p> <p><b>Chave Estrangeira</b> diaSemana <b>referencia</b> Ementa (diaSemana)</p>
<p><b>Confeccionado</b> (produtoNo, prodNome, prodPreco, prodTipoTaxa, prodDescricao, prodCalorias, prodTipo, prodComGluten, confProcedimento, ativo)</p> <p><b>Chave Primária</b> produtoNo</p>	<p><b>Ingrediente</b> (ingredienteNo, ingNome, ingGruAlim, ativo)</p> <p><b>Chave Primária</b> ingredienteNo</p>
<p><b>ConfeccionadoIngrediente</b> (produtoNo, ingredienteNo, ingQuantidade, ingUnidadeMedida)</p> <p><b>Chave Primária</b> confeccionadoNo, ingredienteNo</p> <p><b>Chave Estrangeira</b> produtoNo <b>referencia</b> Confeccionado (produtoNo)</p> <p><b>Chave Estrangeira</b> ingredienteNo <b>referencia</b> Ingrediente (ingredienteNo)</p>	

miro

Figura 9 - Resultado das relações para o Modelo Lógico

## **2.2.2. Validar relações com recurso à normalização**

A normalização é uma técnica para produzir um conjunto de relações com propriedades desejáveis dados os requisitos de dados de um negócio.

### **Forma não normalizada:**

Relação onde pode existir mais do que um valor na interseção de uma coluna com uma linha.

### **Primeira Forma normal:**

Relação onde existe unicamente um valor na interseção de uma coluna com uma linha.

### **Segunda Forma normal:**

Uma relação que está na Primeira Forma normal e todos os atributos que não são chaves primárias apresentam uma dependência funcional total da chave primária. Por outras palavras removemos os atributos parcialmente dependentes da relação, colocando-os numa nova relação com o seu determinante.

### **Terceira Forma normal:**

Uma relação que está na Primeira e Segunda Formas normais e que nenhum atributo que não é chave primária apresenta uma dependência transitiva da chave primária. Por outras palavras, a passagem para a Terceira Forma normal consiste na remoção das dependências transitivas.

## Documento 'Produtos da Ementa'

Produtos da Ementa de Domingo	
<b>Número do Confeccionado:</b> 12 <b>Nome:</b> Arroz de Pato <b>Calorias:</b> 700 Kcal <b>Tipo:</b> Prato <b>Gluten:</b> Sim	<b>Descrição:</b> Especialidade da casa
<b>Número do Artigo:</b> 600 <b>Nome:</b> Gelado Olá de Morango <b>Calorias:</b> 200 Kcal <b>Tipo:</b> Sobremesa <b>Gluten:</b> Sim	<b>Descrição:</b> Marca - Olá Gelados Sabor - Morango
<b>Número do Artigo:</b> 700 <b>Nome:</b> Maduro Tinto EA <b>Calorias:</b> 125 Kcal <b>Tipo:</b> Bebida <b>Gluten:</b> Não	<b>Descrição:</b> Marca - EA - Vinho Regional Alentejano Cor - Reserva tinto Capacidade - 750ml Ano de colheita - 2017
<b>Número do Confeccionado:</b> 28 <b>Nome:</b> Moelas <b>Calorias:</b> 125 Kcal <b>Tipo:</b> Entrada <b>Gluten:</b> Sim	<b>Descrição:</b> Iguaria confeccionada no restaurante. Base com um pequeno refogado de cebola e tomate a que se juntam as moelas de galinha ou de pato.
<b>Observações</b> Tempo médio de preparação: 4h Ementa direcionada às carnes	

*Figura 10 - Documento 'Produtos da Ementa'*

### Forma não normalizada:

diaSemana	eObs	produtoNo	prodNome	prodCalorias	prodTipo	prodTipoNome	prodGluten	prodDescricao
7	Tempo médio de...	12	Arroz de Pato	700	3	Prato	Sim	Especialidade...
		600	Gelado Olá de Morango	200	4	Sobremesa	Sim	Marca – Olá...
		700	Maduro Tinto EA	125	5	Bebida	Não	Marca – EA...
		28	Moelas	125	1	Entrada	Sim	Iguaria conf...

*Tabela 4 - Forma não normalizada 'Produtos da ementa'*

### Primeira Forma normal:

diaSemana	eObs	produtoNo	prodNome	prodCalorias	prodTipo	prodTipoNome	prodGluten	prodDescricao
7	Tempo médio de...	12	Arroz de Pato	700	3	Prato	Sim	Especialidade...
7	Tempo médio de...	600	Gelado Olá de Morango	200	4	Sobremesa	Sim	Marca – Olá...
7	Tempo médio de...	700	Maduro Tinto EA	125	5	Bebida	Não	Marca – EA...
7	Tempo médio de...	28	Moelas	200	1	Entrada	Sim	Iguaria conf...

Tabela 5 - Primeira Forma normal 'Produtos da ementa'

### Diagrama de Dependências:

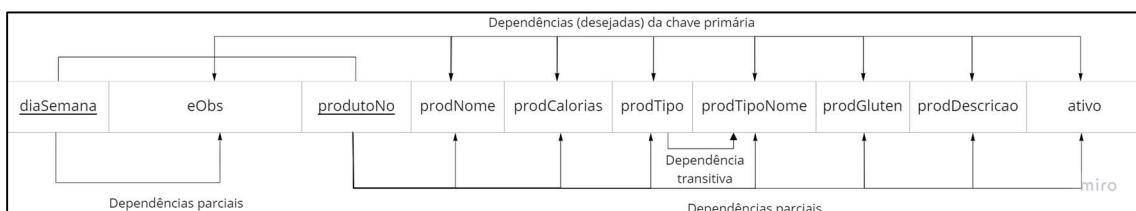


Figura 11 - Diagrama de dependências 'Produtos da ementa'

### Segunda Forma normal:

#### **Ementa**

diaSemana	eObs
7	Tempo médio de...

Tabela 6 - Segunda Forma normal 'Produtos da ementa' - Ementa

## Produto

produtoNo	prodNome	prodCalorias	prodTipo	prodTipoNome	prodGluten	prodDescricao
12	Arroz de Pato	700	3	Prato	Sim	Especialidade...
600	Gelado Olá de Morango	200	4	Sobremesa	Sim	Marca – Olá...
700	Maduro Tinto EA	125	5	Bebida	Não	Marca – EA...
28	Moelas	200	1	Entrada	Sim	Iguaria conf...

Tabela 7 - Segunda Forma normal 'Produtos da ementa' - Produto

## EmentaProduto

diaSemana	produtoNo
7	12
7	600
7	700
7	28

Tabela 8 - Segunda Forma normal 'Produtos da ementa' - EmentaProduto

## Terceira Forma normal:

### Ementa

diaSemana	eObs
7	Tempo médio de...

Tabela 9 - Terceira Forma normal 'Produtos da ementa' - Ementa

## Produto

<u>produtoNo</u>	<u>prodNome</u>	<u>prodCalorias</u>	<u>prodTipo</u>	<u>prodGluten</u>	<u>prodDescricao</u>	<u>ativo</u>
12	Arroz de Pato	700	3	Sim	Especialidade...	1
600	Gelado Olá de Morango	200	4	Sim	Marca – Olá...	1
700	Maduro Tinto EA	125	5	Não	Marca – EA...	1
28	Moelas	200	1	Sim	Iguaria conf...	1

Tabela 10 - Terceira Forma normal 'Produtos da ementa' - Produto

## EmentaProduto

<u>diaSemana</u>	<u>produtoNo</u>
7	12
7	600
7	700
7	28

Tabela 11 - Terceira Forma normal 'Produtos da ementa' - EmentaProduto

## TipoProduto

<u>prodTipo</u>	<u>prodTipoNome</u>	<u>ativo</u>
3	Prato	1
4	Sobremesa	1
5	Bebida	1
1	Entrada	1

Tabela 12 - Terceira Forma normal 'Produtos da ementa' - TipoProduto

## Documento ‘Receita do Confeccionado’

**Receita do Confeccionado nº12**

<p><b>Ingredientes</b></p> <div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;">           Número do Ingrediente: 12            Nome: Pato            Grupo alimentar: 5            Quantidade: 0.10            Unidade de medida: Kg         </div> <div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;">           Número do Ingrediente: 18            Nome: Arroz            Grupo alimentar: 1            Quantidade: 0.05            Unidade de medida: Kg         </div> <div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;">           Número do Ingrediente: 6            Nome: Presunto            Grupo alimentar: 5            Quantidade: 0.03            Unidade de medida: Kg         </div> <div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;">           Número do Ingrediente: 24            Nome: Vinho do Porto            Grupo alimentar: 7            Quantidade: 0.02            Unidade de medida: L         </div> <p>Calorias: 500Kcal Glúten: Sim</p>	<p><b>Procedimento</b></p> <ol style="list-style-type: none"> <li>1. Numa panela colocar o pato em pedaços e acrescentar presunto.</li> <li>2. Juntar 0.02L de Vinho do Porto.</li> <li>3. Regue com água e levar ao lume durante 45 minutos.</li> <li>4. Quando estiver bem cozinhado, retirar o pato do lume e desfiar.</li> </ol> <p>...</p>
--	---

*Figura 12 - Documento ‘Receita do Confeccionado’*

### Forma não normalizada:

prodNo	confProcedimento	prodCalorias	prodComGluten	ingredienteNo	ingNome	ingGruAlim	ingGruAlimNome	ingQuant	ingUniMed
12	1. Numa panela...	500	Sim	12	Pato	5	Carnes e Ovos	0.10	Kg
		500	Sim	18	Arroz	1	Carboidratos	0.05	Kg
		500	Sim	6	Presunto	5	Carnes e Ovos	0.03	Kg
		500	Sim	24	Vinho do Porto	7	Óleos e Gorduras	0.02	L

*Tabela 13 - Forma não normalizada ‘Receita do Confeccionado’*

### Primeira Forma normal:

<u>prodNo</u>	<u>confProcedimento</u>	<u>prodCalorias</u>	<u>prodComGluten</u>	<u>ingredienteNo</u>	<u>ingNome</u>	<u>ingGrupAlim</u>	<u>ingGruAlimNome</u>	<u>ingQuant</u>	<u>ingUniMed</u>
12	1. Numa panela...	500	Sim	12	Pato	5	Carnes e Ovos	0.10	Kg
12	1. Numa panela...	500	Sim	18	Arroz	1	Carboidratos	0.05	Kg
12	1. Numa panela...	500	Sim	6	Presunto	5	Carnes e Ovos	0.03	Kg
12	1. Numa panela...	500	Sim	24	Vinho do Porto	7	Óleos e Gorduras	0.02	L

Tabela 14 - Primeira Forma normal 'Receita do Confeccionado'

### Diagrama de Dependências:

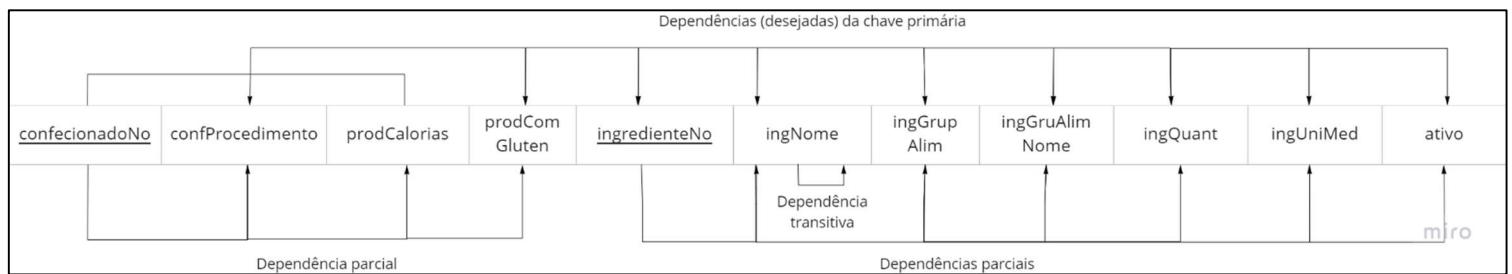


Figura 13 - Diagrama de dependências 'Receita do Confeccionado'

### Segunda Forma normal:

#### Confeccionado

<u>prodNo</u>	<u>confProcedimento</u>
12	1. Numa panela...

Tabela 15 - Segunda Forma normal 'Receita do Confeccionado' - Confeccionado

## Ingrediente

<u>ingredienteNo</u>	ingNome	ingGruAlim	ingGruAlimNome	ingQuant	ingUniMed
12	Pato	5	Carnes e Ovos	0.10	Kg
18	Arroz	1	Carboidratos	0.05	Kg
6	Presunto	5	Carnes e Ovos	0.03	Kg
24	Vinho do Porto	7	Óleos e Gorduras	0.02	L

Tabela 16 - Segunda Forma normal 'Receita do Confeccionado' - Ingrediente

## ConfeccionadoIngrediente

<u>prodNo</u>	<u>ingredienteNo</u>	ingQuant	ingUniMed
12	12	0.10	Kg
12	18	0.05	Kg
12	6	0.03	Kg
12	24	0.02	L

Tabela 17 - Segunda Forma normal 'Receita do Confeccionado' - ConfeccionadoIngrediente

## Terceira Forma normal:

### Confeccionado

<u>prodNo</u>	<u>confProcedimento</u>	<u>prodCalorias</u>	<u>prodComGluten</u>	<u>ativo</u>
12	1. Numa panela...	500	Sim	1

Tabela 18 - Terceira Forma normal 'Receita do Confeccionado' - Confeccionado

## Ingrediente

ingredienteNo	ingNome	ingGruAlim	ingQuant	ingUniMed	ativo
12	Pato	5	0.10	Kg	1
18	Arroz	1	0.05	Kg	1
6	Presunto	5	0.03	Kg	1
24	Vinho do Porto	7	0.02	L	1

Tabela 19 - Terceira Forma normal 'Receita do Confeccionado' - Ingrediente

## ConfeccionadoIngrediente

prodNo	ingredienteNo	ingQuant	ingUniMed
12	12	0.10	Kg
12	18	0.05	Kg
12	6	0.03	Kg
12	24	0.02	L

Tabela 20 - Terceira Forma normal 'Receita do Confeccionado' - ConfeccionadoIngrediente

## GrupoAlimentar

ingGruAlim	ingGruAlimNome	ativo
5	Carnes e Ovos	1
1	Carboidratos	1
7	Óleos e Gorduras	1

Tabela 21 - Terceira Forma normal 'Receita do Confeccionado' – GrupoAlimentar

## Documento 'Recursos Humanos'

<p><b>Recursos Humanos</b></p> <p><b>Rogério Fernandes</b> <i>rfernandesfcp@gmail.com</i></p> <p><b><u>Cozinheiro</u></b></p> <p><b>NIF:</b> 334893454  <b>Data de nascimento:</b> 20-05-1980  <b>Morada:</b> Rua das Fontainhas Nº302, S. Torcato - Porto 4798-332  <b>Contacto:</b> 252 334 543                            918 657 533  <b>Género:</b> Masculino  <b>Estado Civil:</b> Solteiro  <b>Salário por hora:</b> 7.50€</p> <p><b>Paula Silva</b> <i>psilvafcf@gmail.com</i></p> <p><b><u>Empregada de mesa</u></b></p> <p><b>NIF:</b> 748274638  <b>Data de nascimento:</b> 31-10-1985  <b>Morada:</b> Rua Pedro Pires Nº10, Campanhã - Porto 2244-258  <b>Contacto:</b> 252 333 581                            913 001 230  <b>Género:</b> Feminino  <b>Estado Civil:</b> Casada  <b>Salário por hora:</b> 5.50€</p>
--

*Figura 14 - Documento 'Recursos Humanos'*

### Forma não normalizada:

fNif	fPrimeiroNome	fUltimoNome	fDob	fRua	fNumPorta	fFreguesia	fCidade	
334893454	Rogério	Fernandes	20/05/1980	Fontainhas	302	S. Torcato	Porto	
748274638	Paula	Silva	31/10/1985	Pedro Pires	10	Campanhã	Porto	
fCodPostal	fEmail	FNumTelemovel	fSalarioHora	fCargo	fCargo Nome	fGenero	fEstadoCivil	fEstado CivilNome
4798-332	rfernandesfcp@gmail.com	252334543	7.50	3	Cozinheiro	Masculino	1	Solteiro
2244-258	psilvafcf@gmail.com	252333581	5.50	4	Empregado de mesa	Feminino	2	Casado

*Tabela 22 - Forma não normalizada 'Recursos Humanos'*

### Primeira Forma normal:

fNif	fPrimeiroNome	fUltimoNome	fDob	fRua	fNumPorta	fFreguesia	fCidade	
334893454	Rogério	Fernandes	20/05/1980	Fontainhas	302	S. Torcato	Porto	
748274638	Paula	Silva	31/10/1985	Pedro Pires	10	Campanhã	Porto	
fCodPostal	fEmail	fNumTelemovel	fSalarioHora	fCargo	fCargo Nome	fGenero	fEstadoCivil	fEstado Civil Nome
4798-332	rfernandesfcp@gmail.com	918657533	7.50	3	Cozinheiro	Masculino	1	Solteiro
2244-258	psilvafcf@gmail.com	913001230	5.50	4	Empregado de mesa	Feminino	2	Casado

Tabela 23 - Primeira Forma normal 'Recursos Humanos'

### Diagrama de Dependências:

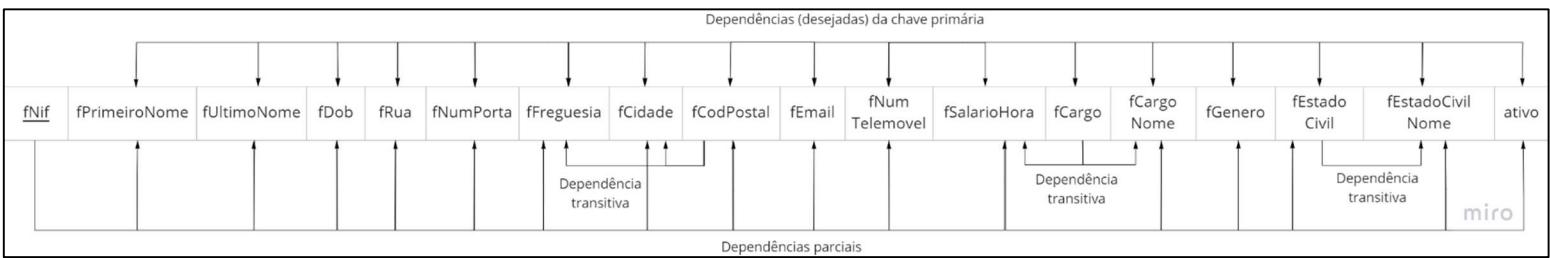


Figura 15 - Diagrama de dependências 'Recursos Humanos'

### Segunda Forma normal:

Dado que só existe um atributo que é chave primária, já nos encontramos na segunda forma normal.

### **Terceira Forma normal:**

#### **Funcionario**

<b>fNif</b>	<b>fPrimeiroNome</b>	<b>fUltimoNome</b>	<b>fDob</b>	<b>fRua</b>	<b>fNumPorta</b>	
334893454	Rogério	Fernandes	20/05/1980	Fontainhas	302	
748274638	Paula	Silva	31/10/1985	Pedro Pires	10	
<b>fCodPostal</b>	<b>fEmail</b>	<b>fNumTelemovel</b>	<b>fCargo</b>	<b>fGenero</b>	<b>fEstadoCivil</b>	<b>ativo</b>
4798-332	rfernandesfcp@gmail.com	918657533	3	Masculino	1	1
2244-258	psilvafcf@gmail.com	913001230	4	Feminino	2	1

*Tabela 24 - Terceira Forma normal 'Recursos Humanos' – Funcionario*

#### **Localizacao**

<b>fCodPostal</b>	<b>fFreguesia</b>	<b>fCidade</b>	<b>ativo</b>
4798-332	S. Torcato	Porto	1
2244-258	Campanhã	Porto	1

*Tabela 25 - Terceira Forma normal 'Recursos Humanos' – Localizacao*

#### **Cargo**

<b>fCargo</b>	<b>fSalarioHora</b>	<b>fCargoNome</b>	<b>ativo</b>
3	7.50	Cozinheiro	1
4	5.50	Empregado de mesa	1

*Tabela 26 - Terceira Forma normal 'Recursos Humanos' – Cargo*

#### **EstadoCivil**

<b>fEstadoCivil</b>	<b>fEstadoCivilNome</b>	<b>ativo</b>
1	Solteiro	1
2	Casado	1

*Tabela 27 - Terceira Forma normal 'Recursos Humanos' – EstadoCivil*

## Documento 'Pedido'

<p><b>Pedido nº108</b></p> <p>Número mesa: 5 Estado: Fechado Funcionário : 748274638</p> <p><b>Produtos:</b></p> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 10px;"> <tr><td>Número do Confeccionado: 28 Quantidade: 2</td></tr> <tr><td>Número do Confeccionado: 12 Quantidade: 1</td></tr> <tr><td>Número do Airtigo: 700 Quantidade: 1</td></tr> <tr><td>Número do Airtigo: 600 Quantidade: 1</td></tr> </table> <p><b>Observações</b> ...</p>	Número do Confeccionado: 28 Quantidade: 2	Número do Confeccionado: 12 Quantidade: 1	Número do Airtigo: 700 Quantidade: 1	Número do Airtigo: 600 Quantidade: 1	<p><b>Fatura do Pedido nº108</b></p> <p>Data de criação: 19/05/2022 Método Pagamento: MBWay Número de contribuinte: 521478523</p> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 10px;"> <thead> <tr> <th>Produto</th> <th>Taxa</th> <th>Preço</th> </tr> </thead> <tbody> <tr><td>Moelas</td><td>13%</td><td>3,00 €</td></tr> <tr><td>Airoz de Pato</td><td>13%</td><td>8,20 €</td></tr> <tr><td>Maduro Tinto EA</td><td>23%</td><td>7,50 €</td></tr> <tr><td>Gelado Olá de Morango</td><td>13%</td><td>1,50 €</td></tr> </tbody> </table> <p style="text-align: right;"><b>Total: 20,20 €</b></p> <p style="text-align: right;">miro</p>	Produto	Taxa	Preço	Moelas	13%	3,00 €	Airoz de Pato	13%	8,20 €	Maduro Tinto EA	23%	7,50 €	Gelado Olá de Morango	13%	1,50 €
Número do Confeccionado: 28 Quantidade: 2																				
Número do Confeccionado: 12 Quantidade: 1																				
Número do Airtigo: 700 Quantidade: 1																				
Número do Airtigo: 600 Quantidade: 1																				
Produto	Taxa	Preço																		
Moelas	13%	3,00 €																		
Airoz de Pato	13%	8,20 €																		
Maduro Tinto EA	23%	7,50 €																		
Gelado Olá de Morango	13%	1,50 €																		

*Figura 16 - Documento 'Pedido'*

### Forma não normalizada:

pedidoNo	pedNumeroMesa	pedEstado	pedObs	fNif	pedDataCriacao	pedMetodoPagamento	pedMetodoPagamentoNome
108	5	Fechado	...	748274638	19/05/2022	3	MBWay
pedPreco	pedNumeroContribuinte	prodNo	prodQuantidade	prodPreco	prodTipoTaxa	prodTaxaValor	
20.20	521478523	28	2	3.00	Intermédia	13	
		12	1	8.20	Intermédia	13	
		700	1	7.50	Normal	23	
		600	1	1.50	Intermédia	13	

*Tabela 28 - Forma não normalizada 'Pedido'*

### Primeira Forma normal:

<u>pedidoNo</u>	<u>pedNumeroMesa</u>	<u>pedEstado</u>	<u>pedObs</u>	<u>fNif</u>	<u>pedDataCriacao</u>	<u>pedMetodoPagamento</u>	<u>pedMetodoPagamentoNome</u>
108	5	Fechado	...	748274638	19/05/2022	3	MBWay
108	5	Fechado	...	748274638	19/05/2022	3	MBWay
108	5	Fechado	...	748274638	19/05/2022	3	MBWay
108	5	Fechado	...	748274638	19/05/2022	3	MBWay
<u>pedPreco</u>	<u>pedNumeroContribuinte</u>	<u>prodNo</u>	<u>prodQuantidade</u>	<u>prodPreco</u>	<u>prodTipoTaxa</u>	<u>prodTaxaValor</u>	
20.20	521478523	28	2	3.00	Intermédia	13	
20.20	521478523	12	1	8.20	Intermédia	13	
20.20	521478523	700	1	7.50	Normal	23	
20.20	521478523	600	1	1.50	Intermédia	13	

Tabela 29 - Primeira Forma normal 'Pedido'

### Diagrama de Dependências:

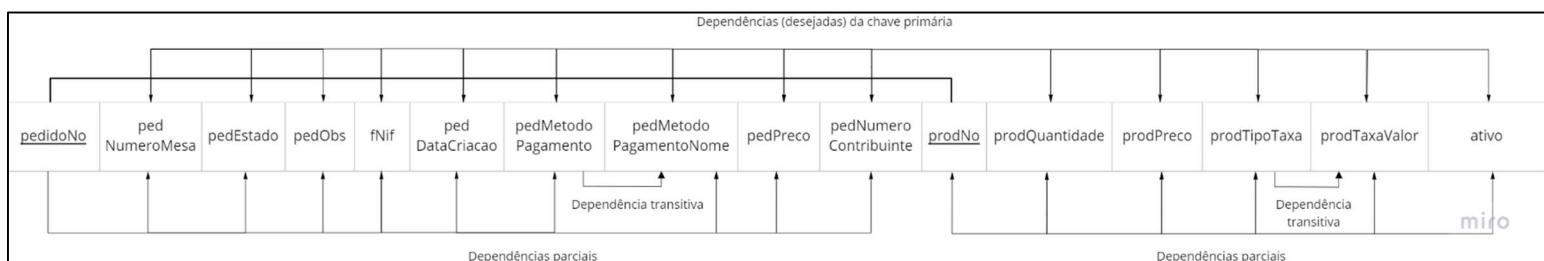


Figura 17 - Diagrama de dependências 'Pedido'

### Segunda Forma normal:

#### Pedido

<u>pedidoNo</u>	<u>pedNumeroMesa</u>	<u>pedEstado</u>	<u>pedObs</u>	<u>pedDataCriacao</u>	<u>pedMetodoPagamento</u>	<u>pedPreco</u>	<u>pedNumeroContribuinte</u>
108	5	Fechado	...	19/05/2022	MBWay	20.20	521478523

Tabela 30 - Segunda Forma normal 'Pedido' - Pedido

## Produto

<u>prodNo</u>	<u>prodPreco</u>	<u>prodTipoTaxa</u>	<u>prodTaxaValor</u>
28	3.00	Intermédia	13
12	8.20	Intermédia	13
700	7.50	Normal	23
600	1.50	Intermédia	13

Tabela 31 - Segunda Forma normal 'Pedido' - Produto

## PedidoProduto

<u>pedidoNo</u>	<u>produtoNo</u>	<u>prodQuantidade</u>
108	28	2
108	12	1
108	700	1
108	600	1

Tabela 32 - Segunda Forma normal 'Pedido' - PedidoProduto

## Terceira Forma normal:

### Pedido

<u>pedidoNo</u>	<u>pedNumeroMesa</u>	<u>pedEstado</u>	<u>pedObs</u>	<u>pedDataCriacao</u>	<u>pedMetodoPagamento</u>	<u>pedPreco</u>	<u>pedNumeroContribuinte</u>	<u>ativo</u>
108	5	Fechado	...	19/05/2022	3	20.20	521478523	1

Tabela 33 - Terceira Forma normal 'Pedido' - Pedido

## Produto

<u>prodNo</u>	<u>prodPreco</u>	<u>prodTipoTaxa</u>
28	3.00	Intermédia
12	8.20	Intermédia
700	7.50	Normal
600	1.50	Intermédia

Tabela 34 - Terceira Forma normal 'Pedido' - Produto

## PedidoProduto

pedidoNo	produtoNo	prodQuantidade
108	28	2
108	12	1
108	700	1
108	600	1

Tabela 35 - Terceira Forma normal 'Pedido' - PedidoProduto

## TipoTaxa

prodTipoTaxa	prodTaxaValor	ativo
Intermédia	13	1
Normal	23	1

Tabela 36 - Terceira Forma normal 'Pedido' – TipoTaxa

## MetodoPagamento

pedMetodoPagamento	pedMetodoPagamentoNome	ativo
3	MBWay	1

Tabela 37 - Terceira Forma normal 'Pedido' – MetodoPagamento

## 2.2.3. Diagrama do Modelo Lógico

É de notar que as relações Cargo, EstadoCivil, Localizacao, MetodoPagamento, TipoTaxa, TipoProduto e GrupoAlimentar surgem do processo de normalização.

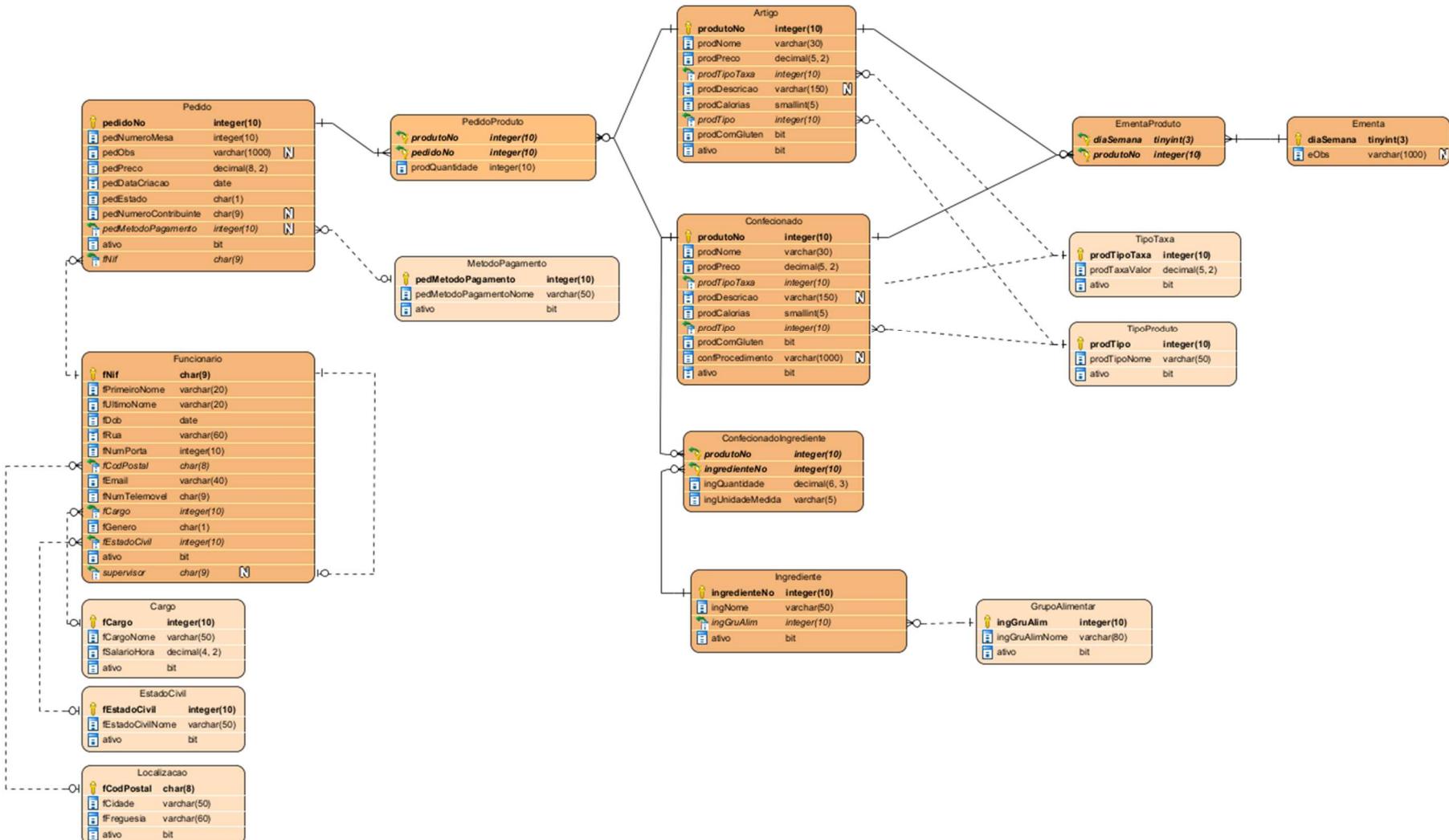


Figura 18 – Resultado das relações para o modelo Lógico

## 2.2.4. Validar relações em relação a transações do utilizador

Nesta etapa, verificamos que as relações criadas na etapa anterior também suportam as transações do utilizador descritas no ponto 2.1.6. Validar modelo conceptual em relação a transações do utilizador, e, assim, garantimos que nenhum erro foi introduzido durante a criação das relações.

## 2.2.5. Verificar restrições de integridade

### Referential integrity Constraints:

EmentaProduto (produtoNo, ementaNo) <b>Chave Primária</b> produtoNo, ementaNo <b>Chave Estrangeira</b> produtoNo <b>referencia</b> Produto (produtoNo) ON UPDATE CASCADE ON DELETE NO ACTION <b>Chave Estrangeira</b> ementaNo <b>referencia</b> Ementa (ementaNo) ON UPDATE CASCADE ON DELETE NO ACTION
Pedido (pedidoNo, pedNumeroMesa, pedObs, pedPreco, pedDataCriacao, pedEstado, pedNumeroContribuinte, pedMetodoPagamento, fNif) <b>Chave Primária</b> pedidoNo <b>Chave Estrangeira</b> fNif <b>referencia</b> Funcionario (fNif) ON UPDATE CASCADE ON DELETE NO ACTION <b>Chave Estrangeira</b> pedMetodoPagamento <b>referencia</b> MetodoPagamento (pedMetodoPagamento) ON UPDATE CASCADE ON DELETE NO ACTION
ConfeccionadoIngrediente (produtoNo, ingredienteNo, quantidade) <b>Chave Primária</b> confeccionadoNo, ingredienteNo <b>Chave Estrangeira</b> produtoNo <b>referencia</b> Confeccionado (produtoNo) ON UPDATE CASCADE ON DELETE NO ACTION <b>Chave Estrangeira</b> ingredienteNo <b>referencia</b> Ingrediente (ingredienteNo) ON UPDATE CASCADE ON DELETE NO ACTION
PedidoProduto (pedidoNo, produtoNo, quantidade) <b>Chave Primária</b> pedidoNo, produtoNo <b>Chave Estrangeira</b> pedidoNo <b>referencia</b> Pedido (pedidoNo) ON UPDATE CASCADE ON DELETE NO ACTION <b>Chave Estrangeira</b> produtoNo <b>referencia</b> Produto (produtoNo) ON UPDATE CASCADE ON DELETE NO ACTION
Funcionario (fNif, fPrimeiroNome, fUltimoNome, fDob, fRua, fNumPorta, fFreguesia, fCidade, fCodPostal, fEmail, fNumTelemovel, fSalarioHora, fCargo, fGenero, fEstadoCivil, supervisor) <b>Chave Primária</b> fNif <b>Chave Alternativa</b> fEmail <b>Chave Estrangeira</b> supervisor <b>referencia</b> Funcionario (fNif) ON UPDATE CASCADE ON DELETE NO ACTION <b>Chave Estrangeira</b> fCargo <b>referencia</b> Cargo (fCargo) ON UPDATE CASCADE ON DELETE NO ACTION <b>Chave Estrangeira</b> fEstadoCivil <b>referencia</b> EstadoCivil (fEstadoCivil) ON UPDATE CASCADE ON DELETE NO ACTION <b>Chave Estrangeira</b> fcodPostal <b>referencia</b> Localizacao (fcodPostal) ON UPDATE CASCADE ON DELETE NO ACTION
Produto (produtoNo, prodNome, prodPreco, prodTipoTaxa, prodDescricao, prodCalorias, prodTipo, prodComGluten, ativo) <b>Chave Primária</b> produtoNo <b>Chave Estrangeira</b> prodTipoTaxa <b>referencia</b> TipoTaxa (prodTipoTaxa) ON UPDTE CASCADE ON DELETE NO ACTION <b>Chave Estrangeira</b> prodTipo <b>referencia</b> TipoProduto (prodTipo) ON UPDATE CASCADE ON DELETE NO ACTION
Ingrediente (ingredienteNo, ingNome, ingGruAlim, ativo) <b>Chave Primária</b> ingredienteNo <b>Chave Estrangeira</b> ingGruAlim <b>referencia</b> GrupoAlimentar (ingGruAlim) ON UPDATE CASCADE ON DELETE NO ACTION

Figura 19 - Restrições de Integridade Referencial

miro

**General Constraints:**

Pedido:

- Produtos não podem ser adicionados a pedidos eliminados.
- Produtos não podem ser adicionados a pedidos fechados.
- Produtos “eliminados” (ativo = 0) não podem ser adicionados a pedidos.
- Confeccionados com 0 ingredientes não podem ser adicionados a pedidos.
- Pedidos previamente “eliminados” (ativo = 0) não podem ser fechados.
- Pedidos com 0 produtos associados não podem ser fechados.
- Métodos de Pagamento “eliminados” (ativo = 0) não podem ser associados a pedidos.
- Pedidos fechados não podem ser eliminados/atualizados.
- Funcionários “eliminados” (ativo = 0) não podem ser associados a pedidos.
- Não deve ser possível remover/atualizar produtos de pedidos fechados.
- Não podem existir 2 pedidos abertos ao mesmo tempo com a mesma mesa atribuída.

Funcionario:

- Não deve ser possível eliminar um funcionário responsável por um pedido aberto no momento.
- Um Funcionario não pode ser supervisor dele próprio;
- Um Funcionario pode supervisionar no máximo 10 funcionários;
- Códigos Postais “eliminados” (ativo = 0) não podem ser associados a Funcionarios.
- Estados Civis “eliminados” (ativo = 0) não podem ser associados a Funcionarios.
- Cargos “eliminados” (ativo = 0) não podem ser associados a Funcionarios.

Artigo:

- Tipos de taxa “eliminados” (ativo = 0) não podem ser associados a Artigos.
- Tipos de produto “eliminados” (ativo = 0) não podem ser associados a Artigos.
- Artigos associados a pedidos abertos não podem ser eliminados.

## 2.3. Modelo Físico

### 2.3.1. Diagrama do Modelo Físico

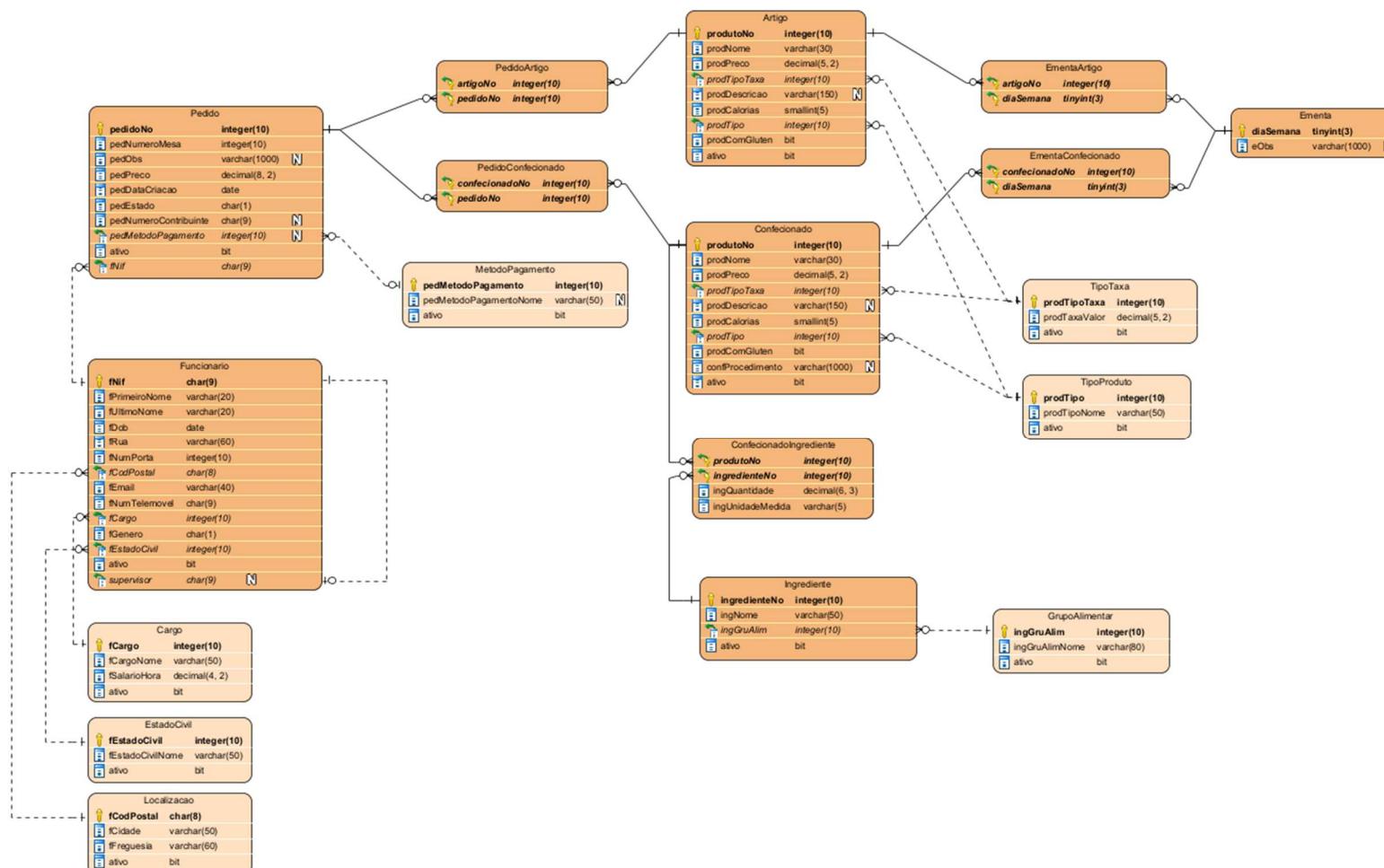


Figura 20 - Diagrama do Modelo Físico

No [diagrama do Modelo Lógico](#), na relação EmentaProduto surge um pequeno problema, uma vez que temos duas entidades distintas que são consideradas Produtos. Ou seja, queremos que a chave estrangeira (produtoNo) da entidade EmentaProduto faça referência a duas relações (Confeccionado e Artigo).

Com isto, ao adicionarmos um Artigo a uma Ementa poderia existir uma violação de chave estrangeira dado que o produtoNo do Artigo teria de estar nas relações Artigo e Confeccionado.

Para solucionar este problema decidiu-se decompor a relação EmentaProduto em EmentaArtigo e EmentaConfeccionado.

Este problema também se encontrava do lado do Pedido e usou-se o mesmo raciocínio decompondo a relação PedidoProduto em PedidoArtigo e PedidoConfeccionado.

O resultado do assunto descrito previamente é visível no [diagrama do Modelo Físico](#).

## 2.3.1. Traduzir o modelo Lógico de dados para o DBMS destino

### 2.3.1.1. Desenho de Relações Base e *General Constraints*

#### Artigo:

```
USE [MichelinStar]
GO

/******** Object: Table [dbo].[Artigo]    Script Date: 10/06/2022 01:14:51 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Artigo](
    [produtoNo] [int] IDENTITY(1,1) NOT NULL,
    [prodNome] [varchar](30) NOT NULL,
    [prodPreco] [decimal](5, 2) NOT NULL,
    [prodTipoTaxa] [int] NOT NULL,
    [prodDescricao] [varchar](150) NULL,
    [prodCalorias] [smallint] NOT NULL,
    [prodTipo] [int] NOT NULL,
    [prodComGluten] [bit] NOT NULL,
    [ativo] [bit] NOT NULL,
    CONSTRAINT [PK_Artigo] PRIMARY KEY CLUSTERED
(
    [produtoNo] 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].[Artigo]
ADD CONSTRAINT [DF_Artigo_ativo] DEFAULT ((1)) FOR [ativo]
GO

ALTER TABLE [dbo].[Artigo] WITH CHECK
ADD CONSTRAINT [FK_Artigo_TipoProduto]
FOREIGN KEY([prodTipo])
REFERENCES [dbo].[TipoProduto] ([prodTipo])
ON UPDATE CASCADE
GO

ALTER TABLE [dbo].[Artigo] CHECK CONSTRAINT [FK_Artigo_TipoProduto]
GO
```

Figura 21 – DBDL para Artigo pt. 1

```

ALTER TABLE [dbo].[Artigo] WITH CHECK
ADD CONSTRAINT [FK_Artigo_TipoTaxa] FOREIGN KEY([prodTipoTaxa])
REFERENCES [dbo].[TipoTaxa] ([prodTipoTaxa])
ON UPDATE CASCADE
GO

ALTER TABLE [dbo].[Artigo] CHECK CONSTRAINT [FK_Artigo_TipoTaxa]
GO

ALTER TABLE [dbo].[Artigo] WITH CHECK
ADD CONSTRAINT [CK_prodCaloriasArtigo]
CHECK (([prodCalorias]>=(0) AND [prodCalorias]<=(9999)))
GO

ALTER TABLE [dbo].[Artigo] CHECK CONSTRAINT [CK_prodCaloriasArtigo]
GO

ALTER TABLE [dbo].[Artigo] WITH CHECK
ADD CONSTRAINT [CK_prodPrecoArtigo] CHECK (([prodPreco]>=(0)))
GO

ALTER TABLE [dbo].[Artigo] CHECK CONSTRAINT [CK_prodPrecoArtigo]
GO

ALTER TABLE [dbo].[Artigo] WITH CHECK
ADD CONSTRAINT [CK_produtoNo] CHECK (([produtoNo]<(5000)))
GO

ALTER TABLE [dbo].[Artigo] CHECK CONSTRAINT [CK_produtoNo]
GO

EXEC sys.sp_addextendedproperty @name=N'MS_Description',
@value=N'Constraint para restringir as prodCalorias entre 0 e 9999 inclusive' ,
@level0type=N'SCHEMA',@level0name=N'dbo', @level1type=N'TABLE',
@level1name=N'Artigo', @level2type=N'CONSTRAINT',
@level2name=N'CK_prodCaloriasArtigo'
GO

EXEC sys.sp_addextendedproperty @name=N'MS_Description',
@value=N'Constraint para restringir o prodPreco a ser maior ou igual a 0' ,
@level0type=N'SCHEMA',@level0name=N'dbo', @level1type=N'TABLE',
@level1name=N'Artigo', @level2type=N'CONSTRAINT',@level2name=N'CK_prodPrecoArtigo'
GO

```

Figura 22 – DBDL para Artigo pt. 2

```

EXEC sys.sp_addextendedproperty @name=N'MS_Description',
@value=N'Constraint que restringe o produtoNo a ser menor que 5000' ,
@level0type=N'SCHEMA',@level0name=N'dbo', @level1type=N'TABLE',
@level1name=N'Artigo', @level2type=N'CONSTRAINT',@level2name=N'CK_produtoNo'
GO

```

Figura 23 - DBDL para Artigo pt. 3

**Cargo:**

```
USE [MichelinStar]
GO

/******** Object: Table [dbo].[Cargo]      Script Date: 10/06/2022 01:28:10 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Cargo](
    [fCargo] [int] IDENTITY(1,1) NOT NULL,
    [fCargoNome] [varchar](50) NOT NULL,
    [fSalarioHora] [decimal](4, 2) NOT NULL,
    [ativo] [bit] NOT NULL,
    CONSTRAINT [PK_Cargo] PRIMARY KEY CLUSTERED
(
    [fCargo] 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].[Cargo]
ADD CONSTRAINT [DF_Cargo_ativo] DEFAULT ((1)) FOR [ativo]
GO

ALTER TABLE [dbo].[Cargo] WITH CHECK
ADD CONSTRAINT [CK_fSalarioHora] CHECK ((([fSalarioHora]>=(0)))
GO

ALTER TABLE [dbo].[Cargo] CHECK CONSTRAINT [CK_fSalarioHora]
GO

EXEC sys.sp_addextendedproperty @name=N'MS_Description',
@value=N'Constraint que restringe o fSalarioHora a ser maior ou igual a 0' ,
@level0type=N'SCHEMA', @level0name=N'dbo', @level1type=N'TABLE',
@level1name=N'Cargo', @level2type=N'CONSTRAINT',
@level2name=N'CK_fSalarioHora'
GO
```

Figura 24 - DBDL para Cargo

### Confeccionado:

```
USE [MichelinStar]
GO

/******** Object: Table [dbo].[Confeccionado]    Script Date: 10/06/2022 01:29:33 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Confeccionado](
    [produtoNo] [int] IDENTITY(5000,1) NOT NULL,
    [prodNome] [varchar](30) NOT NULL,
    [prodPreco] [decimal](5, 2) NOT NULL,
    [prodTipoTaxa] [int] NOT NULL,
    [prodDescricao] [varchar](150) NULL,
    [prodCalorias] [smallint] NOT NULL,
    [prodTipo] [int] NOT NULL,
    [prodComGluten] [bit] NOT NULL,
    [confProcedimento] [varchar](1000) NULL,
    [ativo] [bit] NOT NULL,
    CONSTRAINT [PK_Confeccionado] PRIMARY KEY CLUSTERED
(
    [produtoNo] 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].[Confeccionado]
ADD CONSTRAINT [DF_Confeccionado_ativo] DEFAULT ((1)) FOR [ativo]
GO

ALTER TABLE [dbo].[Confeccionado] WITH CHECK
ADD CONSTRAINT [FK_Confeccionado_TipoProduto] FOREIGN KEY([prodTipo])
REFERENCES [dbo].[TipoProduto] ([prodTipo])
ON UPDATE CASCADE
GO

ALTER TABLE [dbo].[Confeccionado] CHECK CONSTRAINT [FK_Confeccionado_TipoProduto]
GO
```

Figura 25 - DBDL para Confeccionado pt.1

```

ALTER TABLE [dbo].[Confucionado] WITH CHECK
ADD CONSTRAINT [FK_Confucionado_TipoTaxa] FOREIGN KEY([prodTipoTaxa])
REFERENCES [dbo].[TipoTaxa] ([prodTipoTaxa])
ON UPDATE CASCADE
GO

ALTER TABLE [dbo].[Confucionado] CHECK CONSTRAINT [FK_Confucionado_TipoTaxa]
GO

ALTER TABLE [dbo].[Confucionado] WITH CHECK
ADD CONSTRAINT [CK_prodCaloriasConfucionado]
CHECK (([prodCalorias]>(0) AND [prodCalorias]<=(9999)))
GO

ALTER TABLE [dbo].[Confucionado] CHECK CONSTRAINT [CK_prodCaloriasConfucionado]
GO

ALTER TABLE [dbo].[Confucionado] WITH CHECK
ADD CONSTRAINT [CK_prodPrecoConfucionado] CHECK (([prodPreco]>=(0)))
GO

ALTER TABLE [dbo].[Confucionado] CHECK CONSTRAINT [CK_prodPrecoConfucionado]
GO

EXEC sys.sp_addextendedproperty @name=N'MS_Description',
@value=N'Constraint para restringir as prodCalorias entre 0 e 9999 inclusive' ,
@level0type=N'SCHEMA',@level0name=N'dbo', @level1type=N'TABLE',
@level1name=N'Confucionado', @level2type=N'CONSTRAINT',
@level2name=N'CK_prodCaloriasConfucionado'
GO

EXEC sys.sp_addextendedproperty @name=N'MS_Description',
@value=N'Constraint para restringir o prodPreco a ser maior ou igual a 0' ,
@level0type=N'SCHEMA',@level0name=N'dbo', @level1type=N'TABLE',
@level1name=N'Confucionado', @level2type=N'CONSTRAINT',
@level2name=N'CK_prodPrecoConfucionado'
GO

```

Figura 26 - DBDL para Confucionado pt.2

### **ConfeccionadoIngrediente:**

```
USE [MichelinStar]
GO

/******** Object: Table [dbo].[ConfeccionadoIngrediente]      Script Date: 10/06/2022 01:32:00 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[ConfeccionadoIngrediente](
    [produtoNo] [int] NOT NULL,
    [ingredienteNo] [int] NOT NULL,
    [ingQuantidade] [decimal](6, 3) NOT NULL,
    [ingUnidadeMedida] [varchar](5) NOT NULL,
    CONSTRAINT [PK_ConfeccionadoIngrediente] PRIMARY KEY CLUSTERED
(
    [produtoNo] ASC,
    [ingredienteNo] 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].[ConfeccionadoIngrediente] WITH CHECK
ADD CONSTRAINT [FK_ConfeccionadoIngrediente_Confeccionado] FOREIGN KEY([produtoNo])
REFERENCES [dbo].[Confeccionado] ([produtoNo])
ON UPDATE CASCADE
GO

ALTER TABLE [dbo].[ConfeccionadoIngrediente]
CHECK CONSTRAINT [FK_ConfeccionadoIngrediente_Confeccionado]
GO

ALTER TABLE [dbo].[ConfeccionadoIngrediente] WITH CHECK
ADD CONSTRAINT [FK_ConfeccionadoIngrediente_Ingrediente] FOREIGN KEY([ingredienteNo])
REFERENCES [dbo].[Ingrediente] ([ingredienteNo])
ON UPDATE CASCADE
GO

ALTER TABLE [dbo].[ConfeccionadoIngrediente]
CHECK CONSTRAINT [FK_ConfeccionadoIngrediente_Ingrediente]
GO
```

Figura 27 - DBDL para ConfeccionadoIngrediente pt.1

```

ALTER TABLE [dbo].[ConfeccionadoIngrediente] WITH CHECK
ADD CONSTRAINT [CK_ingQuantidade] CHECK (([ingQuantidade]>(0)))
GO

ALTER TABLE [dbo].[ConfeccionadoIngrediente] CHECK CONSTRAINT [CK_ingQuantidade]
GO

ALTER TABLE [dbo].[ConfeccionadoIngrediente] WITH CHECK
ADD CONSTRAINT [CK_ingUnidadeMedida]
CHECK (([ingUnidadeMedida]='Kg' OR [ingUnidadeMedida]='L' OR [ingUnidadeMedida]='Un'))
GO

ALTER TABLE [dbo].[ConfeccionadoIngrediente] CHECK CONSTRAINT [CK_ingUnidadeMedida]
GO

EXEC sys.sp_addextendedproperty @name=N'MS_Description',
@value=N'Constraint que restringe a ingQuantidade a ser maior que 0' ,
@level0type=N'SCHEMA',@level0name=N'dbo', @level1type=N'TABLE',
@level1name=N'ConfeccionadoIngrediente', @level2type=N'CONSTRAINT',
@level2name=N'CK_ingQuantidade'
GO

EXEC sys.sp_addextendedproperty @name=N'MS_Description',
@value=N'Constraint que restringe o ingUnidadeMedida a ser igual a ''Kg'', ''L'' ou ''Un''' ,
@level0type=N'SCHEMA',@level0name=N'dbo', @level1type=N'TABLE',
@level1name=N'ConfeccionadoIngrediente', @level2type=N'CONSTRAINT',
@level2name=N'CK_ingUnidadeMedida'
GO

```

Figura 28 - DBDL para ConfeccionadoIngrediente pt.2

### Ementa:

```
USE [MichelinStar]
GO

/****** Object: Table [dbo].[Ementa]      Script Date: 10/06/2022 01:34:22 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Ementa](
    [diaSemana] [tinyint] NOT NULL,
    [eObs] [varchar](1000) NULL,
    CONSTRAINT [PK_Ementa] PRIMARY KEY CLUSTERED
(
    [diaSemana] 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].[Ementa] WITH CHECK
ADD CONSTRAINT [CK_diaSemana] CHECK (([diaSemana]>=(1) AND [diaSemana]<=(7)))
GO

ALTER TABLE [dbo].[Ementa] CHECK CONSTRAINT [CK_diaSemana]
GO

EXEC sys.sp_addextendedproperty @name=N'MS_Description',
@value=N'Constraint que restringe o diaSemana a estar compreendido entre 1 e 7 inclusive' ,
@level0type=N'SCHEMA',@level0name=N'dbo', @level1type=N'TABLE',
@level1name=N'Ementa', @level2type=N'CONSTRAINT',@level2name=N'CK_diaSemana'
GO
```

Figura 29 - DBDL para Ementa

### EmentaArtigo:

```
USE [MichelinStar]
GO

/******** Object: Table [dbo].[EmentaArtigo]    Script Date: 10/06/2022 01:35:39 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[EmentaArtigo](
    [diaSemana] [tinyint] NOT NULL,
    [artigoNo] [int] NOT NULL,
    CONSTRAINT [PK_EmentaArtigo] PRIMARY KEY CLUSTERED
(
    [artigoNo] ASC,
    [diaSemana] 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].[EmentaArtigo] WITH CHECK
ADD CONSTRAINT [FK_EmentaArtigo_Artigo] FOREIGN KEY([artigoNo])
REFERENCES [dbo].[Artigo] ([produtoNo])
ON UPDATE CASCADE
GO

ALTER TABLE [dbo].[EmentaArtigo]
CHECK CONSTRAINT [FK_EmentaArtigo_Artigo]
GO

ALTER TABLE [dbo].[EmentaArtigo] WITH CHECK
ADD CONSTRAINT [FK_EmentaArtigo_Ementa] FOREIGN KEY([diaSemana])
REFERENCES [dbo].[Ementa] ([diaSemana])
ON UPDATE CASCADE
GO

ALTER TABLE [dbo].[EmentaArtigo]
CHECK CONSTRAINT [FK_EmentaArtigo_Ementa]
GO
```

Figura 30 - DBDL para EmentaArtigo

### EmentaConfeccionado:

```
USE [MichelinStar]
GO

/******** Object: Table [dbo].[EmentaConfeccionado]      Script Date: 10/06/2022 01:36:42 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[EmentaConfeccionado](
    [diaSemana] [tinyint] NOT NULL,
    [confeccionadoNo] [int] NOT NULL,
    CONSTRAINT [PK_EmentaConfeccionado] PRIMARY KEY CLUSTERED
(
    [confeccionadoNo] ASC,
    [diaSemana] 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].[EmentaConfeccionado] WITH CHECK
ADD CONSTRAINT [FK_EmentaConfeccionado_Confeccionado]
FOREIGN KEY([confeccionadoNo])
REFERENCES [dbo].[Confeccionado] ([produtoNo])
ON UPDATE CASCADE
GO

ALTER TABLE [dbo].[EmentaConfeccionado]
CHECK CONSTRAINT [FK_EmentaConfeccionado_Confeccionado]
GO

ALTER TABLE [dbo].[EmentaConfeccionado] WITH CHECK
ADD CONSTRAINT [FK_EmentaConfeccionado_Ementa] FOREIGN KEY([diaSemana])
REFERENCES [dbo].[Ementa] ([diaSemana])
ON UPDATE CASCADE
GO

ALTER TABLE [dbo].[EmentaConfeccionado]
CHECK CONSTRAINT [FK_EmentaConfeccionado_Ementa]
GO
```

Figura 31 - DBDL para EmentaConfeccionado

### EstadoCivil:

```
USE [MichelinStar]
GO

/******** Object: Table [dbo].[EstadoCivil]    Script Date: 10/06/2022 01:37:56 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[EstadoCivil](
    [fEstadoCivil] [int] IDENTITY(1,1) NOT NULL,
    [fEstadoCivilNome] [varchar](50) NOT NULL,
    [ativo] [bit] NOT NULL,
    CONSTRAINT [PK_EstadoCivil] PRIMARY KEY CLUSTERED
    (
        [fEstadoCivil] 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].[EstadoCivil]
ADD CONSTRAINT [DF_EstadoCivil_ativo] DEFAULT ((1)) FOR [ativo]
GO
```

Figura 32 - DBDL para EstadoCivil

### Funcionario:

```
USE [MichelinStar]
GO

/******** Object: Table [dbo].[Funcionario]      Script Date: 10/06/2022 01:38:47 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Funcionario](
    [fNif] [char](9) NOT NULL,
    [fPrimeiroNome] [varchar](20) NOT NULL,
    [fUltimoNome] [varchar](20) NOT NULL,
    [fDob] [date] NOT NULL,
    [fRua] [varchar](60) NOT NULL,
    [fNumPorta] [int] NOT NULL,
    [fCodPostal] [char](8) NOT NULL,
    [fEmail] [varchar](40) NOT NULL,
    [fNumTelemovel] [char](9) NOT NULL,
    [fCargo] [int] NOT NULL,
    [fGenero] [char](1) NOT NULL,
    [fEstadoCivil] [int] NOT NULL,
    [ativo] [bit] NOT NULL,
    [supervisor] [char](9) NULL,
    CONSTRAINT [PK_Funcionario] PRIMARY KEY CLUSTERED
    (
        [fNif] 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].[Funcionario]
ADD CONSTRAINT [DF_Funcionario_ativo] DEFAULT ((1)) FOR [ativo]
GO

ALTER TABLE [dbo].[Funcionario] WITH CHECK
ADD CONSTRAINT [FK_Funcionario_Cargo] FOREIGN KEY([fCargo])
REFERENCES [dbo].[Cargo] ([fCargo])
ON UPDATE CASCADE
GO
```

Figura 33 - DBDL para Funcionario pt.1

```

ALTER TABLE [dbo].[Funcionario]
CHECK CONSTRAINT [FK_Funcionario_Cargo]
GO

ALTER TABLE [dbo].[Funcionario] WITH CHECK
ADD CONSTRAINT [FK_Funcionario_EstadoCivil] FOREIGN KEY([fEstadoCivil])
REFERENCES [dbo].[EstadoCivil] ([fEstadoCivil])
ON UPDATE CASCADE
GO

ALTER TABLE [dbo].[Funcionario] CHECK CONSTRAINT [FK_Funcionario_EstadoCivil]
GO

ALTER TABLE [dbo].[Funcionario] WITH CHECK
ADD CONSTRAINT [FK_Funcionario_Localizacao] FOREIGN KEY([fCodPostal])
REFERENCES [dbo].[Localizacao] ([fCodPostal])
ON UPDATE CASCADE
GO

ALTER TABLE [dbo].[Funcionario] CHECK CONSTRAINT [FK_Funcionario_Localizacao]
GO

ALTER TABLE [dbo].[Funcionario] WITH CHECK
ADD CONSTRAINT [CK_fEmail]
CHECK (([fEmail] like '%@___.__%') AND patindex('%[^a-z,0-9,@,.,_,\-%]',[fEmail])=(0))
GO

ALTER TABLE [dbo].[Funcionario] CHECK CONSTRAINT [CK_fEmail]
GO

ALTER TABLE [dbo].[Funcionario] WITH CHECK
ADD CONSTRAINT [CK_fGenero] CHECK (([fGenero]='M' OR [fGenero]='F' OR [fGenero]='O'))
GO

ALTER TABLE [dbo].[Funcionario] CHECK CONSTRAINT [CK_fGenero]
GO

ALTER TABLE [dbo].[Funcionario] WITH CHECK
ADD CONSTRAINT [CK_fNif]
CHECK (([fNif] like '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'))
GO

ALTER TABLE [dbo].[Funcionario] CHECK CONSTRAINT [CK_fNif]
GO

ALTER TABLE [dbo].[Funcionario] WITH CHECK
ADD CONSTRAINT [CK_fNumPorta] CHECK ((([fNumPorta]>(0) AND [fNumPorta]<=(9999)))
GO

```

Figura 34 - DBDL para Funcionario pt.2

```

ALTER TABLE [dbo].[Funcionario] CHECK CONSTRAINT [CK_fNumPorta]
GO

ALTER TABLE [dbo].[Funcionario] WITH CHECK
ADD CONSTRAINT [CK_Funcionario]
CHECK (([fNumTeleovel] like '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'))
GO

ALTER TABLE [dbo].[Funcionario]
CHECK CONSTRAINT [CK_Funcionario]
GO

EXEC sys.sp_addextendedproperty @name=N'MS_Description',
@value=N'Constraint que valida o e-mail',
@level0type=N'SCHEMA',@level0name=N'dbo',
@level1type=N'TABLE',@level1name=N'Funcionario',
@level2type=N'CONSTRAINT',@level2name=N'CK_fEmail'
GO

EXEC sys.sp_addextendedproperty @name=N'MS_Description',
@value=N'Constraint que restringe o fGenero a ser igual a ''M'' (Masculino), ''F'' (Feminino) ou ''O'' (Outro)',
@level0type=N'SCHEMA',@level0name=N'dbo', @level1type=N'TABLE',
@level1name=N'Funcionario', @level2type=N'CONSTRAINT',@level2name=N'CK_fGenero'
GO

EXEC sys.sp_addextendedproperty @name=N'MS_Description',
@value=N'Constraint que restringe o fnif a ter exatamente 9 dígitos',
@level0type=N'SCHEMA',@level0name=N'dbo', @level1type=N'TABLE',
@level1name=N'Funcionario', @level2type=N'CONSTRAINT',@level2name=N'CK_fnif'
GO

EXEC sys.sp_addextendedproperty @name=N'MS_Description',
@value=N'Constraint que restringe o fNumPorta a ser um valor inteiro compreendido entre 0 e 10000',
@level0type=N'SCHEMA',@level0name=N'dbo', @level1type=N'TABLE',@level1name=N'Funcionario',
@level2type=N'CONSTRAINT',@level2name=N'CK_fNumPorta'
GO

EXEC sys.sp_addextendedproperty @name=N'MS_Description',
@value=N'Constraint que restringe o fNumTeleovel a ter exatamente 9 dígitos',
@level0type=N'SCHEMA',@level0name=N'dbo', @level1type=N'TABLE',@level1name=N'Funcionario',
@level2type=N'CONSTRAINT',@level2name=N'CK_Funcionario'
GO

```

Figura 35 - DBDL para Funcionario pt.3

### GrupoAlimentar:

```
USE [MichelinStar]
GO

/******** Object: Table [dbo].[GrupoAlimentar]      Script Date: 10/06/2022 01:47:32 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[GrupoAlimentar](
    [ingGruAlim] [int] IDENTITY(1,1) NOT NULL,
    [ingGruAlimNome] [varchar](80) NOT NULL,
    [ativo] [bit] NOT NULL,
    CONSTRAINT [PK_GrupoAlimentar] PRIMARY KEY CLUSTERED
    (
        [ingGruAlim] 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].[GrupoAlimentar]
ADD CONSTRAINT [DF_GrupoAlimentar_ativo] DEFAULT ((1)) FOR [ativo]
GO
```

Figura 36 - DBDL para GrupoAlimentar

**Ingrediente:**

```
USE [MichelinStar]
GO

/******** Object: Table [dbo].[Ingrediente]    Script Date: 10/06/2022 01:48:22 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Ingrediente](
    [ingredienteNo] [int] IDENTITY(1,1) NOT NULL,
    [ingNome] [varchar](50) NOT NULL,
    [ingGruAlim] [int] NOT NULL,
    [ativo] [bit] NOT NULL,
    CONSTRAINT [PK_Ingrediente] PRIMARY KEY CLUSTERED
    (
        [ingredienteNo] 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].[Ingrediente]
ADD CONSTRAINT [DF_Ingrediente_ativo] DEFAULT ((1)) FOR [ativo]
GO

ALTER TABLE [dbo].[Ingrediente] WITH CHECK
ADD CONSTRAINT [FK_Ingrediente_GrupoAlimentar] FOREIGN KEY([ingGruAlim])
REFERENCES [dbo].[GrupoAlimentar] ([ingGruAlim])
ON UPDATE CASCADE
GO

ALTER TABLE [dbo].[Ingrediente] CHECK CONSTRAINT [FK_Ingrediente_GrupoAlimentar]
GO
```

Figura 37 - DBDL para Ingrediente

## Localizacao:

```
USE [MichelinStar]
GO

/******** Object: Table [dbo].[Localizacao]    Script Date: 10/06/2022 01:49:25 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Localizacao](
    [fCodPostal] [char](8) NOT NULL,
    [fCidade] [varchar](50) NOT NULL,
    [fFreguesia] [varchar](60) NOT NULL,
    [ativo] [bit] NOT NULL,
    CONSTRAINT [PK_Localizacao] PRIMARY KEY CLUSTERED
    (
        [fCodPostal] 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].[Localizacao]
ADD CONSTRAINT [DF_Localizacao_ativo] DEFAULT ((1)) FOR [ativo]
GO

ALTER TABLE [dbo].[Localizacao] WITH CHECK
ADD CONSTRAINT [CK_fCodPostal]
CHECK (([fCodPostal] like '[0-9][0-9][0-9][0-9][-][0-9][0-9][0-9]'))
GO

ALTER TABLE [dbo].[Localizacao] CHECK CONSTRAINT [CK_fCodPostal]
GO

EXEC sys.sp_addextendedproperty @name=N'MS_Description',
@value=N'Constraint que restringe o fCodPostal a ter 4 dígitos e 3 dígitos respetivamente separados por 1 hifen' ,
@level0type=N'SCHEMA',@level0name=N'dbo', @level1type=N'TABLE',
@level1name=N'Localizacao', @level2type=N'CONSTRAINT',@level2name=N'CK_fCodPostal'
GO
```

Figura 38 - DBDL para Localizacao

### **MetodoPagamento:**

```
USE [MichelinStar]
GO

/******** Object: Table [dbo].[MetodoPagamento]     Script Date: 10/06/2022 01:50:31 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[MetodoPagamento](
    [ped MetodoPagamento] [int] IDENTITY(1,1) NOT NULL,
    [ped MetodoPagamentoNome] [varchar](50) NOT NULL,
    [ativo] [bit] NOT NULL,
    CONSTRAINT [PK_MetodoPagamento] PRIMARY KEY CLUSTERED
(
    [ped MetodoPagamento] 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].[MetodoPagamento]
ADD CONSTRAINT [DF_MetodoPagamento_ativo] DEFAULT ((1)) FOR [ativo]
GO
```

*Figura 39 - DBDL para MetodoPagamento*

**Pedido:**

```
USE [MichelinStar]
GO

/******** Object: Table [dbo].[Pedido]      Script Date: 10/06/2022 01:51:58 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Pedido](
    [pedidoNo] [int] IDENTITY(1,1) NOT NULL,
    [pedNumeroMesa] [int] NOT NULL,
    [pedObs] [varchar](1000) NULL,
    [pedPreco] [dbo].[preco] NOT NULL,
    [pedDataCriacao] [date] NOT NULL,
    [pedEstado] [char](1) NOT NULL,
    [pedNumeroContribuinte] [char](9) NULL,
    [pedMetodoPagamento] [int] NULL,
    [ativo] [bit] NOT NULL,
    [fnif] [char](9) NOT NULL,
    CONSTRAINT [PK_Pedido] PRIMARY KEY CLUSTERED
(
    [pedidoNo] 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_pedPreco] DEFAULT ((0)) FOR [pedPreco]
GO

ALTER TABLE [dbo].[Pedido]
ADD CONSTRAINT [DF_Pedido_pedEstado] DEFAULT ('A') FOR [pedEstado]
GO

ALTER TABLE [dbo].[Pedido]
ADD CONSTRAINT [DF_Pedido_ativo] DEFAULT ((1)) FOR [ativo]
GO
```

Figura 40 - DBDL para Pedido pt.1

```

ALTER TABLE [dbo].[Pedido] WITH CHECK
ADD CONSTRAINT [FK_Pedido_Funcionario] FOREIGN KEY([fNif])
REFERENCES [dbo].[Funcionario] ([fNif])
ON UPDATE CASCADE
GO

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

ALTER TABLE [dbo].[Pedido] WITH CHECK
ADD CONSTRAINT [FK_Pedido_MetodoPagamento] FOREIGN KEY([pedMetodoPagamento])
REFERENCES [dbo].[MetodoPagamento] ([pedMetodoPagamento])
ON UPDATE CASCADE
GO

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

ALTER TABLE [dbo].[Pedido] WITH CHECK
ADD CONSTRAINT [CK_pedEstado] CHECK (((pedEstado)='A' OR [pedEstado]='F'))
GO

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

ALTER TABLE [dbo].[Pedido] WITH CHECK
ADD CONSTRAINT [CK_pedNumeroContribuinte]
CHECK (((pedNumeroContribuinte) like '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'))
GO

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

ALTER TABLE [dbo].[Pedido] WITH CHECK
ADD CONSTRAINT [CK_pedNumeroMesa] CHECK (((pedNumeroMesa)>=(0)))
GO

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

ALTER TABLE [dbo].[Pedido] WITH CHECK
ADD CONSTRAINT [CK_pedPreco] CHECK (((pedPreco)>=(0.00) AND [pedPreco]<=(100000.00)))
GO

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

```

Figura 41 - DBDL para Pedido pt.2

### **PedidoArtigo:**

```
USE [MichelinStar]
GO

/******** Object: Table [dbo].[PedidoArtigo]    Script Date: 10/06/2022 01:54:17 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[PedidoArtigo](
    [pedidoNo] [int] NOT NULL,
    [artigoNo] [int] NOT NULL,
    [quantidade] [smallint] NOT NULL,
    CONSTRAINT [PK_PedidoArtigo] PRIMARY KEY CLUSTERED
(
    [artigoNo] ASC,
    [pedidoNo] 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].[PedidoArtigo] WITH CHECK
ADD CONSTRAINT [FK_PedidoArtigo_Artigo] FOREIGN KEY([artigoNo])
REFERENCES [dbo].[Artigo] ([produtoNo])
ON UPDATE CASCADE
GO

ALTER TABLE [dbo].[PedidoArtigo]
CHECK CONSTRAINT [FK_PedidoArtigo_Artigo]
GO

ALTER TABLE [dbo].[PedidoArtigo] WITH CHECK
ADD CONSTRAINT [FK_PedidoArtigo_Pedido] FOREIGN KEY([pedidoNo])
REFERENCES [dbo].[Pedido] ([pedidoNo])
ON UPDATE CASCADE
GO

ALTER TABLE [dbo].[PedidoArtigo] CHECK CONSTRAINT [FK_PedidoArtigo_Pedido]
GO
```

Figura 42 - DBDL para PedidoArtigo pt.1

```
ALTER TABLE [dbo].[PedidoArtigo] WITH CHECK  
ADD CONSTRAINT [CK_artigoQuantidade] CHECK (([quantidade]>(0)))  
GO  
  
ALTER TABLE [dbo].[PedidoArtigo] CHECK CONSTRAINT [CK_artigoQuantidade]  
GO  
  
EXEC sys.sp_addextendedproperty @name=N'MS_Description',  
@value=N'Constraint que restringe a quantidade a ser superior a 0' ,  
@level0type=N'SCHEMA',@level0name=N'dbo', @level1type=N'TABLE',  
@level1name=N'PedidoArtigo', @level2type=N'CONSTRAINT',  
@level2name=N'CK_artigoQuantidade'  
GO
```

Figura 43 - DBDL para PedidoArtigo pt.2

### PedidoConfeccionado:

```
USE [MichelinStar]
GO

/******** Object: Table [dbo].[PedidoConfeccionado]    Script Date: 10/06/2022 01:55:46 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[PedidoConfeccionado](
    [pedidoNo] [int] NOT NULL,
    [confeccionadoNo] [int] NOT NULL,
    [quantidade] [smallint] NOT NULL,
    CONSTRAINT [PK_PedidoConfeccionado] PRIMARY KEY CLUSTERED
    (
        [confeccionadoNo] ASC,
        [pedidoNo] 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].[PedidoConfeccionado] WITH CHECK
ADD CONSTRAINT [FK_PedidoConfeccionado_Confeccionado] FOREIGN KEY([confeccionadoNo])
REFERENCES [dbo].[Confeccionado] ([produtoNo])
ON UPDATE CASCADE
GO

ALTER TABLE [dbo].[PedidoConfeccionado] CHECK CONSTRAINT [FK_PedidoConfeccionado_Confeccionado]
GO

ALTER TABLE [dbo].[PedidoConfeccionado] WITH CHECK
ADD CONSTRAINT [FK_PedidoConfeccionado_Pedido] FOREIGN KEY([pedidoNo])
REFERENCES [dbo].[Pedido] ([pedidoNo])
ON UPDATE CASCADE
GO

ALTER TABLE [dbo].[PedidoConfeccionado] CHECK CONSTRAINT [FK_PedidoConfeccionado_Pedido]
GO
```

Figura 44 - DBDL para PedidoConfeccionado pt.1

```

ALTER TABLE [dbo].[PedidoConfeccionado] WITH CHECK
ADD CONSTRAINT [CK_confeccionadoQuantidade] CHECK (([quantidade]>(0)))
GO

ALTER TABLE [dbo].[PedidoConfeccionado] CHECK CONSTRAINT [CK_confeccionadoQuantidade]
GO

EXEC sys.sp_addextendedproperty @name=N'MS_Description',
@value=N'Constraint que restringe quantidade a ser superior a 0' ,
@level0type=N'SCHEMA', @level0name=N'dbo', @level1type=N'TABLE',
@level1name=N'PedidoConfeccionado', @level2type=N'CONSTRAINT',
@level2name=N'CK_confeccionadoQuantidade'
GO

```

Figura 45 – DBDL para PedidoConfeccionado pt.2

### TipoProduto:

```

USE [MichelinStar]
GO

***** Object: Table [dbo].[TipoProduto]    Script Date: 10/06/2022 01:57:04 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[TipoProduto](
    [prodTipo] [int] IDENTITY(1,1) NOT NULL,
    [prodTipoNome] [varchar](50) NOT NULL,
    [ativo] [bit] NOT NULL,
    CONSTRAINT [PK_TipoProduto] PRIMARY KEY CLUSTERED
(
    [prodTipo] 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].[TipoProduto]
ADD CONSTRAINT [DF_TipoProduto_ativo] DEFAULT ((1)) FOR [ativo]
GO

```

Figura 46 - DBDL para TipoProduto

### Tipotaxa:

```
USE [MichelinStar]
GO

/******** Object: Table [dbo].[Tipotaxa]    Script Date: 10/06/2022 01:58:25 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Tipotaxa](
    [prodTipoTaxa] [int] IDENTITY(1,1) NOT NULL,
    [prodTaxaValor] [decimal](5, 2) NOT NULL,
    [ativo] [bit] NOT NULL,
    CONSTRAINT [PK_TipoTaxa] PRIMARY KEY CLUSTERED
(
    [prodTipoTaxa] 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].[Tipotaxa]
ADD CONSTRAINT [DF_TipoTaxa_ativo] DEFAULT ((1)) FOR [ativo]
GO

ALTER TABLE [dbo].[Tipotaxa] WITH CHECK
ADD CONSTRAINT [CK_prodTipoTaxa]
CHECK (([prodTaxaValor]>=(0) AND [prodTaxaValor]<=(100)))
GO

ALTER TABLE [dbo].[Tipotaxa] CHECK CONSTRAINT [CK_prodTipoTaxa]
GO

EXEC sys.sp_addextendedproperty @name=N'MS_Description',
@value=N'Constraint que restringe o prodTaxaValor entre 0 e 100 inclusive' ,
@level0type=N'SCHEMA', @level0name=N'dbo', @level1type=N'TABLE',
@level1name=N'TipoTaxa', @level2type=N'CONSTRAINT', @level2name=N'CK_prodTipoTaxa'
GO
```

Figura 47 - DBDL para Tipotaxa

### 2.3.1.2. Desenho da representação de dados derivados

Atributos cujo seu valor depende de outros atributos são denominados atributos calculados ou derivados.

Um atributo desta natureza pode ser encontrado na relação “Pedido”, visto que o “pedPreco” depende da soma dos preços dos produtos contidos num “Pedido”.

Para satisfação do referido anteriormente, foram criados os seguintes *Triggers*:

#### PedidoArtigo:

```
USE [MichelinStar]
GO

***** Object: Trigger [dbo].[PedidoArtigo_HandleRowDelete]      Script Date: 10/06/2022 02:39:06 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

<-- =====
<-- Author: Simão Santos
<-- Create date: 04/06/2022
<-- Description: Trigger that handles the following rules:
<-- -Updates the pedPreco field on the Pedido relation
<-- =====
CREATE TRIGGER [dbo].[PedidoArtigo_HandleRowDelete]
ON [dbo].[PedidoArtigo]
AFTER DELETE
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @artigoNo INT
    DECLARE @pedidoNo INT
    DECLARE @previousQuantidade SMALLINT
    DECLARE @prodPreco DECIMAL(6,2)
    DECLARE @taxa DECIMAL(5,2)

    SELECT @artigoNo = deleted.artigoNo,
           @pedidoNo = deleted.pedidoNo,
           @previousQuantidade = deleted.quantidade
    FROM deleted

    SELECT @prodPreco = prodPreco
    FROM Artigo
    WHERE produtoNo = @artigoNo

    SELECT @taxa = TipoTaxa.prodTaxaValor
    FROM Artigo INNER JOIN TipoTaxa ON Artigo.prodTipoTaxa = TipoTaxa.prodTipoTaxa

    -- Decrease price
    UPDATE Pedido
    SET pedPreco = pedPreco - (@prodPreco * @previousQuantidade) -
        ((@prodPreco * @previousQuantidade) * @taxa / 100)
    WHERE pedidoNo = @pedidoNo
END
GO
```

Figura 48 - Trigger PedidoArtigo - HandleRowDelete

```

USE [MichelinStar]
GO

/****** Object: Trigger [dbo].[PedidoArtigo_HandleRowInsert] Script Date: 10/06/2022 02:40:22 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

<-- =====
-- Author: Simão
-- Create date: 04/06/2022
-- Description: Trigger that handles the following rules:
-- -Updates the pedPreco field on the Pedido relation
<-- =====

CREATE TRIGGER [dbo].[PedidoArtigo_HandleRowInsert]
ON [dbo].[PedidoArtigo]
AFTER INSERT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @artigoNo INT
    DECLARE @pedidoNo INT
    DECLARE @quantidade SMALLINT
    DECLARE @prodPreco DECIMAL(6,2)
    DECLARE @taxa DECIMAL(5,2)

    SELECT @artigoNo = inserted.artigoNo,
           @pedidoNo = inserted.pedidoNo,
           @quantidade = inserted.quantidade
    FROM inserted

    SELECT @prodPreco = prodPreco
    FROM Artigo
    WHERE produtoNo = @artigoNo

    SELECT @taxa = TipoTaxa.prodTaxaValor
    FROM Artigo INNER JOIN TipoTaxa ON Artigo.prodTipoTaxa = TipoTaxa.prodTipoTaxa

    -- Increase price
    UPDATE Pedido
    SET pedPreco = pedPreco + (@prodPreco * @quantidade) +
        (@prodPreco * @quantidade) * @taxa / 100
    WHERE pedidoNo = @pedidoNo
END
GO

ALTER TABLE [dbo].[PedidoArtigo] ENABLE TRIGGER [PedidoArtigo_HandleRowInsert]
GO

```

Figura 49 - Trigger PedidoArtigo - HandleRowInsert

```

USE [MichelinStar]
GO

/* Object: Trigger [dbo].[PedidoArtigo_HandleRowUpdate] Script Date: 10/06/2022 02:40:55 */
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

<-- -----
<-- Author: Simão Santos
<-- Create date: 04/06/2022
<-- Description: Trigger that handles the following rules:
<-- -Blocks the artigoNo field update
<-- -Blocks the pedidoNo field update
<-- -Updates the pedPreco field on the Pedido relation
<-- -----
CREATE TRIGGER [dbo].[PedidoArtigo_HandleRowUpdate]
    ON [dbo].[PedidoArtigo]
    AFTER UPDATE
AS
BEGIN
    SET NOCOUNT ON;

    IF UPDATE(artigoNo)
        BEGIN
            RAISERROR('Campo artigoNo não pode ser atualizado.', 16, 1)
        END

    IF UPDATE(pedidoNo)
        BEGIN
            RAISERROR('Campo pedidoNo não pode ser atualizado.', 16, 1)
        END

    IF UPDATE(quantidade)
        BEGIN
            DECLARE @artigoNo INT
            DECLARE @pedidoNo INT
            DECLARE @newQuantidade SMALLINT
            DECLARE @previousQuantidade SMALLINT
            DECLARE @prodPreco DECIMAL(6,2)
            DECLARE @taxa DECIMAL(5,2)

            SELECT @artigoNo = inserted.artigoNo,
                   @pedidoNo = inserted.pedidoNo,
                   @newQuantidade = inserted.quantidade
            FROM inserted

            SELECT @previousQuantidade = deleted.quantidade
            FROM deleted

            SELECT @prodPreco = prodPreco
            FROM Artigo
            WHERE produtoNo = @artigoNo

            SELECT @taxa = TipoTaxa.prodTaxaValor
            FROM Artigo INNER JOIN TipoTaxa ON Artigo.prodTipoTaxa = TipoTaxa.prodTipoTaxa

```

Figura 50 – Trigger PedidoArtigo - HandleRowUpdate pt.1

```
-- Decrease price
UPDATE Pedido
SET pedPreco = pedPreco - (@prodPreco * @previousQuantidade) -
    ((@prodPreco * @previousQuantidade) * @taxa / 100)
WHERE pedidoNo = @pedidoNo

-- Increase price
UPDATE Pedido
SET pedPreco = pedPreco + (@prodPreco * @newQuantidade) +
    ((@prodPreco * @newQuantidade) * @taxa / 100)
WHERE pedidoNo = @pedidoNo
END
GO

ALTER TABLE [dbo].[PedidoArtigo] ENABLE TRIGGER [PedidoArtigo_HandleRowUpdate]
GO
```

Figura 51 - Trigger PedidoArtigo - HandleRowUpdate pt.2

### PedidoConfeccionado:

```
USE [MichelinStar]
GO

***** Object: Trigger [dbo].[PedidoConfeccionado_HandleRowDelete]    Script Date: 10/06/2022 02:42:17 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

-- =====
-- Author: Simão Santos
-- Create date: 04/06/2022
-- Description: Trigger that handles the following rules:
-- -Updates the pedPreco field on the Pedido relation
-- =====

CREATE TRIGGER [dbo].[PedidoConfeccionado_HandleRowDelete]
ON [dbo].[PedidoConfeccionado]
AFTER DELETE
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @confeccionadoNo INT
    DECLARE @pedidoNo INT
    DECLARE @previousQuantidade SMALLINT
    DECLARE @prodPreco DECIMAL(6,2)
    DECLARE @taxa DECIMAL(5,2)

    SELECT @confeccionadoNo = deleted.confeccionadoNo,
           @pedidoNo = deleted.pedidoNo,
           @previousQuantidade = deleted.quantidade
    FROM deleted

    SELECT @prodPreco = prodPreco
    FROM Confeccionado
    WHERE produtoNo = @confeccionadoNo

    SELECT @taxa = TipoTaxa.prodTaxaValor
    FROM Confeccionado INNER JOIN TipoTaxa ON Confeccionado.prodTipoTaxa = TipoTaxa.prodTipoTaxa

    -- Decrease price
    UPDATE Pedido
    SET pedPreco = pedPreco - (@prodPreco * @previousQuantidade)
        - ((@prodPreco * @previousQuantidade) * @taxa / 100)
    WHERE pedidoNo = @pedidoNo
END
GO

ALTER TABLE [dbo].[PedidoConfeccionado] ENABLE TRIGGER [PedidoConfeccionado_HandleRowDelete]
GO
```

Figura 52 - Trigger PedidoConfeccionado - HandleRowDelete

```

USE [MichelinStar]
GO

***** Object: Trigger [dbo].[PedidoConfeccionado_HandleRowInsert] Script Date: 10/06/2022 02:45:02 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

<-- =====
-- Author: Simão
-- Create date: 04/06/2022
-- Description: Trigger that handles the following rules:
-- -Updates the pedPreco field on the Pedido relation
<-- =====

CREATE TRIGGER [dbo].[PedidoConfeccionado_HandleRowInsert]
ON [dbo].[PedidoConfeccionado]
AFTER INSERT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @confeccionadoNo INT
    DECLARE @pedidoNo INT
    DECLARE @quantidade SMALLINT
    DECLARE @prodPreco DECIMAL(6,2)
    DECLARE @taxa DECIMAL(5,2)

    SELECT @confeccionadoNo = inserted.confeccionadoNo,
           @pedidoNo = inserted.pedidoNo,
           @quantidade = inserted.quantidade
    FROM inserted

    SELECT @prodPreco = prodPreco
    FROM Confeccionado
    WHERE produtoNo = @confeccionadoNo

    SELECT @taxa = TipoTaxa.prodTaxaValor
    FROM Confeccionado INNER JOIN TipoTaxa ON Confeccionado.prodTipoTaxa = TipoTaxa.prodTipoTaxa

    -- Increase price
    UPDATE Pedido
    SET pedPreco = pedPreco + (@prodPreco * @quantidade) + ((@prodPreco * @quantidade) * @taxa / 100)
    WHERE pedidoNo = @pedidoNo
END
GO

ALTER TABLE [dbo].[PedidoConfeccionado] ENABLE TRIGGER [PedidoConfeccionado_HandleRowInsert]
GO

```

*Figura 53 - Trigger PedidoConfeccionado - HandleRowInsert*

```

USE [MichelinStar]
GO

/****** Object: Trigger [dbo].[PedidoConfeccionado_HandleRowUpdate] Script Date: 10/06/2022 02:45:19 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

<-- ****
-- Author: Simão Santos
-- Create date: 04/06/2022
-- Description: Trigger that handles the following rules:
-- -Blocks the artigoNo field update
-- -Blocks the pedidoNo field update
-- -Updates the pedPreco field on the Pedido relation
-- ****
CREATE TRIGGER [dbo].[PedidoConfeccionado_HandleRowUpdate]
    ON [dbo].[PedidoConfeccionado]
    AFTER UPDATE
AS
BEGIN
    SET NOCOUNT ON;

    IF UPDATE(confeccionadoNo)
        BEGIN
            RAISERROR('Campo confeccionadoNo não pode ser atualizado.', 16, 1)
        END

    IF UPDATE(pedidoNo)
        BEGIN
            RAISERROR('Campo pedidoNo não pode ser atualizado.', 16, 1)
        END

    IF UPDATE(quantidade)
        BEGIN
            DECLARE @confeccionadoNo INT
            DECLARE @pedidoNo INT
            DECLARE @newQuantidade SMALLINT
            DECLARE @previousQuantidade SMALLINT
            DECLARE @prodPreco DECIMAL(6,2)
            DECLARE @taxa DECIMAL(5,2)

            SELECT @confeccionadoNo = inserted.confeccionadoNo,
                   @pedidoNo = inserted.pedidoNo,
                   @newQuantidade = inserted.quantidade
            FROM inserted

            SELECT @previousQuantidade = deleted.quantidade
            FROM deleted

            SELECT @prodPreco = prodPreco
            FROM Confeccionado
            WHERE produtoNo = @confeccionadoNo

            SELECT @taxa = TipoTaxa.prodTaxaValor
            FROM Confeccionado INNER JOIN TipoTaxa ON Confeccionado.prodTipoTaxa = TipoTaxa.prodTipoTaxa
        END

```

Figura 54 - Trigger PedidoConfeccionado – HandleRowUpdate pt.1

```
-- Decrease price
UPDATE Pedido
SET pedPreco = pedPreco - (@prodPreco * @previousQuantidade)
    - ((@prodPreco * @previousQuantidade) * @taxa / 100)
WHERE pedidoNo = @pedidoNo

-- Increase price
UPDATE Pedido
SET pedPreco = pedPreco + (@prodPreco * @newQuantidade)
    + ((@prodPreco * @newQuantidade) * @taxa / 100)
WHERE pedidoNo = @pedidoNo
END
GO

ALTER TABLE [dbo].[PedidoConfeccionado] ENABLE TRIGGER [PedidoConfeccionado_HandleRowUpdate]
GO
```

Figura 55 - Trigger PedidoConfeccionado - HandleRowUpdate pt.2

### 2.3.2. Análise de Transações

(A) Inserir os detalhes de um Pedido (número de mesa, observações e o funcionário associado ao Pedido)	}	Pedido View
(B) Atualizar/Eliminar/Fehar um Pedido		
(C) Adicionar/Atualizar/Remover um Artigo/Confeccionado a um Pedido		
(D) Obter a Fatura de um Pedido		

Figura 56 – Transações de Utilizador associadas à View Pedido

Transação/Relação	(A)				(B)				(C)				(D)			
	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D
Pedido	X					X	X	X		X				X		
Funcionario		X														
PedidoArtigo									X	X	X	X			X	
PedidoConfecionado									X	X	X	X			X	
MetodoPagamento						X										

Tabela 38 - Transações e relações de referência cruzada

### 2.3.2.1. Stored Procedures e Funções que satisfazem as transações da View Pedido

#### SP InsertPedido:

```
USE [MichelinStar]
GO

/****** Object:  StoredProcedure [dbo].[spPedido_InsertPedido]    Script Date: 10/06/2022 13:35:59 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

<-- -----
-- Author:  Simão Santos
-- Create date: 05/06/2022
-- Description: Insert Pedido
<-- ----->

CREATE PROCEDURE [dbo].[spPedido_InsertPedido]
    @pedNumeroMesa INT,
    @pedObs VARCHAR(1000) = NULL,
    @fnif CHAR(9)
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @tableIsOccupied INT
    DECLARE @fnifIsActive CHAR(9)

    SELECT @tableIsOccupied = COUNT(pedidoNo)
    FROM Pedido
    WHERE pedNumeroMesa = @pedNumeroMesa AND pedEstado = 'A' AND ativo = 1

    SELECT @fnifIsActive = ativo
    FROM Funcionario
    WHERE fnif = @fnif

    -- Check if Funcionario is active
    IF(@fnifIsActive = 0)
        BEGIN
            RAISERROR('O Funcionário é inválido, já se encontra eliminado.', 16, 1)
            RETURN
        END

    -- Check if Table is occupied
    IF(@tableIsOccupied != 0)
        BEGIN
            RAISERROR('Já existe um Pedido aberto com esta mesa atribuída.', 16, 1)
            RETURN
        END

```

Figura 57 – Stored Procedure InsertPedido pt.1

```
-- Insert Pedido
BEGIN TRY
    INSERT INTO Pedido(pedNumeroMesa, pedObs, pedPreco, pedDataCriacao,
    pedEstado, pedNumeroContribuinte, pedMetodoPagamento, ativo, fnif)
    VALUES(@pedNumeroMesa, @pedObs, 0, GETDATE(), 'A', NULL, NULL, 1, @fnif)
    PRINT 'Pedido inserido corretamente.'
END TRY
BEGIN CATCH
    DECLARE @ErrorMessage NVARCHAR(4000);
    DECLARE @ErrorSeverity INT;
    DECLARE @ErrorState INT;

    SELECT @ErrorMessage = ERROR_MESSAGE(),
    @ErrorSeverity = ERROR_SEVERITY(),
    @ErrorState = ERROR_STATE()

    RAISERROR (@ErrorMessage,
    @ErrorSeverity,
    @ErrorState)
END CATCH
END
GO
```

Figura 58 - Stored Procedure InsertPedido pt.2

### **SP ClosePedido:**

```
USE [MichelinStar]
GO

***** Object: StoredProcedure [dbo].[spPedido_ClosePedido]    Script Date: 10/06/2022 13:37:53 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

<-- -----
-- Author: Simão Santos
-- Create date: 05/06/2022
-- Description: Close Pedido
-- -----

CREATE PROCEDURE [dbo].[spPedido_ClosePedido]
    @pedidoNo INT,
    @pedNumeroContribuinte CHAR(9) = NULL,
    @pedMetodoPagamento INT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @products INT
    DECLARE @ativo BIT
    DECLARE @pedEstado CHAR(1)
    DECLARE @pedMetodoPagamentoIsActive BIT

    SELECT @ativo = ativo, @pedEstado = pedEstado
    FROM Pedido
    WHERE pedidoNo = @pedidoNo

    SELECT @products = COUNT(artigoNo)
    FROM PedidoArtigo
    WHERE pedidoNo = @pedidoNo

    SELECT @pedMetodoPagamentoIsActive = ativo
    FROM MetodoPagamento
    WHERE pedMetodoPagamento = @pedMetodoPagamento

    -- Check if Pedido exists
    IF(@ativo IS NULL)
        BEGIN
            RAISERROR('O Pedido não existe.', 16, 1)
            RETURN
        END

    -- Check if Pedido is active
    IF (@ativo = 0)
        BEGIN
            RAISERROR('Pedidos eliminados não podem ser fechados.', 16, 1)
            RETURN
        END
```

Figura 59 - Stored Procedure ClosePedido pt.1

```

-- Check if Pedido is already closed
IF(@pedEstado = 'F')
    BEGIN
        RAISERROR('O Pedido já se econtra fechado.', 16, 1)
        RETURN
    END

-- Check if Pedido has products
IF (@products = 0)
    BEGIN
        RAISERROR('Pedidos com 0 produtos associados não podem ser fechados.', 16, 1)
        RETURN
    END

-- Check if MetodoPagamento is active
IF (@pedMetodoPagamentoIsActive = 0)
    BEGIN
        RAISERROR('O Método de Pagamento não é válido, visto que foi previamente eliminado.', 16, 1)
        RETURN
    END

-- Close Pedido
BEGIN TRY
    UPDATE Pedido
    SET pedNumeroContribuinte = @pedNumeroContribuinte,
        pedMetodoPagamento = @pedMetodoPagamento,
        pedEstado = 'F'
    WHERE pedidoNo = @pedidoNo
    PRINT 'Pedido fechado corretamente.'
END TRY
BEGIN CATCH
    DECLARE @ErrorMessage NVARCHAR(4000);
    DECLARE @ErrorSeverity INT;
    DECLARE @ErrorState INT;

    SELECT @ErrorMessage = ERROR_MESSAGE(),
    @ErrorSeverity = ERROR_SEVERITY(),
    @ErrorState = ERROR_STATE()

    RAISERROR (@ErrorMessage,
    @ErrorSeverity,
    @ErrorState)
END CATCH
END
GO

```

*Figura 60 - Stored Procedure ClosePedido pt.2*

### **SP DeletePedido:**

```
USE [MichelinStar]
GO

***** Object: StoredProcedure [dbo].[spPedido_DeletePedido]    Script Date: 10/06/2022 13:42:18 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

<-- -----
-- Author: Simão Santos
-- Create date: 05/06/2022
-- Description: Delete Pedido
<-- -----

CREATE PROCEDURE [dbo].[spPedido_DeletePedido]
    @pedidoNo INT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @pedEstado CHAR(1)
    DECLARE @ativo BIT

    SELECT @pedEstado = pedEstado, @ativo = ativo
    FROM Pedido
    WHERE pedidoNo = @pedidoNo

    -- Check if Pedido exists
    IF(@ativo IS NULL)
        BEGIN
            RAISERROR('O Pedido não existe.', 16, 1)
            RETURN
        END

    -- Check if Pedido is active
    IF(@ativo = 0)
        BEGIN
            RAISERROR('O Pedido já se encontra eliminado.', 16, 1)
            RETURN
        END

    -- Check if Pedido is closed
    IF(@pedEstado = 'F')
        BEGIN
            RAISERROR('Pedidos fechados não podem ser eliminados.', 16, 1)
            RETURN
        END
```

Figura 61 - Stored Procedure DeletePedido pt.1

```
-- Delete Pedido and Delete Artigos From Pedido
BEGIN TRY
    UPDATE Pedido
    SET ativo = 0
    WHERE pedidoNo = @pedidoNo

    DELETE FROM PedidoArtigo
    WHERE pedidoNo = @pedidoNo
    PRINT 'Pedido eliminado corretamente.'
END TRY
BEGIN CATCH
    DECLARE @ErrorMessage NVARCHAR(4000);
    DECLARE @ErrorSeverity INT;
    DECLARE @ErrorState INT;

    SELECT @ErrorMessage = ERROR_MESSAGE(),
    @ErrorSeverity = ERROR_SEVERITY(),
    @ErrorState = ERROR_STATE()

    RAISERROR (@ErrorMessage,
    @ErrorSeverity,
    @ErrorState)
END CATCH
END
GO
```

Figura 62 – Stored Procedure DeletePedido pt.2

### **SP AddConfeccionadoToPedido:**

```
USE [MichelinStar]
GO

***** Object: StoredProcedure [dbo].[spPedido_AddConfeccionadoToPedido]    Script Date: 10/06/2022 13:43:11 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

<-- -----
-- Author: Simão Santos
-- Create date: 06/06/2022
-- Description: Add Confeccionado to Pedido
<-- -----

CREATE PROCEDURE [dbo].[spPedido_AddConfeccionadoToPedido]
    @pedidoNo INT,
    @confeccionadoNo INT,
    @quantidade INT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @pedEstado CHAR(1)
    DECLARE @ativo BIT
    DECLARE @confeccionadoIsActive BIT
    DECLARE @confeccionadoHasIngredientes INT

    SELECT @pedEstado = pedEstado, @ativo = ativo
    FROM Pedido
    WHERE pedidoNo = @pedidoNo

    SELECT @confeccionadoIsActive = ativo
    FROM Confeccionado
    WHERE produtoNo = @confeccionadoNo

    SELECT @confeccionadoHasIngredientes = COUNT(produtoNo)
    FROM ConfeccionadoIngrediente
    WHERE produtoNo = @confeccionadoNo

    -- Check if Pedido is active
    IF(@ativo = 0)
        BEGIN
            RAISERROR('Confeccionados não podem ser adicionados a Pedidos eliminados.', 16, 1)
            RETURN
        END

    -- Check if Pedido is closed
    IF(@pedEstado = 'F')
        BEGIN
            RAISERROR('Confeccionados não podem ser adicionados a Pedidos fechados.', 16, 1)
            RETURN
        END
```

Figura 63 - Stored Procedure AddConfeccionadoToPedido pt. 1

```

-- Check if Confeccionado is active
IF(@confeccionadoIsActive = 0)
    BEGIN
        RAISERROR('O Confeccionado não é válido, visto que foi previamente eliminado.', 16, 1)
        RETURN
    END

-- Check if Confeccionado has Ingredientes
IF(@confeccionadoHasIngredientes = 0)
    BEGIN
        RAISERROR('Confeccionados com 0 Ingredientes são inválidos.', 16, 1)
        RETURN
    END

-- Add Confeccionado to Pedido
BEGIN TRY
    INSERT INTO PedidoConfeccionado(pedidoNo, confeccionadoNo, quantidade)
    VALUES(@pedidoNo, @confeccionadoNo, @quantidade)
    PRINT 'Confeccionado adicionado ao Pedido corretamente.'
END TRY
BEGIN CATCH
    DECLARE @ErrorMessage NVARCHAR(4000);
    DECLARE @ErrorSeverity INT;
    DECLARE @ErrorState INT;

    SELECT @ErrorMessage = ERROR_MESSAGE(),
    @ErrorSeverity = ERROR_SEVERITY(),
    @ErrorState = ERROR_STATE()

    RAISERROR (@ErrorMessage,
    @ErrorSeverity,
    @ErrorState)
END CATCH
END
GO

```

Figura 64 - Stored Procedure AddConfeccionadoToPedido pt.2

### **SP addArtigoToPedido:**

```
USE [MichelinStar]
GO

/****** Object: StoredProcedure [dbo].[spPedido_AddArtigoToPedido]      Script Date: 10/06/2022 13:44:44 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

-----
-- Author: Simão Santos
-- Create date: 05/06/2022
-- Description: Add Artigo to Pedido
-----

CREATE PROCEDURE [dbo].[spPedido_AddArtigoToPedido]
    @pedidoNo INT,
    @artigoNo INT,
    @quantidade INT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @pedEstado CHAR(1)
    DECLARE @ativo BIT
    DECLARE @artigoIsActive BIT

    SELECT @pedEstado = pedEstado, @ativo = ativo
    FROM Pedido
    WHERE pedidoNo = @pedidoNo

    SELECT @artigoIsActive = ativo
    FROM Artigo
    WHERE produtoNo = @artigoNo

    -- Check if Pedido is active
    IF(@ativo = 0)
        BEGIN
            RAISERROR('Artigos não podem ser adicionados a Pedidos eliminados.', 16, 1)
            RETURN
        END

    -- Check if Pedido is closed
    IF(@pedEstado = 'F')
        BEGIN
            RAISERROR('Artigos não podem ser adicionados a Pedidos fechados.', 16, 1)
            RETURN
        END
```

Figura 65 - Stored Procedure AddArtigoToPedido pt.1

```

-- Check if Artigo is active
IF (@artigoIsActive = 0)
BEGIN
    RAISERROR('O Artigo não é válido, visto que foi previamente eliminado.', 16, 1)
    RETURN
END

-- Add artigo to Pedido
BEGIN TRY
    INSERT INTO PedidoArtigo(pedidoNo, artigoNo, quantidade)
    VALUES(@pedidoNo, @artigoNo, @quantidade)
    PRINT 'Artigo adicionado ao Pedido corretamente.'
END TRY
BEGIN CATCH
    DECLARE @ErrorMessage NVARCHAR(4000);
    DECLARE @ErrorSeverity INT;
    DECLARE @ErrorState INT;

    SELECT @ErrorMessage = ERROR_MESSAGE(),
    @ErrorSeverity = ERROR_SEVERITY(),
    @ErrorState = ERROR_STATE()

    RAISERROR (@ErrorMessage,
    @ErrorSeverity,
    @ErrorState)
END CATCH
END
GO

```

*Figura 66 - Stored Procedure AddArtigoToPedido pt.2*

### **Função getFaturaFromPedido:**

```
USE [MichelinStar]
GO

***** Object: UserDefinedFunction [dbo].[getFaturaFromPedido] Script Date: 10/06/2022 14:10:28 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

<-- -----
-- Author: Simão Santos
-- Create date: 09/06/2022
-- Description: get the Fatura from Pedido
-- -----

CREATE FUNCTION [dbo].[getFaturaFromPedido]
(
    @pedido INT
)
RETURNS
@fatura TABLE
(
    produtoNo INT,
    quantidade SMALLINT,
    preco VARCHAR(30),
    iva VARCHAR(30),
    total DECIMAL(7,2)
)
AS
BEGIN
    INSERT INTO @fatura
    SELECT Artigo.produtoNo, PedidoArtigo.quantidade, CONCAT(CONVERT(VARCHAR(30), Artigo.prodPreco), '€'),
           CONCAT(CONVERT(VARCHAR(30), TipoTaxa.prodTaxaValor), '%'),
           (Artigo.prodPreco * PedidoArtigo.quantidade) + ((Artigo.prodPreco * PedidoArtigo.quantidade) *
           TipoTaxa.prodTaxaValor / 100)
    FROM PedidoArtigo
    INNER JOIN Pedido ON PedidoArtigo.pedidoNo = Pedido.pedidoNo
    INNER JOIN Artigo ON Artigo.produtoNo = PedidoArtigo.artigoNo
    INNER JOIN TipoTaxa ON TipoTaxa.prodTipoTaxa = Artigo.prodTipoTaxa
    WHERE Pedido.pedidoNo = @pedido

    INSERT INTO @fatura
    SELECT Confeccionado.produtoNo, PedidoConfeccionado.quantidade,
           CONCAT(CONVERT(VARCHAR(30), Confeccionado.prodPreco), '€'),
           CONCAT(CONVERT(VARCHAR(30), TipoTaxa.prodTaxaValor), '%'),
           (Confeccionado.prodPreco * PedidoConfeccionado.quantidade) +
           ((Confeccionado.prodPreco * PedidoConfeccionado.quantidade) *
           TipoTaxa.prodTaxaValor / 100)
    FROM PedidoConfeccionado
    INNER JOIN Pedido ON PedidoConfeccionado.pedidoNo = Pedido.pedidoNo
    INNER JOIN Confeccionado ON Confeccionado.produtoNo = PedidoConfeccionado.confeccionadoNo
    INNER JOIN TipoTaxa ON TipoTaxa.prodTipoTaxa = Confeccionado.prodTipoTaxa
    WHERE Pedido.pedidoNo = @pedido
```

Figura 67 – Função getFaturaFromPedido pt.1

### 2.3.2.2. Outras Transações de Utilizador relevantes para o negócio

#### Função getEmentaAmanha:

```
USE [MichelinStar]
GO

***** Object: UserDefinedFunction [dbo].[getEmentaAmanha]     Script Date: 10/06/2022 14:29:19 *****
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

-- =====
-- Author: Simão e Rui
-- Create date: 09/06/2022
-- Description: get the tomorrow's Ementa
-- =====

CREATE FUNCTION [dbo].[getEmentaAmanha]
(
)
RETURNS
@ementa TABLE (
    diaSemana VARCHAR(30),
    produtoNo INT,
    nome VARCHAR(30),
    prodTipo VARCHAR(50),
    descricao VARCHAR(150),
    calorias VARCHAR(50),
    comGluten BIT)
AS
BEGIN
    INSERT INTO @ementa
    SELECT DATENAME(WEEKDAY, GETDATE() + 1) AS diaSemana, Artigo.productoNo, Artigo.prodNome,
           TipoProduto.prodTipoNome AS prodTipo, Artigo.prodDescricao,
           CONCAT(CONVERT(VARCHAR(30), Artigo.prodCalorias), 'Kcal') AS prodCalorias,
           Artigo.prodComGluten
    FROM EmentaArtigo
    INNER JOIN Artigo ON EmentaArtigo.artigoNo = Artigo.productoNo
    INNER JOIN TipoProduto ON Artigo.prodTipo = TipoProduto.prodTipo
    WHERE EmentaArtigo.diaSemana = DATEPART(dw, GETDATE()) + 1 AND Artigo.ativo = 1

    INSERT INTO @ementa
    SELECT DATENAME(WEEKDAY, GETDATE() + 1) AS diaSemana, Confeccionado.productoNo,
           Confeccionado.prodNome, TipoProduto.prodTipoNome AS prodTipo,
           Confeccionado.prodDescricao, CONCAT(CONVERT(VARCHAR(30),
           Confeccionado.prodCalorias), 'Kcal') AS prodCalorias,
           Confeccionado.prodComGluten
    FROM EmentaConfeccionado
    INNER JOIN Confeccionado ON EmentaConfeccionado.confeccionadoNo = Confeccionado.productoNo
    INNER JOIN TipoProduto ON Confeccionado.prodTipo = TipoProduto.prodTipo
    WHERE EmentaConfeccionado.diaSemana = DATEPART(dw, GETDATE()) + 1 AND Confeccionado.ativo = 1

    RETURN
END
GO
```

Figura 68 - Função getEmentaAmanha

	diaSemana	produtoNo	nome	prodTipo	descricao	calorias	comGluten
1	Saturday	9	Coca-Cola	Bebida	NULL	150Kcal	1
2	Saturday	10	Vinho da Casa	Bebida	NULL	200Kcal	0
3	Saturday	11	Gelado Morango Olá	Sobremesa	NULL	170Kcal	1
4	Saturday	13	Pão	Entrada	NULL	230Kcal	1
5	Saturday	15	Áqua Luso	Bebida	NULL	0Kcal	0
6	Saturday	16	Rissol	Entrada	NULL	90Kcal	1
7	Saturday	17	Moet & Chandon	Bebida	NULL	250Kcal	1
8	Saturday	18	Camaraao	Entrada	NULL	99Kcal	0
9	Saturday	19	Polvo	Entrada	NULL	150Kcal	0
10	Saturday	21	Vinho Tinto E.A.	Bebida	NULL	210Kcal	1
11	Saturday	22	Vinho Branco E.A.	Bebida	NULL	205Kcal	1
12	Saturday	23	Gelado Chocolate Olá	Sobremesa	NULL	180Kcal	1
13	Saturday	25	Bolo de Bolacha	Sobremesa	NULL	280Kcal	1
14	Saturday	26	Baba de Camelo	Sobremesa	NULL	260Kcal	1
15	Saturday	27	CheeseCake de Frutos Vermelhos	Sobremesa	NULL	230Kcal	1
16	Saturday	29	Cerveja SuperBock	Bebida	NULL	150Kcal	1
17	Saturday	5006	Arroz de pato	Prato de Carne	NULL	460Kcal	1
18	Saturday	5013	Açorda com Molho de Tomate	Prato de Peixe	NULL	510Kcal	0

Figura 69 - Output da Função getEmentaAmanha

### Função getEmentaHoje:

```
USE [MichelinStar]
GO

/* Object: UserDefinedFunction [dbo].[getEmentaHoje] Script Date: 10/06/2022 14:31:23 */
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

-- =====
-- Author: Simão e Rui
-- Create date: 08/06/2022
-- Description: get the today's Ementa
-- =====

CREATE FUNCTION [dbo].[getEmentaHoje]
(
)
RETURNS
@ementa TABLE (
    diaSemana VARCHAR(30),
    produtoNo INT,
    nome VARCHAR(30),
    prodTipo VARCHAR(50),
    descricao VARCHAR(150),
    calorias VARCHAR(50),
    comGluten BIT)
AS
BEGIN
    INSERT INTO @ementa
    SELECT DATENAME(WEEKDAY, GETDATE()) AS diaSemana, Artigo.produtoNo, Artigo.prodNome,
           TipoProduto.prodTipoNome AS prodTipo, Artigo.prodDescricao,
           CONCAT(CONVERT(VARCHAR(30), Artigo.prodCalorias), 'Kcal') AS prodCalorias,
           Artigo.prodComGluten
    FROM EmentaArtigo
    INNER JOIN Artigo ON EmentaArtigo.artigoNo = Artigo.produtoNo
    INNER JOIN TipoProduto ON Artigo.prodTipo = TipoProduto.prodTipo
    WHERE EmentaArtigo.diaSemana = DATEPART(dw, GETDATE()) AND Artigo.ativo = 1

    INSERT INTO @ementa
    SELECT DATENAME(WEEKDAY, GETDATE()), Confeccionado.produtoNo, Confeccionado.prodNome,
           TipoProduto.prodTipoNome AS prodTipo, Confeccionado.prodDescricao,
           CONCAT(CONVERT(VARCHAR(30), Confeccionado.prodCalorias), 'Kcal') AS prodCalorias,
           Confeccionado.prodComGluten
    FROM EmentaConfeccionado
    INNER JOIN Confeccionado ON EmentaConfeccionado.confeccionadoNo = Confeccionado.produtoNo
    INNER JOIN TipoProduto ON Confeccionado.prodTipo = TipoProduto.prodTipo
    WHERE EmentaConfeccionado.diaSemana = DATEPART(dw, GETDATE()) AND Confeccionado.ativo = 1

    RETURN
END
GO
```

Figura 70 - Função getEmentaHoje

	Results	Messages					
	diaSemana	produtoNo	nome	prodTipo	descricao	calorias	comGluten
1	Friday	9	Coca-Cola	Bebida	NULL	150Kcal	1
2	Friday	10	Vinho da Casa	Bebida	NULL	200Kcal	0
3	Friday	13	Pão	Entrada	NULL	230Kcal	1
4	Friday	15	Água Luso	Bebida	NULL	0Kcal	0
5	Friday	16	Rissol	Entrada	NULL	90Kcal	1
6	Friday	18	Camarao	Entrada	NULL	99Kcal	0
7	Friday	19	Polvo	Entrada	NULL	150Kcal	0
8	Friday	21	Vinho Tinto E.A.	Bebida	NULL	210Kcal	1
9	Friday	22	Vinho Branco E.A.	Bebida	NULL	205Kcal	1
10	Friday	23	Gelado Chocolate Olá	Sobremesa	NULL	180Kcal	1
11	Friday	24	Gelado Baunilha Olá	Sobremesa	NULL	180Kcal	1
12	Friday	25	Bolo de Bolacha	Sobremesa	NULL	280Kcal	1
13	Friday	26	Baba de Camelo	Sobremesa	NULL	260Kcal	1
14	Friday	27	CheeseCake de Frutos Vermelhos	Sobremesa	NULL	230Kcal	1
15	Friday	29	Cerveja SuperBock	Bebida	NULL	150Kcal	1
16	Friday	5005	Francesinha	Prato de Carne	NULL	590Kcal	1
17	Friday	5012	Cavalas no Forno em Tomate	Prato de Peixe	NULL	470Kcal	0

Figura 71 - Output da Função `getEmentaHoje`

### **Função getFaturaFromPedido:**

```
USE [MichelinStar]
GO

***** Object: UserDefinedFunction [dbo].[getFaturaFromPedido]    Script Date: 10/06/2022 14:44:17 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

<-- =====
-- Author: Simão Santos
-- Create date: 09/06/2022
-- Description: get the Fatura from Pedido
-- =====
CREATE FUNCTION [dbo].[getFaturaFromPedido]
(
    @pedido INT
)
RETURNS
@fatura TABLE
(
    produtoNo INT,
    quantidade SMALLINT,
    preco VARCHAR(30),
    iva VARCHAR(30),
    total DECIMAL(7,2)
)
AS
BEGIN
    INSERT INTO @fatura
    SELECT Artigo.productoNo, PedidoArtigo.quantidade, CONCAT(CONVERT(VARCHAR(30), Artigo.prodPreco), '€'),
           CONCAT(CONVERT(VARCHAR(30), TipoTaxa.prodTaxaValor), '%'),
           ((Artigo.prodPreco * PedidoArtigo.quantidade) + ((Artigo.prodPreco * PedidoArtigo.quantidade) * TipoTaxa.prodTaxaValor / 100))
    FROM PedidoArtigo
    INNER JOIN Pedido ON PedidoArtigo.pedidoNo = Pedido.pedidoNo
    INNER JOIN Artigo ON Artigo.productoNo = PedidoArtigo.artigoNo
    INNER JOIN TipoTaxa ON TipoTaxa.prodTipoTaxa = Artigo.prodTipoTaxa
    WHERE Pedido.pedidoNo = @pedido

    INSERT INTO @fatura
    SELECT Confeccionado.productoNo, PedidoConfeccionado.quantidade,
           CONCAT(CONVERT(VARCHAR(30), Confeccionado.prodPreco), '€'),
           CONCAT(CONVERT(VARCHAR(30), TipoTaxa.prodTaxaValor), '%'),
           ((Confeccionado.prodPreco * PedidoConfeccionado.quantidade) +
           ((Confeccionado.prodPreco * PedidoConfeccionado.quantidade) * TipoTaxa.prodTaxaValor / 100))
    FROM PedidoConfeccionado
    INNER JOIN Pedido ON PedidoConfeccionado.pedidoNo = Pedido.pedidoNo
    INNER JOIN Confeccionado ON Confeccionado.productoNo = PedidoConfeccionado.confeccionadoNo
    INNER JOIN TipoTaxa ON TipoTaxa.prodTipoTaxa = Confeccionado.prodTipoTaxa
    WHERE Pedido.pedidoNo = @pedido

    RETURN
END
GO
```

Figura 72 - Função getFaturaFromPedido

	produtoNo	quantidade	preco	iva	total
1	9	4	1.00€	23.00%	4.92
2	11	2	3.50€	23.00%	8.61
3	13	5	0.30€	6.00%	1.59
4	5004	1	12.00€	13.00%	13.56
5	5011	1	28.00€	13.00%	31.64

Figura 73 - Output da Função getFaturaFromPedido

### Função getIngredientesAmanha:

```

USE [MichelinStar]
GO

***** Object: UserDefinedFunction [dbo].[getIngredientesAmanha]      Script Date: 10/06/2022 14:46:55 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

<-- =====
-- Author: Simão
-- Create date: 09/06/2022
-- Description: get tomorrow's Ingredientes
-- =====
CREATE FUNCTION [dbo].[getIngredientesAmanha]
(
)
RETURNS TABLE
AS
RETURN
(
    SELECT DATENAME(WEEKDAY, GETDATE() + 1) AS diaSemana, Confeccionado.productoNo, Confeccionado.prodNome,
           Ingrediente.ingredienteNo, Ingrediente.ingNome, CONCAT(CONVERT(VARCHAR(30),
           ConfeccionadoIngrediente.ingQuantidade), CONVERT(VARCHAR(30),
           ConfeccionadoIngrediente.ingUnidadeMedida)) AS quantidadePorPrato
    FROM Ementa
        INNER JOIN EmentaConfeccionado
            ON Ementa.diaSemana = EmentaConfeccionado.diaSemana
        INNER JOIN Confeccionado
            ON EmentaConfeccionado.confecionadoNo = Confeccionado.productoNo
        INNER JOIN ConfeccionadoIngrediente
            ON Confeccionado.productoNo = ConfeccionadoIngrediente.productoNo
        INNER JOIN Ingrediente
            ON ConfeccionadoIngrediente.ingredienteNo = Ingrediente.ingredienteNo
    WHERE Ementa.diaSemana = DATEPART(dw, GETDATE()) + 1 AND Confeccionado.ativo = 1 AND Ingrediente.ativo = 1
)
GO

```

Figura 74 - Função getIngredientesAmanha

	diaSemana	produtoNo	prodNome	ingredienteNo	ingNome	quantidadePorPrato
1	Saturday	5006	Arroz de pato	5	Arroz	0.200Kg
2	Saturday	5006	Arroz de pato	23	Pato	0.150Kg
3	Saturday	5006	Arroz de pato	24	Presunto	0.050Kg
4	Saturday	5006	Arroz de pato	29	Chourica	0.020Kg
5	Saturday	5013	Açorda com Molho de Tomate	32	Molho de Tomate	0.050Kg
6	Saturday	5013	Açorda com Molho de Tomate	38	Limão	1.000Un
7	Saturday	5013	Açorda com Molho de Tomate	40	Alho	1.000Un
8	Saturday	5013	Açorda com Molho de Tomate	42	Açorda	0.150Kg

Figura 75 - Output da Função getIngredientesAmanha

### Função whatWasSoldBetweenTwoDates:

```
USE [MichelinStar]
GO
/****** Object: UserDefinedFunction [dbo].[whatWasSoldBetweenTwoDates] Script Date: 10/06/2022 15:03:47 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
<-- ****
-- Author: Rui Neto
-- Create date: 09/06/2022
-- Description: What products of a certain type
-- did we sell between two dates
<-- ****
ALTER FUNCTION [dbo].[whatWasSoldBetweenTwoDates]
(
    @prodTipo INT,
    @startDate date,
    @finishDate date
)
RETURNS
@whatWeSell TABLE (
    produtoNo INT,
    prodNome VARCHAR(30)
)
AS
BEGIN

DECLARE @produtos AS TABLE(produtoNo INT, prodNome VARCHAR(30))

INSERT INTO @produtos
SELECT PedidoConfeccionado.confeccionadoNo, Confeccionado.prodNome
FROM Pedido INNER JOIN PedidoConfeccionado ON Pedido.pedidoNo = PedidoConfeccionado.pedidoNo
RIGHT JOIN Confeccionado ON PedidoConfeccionado.confeccionadoNo = Confeccionado.produtoNo
INNER JOIN TipoProduto ON TipoProduto.prodTipo = Confeccionado.prodTipo
WHERE Pedido.pedDataCriacao >= @startDate AND Pedido.pedDataCriacao <= @finishDate AND Confeccionado.prodTipo = @prodTipo
GROUP BY PedidoConfeccionado.confeccionadoNo, Confeccionado.prodNome

INSERT INTO @produtos
SELECT PedidoArtigo.artigoNo, Artigo.prodNome
FROM Pedido INNER JOIN PedidoArtigo ON Pedido.pedidoNo = PedidoArtigo.pedidoNo
RIGHT JOIN Artigo ON PedidoArtigo.artigoNo = Artigo.produtoNo
INNER JOIN TipoProduto ON TipoProduto.prodTipo = Artigo.prodTipo
WHERE Pedido.pedDataCriacao >= @startDate AND Pedido.pedDataCriacao <= @finishDate AND Artigo.prodTipo = @prodTipo
GROUP BY PedidoArtigo.artigoNo, Artigo.prodNome

INSERT INTO @whatWeSell
SELECT produtoNo, prodNome
FROM @produtos
GROUP BY produtoNo, prodNome

RETURN
END
```

Figura 76 – Função whatWasSoldBetweenTwoDates

Results		Messages
	produtoNo	prodNome
1	11	Gelado Morango Olá
2	23	Gelado Chocolate Olá
3	24	Gelado Baunilha Olá
4	25	Bolo de Bolacha
5	26	Baba de Camelô

Figura 77 - Output da Função whatWasSoldBetweenTwoDates

### Função onWichDayWereSoldTogether:

```
USE [MichelinStar]
GO

/* Object: UserDefinedFunction [dbo].[onWichDaysWereSoldTogether] Script Date: 10/06/2022 15:08:55 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

<-- =====
-- Author: Simão
-- Create date: 09/06/2022
-- Description: On which days 2 specific products were sold
<-- =====
CREATE FUNCTION [dbo].[onWichDaysWereSoldTogether]
(
    @produtoNo1 INT,
    @produtoNo2 INT
)
RETURNS
@datas TABLE
(
    dia DATE
)
AS
BEGIN
    DECLARE @pedidoProduto TABLE(pedidoNo INT, produtoNo INT)
    DECLARE @datasTemp TABLE(dia DATE, produtoNo INT)

    INSERT INTO @pedidoProduto
    SELECT PedidoArtigo.pedidoNo, PedidoArtigo.artigoNo
    FROM PedidoArtigo
    WHERE PedidoArtigo.artigoNo = @produtoNo1 OR PedidoArtigo.artigoNo = @produtoNo2

    INSERT INTO @pedidoProduto
    SELECT PedidoConfeccionado.pedidoNo, PedidoConfeccionado.confecionadoNo
    FROM PedidoConfeccionado
    WHERE PedidoConfeccionado.confecionadoNo = @produtoNo1 OR PedidoConfeccionado.confecionadoNo = @produtoNo2

    INSERT INTO @datasTemp
    SELECT DISTINCT Pedido.pedDataCriacao, [@pedidoProduto].produtoNo
    FROM @pedidoProduto INNER JOIN Pedido ON [@pedidoProduto].pedidoNo = Pedido.pedidoNo

    INSERT INTO @datas
    SELECT [@datasTemp].dia
    FROM @datasTemp
    GROUP BY [@datasTemp].dia
    HAVING COUNT([@datasTemp].dia) = 2

    RETURN
END
GO
```

Figura 78 – Função onWichDayWereSoldTogether

```
select *  
from dbo.onWichDaysWereSoldTogether(9, 10)
```

Figura 79 - Select da Função onWichDayWereSoldTogether

	dia
1	2022-06-09
2	2022-06-10

Figura 80 - Output da Função onWichDayWereSoldTogether

### 2.3.3. Desenho de User Views

Informação	Tipo de acesso	Empregado de mesa	Supervisor	Cozinheiro
Funcionario	Gerir		X	
	Visualizar		X	
Pedido	Gerir		X	
	Visualizar		X	
Pedido Aberto	Gerir	X	X	
	Visualizar	X	X	X
Artigo	Gerir		X	
	Visualizar	X	X	X
Confeccionado	Gerir		X	
	Visualizar	X	X	X
Ingrediente	Gerir		X	
	Visualizar	X	X	X
Ementa	Gerir		X	X
	Visualizar	X	X	X
Localizacao	Gerir		X	
	Visualizar		X	
EstadoCivil	Gerir		X	
	Visualizar		X	
Cargo	Gerir		X	
	Visualizar		X	
GrupoAlimentar	Gerir		X	
	Visualizar	X	X	X
TipoProduto	Gerir		X	
	Visualizar	X	X	X
TipoTaxa	Gerir		X	
	Visualizar		X	
MetodoPagamento	Gerir		X	
	Visualizar	X	X	

Tabela 39 - User Views principais da Base de dados Michelin Star

## Artigos:

```
USE [MichelinStar]
GO

***** Object: View [dbo].[Artigos]    Script Date: 10/06/2022 03:05:54 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE VIEW [dbo].[Artigos]
AS
SELECT TOP (100) PERCENT dbo.Artigo.productoNo, dbo.Artigo.prodNome,
       { fn CONCAT(CONVERT(varchar(30), dbo.Artigo.prodPreco), '€') } AS prodPreco, dbo.Artigo.prodDescricao,
       { fn CONCAT(CONVERT(varchar(30), dbo.Artigo.prodCalorias), 'Kcal') } AS prodCalorias,
       dbo.Artigo.prodComGluten, dbo.TipoProduto.prodTipolome AS prodTipo,
       { fn CONCAT(CONVERT(varchar(30), dbo.TipoTaxa.prodTaxaValor), '%') } AS taxa
FROM   dbo.Artigo INNER JOIN
       dbo.TipoProduto ON dbo.Artigo.prodTipo = dbo.TipoProduto.prodTipo INNER JOIN
       dbo.TipoTaxa ON dbo.Artigo.prodTipoTaxa = dbo.TipoTaxa.prodTipoTaxa
WHERE  (dbo.Artigo.ativo = 1)
GO
```

Figura 81 - User View Artigos

	produtoNo	prodNome	prodPreco	prodDescricao	prodCalorias	prodComGluten	prodTipo	taxa
1	9	Coca-Cola	1.00€	NULL	150Kcal	1	Bebida	23.00%
2	10	Vinho da Casa	2.00€	NULL	200Kcal	0	Bebida	23.00%
3	11	Gelado Morango Olá	3.50€	NULL	170Kcal	1	Sobremesa	23.00%
4	13	Pão	0.30€	NULL	230Kcal	1	Entrada	6.00%
5	15	Água Luso	1.00€	NULL	0Kcal	0	Bebida	6.00%
6	16	Rissol	0.50€	NULL	90Kcal	1	Entrada	13.00%
7	17	Moet & Chandon	131.00€	NULL	250Kcal	1	Bebida	23.00%
8	18	Camarao	1.00€	NULL	99Kcal	0	Entrada	23.00%
9	19	Polvo	4.50€	NULL	150Kcal	0	Entrada	13.00%
10	20	Pão de alho	3.50€	NULL	125Kcal	1	Entrada	13.00%
11	21	Vinho Tinto E.A.	5.00€	NULL	210Kcal	1	Bebida	23.00%
12	22	Vinho Branco E.A.	7.00€	NULL	205Kcal	1	Bebida	23.00%
13	23	Gelado Chocolate Olá	3.50€	NULL	180Kcal	1	Sobremesa	23.00%
14	24	Gelado Baunilha Olá	3.50€	NULL	180Kcal	1	Sobremesa	23.00%
15	25	Bolo de Bolacha	4.00€	NULL	280Kcal	1	Sobremesa	23.00%
16	26	Baba de Camelô	6.00€	NULL	260Kcal	1	Sobremesa	23.00%
17	27	CheeseCake de Frutos Vermelhos	5.40€	NULL	230Kcal	1	Sobremesa	23.00%
18	28	Folhado de Leitão	1.50€	NULL	100Kcal	1	Entrada	13.00%
19	29	Cerveja SuperBock	1.50€	NULL	150Kcal	1	Bebida	23.00%

Figura 82 - Output User View Artigos

## Funcionarios:

```

USE [MichelinStar]
GO

***** Object: View [dbo].[Funcionarios]    Script Date: 10/06/2022 03:39:33 *****
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE VIEW [dbo].[Funcionarios]
AS
SELECT TOP (100) PERCENT dbo.Funcionario.fNif, dbo.Funcionario.fPrimeiroNome, dbo.Funcionario.fUltimoNome,
dbo.Funcionario.fDob, dbo.Funcionario.fRua, dbo.Funcionario.fNumPorta,
dbo.Funcionario.fNumTeleovel, dbo.Funcionario.fEmail,
dbo.Funcionario.fGenero, dbo.Funcionario.supervisor, dbo.Cargo.fCargoNome AS Cargo,
{ fn CONCAT(CONVERT(VARCHAR(30), dbo.Cargo.fSalarioHora), '€') } AS fSalarioHora,
dbo.EstadoCivil.fEstadoCivilNome AS [Estado Civil], dbo.Localizacao.fCodPostal AS CodPostal,
dbo.Localizacao.fCidade, dbo.Localizacao.fFreguesia
FROM    dbo.Funcionario INNER JOIN
        dbo.Cargo ON dbo.Funcionario.fCargo = dbo.Cargo.fCargo INNER JOIN
        dbo.EstadoCivil ON dbo.Funcionario.fEstadoCivil = dbo.EstadoCivil.fEstadoCivil INNER JOIN
        dbo.Localizacao ON dbo.Funcionario.fCodPostal = dbo.Localizacao.fCodPostal
WHERE   (dbo.Funcionario.ativo = 1)
GO

```

Figura 83 - User View Funcionarios

	INif	fPrimeiroNome	fUltimoNome	fDob	fRua	fNumPorta	fNumTeleovel	fEmail	fGenero	supervisor	Cargo	fSalarioHora	Estado Civil	CodPostal	fCidade	fFreguesia
1	256545251	Rui	Neto	2002-02-20	Rua das Pombinhas	404	918589899	rneto@gmail.com	M	NULL	Empregado de Limpeza	4.50€	Solteiro	1000-820	Lisboa	Avenidas Novas
2	256784000	Miguel	Gonçalves	2000-10-27	Rua das Andorinhas	127	938367830	mgoncalves@gmail.com	M	256545251	Empregado de Mesa	5.50€	Solteiro	4850-070	Eira Vedra	Minho
3	256789123	Carolina	Bessa	2003-12-17	Rua de Bairro	4456	939064567	cbessa@gmail.com	F	256545251	Cozinheiro	7.50€	Solteiro	4850-010	Eira Vedra	Ventosa
4	256890409	Luis	Miranda	1990-11-22	Rua das Pedras Caídas	14	918450340	lmiranda@gmail.com	M	256545251	Chefe de Cozinha	10.00€	Casado	2780-995	Vila Real de Santo António	Monte Gordo
5	256901123	Leonor	Araújo	1988-05-31	Rua São João	4045	919955678	laraujo@gmail.com	F	257434567	Empregado de Mesa	5.50€	Solteiro	4795-679	Santo Tirso	S. Tomé de Negrelos
6	256963867	Maria	Oliveira	2000-02-13	Rua da Curvinha	202	917649764	moliveira@gmail.com	F	257434567	Empregado de Mesa	5.50€	Viúvo	4200-047	Porto	Paranhos
7	256986754	José	Freitas	1980-07-04	Rua da Porta Nova	400	910654673	jfreitas@gmail.com	M	257434567	Empregado de Mesa	5.50€	Casado	4300-144	Porto	Campanhã
8	257434567	Simão	Santos	2001-12-03	Rua Comércio Indústria	125	917533668	ssantos@gmail.com	M	NULL	Cozinheiro	7.50€	Solteiro	1000-332	Lisboa	Areeiro
9	257893005	André	Ribeiro	1994-12-12	Rua da Portadela de Cima	2320	910065783	arieiro@gmail.com	M	257434567	Cozinheiro	7.50€	Casado	4490-003	Póvoa de Varzim	Aver-o-Mar
10	258450898	Sofia	Monteiro	1999-10-31	Rua das Luvírias	2222	910608608	smonteiro@gmail.com	M	256545251	Cozinheiro	7.50€	Divorciado	2400-030	Leiria	Amor

Figura 84 - Output User View Funcionarios

## Pedidos Fechados:

```

USE [MichelinStar]
GO

***** Object: View [dbo].[PedidosFechados]    Script Date: 10/06/2022 03:41:24 *****
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE VIEW [dbo].[PedidosFechados]
AS
SELECT TOP (100) PERCENT dbo.Pedido.pedidoNo, dbo.Pedido.pedNumeroMesa, dbo.Pedido.pedObs,
{ fn CONCAT(CONVERT(VARCHAR(60), dbo.Pedido.pedPreco), '€') } AS pedPreco,
dbo.Pedido.pedDataCriacao, dbo.Pedido.pedEstado,
dbo.Pedido.pedNumeroContribuinte, dbo.MetodoPagamento.pedMetodoPagamentoNome,
dbo.Pedido.fNif
FROM    dbo.Pedido INNER JOIN
        dbo.MetodoPagamento ON dbo.Pedido.pedMetodoPagamento = dbo.MetodoPagamento.pedMetodoPagamento
WHERE   (dbo.Pedido.ativo = 1) AND (dbo.Pedido.pedEstado = 'F')
ORDER BY dbo.Pedido.fNif, dbo.Pedido.pedDataCriacao
GO

```

Figura 85 - User View Pedidos Fechados

	pedidoNo	pedNumeroMesa	pedObs	pedPreco	pedDataCriacao	pedEstado	pedNumeroContribuinte	ped Metodo Pagamento Nome	fNif
1	12	1	NULL	60.58€	2022-06-09	F	256123456	Bitcoin	256545251
2	13	2	NULL	361.68€	2022-06-09	F	NULL	Bitcoin	256545251
3	14	6	NULL	217.86€	2022-06-09	F	NULL	VISA	257434567
4	15	4	NULL	22.17€	2022-06-09	F	256987456	Mastercard	257434567
5	16	7	NULL	78.78€	2022-06-09	F	NULL	Dinheiro	257434567

Figura 86 - Output User View Pedidos Fechados

### Faturado Mensalmente:

```
USE [MichelinStar]
GO

/******** Object: View [dbo].[FaturadoMensalmente]    Script Date: 10/06/2022 03:44:00 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE VIEW [dbo].[FaturadoMensalmente]
AS
SELECT TOP (100) PERCENT YEAR(pedDataCriacao) AS ano,
       FORMAT(pedDataCriacao, 'MMMM', 'pt-PT') AS mes,
       { fn CONCAT(CONVERT(varchar(60), SUM(pedPreco)), '€') } AS faturado,
       COUNT(*) AS nPedidos
FROM      dbo.Pedido
WHERE     (ativo = 1) AND (pedEstado = 'F')
GROUP BY FORMAT(pedDataCriacao, 'MMMM', 'pt-PT'), YEAR(pedDataCriacao)
ORDER BY ano, mes
GO
```

Figura 87 - User View Faturado Mensalmente

	ano	mes	faturado	nPedidos
1	2022	junho	741.07€	5

Figura 88 - Output User View Faturado Mensalmente

### 3. Conclusões e Trabalho Futuro

Com a realização deste trabalho conseguimos colocar em prática todos os conteúdos lecionados nas aulas ao longo do semestre. Graças a este fator, ficamos a conhecer os passos, regras, cuidados e boas práticas que devem ser abordadas e utilizadas na criação de uma base de dados.

Também foi importante para o aumento da percepção de que a boa implementação de uma base de dados é algo de enorme importância para uma empresa, visto que, no caso de a mesma possuir erros, poderá trazer várias consequências negativas para os negócios.

Acreditamos que, num trabalho futuro, como estamos mais familiarizados com as técnicas e a metodologia para a construção de uma base de dados, irá ser mais fácil o seu desenho e implementação.

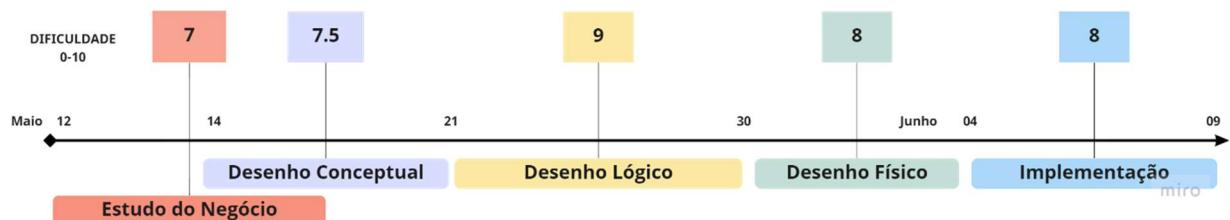


Figura 89 - Diagrama Cronológico e Dificuldades do Grupo

## **4. Bibliografia**

Livro “Database Systems - A Practical Approach to Design, Implementation and Management”

## 5. Referências WWW

[01] <https://moodle2.estg.ipp.pt>

Moodle da Escola Superior de Tecnologia e Gestão. Contem *powerpoints* e fichas de apoio aos conteúdos lecionados nas aulas.

[02] <https://docs.microsoft.com/en-us/sql/t-sql/language-reference?view=sql-server-ver16>

[03] <https://stackoverflow.com>

## **6. Lista de Siglas e Acrónimos**

**DBDL** – Database Definition Language

**DBMS** – Database Management Systems