ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

Disciplina de Base de Dados

Ano Letivo de 2022/2023

Relatório do trabalho prático de base de dados do grupo 206

Jorge Correia - 8200592

Nuno Castro - 8200591

Junho, 2023

Data de Receção	
Responsável	
Avaliação	
Observações	

Relatório do trabalho prático de base de dados do grupo 206

Jorge Correia - 8200592

Nuno Castro - 8200591

Junho, 2023

Agradecimentos

Gostaríamos de agradecer ao professor Vasco Santos pelo esclarecimento das dúvidas que foram surgindo ao longo do desenvolvimento do projeto e pela sua disponibilidade para nos ajudar.

Resumo

Este projeto tem como objetivo desenvolver de uma Base de Dados (BD) para uma loja de roupa, visando a informatização de alguns serviços para facilitar suas operações. O trabalho abrange diversos conceitos estudados na disciplina de Base de Dados, incluindo a modelagem de uma BD, o processo de normalização, criação de estruturas de BD, restrições de integridade, entre outros. Também serão utilizadas Queries, Views e Stored Procedures para finalizar o projeto.
A BD criada neste trabalho prático possui Views e Stored Procedures que respondem aos pedidos existentes no enunciado e utilizando os conhecimentos obtidos ao longo do semestre.

Índice

Índice de Figuras

Índice de Tabelas

1. Introdução

1.1 Contextualização

Este trabalho foi desenvolvido no âmbito da disciplina de Base de Dados e tem como objetivo utilizar os conhecimentos obtidos durante o semestre para resolver o problema da loja de roupa descrito no enunciado do trabalho prático.

1.2 Apresentação do Caso de Estudo

O gerente de uma loja de roupa, insatisfeito com a aplicação informática existente que gere a loja, necessita que seja desenhada uma nova base de dados que permita uma melhor e mais eficiente gestão da loja e armazém.

1.3 Motivação e Objetivos

O problema existente consiste na criação de uma Base de Dados (BD) que dê suporte aos processos descritos pela loja de roupa. Estes processos são:

- A loja está fisicamente organizada por secções (Criança, Homem e Mulher) para facilitar aos clientes a procura das peças que pretendem. E em cada secção existem 10 escaparates onde estão dispostas as peças.
- A loja tem um pequeno armazém onde estão guardadas peças para reposição de stock da loja.
- A loja funciona das 10h até às 22h, com funcionários que fazem turnos de 6h por dia. Estes funcionários têm como principais tarefas atender os clientes e registar as vendas na aplicação.
- Deverá ser possível pesquisar em que escaparate está determinada peça.
- Se um funcionário detetar um defeito numa peça, esta deverá ser catalogada e armazenada no armazém como peça defeituosa.

Para além destes processos a gerência do restaurante pretende conseguir obter rapidamente as seguintes informações:

- Qual o total faturado no dia anterior, semana anterior, mês anterior.
- Lista das peças vendidas no dia anterior.
- Lista dos funcionários por ordem decrescente do valor de vendas do mês anterior.
- Qual a peça mais vendida.

- Qual o valor de peças defeituosas identificadas numa determinada semana.
- Listar as faturas de um cliente.

Também pertendemos obter a aprovação à cadeira.

1.4 Estrutura do Relatório

Este relatório esta dividir em três partes que são a Introdução, Desenvolvimento e Conclusão. Na Introdução é introduzido o problema, o que pretendemos realizar e o que pretendemos obter com este projecto. No desenvolvimento começámos por indicar o processo que usamos para realizar a base de dados, como as decisões tomadas e parte da implementação realizada. Na conclusão reflectimos sobre o que fizemos e indicamos trabalho futuro.

2. Desenvolvimento

Para que conseguíssemos desenvolver uma estrutura mais robusta para o projeto, começamos por interpretar o enunciado e discutir ideias importantes para as decisões que teríamos de tomar futuramente.

Nesta secção serão explicados os passos e os procedimentos seguidos de modo a desenhar a estrutura da BD do projeto de forma mais eficaz.

De modo a desenhar a BD de uma maneira mais correta decidimos utilizar a metodologia de desenho de BD que consiste numa forma estruturada, com determinados passos, técnicas, ferramentas e documentação que dão suporte ao processo de desenho de BD.

A metodologia de desenho de BD utilizada possui 3 fases principais:

- Desenho Conceptual;
- Desenho Lógico;
- Desenho Físico.

2.1 Desenho Conceptual

O desenho conceptual não depende de considerações físicas e consiste no processo de criação de um modelo de informação.

Inicialmente foram identificados os tipos de entidades pertencentes a uma loja de roupa. Cada entidade contém, para além da designação, a respetiva descrição, acrónimo e ocorrência.

Nome da entidade	Descrição	Acrónimo	Ocorrência
Stock	Descrição do stock das peças	STO	Cada stock contém zero ou mais defeitos
Defeito	Descrição dos defeitos	DEF	Cada defeito pertence a um stock e a um funcionário
Peça	Descrição de todos as peças de roupa	ITM	Cada peça pertence apenas a uma secção e a zero ou mais escaparates.
Seccao	Descrição de todos as secções	SEC	Cada secção contém zero ou mais peças e zero ou mais escaparates.
Escaparate	Descrição de todos os escaparates	ESC	Cada escaparate pertence a apenas uma secção e contém zero ou mais peças
Funcionario	Descrição de todos os funcionários	FUN	Cada funcionário pode tratar de zero ou mais faturas e pode registar zero ou mais defeitos
Fatura	Descrição geral de todas as faturas	FAT	Cada fatura contém uma ou mais peças, é tratada por apenas um funcionário e pertencer a um cliente.
Cliente	Descrição de todos os clientes	CLI	Cada cliente contém zero ou mais faturas

Tabela 1- Identificação do tipo de entidades.

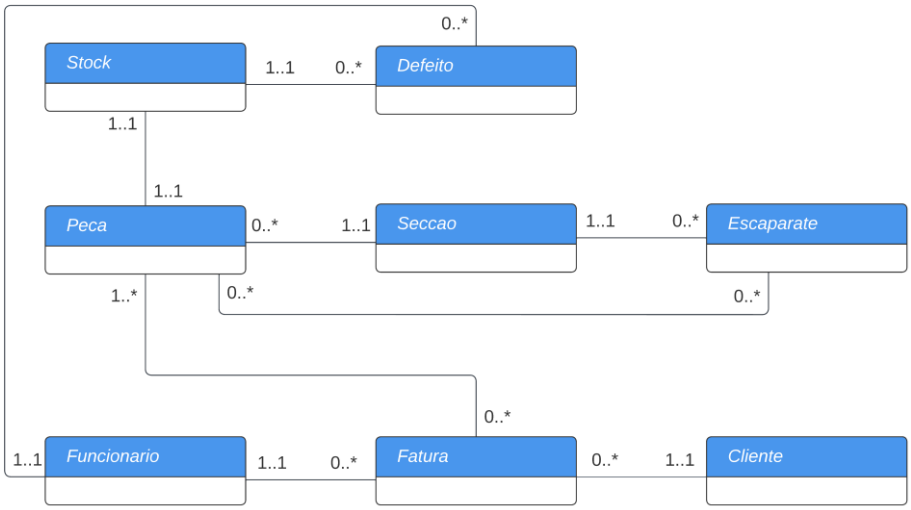
Com as entidades identificadas, é necessário verificar o que é que as relacionam e que tipo de relacionamento é que possuem. Para representar os relacionamentos foi criada uma tabela onde, para cada relacionamento, é descrito o tipo de relação, as entidades envolvidas e a multiplicidade para cada uma delas.

Nome da Entidade	Multiplicidade	Relação	Multiplicidade	Nome da Entidade
Peca	0..*	Pertence	1..1	Seccao
Peca	0..*	Pertence	0..*	Escaparate
Stock	1..1	Contém	1..1	Peca
Stock	1..1	Contém	0..*	Defeito
Escaparate	0..*	Pertence	1..1	Seccao
Funcionario	1..1	Regista	0..*	Fatura
Funcionario	1..1	Regista	0..*	Defeito
Fatura	0..*	Contém	1..*	Peca
Cliente	1..1	Contém	0..*	Fatura

Tabela 2 - Identificação do tipo de relacionamentos.

Com as entidades e os relacionamentos identificados é possível criar o primeiro diagrama ER deste projeto.

Figura 1 - Primeiro diagrama ER



O passo seguinte consiste em descrever as entidades com a identificação dos seus atributos e domínios, para isso identificamos o a entidade, os seus atributos. Cada atributo contém uma pequena descrição, o tipo de dados e o seu tamanho, um campo onde é verificado se o atributo pode ser nulo ou não e se o atributo é multi-valor ou não. O mesmo é feito para os relacionamentos.

Nome da Entidade	Atributos	Descrição	Tipo de dados e tamanho	Nulos	Multi - valor
Peca	codPeca	Único e identifica uma peça	5 caracteres variáveis	Não	Não
	nomePeca	Nome da peça	50 caracteres variáveis	Não	Não
	codSeccao	Único e identifica uma secção	5 caracteres variáveis	Não	Não
	precoUnitario	Preço Unitario da peça	4 dígitos sendo 2 casas decimais	Não	Não
	cores	Cores que as variantes da peça tem	20 caracteres variáveis (Azul, Vermelho, etc)	Não	Sim
	categoria	Categoria a que uma peça pertence	50 caracteres variáveis	Não	Não
	tamanhos	Tamanhos que a peça tem	50 caracteres variáveis	Não	SIm
	obs	Observações adicionais da peca	200 caracteres variáveis	Sim	Não
	cuidados	Cuidados que se tem de ter com a peça	50 caracteres variáveis	Sim	Sim
Stock	codPeca	Único e identifica uma peça	5 caracteres variáveis	Não	Não
	stockArmazem	Quantidade de uma determinada peça em armazem	4 dígitos inteiros	Não	Não
	stockLoja	Quantidade de uma determinada peça em loja	4 dígitos inteiros	Não	Não
	quantDefeitos	Total de peças com defeito	4 dígitos inteiros	Não	Não

Defeito	codDefeito	Único e identifica um defeito	5 caracteres variáveis	Não	Não
	codPeca	Único e identifica uma peça	5 caracteres variáveis	Não	Não
	codFuncionario	NIF do funcionario	9 dígitos inteiros	Não	Não
	quant	Quantidade de peças com defeito	4 dígitos inteiros	Não	Não
	tipoDefeito	Tipo de defeito encontrado na peça	20 carateres variáveis (Manhã, Noite)	Não	Não
	descricao	Descrição do defeito	200 caracteres variáveis	Não	Não
	datahora	Data do registo do defeito	Data	Não	Não
Seccao	codSeccao	Único e identifica uma secção	5 caracteres variáveis	Não	Não
	nomeSeccao	Nome da secção	20 carateres variáveis	Não	Não
	descricao	Descrição da secção	200 caracteres variáveis	Sim	Não
Escaparate	codEscaparate	Único e identifica um escaparate	5 caracteres variáveis	Não	Não
	nomeEscaparate	Nome do escaparate	20 carateres variáveis	Não	Não
	codSeccao	Único e identifica uma secção	5 caracteres variáveis	Não	Não
	descricao	Descrição do escaparate	200 caracteres variáveis	Sim	Não
Funcionario	codFuncionario	NIF do funcionario	9 dígitos inteiros	Não	Não
	nomeFuncionario	Nome do funcionário	50 carateres variáveis	Não	Não

	numeroTelemovel	Número do telemóvel do funcionário	9 dígitos inteiros a começar por 91, 92, 93, 96	Não	Não
	email	Email do funcionário	256 caracteres variáveis e tem de possuir @	Não	Não
	turno	Turno que o funcionário trabalha	20 carateres variáveis (Manhã, Noite)	Não	Não
	dataNascimento	Data de nascimento do funcionario	Data	Não	Não
	dataEntrada	Data de entrada do funcionario na empresa	Data	Não	Não
	dataSaida	Data de saida do funcionario da empresa	Data	Sim	Não
	salario	Salário do funcionario	7 dígitos sendo 2 casas decimais	Não	Não
Fatura	codFatura	Único e identifica uma fatura	5 caracteres variáveis	Não	Não
	codFuncionario	Único e identifica um funcionário	5 caracteres variáveis	Não	Não
	dataHora	Data da criação da fatura	Data	Não	Não
	nif	Número de Identificação Fiscal do cliente	9 dígitos inteiros	Sim	Não
	codCliente	Único e identifica uma fatura	5 caracteres variáveis	Sim	Não
	pontosGastos	Número de pontos do cliente gasto na compra	int	Sim	Não

	subTotal	preço total da compra sem descontos	7 dígitos sendo 2 casas decimais	Não	Não
	desconto	Valor de desconto em euros	7 dígitos sendo 2 casas decimais	Sim	Não
	total	Preço total da compra	7 dígitos sendo 2 casas decimais	Não	Não
	numerario	Dinheiro recebido para efetuar o pagamento	7 dígitos sendo 2 casas decimais	Não	Não
	troco	Troco do pagamento	7 dígitos sendo 2 casas decimais	Não	Não
	tipoPagamento	Tipo de pagamento efetuado	20 carateres variáveis (Dinheiro, Multibanco, MBWAY)	Não	Não
Cliente	codCliente	Único e identifica uma fatura	5 caracteres variáveis	Não	Não
	nome	Nome do cliente	50 carateres variáveis	Não	Não
	numeroTelemovel	Número do telemóvel do cliente	9 dígitos inteiros a começar por 91, 92, 93, 96	Não	Não
	nif	Número de Identificação Fiscal do cliente	9 dígitos inteiros	Sim	Não
	pontos	Número de pontos do cliente	int	Não	Não

Tabela 3 - Identificação de entidades, atributos e seus domínios.

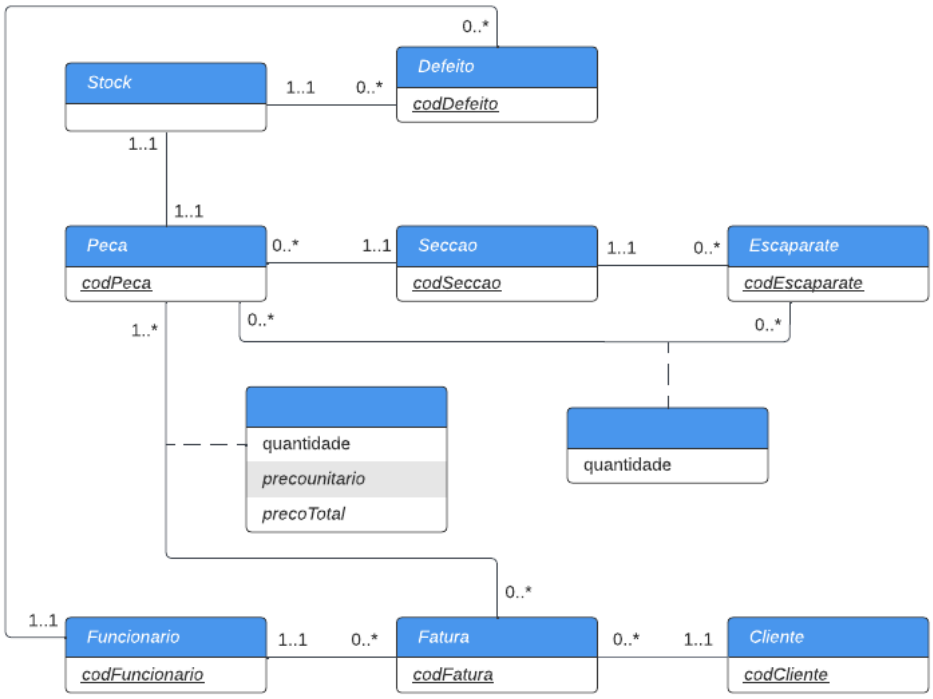
Nome do relacionamento	Atributos	Descrição	Tipo de dados e tamanho	Nulos	Multi - valor
Peca/Escaparate	quantidade	Quantidade de peças por escaparate	2 dígitos inteiros variáveis	Não	Não

Carrinho	quantidade	Quantidade de peças	4 dígitos inteiros	Não	Não
	precoUnitario	Preço Unitario da peça	4 dígitos sendo 2 casas decimais	Não	Não
	precoTotal	Preço total das peças	7 dígitos sendo 2 casas decimais	Não	Não

Tabela 4 - Identificação de relacionamentos, atributos e seus domínios.

Após completar este passo criamos um diagrama ER com as chaves primárias e os atributos dos relacionamentos.

Figura 2 - Diagrama ER com chaves primárias



Para finalizar o desenho conceptual necessitamos de verificar se as entidades e relações criadas são suficientes para efetuar as transações necessárias. As transações são as seguintes:

Transação (a): Qual o total faturado no dia anterior, semana anterior, mês anterior.
Os detalhes dos valores faturados e os detalhes da data da faturação estão contidos na entidade *Fatura*, podemos utilizar a entidade *Fatura* para produzir a lista necessária.

Transação (b): Lista das peças vendidas no dia anterior.
Os detalhes das quantidades de peças vendidas estão na entidade *Fatura*, os detalhes das peças vendidas estão na entidade. *Peca*, podemos utilizar a relação *Fatura* contém *Peca* para produzir a lista necessária.

Transação (c): Lista dos funcionários por ordem decrescente do valor de vendas do mês anterior.
Os detalhes dos funcionários estão na entidade *Funcionário*, os detalhes das vendas estão na entidade *Fatura*, podemos utilizar a relação *Funcionário* regista *Fatura* para produzir a lista necessária.

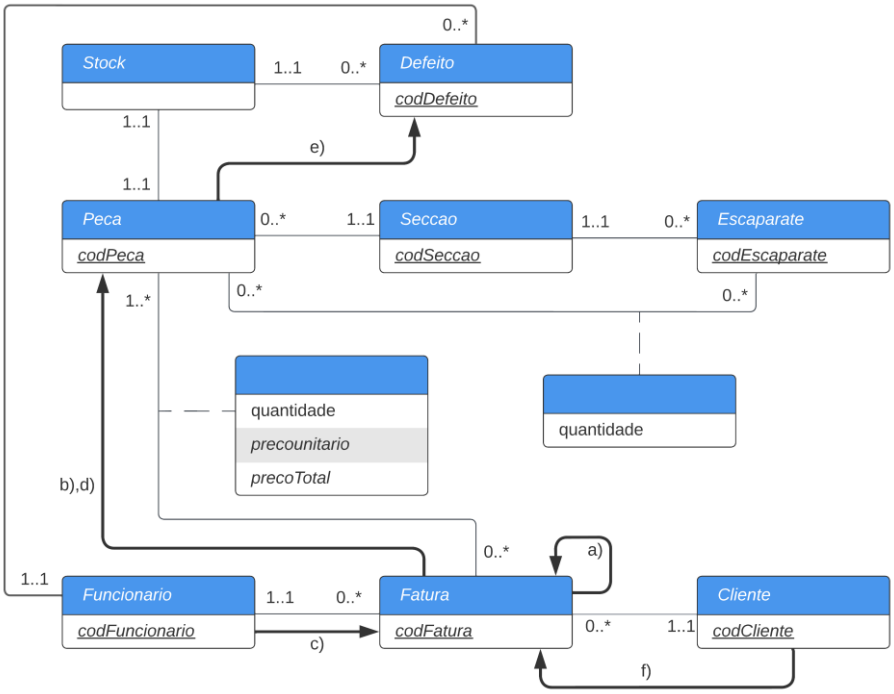
Transação (d): Qual a peça mais vendida.
Os detalhes das quantidades de peças vendidas estão na entidade *Fatura*, os detalhes das peças vendidas estão na entidade *Peca*, podemos utilizar a relação *Fatura* contém *Peca* para produzir a lista necessária.

Transação (e): Qual o valor de peças defeituosas identificadas numa determinada semana.
Os detalhes das peças estão na entidade *Peca*, os detalhes dos defeitos estão na entidade *Defeito*, podemos utilizar a relação *Peca* contém *Defeito* para produzir a lista necessária.

Transação (f): Listar as faturas do cliente.
Os detalhes das clientes estão na entidade *Cliente*, os detalhes das faturas estão na entidade *Fatura*, podemos utilizar a relação *Cliente contém Faturas* para produzir a lista necessária.

Estas transações foram representadas no diagrama ER seguinte.

Figura 3 - Diagrama ER com transações



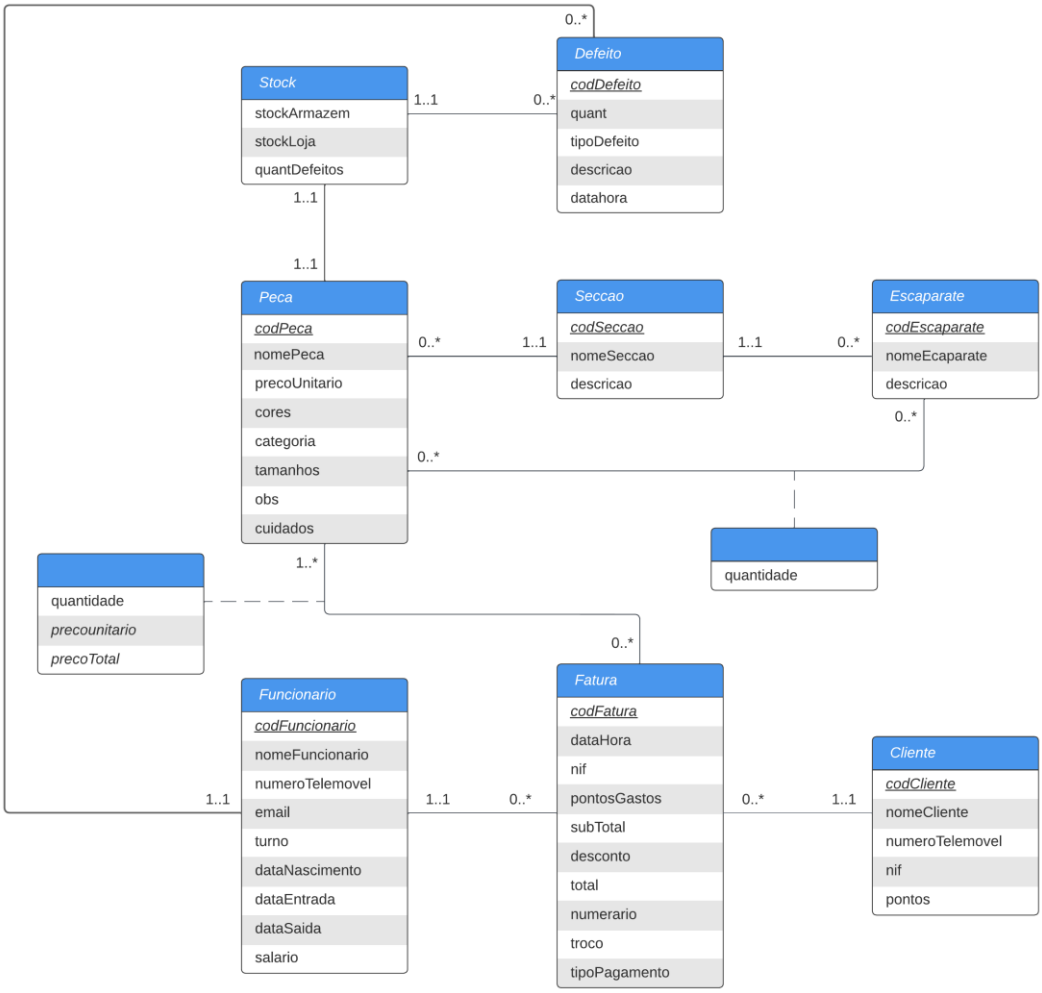
2.2 Desenho Lógico

O desenho lógico não depende de considerações físicas nem de um SGBD e consiste no processo de criação de um modelo de informação baseado num determinado modelo de dados.

O objetivo principal é traduzir o desenho conceptual criado anteriormente para um desenho lógico e verificar se este está estruturado corretamente de modo a responder ao pedido pelo restaurante.

No início do desenho lógico encontramo-nos com o seguinte diagrama ER.

Figura 4 - Diagrama ER no início do desenho lógico



Baseado neste diagrama temos de obter as relações para o modelo lógico de modo a representarmos as entidades, relacionamentos e atributos que tinham sido identificados anteriormente. Começamos por identificar as entidades fortes e fracas.

Entidades fortes são entidades que não dependem de outras entidades, ou seja, entidades que podem existir independentemente se outras entidades existem.

Entidades fracas são entidades em que a sua existência depende de outras entidades.

Entidades fortes

Seccao(codSeccao, nomeSeccao, descricao)
Primary Key codSeccao

Funcionario (codFuncionario, nomeFuncionario, numeroTelemovel, email, turno, dataNascimento, dataEntrada, dataSaida, salario)
Primary Key codFuncionario

Cliente(codCliente, nomeCliente, numeroTelemovel, nif, pontos)
Primary Key codCliente

Entidades fracas

Peca (codPeca, nomePeca, codSeccao, precoUnitario, cores, categoria, tamanhos, obs, cuidados)
Primary Key codPeca

Stock (codPeca, stockArmazem, stockLoja, quantDefeitos)
Primary Key codPeca

Defeito (codDefeito, codPeca, quant, tipoDefeito, descricao, dataHora)
Primary Key codDefeito

Escaparate (codEscaparate, nomeEscaparate, codSeccao, descricao)
Primary Key codEscaparate

Fatura (codFatura, codFuncionario, dataHora, nif, codCliente, pontosGastos, subTotal, desconto, total, numerario, troco, tipoPagamento)
Primary Key codFatura

Tipos de relacionamento binário um-para-muitos (1:*)

Para este tipo de relacionamentos temos de adicionar a chave primária da tabela com relação 1 como chave estrangeira na outra tabela.

Figura 5 - Tipo de relacionamento um para muitos entre Fatura e Cliente

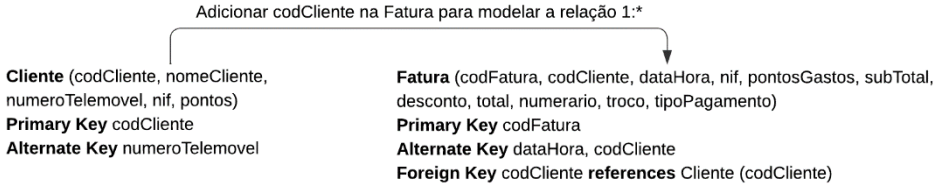


Figura 6 - Tipo de relacionamento um para muitos entre Funcionario e Fatura

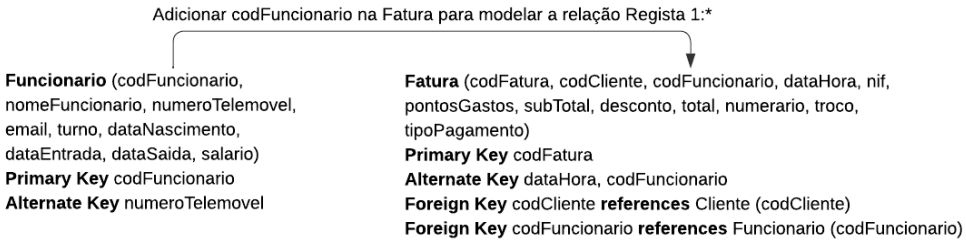


Figura 7 - Tipo de relacionamento um para muitos entre Stock e Defeito

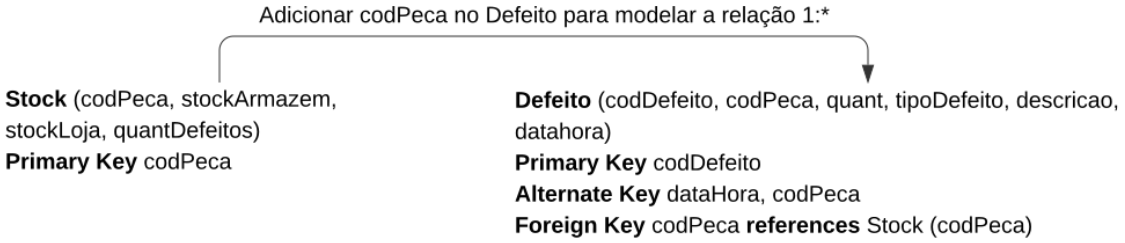


Figura 8 - Tipo de relacionamento um para muitos entre Seccao e Escaparate

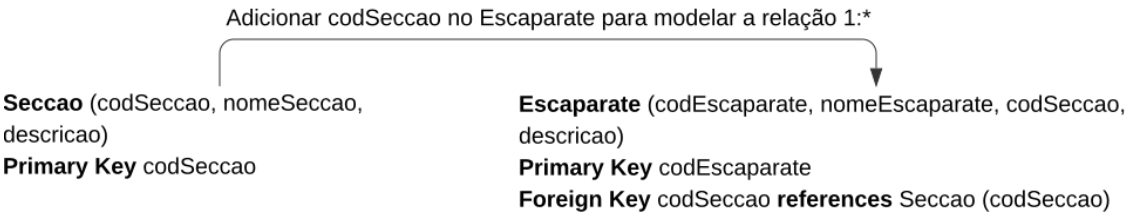


Figura 9 - Tipo de relacionamento um para muitos entre Secca e Peca

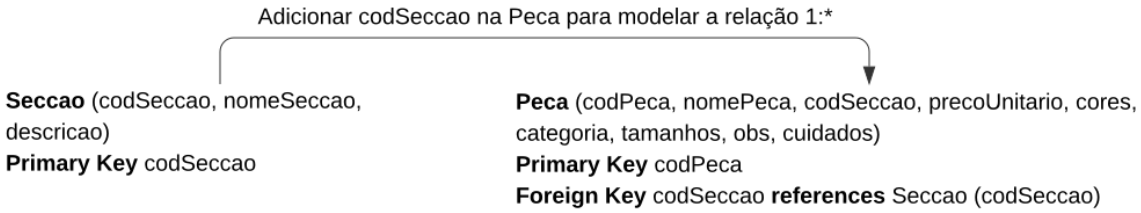
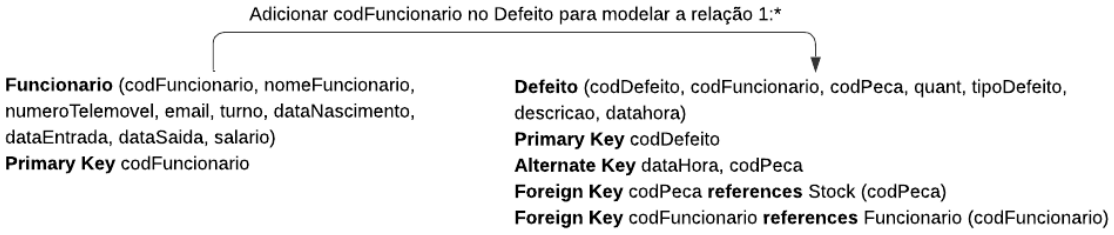


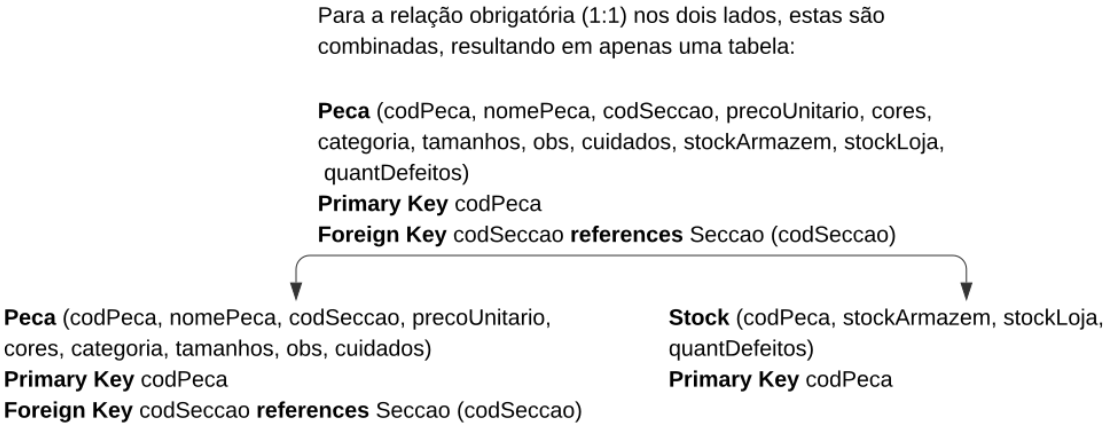
Figura 10 - Tipo de relacionamento um para muitos entre Funcionario e Defeito



Tipo de relacionamento binário um para um (1:1) com participação obrigatória nos dois lados

Para este tipo de relacionamentos temos de combinar as duas tabelas numa única tabela e escolher uma chave primária das entidades originais.

Figura 11 - Tipo de relacionamento um para um obrigatório nos dois lados

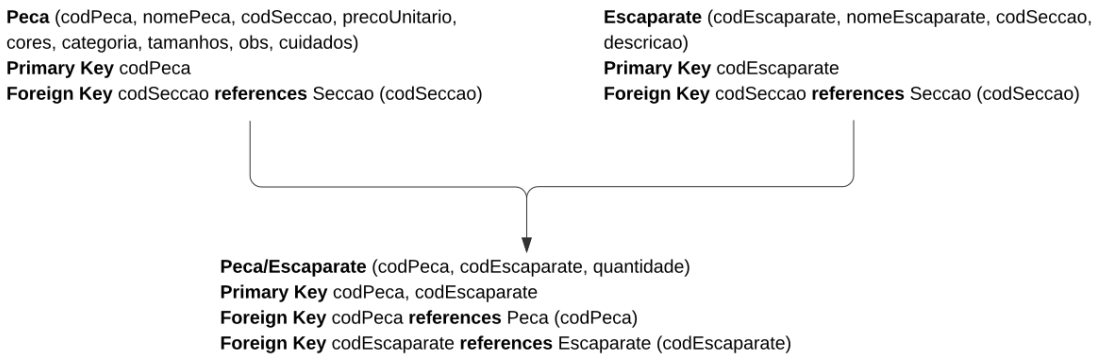


Tipos de relacionamento binário muitos-para-muitos (*:*)

Para os relacionamentos muitos para muitos criamos uma tabela que representa a relação entre as duas entidades. Esta tabela possui qualquer atributo que faça parte da relação e adicionamos as chaves primárias das duas entidades

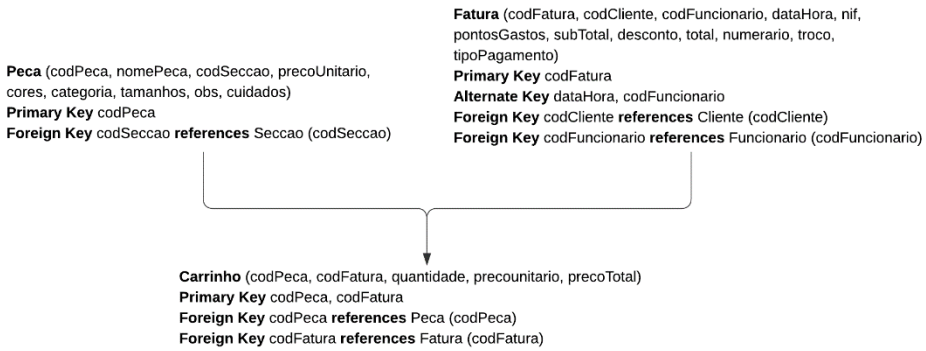
Neste caso é criada a tabela Peca/Escaparate que fica com os atributos da relação (quantidade) e com as chaves da entidade Peca (codPeca) e da entidade Escaparate (codEscaparate).

Figura 12 - Tipo de relacionamento muitos para muitos entre Peca e Escaparate



Neste caso é criada a tabela Carrinho que fica com os atributos da relação (quantidade, precoUnitario, precoTotal) e com as chaves da entidade Peca (codPeca) e da entidade Fatura (codFatura).

Figura 13 - Tipo de relacionamento muitos para muitos entre Peca e Fatura



Atributos multi-valor

Para este tipo de atributos é criada uma tabela para armazenar os atributos multi-valor, contendo uma primary key composta pelo primary key da entidade e o valor do atributo.

Figura 14 - Atributo multi-valor Cor na Tabela Peca

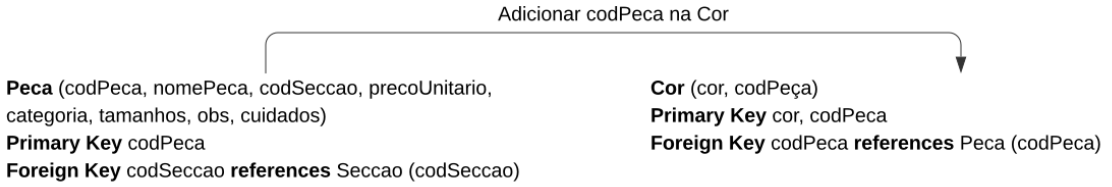


Figura 15 - Atributo multi-valor Tamanho na Tabela Peca

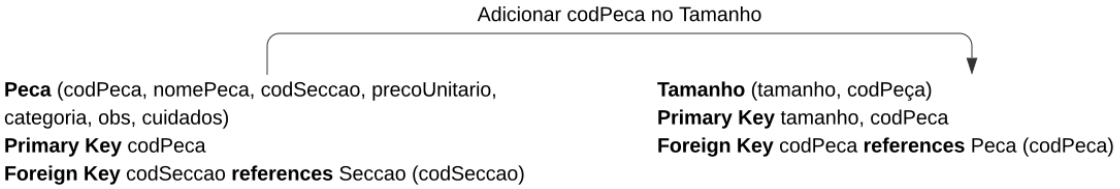
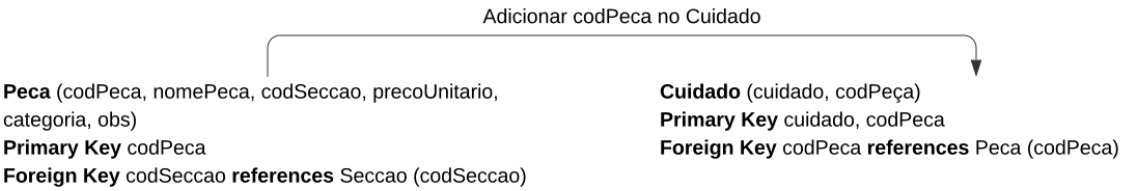
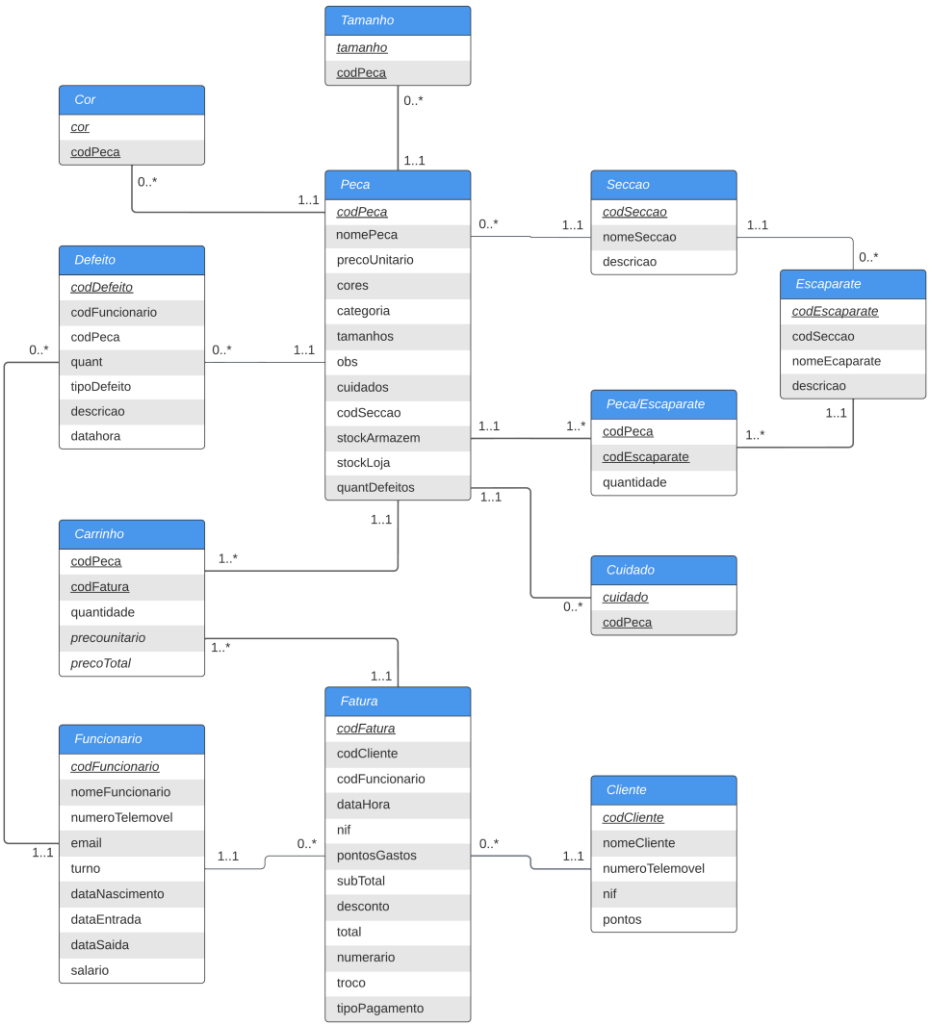


Figura 16 - Atributo multi-valor Cuidado na Tabela Peca



Finalizando estes passos obtemos o seguinte diagrama

Figura 17 - Diagrama ER resultante dos passos iniciais do modelo lógico



De modo a finalizar o desenho lógico é necessário validar esta estrutura através da normalização.

Normalização

A normalização é um processo que pretende validar as relações obtidas no diagrama ER anterior. Através da normalização agrupamos atributos em relações com o objetivo de minimizar a redundância de dados e o espaço de armazenamento necessário.

Ao realizar o processo de normalização o modelo passa por várias fases, mas apenas decidimos utilizar as 4 fases principais:

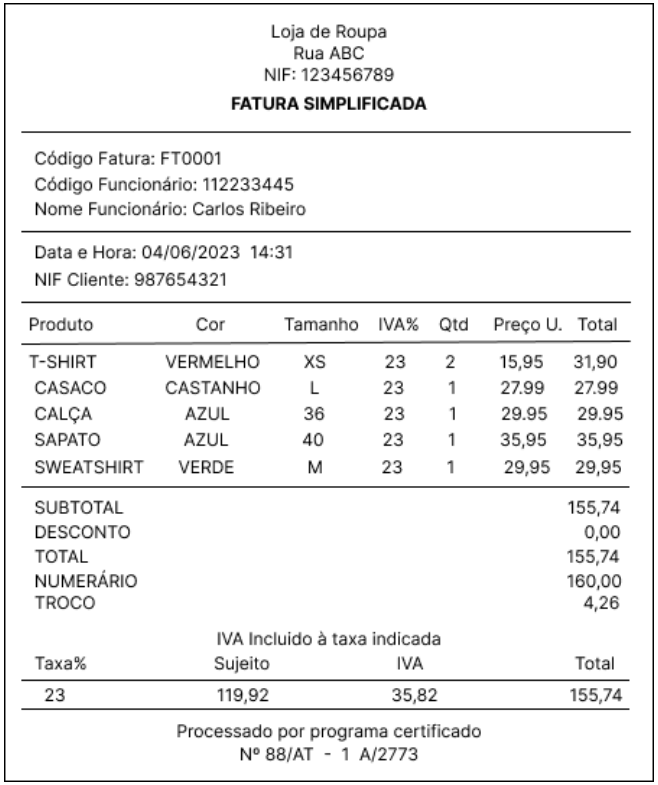
- Forma não normalizada (UNF);
- Primeira Forma Normal (1FN);
- Segunda Forma Normal (2FN);
- Terceira Forma Normal (3FN).

Para iniciar este processo criamos mockups para três documentos que a loja poderá ter. Estes documentos são: uma Fatura, um registo de Defeitos e um documento de reposição de stock.

A normalização é aplicada a cada documento. Após a criação dos mockups é necessário retirar a informação que será armazenada na BD e colocá-la numa tabela. Tendo isto em conta, o modelo irá estar na UNF.

Normalização da Fatura

Na Figura 18 podemos ver o mockup criado para uma fatura, este mockup identifica dados estáticos como por exemplo o nome e a rua da loja que não serão guardados na BD. Também conseguimos identificar os dados dinâmicos que serão guardados na BD como por exemplo o produto pedido pelo cliente. O IVA foi considerado que já se encontra no preço do produto.



Loja de Roupa
Rua ABC
NIF: 123456789

FATURA SIMPLIFICADA

Código Fatura: FT0001
Código Funcionário: 112233445
Nome Funcionário: Carlos Ribeiro

Data e Hora: 04/06/2023 14:31
NIF Cliente: 987654321

Produto	Cor	Tamanho	IVA%	Qtd	Preço U.	Total
T-SHIRT	VERMELHO	XS	23	2	15,95	31,90
CASACO	CASTANHO	L	23	1	27.99	27.99
CALÇA	AZUL	36	23	1	29.95	29.95
SAPATO	AZUL	40	23	1	35,95	35,95
SWEATSHIRT	VERDE	M	23	1	29,95	29,95

SUBTOTAL	155,74
DESCONTO	0,00
TOTAL	155,74
NUMERÁRIO	160,00
TROCO	4,26

IVA Incluido à taxa indicada

Taxa%	Sujeito	IVA	Total
23	119,92	35,82	155,74

Processado por programa certificado
Nº 88/AT - 1 A/2773

Figura 18 - Mockup fatura

Ao retirar os dados necessários do mockup da fatura surge a seguinte tabela.

codFatura	codFuncionario	nomeFuncionario	dataHora	nifCliente	codProduto	nomeProduto	codCor	cor	codTamanho	tamanho	quantidade	precoUnit	totalProduto	subTotal	desconto	total	numerario	troco
FT00001	112233445	Carlos Ribeiro	04/06/2023 14:31	987654321	P0001	T-SHIRT	C01	VERMELHO	T01	XS	2	15,95	31,9	155,74	0	155,74	160	4,26
					P0002	CASACO	C02	CASTANHO	T02	L	1	27,99	27,99					
					P0003	CALÇA	C03	AZUL	T03	36	1	29,95	29,95					
					P0004	SAPATO	C03	AZUL	T04	40	1	35,95	35,95					
					P0005	SWEATSHIRT	C04	VERDE	T05	M	1	29,95	29,95					

Tabela 5 - Tabela da Fatura na UNF

Para passar esta tabela para a 1FN é necessário que a intersecção entre uma linha e uma coluna contenha um e um só valor e contenha uma chave primária válida, para isso foram feitas as seguintes alterações.

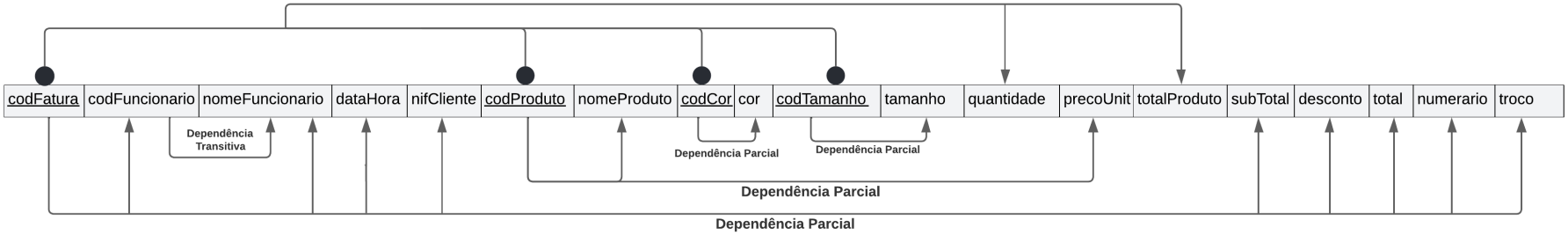
<u>codFatura</u>	codFuncionario	nomeFuncionario	dataHora	nifCliente	<u>codProduto</u>	nomeProduto	<u>codCor</u>	cor	<u>codTamanho</u>	tamanho	quantidade	precoUnit	totalProduto	subTotal	desconto	total	numerario	troco
FT00001	112233445	Carlos Ribeiro	04/06/2023 14:31	987654321	P0001	T-SHIRT	C01	VERMELHO	T01	XS	2	15,95	31,9	155,74	0	155,74	160	4,26
FT00001	112233445	Carlos Ribeiro	04/06/2023 14:31	987654321	P0002	CASACO	C02	CASTANHO	T02	L	1	27,99	27,99	155,74	0	155,74	160	4,26
FT00001	112233445	Carlos Ribeiro	04/06/2023 14:31	987654321	P0003	CALÇA	C03	AZUL	T03	36	1	29,95	29,95	155,74	0	155,74	160	4,26
FT00001	112233445	Carlos Ribeiro	04/06/2023 14:31	987654321	P0004	SAPATO	C03	AZUL	T04	40	1	35,95	35,95	155,74	0	155,74	160	4,26
FT00001	112233445	Carlos Ribeiro	04/06/2023 14:31	987654321	P0005	SWEATSHIRT	C04	VERDE	T05	M	1	29,95	29,95	155,74	0	155,74	160	4,26

Tabela 6 - Tabela da Fatura na 1FN

Para conseguirmos passar esta tabela para a 2FN é necessário verificar se existem dependências funcionais, ou seja, dependências entre atributos e parte da chave. Caso existam dependências parciais é necessário removê-las da tabela inicial e colocá-las numa nova tabela.

De modo a identificar as dependências existentes na tabela criamos o seguinte diagrama de dependências.

Figura 19 - Diagrama de dependências da Fatura



Analisando o diagrama conseguimos verificar que existem dependências parciais, logo, existe a necessidade de passar essas dependências para novas tabelas.

codFatura	codFuncionario	nomeFuncionario	dataHora	nifCliente	subTotal	desconto	total	numerario	troco
FT0001	112233445	Carlos Ribeiro	04/06/2023 14:31	987654321	155,74	0	155,74	160	4,26

Tabela 7 - Tabela Fatura na 2FN

codProduto	nomeProduto	precoUnit
P0001	T-SHIRT	15,95
P0002	CASACO	27,99
P0003	CALÇA	29,95
P0004	SAPATO	35,95

P0005	SWEATSHIRT	29,95

Tabela 8 - Tabela Produto na 2FN

codCor	cor
C01	VERMELHO
C02	CASTANHO
C03	AZUL
C03	AZUL
C04	VERDE

Tabela 9 - Tabela Cor na 2FN

codTamanho	tamanho
T01	XS
T02	L
T03	36
T04	40
T05	M

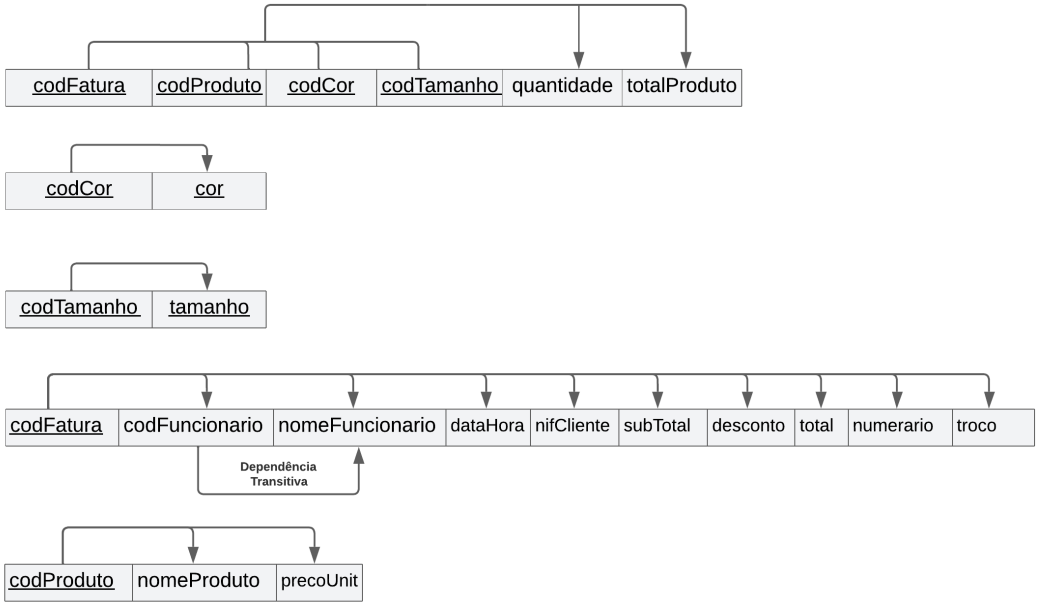
Tabela 10 - Tabela Tamanho na 2FN

codFatura	codProduto	codCor	codTamanho	quantidade	totalProduto
FT0001	P0001	C01	T01	2	31,9
FT0001	P0002	C02	T02	1	27,99
FT0001	P0003	C03	T03	1	29,95
FT0001	P0004	C03	T04	1	35,95
FT0001	P0005	C04	T05	1	29,95

Tabela 11 - Tabela Carinho na 2FN

O último passo é passar as tabelas da 2FN para a 3FN. Para que isto seja possível é necessário verificar se existem dependências transitivas, ou seja, dependências entre atributos não chave. Caso existam, teremos de removê-las, colocando-as numa nova tabela.

Figura 20 - Diagrama resultante da 2FN da Fatura



Como podemos observar, existem dependências transitivas, logo, existe a necessidade de remover essas dependências.

codFatura	codFuncionario	dataHora	nifCliente	subTotal	desconto	total	numerario	troco
FT0001	112233445	04/06/2023 14:31	987654321	155,74	0	155,74	160	4,26

Tabela 12 - Tabela Fatura na 3FN

codProduto	nomeProduto	precoUnit
P0001	T-SHIRT	15,95
P0002	CASACO	27,99
P0003	CALÇA	29,95
P0004	SAPATO	35,95
P0005	SWEATSHIRT	29,95

Tabela 13 - Tabela Produto na 3FN

codCor	cor
C01	VERMELHO
C02	CASTANHO
C03	AZUL
C03	AZUL
C04	VERDE

Tabela 14 - Tabela Cor na 3FN

codTamanho	tamanho
T01	XS
T02	L
T03	36
T04	40
T05	M

Tabela 15 - Tabela Tamanho na 3FN

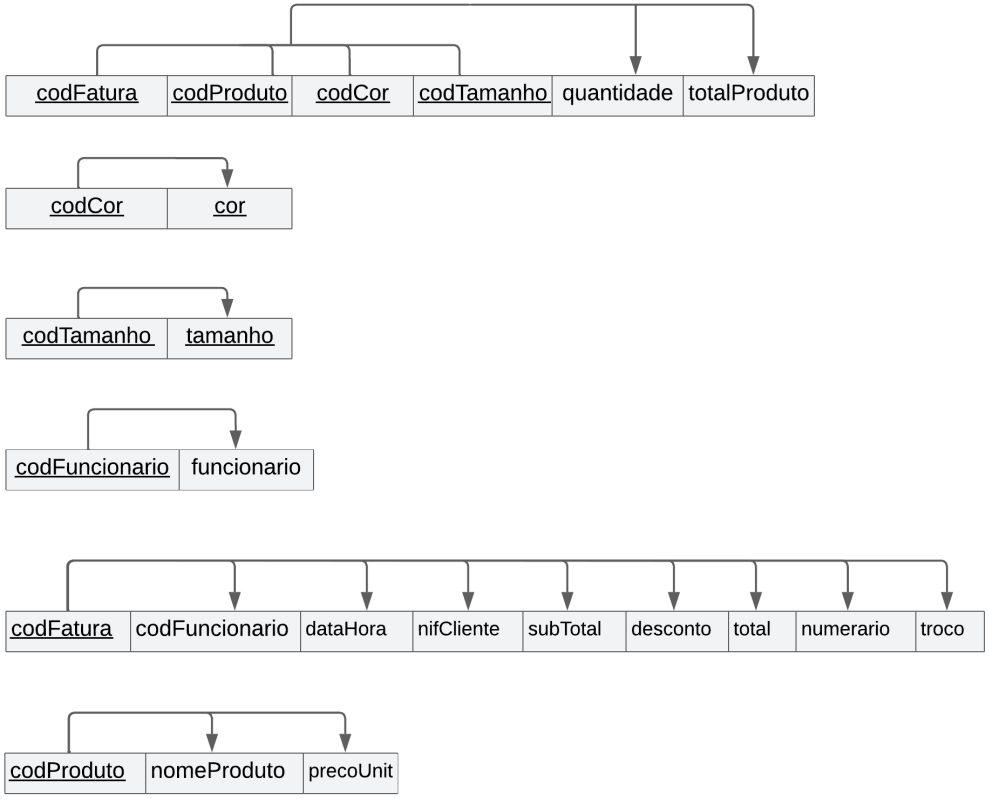
codFatura	codProduto	codCor	codTamanho	quantidade	totalProduto
FT0001	P0001	C01	T01	2	31,9
FT0001	P0002	C02	T02	1	27,99
FT0001	P0003	C03	T03	1	29,95
FT0001	P0004	C03	T04	1	35,95
FT0001	P0005	C04	T05	1	29,95

Tabela 16 - Tabela Carrinho na 3FN

codFuncionario	nomeFuncionario
112233445	Carlos Ribeiro

Tabela 17 - Tabela Funcionário na 3FN

Figura 21 - Diagrama da Fatura na 3FN



Existem mais fases na normalização, mas decidimos apenas fazer os principais.

Normalização do registo de Defeitos

Na Figura 22 podemos ver o mockup criado para o documento de registo de defeitos, este mockup identifica dados estáticos como por exemplo o título do documento e o título de cada secção que não serão guardados na BD. Também conseguimos identificar os dados dinâmicos que serão guardados na BD como por exemplo a peça em que foi identificada um defeito.

Figura 22 - Mockup registo de defeitos

Ao retirar os dados necessários do mockup da fatura surge a seguinte tabela.

CodDefeito	DataHora	CodFun	NomeFuncionario	CodPeca	Peca	CodCor	Cor	CodTamanho	Tamanho	Foto	Quantidade	CodTipoDefeito	TipoDefeito	Descricao
D0001	05/06/2023 17:02	F0103	Carlos Ribeiro	P0001	T-SHIRT	1	Vermelho	1	Médio	/D0001	7	1	Estampa	Estampa da t-shirt mal feita, como é possível ver na imagem, a palavra “BE” quase que não se vê.

Tabela 18 - Tabela registo de defeitos na UNF

Para passar esta tabela para a 1FN é necessário que a intersecção entre uma linha e uma coluna contenha um e um só valor e contenha uma chave primária válida, logo a tabela já se encontra na 1FN.

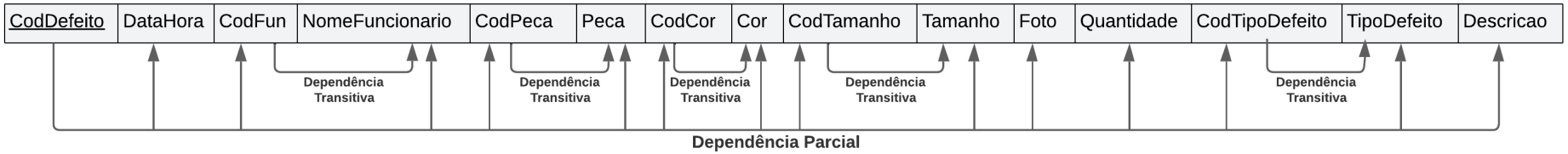
CodDefeito	DataHora	CodFun	NomeFuncionario	CodPeca	Peca	CodCor	Cor	CodTamanho	Tamanho	Foto	Quantidade	CodTipoDefeito	TipoDefeito	Descricao
D0001	05/06/2023 17:02	F0103	Carlos Ribeiro	P0001	T-SHIRT	1	Vermelho	1	Médio	/D0001	7	1	Estampa	Estampa da t-shirt mal feita, como é possível ver na imagem, a palavra “BE” quase que não se vê.

Tabela 19 - Tabela registo de defeitos na 1FN

Para conseguirmos passar esta tabela para a 2FN é necessário verificar se existem dependências funcionais, ou seja, dependências entre atributos e parte da chave. Caso existam dependências parciais é necessário removê-las da tabela inicial e colocá-las numa nova tabela.

De modo a identificar as dependências existentes na tabela criamos o seguinte diagrama de dependências.

Figura 23 - Diagrama de dependências de registo de Defeitos



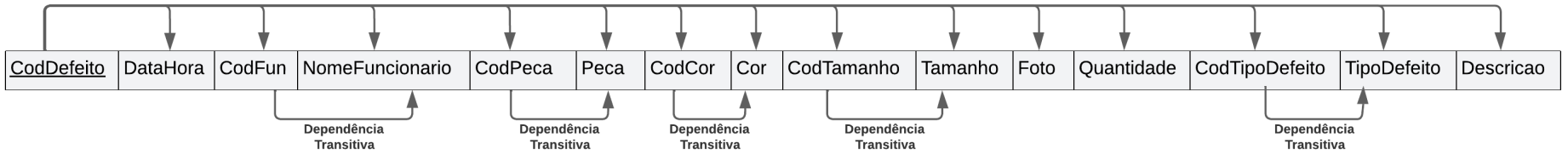
Analisando o diagrama conseguimos verificar que existe apenas uma dependência parcial, logo a tabela já se encontra na 2FN.

CodDefeito	DataHora	CodFun	NomeFuncionario	CodPeca	Peca	CodCor	Cor	CodTamanho	Tamanho	Foto	Quantidade	CodTipoDefeito	TipoDefeito	Descricao
D0001	05/06/2023 17:02	F0103	Carlos Ribeiro	P0001	T-SHIRT	1	Vermelho	1	Médio	/D0001	7	1	Estampa	Estampa da t-shirt mal feita, como é possível ver na imagem, a palavra “BE” quase que não se vê.

Tabela 20 - Tabela registo de defeitos na 2FN

O último passo é passar as tabelas da 2FN para a 3FN. Para que isto seja possível é necessário verificar se existem dependências transitivas, ou seja, dependências entre atributos não chave. Caso existam, teremos de removê-las, colocando-as numa nova tabela.

Figura 24 - Diagrama resultante da 2FN de registo de Defeitos



Como podemos observar, existem dependências transitivas, logo, existe a necessidade de remover essas dependências.

CodDefeito	DataHora	CodFun	CodPeca	CodCor	CodTamanho	Foto	Quantidade	CodTipoDefeito	Descricao
D0001	05/06/2023 17:02	F0103	P0001	1	1	/D0001	7	1	Estampa da t-shirt mal feita, como é possível ver na imagem, a palavra “BE” quase que não se vê.

Tabela 21 - Tabela Defeito na 3FN

CodFun	NomeFuncionario
F0103	Carlos Ribeiro

Tabela 22 - Tabela Funcionario na 3FN

CodPeca	Peca
P0001	T-SHIRT

Tabela 23 - Tabela Peca na 3FN

CodCor	Cor
1	Vermelho

Tabela 24 - Tabela Cor na 3FN

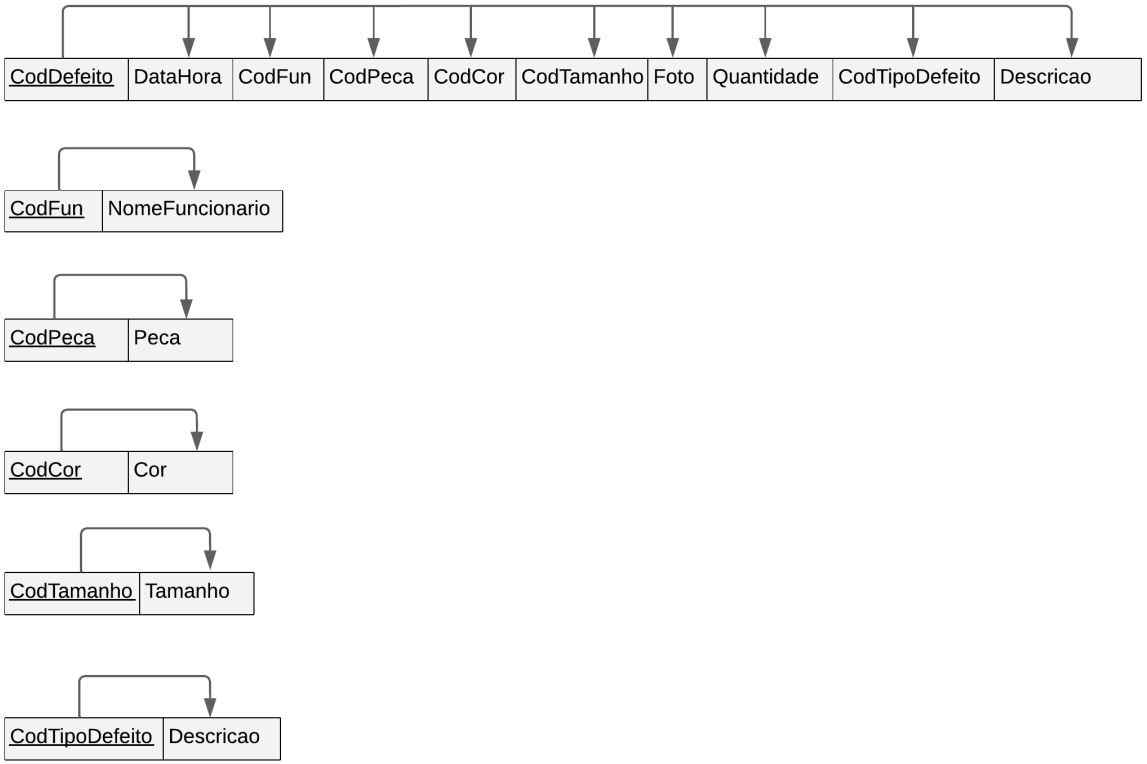
CodTamanho	Tamanho
1	Médio

Tabela 25 - Tabela Tamanho na 3FN

CodTipoDefeito	TipoDefeito
1	Estampa

Tabela 26 - Tabela TipoDefeito na 3FN

Figura 25 - Diagrama de registo de Defeitos na 3FN



Existem mais fases na normalização, mas decidimos apenas fazer os principais.

Normalização da reposição de Stock

Na Figura 26 podemos ver o mockup criado para o documento de reposição de stock, este mockup identifica dados estáticos como por exemplo o título do documento que não será guardado na BD. Também conseguimos identificar os dados dinâmicos que serão guardados na BD como por exemplo os produtos que foram repostos.

Figura 26 - Mockup reposição de stock

Reposição de Stock

Código reposição: R0001

Data e Hora: 05/06/2023 ás 17:02

Código do funcionário: F0103

Nome do funcionário: Carlos Ribeiro

Produto	Cor	Tamanho	Qtd
T-SHIRT	VERMELHO	XS	20
CASACO	CASTANHO	L	17
CALÇA	AZUL	36	15
SAPATO	AZUL	40	10
SWEATSHIRT	VERDE	23	25
TOTAL			87

Ao retirar os dados necessários do mockup da fatura surge a seguinte tabela.

CodReposicao	DataHora	CodFuncionario	NomeFuncionario	CodProduto	Produto	CodCor	Cor	CodTamanho	Tamanho	Quantidade	Total
R0001	05/06/2023 17:02	F0103	Carlos Ribeiro	P0001	T-Shirt	C01	Vermelho	T01	XS	20	87
				P0002	Casaco	C02	Castanho	T02	L	17	
				P0003	Calça	C03	Azul	T03	36	15	
				P0004	Sapato	C03	Azul	T04	40	10	
				P0005	Sweatshirt	C04	Verde	T05	23	25	

Tabela 27 - Tabela reposição de stock na UNF

Para passar esta tabela para a 1FN é necessário que a intersecção entre uma linha e uma coluna contenha um e um só valor e contenha uma chave primária válida, para isso foram feitas as seguintes alterações.

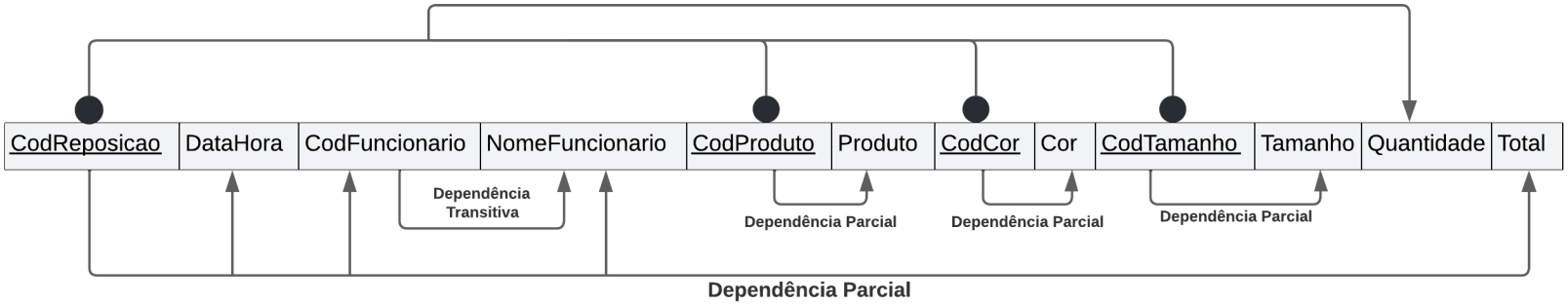
CodReposicao	DataHora	CodFuncionario	NomeFuncionario	CodProduto	Produto	CodCor	Cor	CodTamanho	Tamanho	Quantidade	Total
R0001	05/06/2023 17:02	F0103	Carlos Ribeiro	P0001	T-Shirt	C01	Vermelho	T01	XS	20	87
R0001	05/06/2023 17:02	F0103	Carlos Ribeiro	P0002	Casaco	C02	Castanho	T02	L	17	87
R0001	05/06/2023 17:02	F0103	Carlos Ribeiro	P0003	Calça	C03	Azul	T03	36	15	87
R0001	05/06/2023 17:02	F0103	Carlos Ribeiro	P0004	Sapato	C03	Azul	T04	40	10	87
R0001	05/06/2023 17:02	F0103	Carlos Ribeiro	P0005	Sweatshirt	C04	Verde	T05	23	25	87

Tabela 28 - Tabela reposição de stock na 1FN

Para conseguirmos passar esta tabela para a 2FN é necessário verificar se existem dependências funcionais, ou seja, dependências entre atributos e parte da chave. Caso existam dependências parciais é necessário removê-las da tabela inicial e colocá-las numa nova tabela.

De modo a identificar as dependências existentes na tabela criamos o seguinte diagrama de dependências.

Figura 27 - Diagrama de dependências da reposição de stock



Analisando o diagrama conseguimos verificar que existem dependências parciais, logo, existe a necessidade de passar essas dependências para novas tabelas.

codReposicao	codProduto	codCor	codTamanho	quantidade
R0001	P0001	C01	T01	20
R0001	P0002	C02	T02	17
R0001	P0003	C03	T03	15
R0001	P0004	C03	T04	10
R0001	P0005	C04	T05	25

Tabela 29 - Tabela StockReposicao na 2FN

codCor	Cor
C01	Vermelho
C02	Castanho
C03	Azul
C04	Verde

Tabela 30 - Tabela Cor na 2FN

codTamanho	Tamanho
T01	XS
T02	L
T03	36
T04	40
T05	23

Tabela 31 - Tabela Tamanho na 2FN

codProduto	nomeProduto
P0001	T-Shirt
P0002	Casaco
P0003	Calça
P0004	Sapato
P0005	Sweatshirt

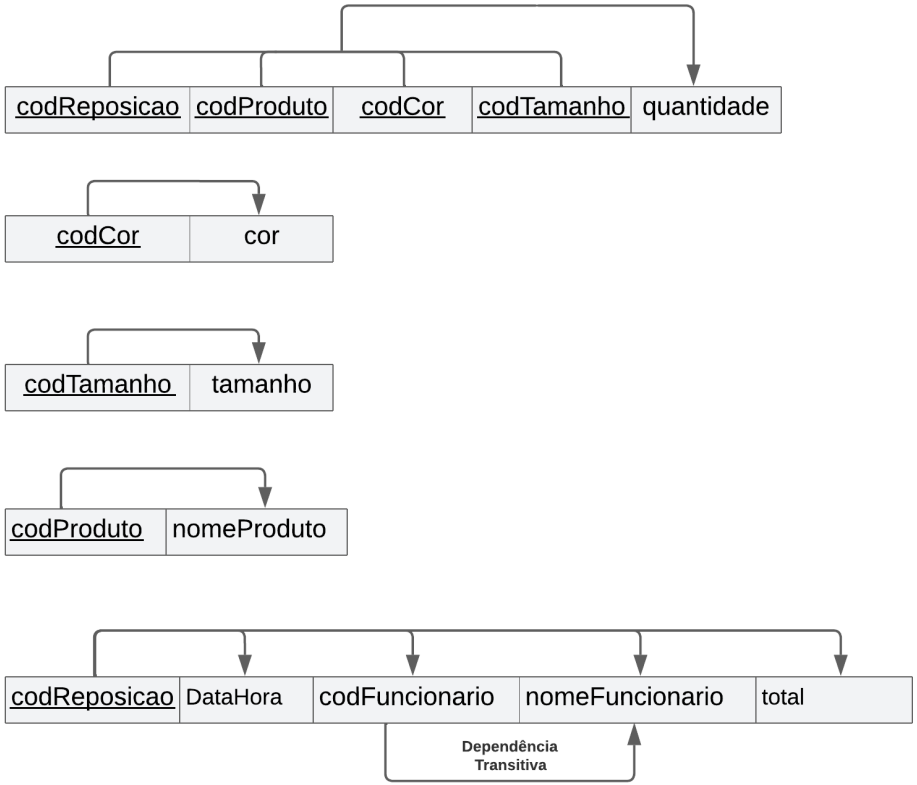
Tabela 32 - Tabela Produto na 2FN

codReposicao	DataHora	codFuncionario	nomeFuncionario	total
R0001	05/06/2023 17:02	F0103	Carlos Ribeiro	87

Tabela 33 - Tabela Reposicao na 2FN

O último passo é passar as tabelas da 2FN para a 3FN. Para que isto seja possível é necessário verificar se existem dependências transitivas, ou seja, dependências entre atributos não chave. Caso existam, teremos de removê-las, colocando-as numa nova tabela.

Figura 28 - Diagrama resultante da 2FN da reposição de stock



Como podemos observar, existem dependências transitivas, logo, existe a necessidade de remover essas dependências.

codReposicao	codProduto	codCor	codTamanho	quantidade
R0001	P0001	C01	T01	20
R0001	P0002	C02	T02	17
R0001	P0003	C03	T03	15
R0001	P0004	C03	T04	10
R0001	P0005	C04	T05	25

Tabela 34 - Tabela StockReposicao na 3FN

codCor	cor
C01	Vermelho
C02	Castanho
C03	Azul
C04	Verde

Tabela 35 - Tabela Cor na 3FN

codTamanho	tamanho
T01	XS
T02	L
T03	36
T04	40
T05	23

Tabela 36 - Tabela Tamanho na 3FN

codProduto	nomeProduto
P0001	T-Shirt
P0002	Casaco
P0003	Calça
P0004	Sapato
P0005	Sweatshirt

Tabela 37 - Tabela Produto na 3FN

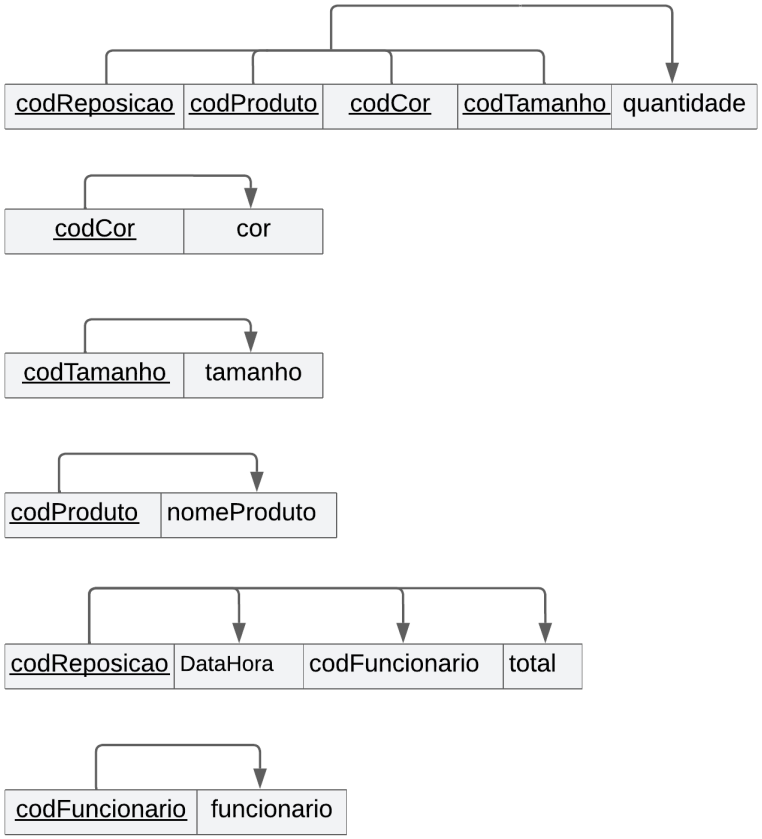
codReposicao	DataHora	codFuncionario	total
R0001	05/06/2023 17:02	F0103	87

Tabela 38 - Tabela Reposicao na 3FN

codFuncionario	nomeFuncionario
F0103	Carlos Ribeiro

Tabela 39 - Tabela Funcionario na 3FN

Figura 29 - Diagrama da reposição de stock na 3FN



Existem mais fases na normalização, mas decidimos apenas fazer os principais.

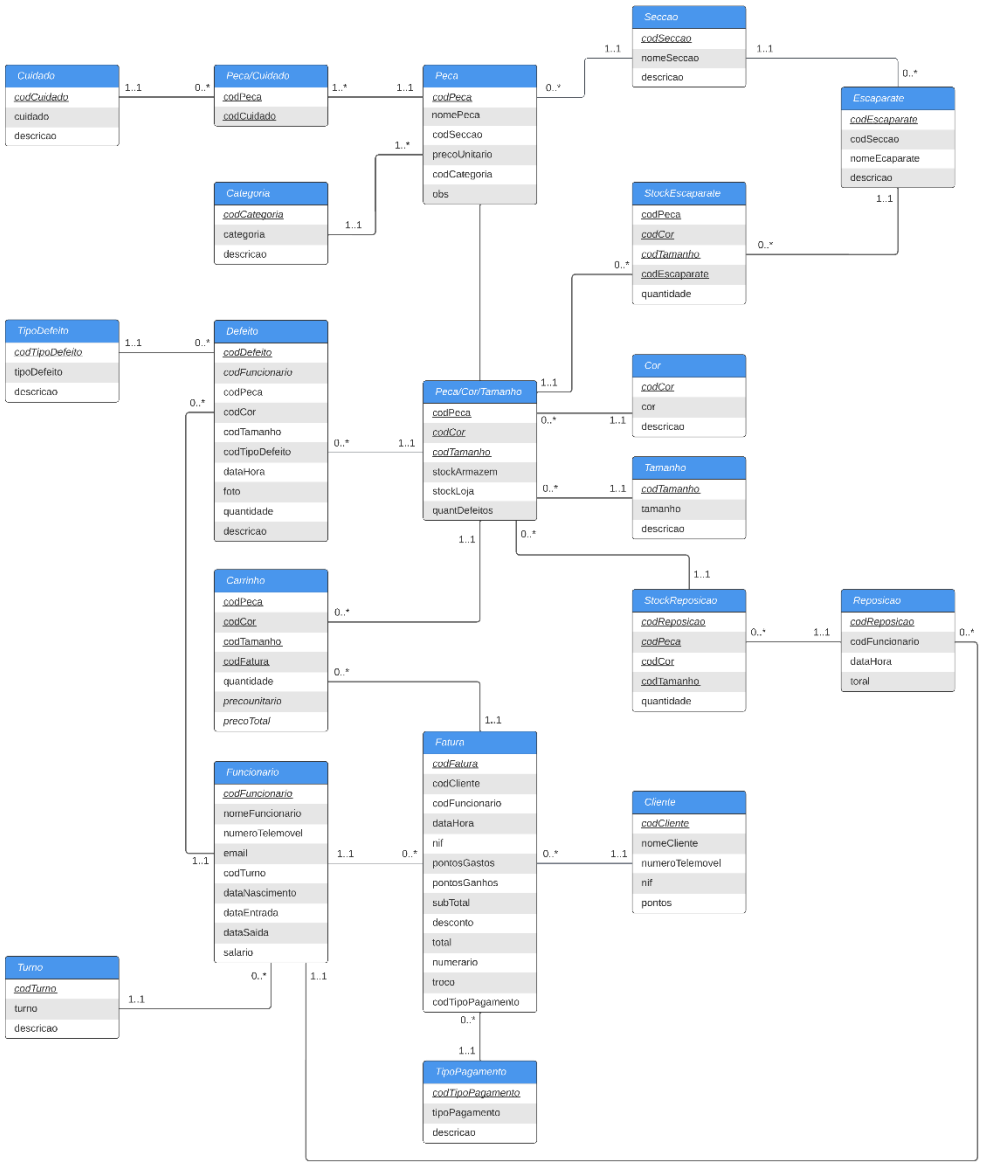
Junção das tabelas

Após termos feito a normalização dos três documentos, verificamos se existiam diferenças. As diferenças que encontramos foram as tabelas de tipo como por exemplo a tabela unidade, secção, etc…

Decidimos implementar essas tabelas visto que reduzem a redundância dos dados. Para além disse existiram tabelas “iguais” como por exemplo a tabela Produto existia em mais do que um dos documentos. Estas tabelas tinham as mesmas chaves primárias, mas uma tinha 1 atributo a mais e ao juntar as duas tabelas, mantendo a chave primária e adicionando atributos não chave, essas tabelas ficaram iguais às tabelas do modelo lógico logo não houve problema.

Tendo isto em conta a estrutura final da BD que será implementada é a seguinte.

Figura 30 - Diagrama final da BD



2.2 Desenho Físico

Dados Obrigatórios

Neste topico identificamos os campos das diversas tabelas que não poderam ser nulos.

Categoria

- codCategoria;
- categoria;

Cor

- codCor;
- cor;

Cuidado

- codCuidado;
- cuidado;

Seccao

- codSeccao;
- nomeSeccao;

Tamanho

- codTamanho;
- tamanho;

TipoDefeito

- codTipoDefeito;
- tipoDefeito;

TipoPagamento

- codTipoPagamento;
- tipoPagamento;

Turno

- codTurno;
- turno;

Cliente

- codCliente;
- nome;
- numeroTelemovel;
- pontos;

Funcionario

- codFuncionario;
- nomeFuncionario;
- numeroTelemovel;

- email;
- codTurno;
- dataNascimento;
- dataEntrada;
- salario;

Peca

- codPeca;
- nomePeca;
- codSeccao;
- precoUnitario;
- codCategoria

Escaparate

- codEscaparate;
- nomeEscaparate;
- codSeccao;

Peca/Cor/Tamanho

- codPeca;
- codCor;
- codTamanho;
- stockArmazem;
- stockLoja;
- quantDefeitos;

StockEscaparate

- codEscaparate;
- codPeca;
- codCor;
- codTamanho;
- quantidade

Defeito

- codDefeito;
- codFuncionario;
- codPeca;
- codCor;
- codTamanho;
- codTipoDefeito;
- dataHora;
- foto;
- quantidade;
- descricao;

Reposicao

- codReposicao;
- DataHora;
- codFuncionario;
- totalAdicionadas;
- totalRemovidas;

StockReposicao

- codReposicao;
- codPeca;
- codCor;
- codTamanho;
- quantidadeAdicionar;
- quantidadeDeduzir;

Fatura

- codFatura;
- codFuncionario;
- dataHora;

Carrinho

- codFatura;
- codPeca;
- codCor;
- codTamanho;
- quantidade;
- precoUnitario**;**

Peca/Cuidado

- codPeca;
- codCuidado;

Restrições de Domínio

Neste tópico identificamos as restrições que os campos das diversas tabelas têm de ter.

Categoria

- codCategoria tem de conter 3 carateres em que o primeiro é a letra C e os restantes 2 carates tem de ser um número de 0 a 9;

Cor

- codCor tem de conter 3 carateres em que o primeiro é a letra O e os restantes 2 carates tem de ser um número de 0 a 9;

Cuidado

- codCuidado tem de conter 3 carateres em que o primeiro é a letra U e os restantes 2 carates tem de ser um número de 0 a 9;

Seccao

- codCuidado tem de conter 3 carateres em que o primeiro é a letra S e os restantes 2 carates tem de ser um número de 0 a 9;

Tamanho

- codCuidado tem de conter 3 carateres em que o primeiro é a letra T e os restantes 2 carates tem de ser um número de 0 a 9;

TipoDefeito

- codCuidado tem de conter 3 carateres em que o primeiro é a letra I e os restantes 2 carates tem de ser um número de 0 a 9;

TipoPagamento

- codCuidado tem de conter 3 carateres em que o primeiro é a letra A e os restantes 2 carates tem de ser um número de 0 a 9;

Turno

- codCuidado tem de conter 3 carateres em que o primeiro é a letra N e os restantes 2 carates tem de ser um número de 0 a 9;

Cliente

- codCliente tem de conter 5 carateres em que o primeiro é a letra L e os restantes 4 carates tem de ser um número de 0 a 9;
- nif que tem de conter 9 dígitos em que o primeiro dígito só pode ser 1, 2 ou 3;
- numeroTelemovel tem de conter 9 dígitos, que os primeiros 2 dígitos só podem ser 91,92,93,96;
- pontos tem de ser maior que 0;

Funcionario

- codFuncionario é o nif do funcionário e tem de conter 9 dígitos em que o primeiro dígito só pode ser 1, 2 ou 3;
- numeroTelemovel tem de conter 9 dígitos, que os primeiros 2 dígitos só podem ser 91,92,93,96;

- email tem de contar o formato de um email;
- dataNascimento tem de ter, mas de 16 ano em relação a data de entrada.
- Salario tem de ser maior que 0;

Peca

- codPeca tem de conter 5 carateres em que o primeiro é a letra P e os restantes 4 carates tem de ser um número de 0 a 9;
- precoUnitario tem de ser maior ou igual a 0;

Escaparate

- codEscaparate tem de conter 3 carateres em que o primeiro é a letra E e os restantes 2 carates tem de ser um número de 0 a 9;

Peca/Cor/Tamanho

- stockArmazem tem de ser maior ou igual a 0;
- stockLoja tem de ser maior ou igual a 0;
- quantidadeDefeitos tem de ser ou igual a que 0;

StockEscaparate

- quantidade tem de ser maior que 0;

Defeito

- codDefeito tem de conter 5 carateres em que o primeiro é a letra D e os restantes 4 carates tem de ser um número de 0 a 9;

Reposicao

- codReposicao tem de conter 10 carateres em que o primeiro é a letra R e os restantes 9 carates tem de ser um número de 0 a 9;
- totalAdicionadas tem de ser ou igual a que 0;
- totalRemovidas tem de ser ou igual a que 0;

StockReposicao

- quantidadeAdicionar tem de ser ou igual a que 0;
- quantidadeDeduzir tem de ser ou igual a que 0;

Fatura

- codFatura tem de conter 10 carateres em que o primeiro é a letra F e os restantes 9 carates tem de ser um número de 0 a 9;
- nifCliente que tem de conter 9 dígitos em que o primeiro dígito só pode ser 1, 2 ou 3;
- pontosGastos tem de ser ou igual a que 0;
- pontosGanhos tem de ser ou igual a que 0;
- subtotal tem de ser ou igual a que 0;
- desconto tem de ser ou igual a que 0;
- total tem de ser o valor do subtotal – desconto;
- numerario tem de ser maior ou igual ao total;
- troco tem de ser o valor do numerário – total;

Carrinho

- quantidade tem de ser ou igual a que 0;
- precoUnitario tem de ser ou igual a que 0;
- precoTotal tem de ser ou igual a que 0;

Integridade de entidades

Neste tópico identificamos as chaves primárias, as chaves estrangeiras e os dados únicos das diversas tabelas.

Categoria

- Chave Primária – codCategoria;
- Unique - categoria;

Cor

- Chave Primária – codCor;
- Unique – cor;

Cuidado

- Chave Primária – codCuidado;
- Unique – cuidado;

Seccao

- Chave Primária – codSeccao ;
- Unique - nomeSeccao;

Tamanho

- Chave Primária – codTamanho;
- Unique – tamanho;

TipoDefeito

- Chave Primária – codTipoDefeito;
- Unique – tipoDefeito;

TipoPagamento

- Chave Primária – codTipoPagamento;
- Unique – tipoPagamento;

Turno

- Chave Primária – codTurno;
- Unique – turno;

Cliente

- Chave Primária – codCliente;
- Unique - numeroTelemovel ;
- Unique - nif;

Funcionario

- Chave Primária – codFuncionario;
- Unique - email;
- Unique - numeroTelemovel;

Peca

- Chave Primária – codPeca;
- Chave estrangeira – codSeccao referente ao campo codSeccaoda tabela Seccao;

- Chave estrangeira – codCategoria referente ao campo codCategoria da tabela Categoria;
- Unique - nomePeca

Escaparate

- Chave Primária – codEscaparate;
- Chave estrangeira – codSeccao referente ao campo codSeccaoda da tabela Seccao;
- Unique - nomeEscaparate

Peca/Cor/Tamanho

- Chave Primária – codPeca, codCor, codTamanho;
- Chave estrangeira – codPeca referente ao campo codPeca tabela Peca;
- Chave estrangeira – codCor referente ao campo codCor tabela Cor;
- Chave estrangeira – codTamanho referente ao campo codTamanho da tabela Tamanho;

StockEscaparate

- Chave Primária – codPeca, codCor, codTamanho;
- Chave estrangeira – codPeca, codCor, codTamanho referente aos campos codPeca, codCor, codTamanho da tabela Peca/Cor/Tamanho;

Defeito

- Chave Primária – codDefeito;
- Chave estrangeira – codPeca, codCor, codTamanho referente aos campos codPeca, codCor, codTamanho da tabela Peca/Cor/Tamanho;
- Chave estrangeira – codFuncionario referente ao campo codFuncionario da tabela Funcionario;
- Chave estrangeira – codTipoDefeito referente ao campo codTipoDefeito da tabela TipoDefeito;

Reposicao

- Chave Primária – codReposicao;
- Chave estrangeira – codFuncionario referente ao campo codFuncionario da tabela Funcionario;

StockReposicao

- Chave Primária – codReposicao, codPeca, codCor, codTamanho;
- Chave estrangeira – codReposicao referente ao campo codReposicao da tabela Reposicao;
- Chave estrangeira – codPeca, codCor, codTamanho referente aos campos codPeca, codCor, codTamanho da tabela Peca/Cor/Tamanho;

Fatura

- Chave Primária – codFatura;
- Chave estrangeira – codCliente referente ao campo codCliente da tabela Cliente;
- Chave estrangeira – codFuncionario referente ao campo codFuncionario da tabela Funcionario;

- Chave estrangeira – codTipoPagamento referente ao campo codTipoPagamento da tabela TipoPagamento;

Carrinho

- Chave Primária – codFatura, codPeca, codCor, codTamanho;
- Chave estrangeira - codFatura referente ao campo codFatura da tabela Fatura;
- Chave estrangeira – codPeca, codCor, codTamanho referente aos campos codPeca, codCor, codTamanho da tabela Peca/Cor/Tamanho;

Peca/Cuidado

- Chave Primária – codPeca, codCuidado;
- Chave estrangeira – codPeca referente ao campo codPeca da tabela Peca;
- Chave estrangeira – codCuidado referente ao campo codCuidado da tabela Cuidado;

Integridade referencial

Para garantir a Integridade Referencial, optamos pela opção NO ACTION, que permite apenas excluir um registo pai se não houver registos filhos relacionados a ele. Isso ajuda a manter uma integridade mais robusta.

Vistas

Para facilitar nas pesquisas mais recorrentes realizadas pela loja de roupa decidimos criar algumas vistas que são:

- Obter faturas do ano anterior;

```
CREATE VIEW [dbo].[FaturasAnoAnterior] AS
SELECT *
FROM [Grupo_206].[dbo].[Fatura]
WHERE YEAR([dataHora]) = YEAR(DATEADD(YEAR, -1, GETDATE()))
```

- Obter faturas do mês anterior;

```
CREATE VIEW [dbo].[FaturasMesAnterior] AS
SELECT *
FROM [Grupo_206].[dbo].[Fatura]
WHERE MONTH([dataHora]) = MONTH(DATEADD(MONTH, -1, GETDATE()))
AND YEAR([dataHora]) = YEAR(DATEADD(MONTH, -1, GETDATE()))
```

- Obter faturas do dia anterior;

```
CREATE VIEW [dbo].[FaturasDiaAnterior] AS
SELECT *
FROM [Grupo_206].[dbo].[Fatura]
WHERE CONVERT(DATE, [dataHora]) = DATEADD(DAY, -1, CONVERT(DATE, GETDATE()))
```

- Obter número de faturas do ano anterior;

```
CREATE VIEW [dbo].[NumeroFaturasAnoAnterior] AS
SELECT COUNT([codFatura]) AS Count
FROM [Grupo_206].[dbo].[FaturasAnoAnterior]
```

- Obter número de faturas do mês anterior;

```
CREATE VIEW [dbo].[NumeroFaturasMesAnterior] AS
SELECT COUNT([codFatura]) AS Count
FROM [Grupo_206].[dbo].[FaturasMesAnterior]
```

- Obter número de faturas do dia anterior;

```
CREATE VIEW [dbo].[NumeroFaturasDiaAnterior] AS
SELECT COUNT([codFatura]) AS Count
FROM [Grupo_206].[dbo].[FaturasDiaAnterior]
```

- Obter funcionários ordenados por dinheiro ganho de forma descendente;

```
CREATE VIEW [dbo].[MelhoresFuncionariosMesAnterior] AS
SELECT TOP 100 PERCENT ROW_NUMBER() OVER (ORDER BY V.total DESC) AS Posicao,
F.*, V.total
FROM (
       SELECT F.codFuncionario, SUM(F.total) AS total
       FROM [Grupo_206].[dbo].[FaturasMesAnterior] AS F
       GROUP BY F.codFuncionario
) AS V
JOIN Funcionario AS F ON V.codFuncionario = F.codFuncionario
ORDER BY V.total DESC
```

- Obter peça mais vendida;

```
CREATE VIEW [dbo].[PecaMaisVendida] AS
SELECT TOP 1 P.*
FROM (
    SELECT TOP 1 C.codPeca, COUNT(C.codPeca) AS Vendidas
    FROM Carrinho AS C
    GROUP BY C.codPeca
    ORDER BY COUNT(C.codPeca) DESC
) AS C
JOIN Peca AS P ON C.codPeca = P.codPeca
```

- Obter peças vendidas no dia anterior;

```
CREATE VIEW [dbo].[PecasVendidasDiaAnterior] AS
SELECT DISTINCT P.*
FROM [Grupo_206].[dbo].[FaturasDiaAnterior] AS F
JOIN [Grupo_206].[dbo].Carrinho as C ON C.codFatura = F.codFatura
JOIN [Grupo_206].[dbo].[Peca] AS P on P.codPeca = C.codPeca
```

- Obter código e nome da Peça, Cor e Tamanho e as suas quantidades em stock;

```
CREATE VIEW [dbo].[PecaCorTamanho] AS
SELECT PCT.[codPeca],
              P.nomePeca
      ,PCT.[codCor]
         ,C.cor
      ,PCT.[codTamanho]
         ,T.tamanho
      ,PCT.[stockArmazem]
      ,PCT.[stockLoja]
      ,PCT.[quantDefeitos]
  FROM [Grupo_206].[dbo].[Peca/Cor/Tamanho] as PCT
  JOIN Peca AS P on PCT.codPeca = P.codPeca
  JOIN Cor AS C on PCT.codCor = c.codCor
  JOIN Tamanho AS T on PCT.codTamanho = T.codTamanho
```

Functions

Para facilitar pesquisas que necessitam do envio de valores e contas a serem realizadas criamos algumas functions.
As functions que criamos são:

- Obter Faturas de um cliente;

```
CREATE FUNCTION [dbo].[FaturasPorCliente] (@codCliente codCliente)
RETURNS TABLE
AS
RETURN
(
    SELECT *
    FROM [Grupo_206].[dbo].[Fatura]
    WHERE [codCliente] = @codCliente
)
```

- Obter o número de peças defeituosas identificadas em uma semana;

```
CREATE FUNCTION [dbo].[NumeroPecasDefeituosasIdentificadasNaSemana]
(
    @data DATE
)
RETURNS INT
AS
BEGIN
    DECLARE @numeroSemana INT = DATEPART(WEEK, @data);
    DECLARE @numeroPecasDefeituosas INT;

    SELECT @numeroPecasDefeituosas = SUM(D.[quantidade])
    FROM [Grupo_206].[dbo].[Defeito] AS D
    WHERE DATEPART(WEEK, D.dataHora) = @numeroSemana;

    RETURN ISNULL(@numeroPecasDefeituosas, 0);
END;
```

- Calcular quantos pontos equivalem a um valor monetário;

```
CREATE FUNCTION [dbo].[PontosParaDinheiro]
(
      @pontos Pontos
)
RETURNS dinheiro
AS
BEGIN
    RETURN @pontos * 0.03;
END;
CREATE VIEW [dbo].[FaturasAnoAnterior] AS
SELECT *
FROM [Grupo_206].[dbo].[Fatura]
WHERE YEAR([dataHora]) = YEAR(DATEADD(YEAR, -1, GETDATE()))
```

- Calcular quanto vale um valor monetário em pontos;

```sql
CREATE FUNCTION [dbo].[DinheiroParaPontos]
(
	@dinheiro dinheiro
)
RETURNS Pontos
AS
BEGIN
    RETURN FLOOR(@dinheiro);
END;
CREATE VIEW [dbo].[FaturasAnoAnterior] AS
SELECT *
FROM [Grupo_206].[dbo].[Fatura]
WHERE YEAR([dataHora]) = YEAR(DATEADD(YEAR, -1, GETDATE()))
```

Stored Procedures

Para facilitar operações mais recorrentes decidimos criar alguns *Stored Procedures*.

Os Stored Procedures que criamos são:

- Adicionar peça a escaparate (Este processo insere um novo registo na tabela 'stockEscaparate' se ele não existir; caso contrário, atualiza a quantidade ao incrementar a variável recebida.)

```
CREATE PROCEDURE [dbo].[AdicionarPecaEscaparate]
@codEscaparate codEscaparate,
@codPeca codPeca,
@codCor codCor,
@codTamanho codTamanho,
@quantidade quantidade
AS
BEGIN
	SET NOCOUNT ON;
	SET XACT_ABORT ON;

	BEGIN TRY
	BEGIN TRANSACTION;
		MERGE INTO [dbo].[StockEscaparate] AS Target
		USING (
		SELECT @codPeca AS codPeca,
		   @codCor AS codCor,
		   @codTamanho AS codTamanho,
		   @codEscaparate AS codEscaparate,
		   @quantidade AS quantidade
		) AS Source
		ON (Target.[codPeca] = Source.codPeca AND Target.[codCor] =
Source.codCor AND Target.[codTamanho] = Source.codTamanho AND
Target.[codEscaparate] = Source.codEscaparate)
		WHEN MATCHED THEN
		UPDATE SET Target.[quantidade] = Target.[quantidade] +
Source.quantidade
		WHEN NOT MATCHED THEN
		INSERT ([codPeca], [codCor], [codTamanho], [codEscaparate],
[quantidade])
		VALUES (Source.codPeca, Source.codCor, Source.codTamanho,
Source.codEscaparate, Source.quantidade);

		COMMIT TRANSACTION;
	END TRY
	BEGIN CATCH
		IF @@TRANCOUNT > 0
			ROLLBACK TRANSACTION;

		DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();
		DECLARE @ErrorSeverity INT = ERROR_SEVERITY();
		DECLARE @ErrorState INT = ERROR_STATE();

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

- Criar fatura

```
CREATE PROCEDURE [dbo].[CriarReposicao]
@codReposicao codReposicao,
@codFuncionario nif
AS
BEGIN
	SET NOCOUNT ON;
	SET XACT_ABORT ON;

	BEGIN TRY
	BEGIN TRANSACTION;
		INSERT INTO [dbo].[Reposicao]
		   ([codReposicao]
		   ,[codFuncionario])
		 VALUES
		   (@codReposicao, @codFuncionario);

		COMMIT TRANSACTION;
	END TRY
	BEGIN CATCH
		IF @@TRANCOUNT > 0
			ROLLBACK TRANSACTION;

		DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();
		DECLARE @ErrorSeverity INT = ERROR_SEVERITY();
		DECLARE @ErrorState INT = ERROR_STATE();

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

- Processo para adicionar uma peça à fatura, validando o stock para verificar se a quantidade solicitada está disponível, atualizando as tabelas de estoque da peça e ajustando o valor total a ser pago na fatura.;

```
CREATE PROCEDURE [dbo].[AdicionarPecaNaFatura]
@codFatura codReposicao,
@codPeca codPeca,
@codCor codCor,
@codTamanho codTamanho,
@quantidade quantidade
AS
BEGIN
	SET NOCOUNT ON;
	SET XACT_ABORT ON;

	BEGIN TRY
	BEGIN TRANSACTION;
		DECLARE @stockLoja quantidade;
		SELECT TOP 1 @stockLoja = [stockLoja]
		FROM [dbo].[Peca/Cor/Tamanho]
		WHERE codPeca = @codPeca AND codCor = @codCor AND codTamanho =
@codTamanho;
```

```
		IF @quantidade <= 0 OR @quantidade > @stockLoja
		BEGIN
			RAISERROR('Quantidade fornecida inválida.', 16, 1);
			RETURN;
		END
		IF EXISTS(
			SELECT TOP 1 codFatura
			FROM [Grupo_206].[dbo].[Fatura]
			WHERE codFatura = @codFatura AND troco IS NOT NULL
		)
		BEGIN
			RAISERROR('Fatura já foi paga.', 16, 1);
			RETURN;
		END

		DECLARE @precoUnitario dinheiro;
		SELECT @precoUnitario = P.precoUnitario
		FROM [dbo].[Peca] as P
		WHERE P.codPeca = @codPeca;

		INSERT INTO [dbo].[Carrinho]
		   ([codFatura]
		   ,[codPeca]
		   ,[codCor]
		   ,[codTamanho]
		   ,[quantidade]
		   ,[precoUnitario])
		 VALUES
		   (@codFatura
		   ,@codPeca
		   ,@codCor
		   ,@codTamanho
		   ,@quantidade
		   ,@precoUnitario);

		EXEC [dbo].[RemoverPecasDosEscaparates] @codPeca = @codPeca,
		@codCor = @codCor,
		@codTamanho = @codTamanho,
		@quantidade = @quantidade;

		UPDATE [dbo].[Peca/Cor/Tamanho]
		SET [stockLoja] = [stockLoja] - @quantidade
		WHERE codPeca = @codPeca AND codCor = @codCor AND codTamanho =
@codTamanho

		UPDATE [dbo].[Fatura]
		SET [subTotal] = ISNULL([subTotal], 0) + (@precoUnitario *
@quantidade)
		WHERE codFatura = @codFatura

		COMMIT TRANSACTION;
	END TRY
	BEGIN CATCH
		IF @@TRANCOUNT > 0
			ROLLBACK TRANSACTION;

		DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();
		DECLARE @ErrorSeverity INT = ERROR_SEVERITY();
		DECLARE @ErrorState INT = ERROR_STATE();

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

- Processo de realização do pagamento, envolve o cálculo e validação do desconto, garantindo que não ultrapasse o valor a pagar, cálculo dos pontos ganhos e atualizando as tabelas relacionadas;

```
CREATE PROCEDURE [dbo].[PagarFatura]
@codFatura codFatura,
@codCliente codCliente = NULL,
@nifCliente nif = NULL,
@pontosGastos Pontos = 0,
@numerario dinheiro,
@codTipoPagamento codTipoPagamento
AS
BEGIN
	SET NOCOUNT ON;
	SET XACT_ABORT ON;

	BEGIN TRY
	BEGIN TRANSACTION;
		DECLARE @desconto dinheiro = 0;
		DECLARE @pontosGanhos Pontos = 0;

		DECLARE @total dinheiro = (
			SELECT SUM(precoTotal)
			FROM [dbo].[Carrinho]
			WHERE codFatura = @codFatura
		);

		IF @total <= 0
		BEGIN
			RAISERROR('Não foram adicionadas peças à fatura.', 16,
1);
			RETURN;
		END

		IF @codCliente IS NOT NULL
		BEGIN
			DECLARE @pontosCliente Pontos = (
				SELECT TOP 1 pontos
				FROM [dbo].[Cliente]
				WHERE codCliente = @codCliente
			);

			IF @pontosGastos <= 0 OR @pontosGastos > @pontosCliente
			BEGIN
				RAISERROR('Pontos Gastos fornecidos inválidos
(Ultrapassam os pontos do cliente).', 16, 1);
				RETURN;
			END

			SET @desconto = dbo.PontosParaDinheiro(@pontosGastos);

			IF @desconto > @total
			BEGIN
				RAISERROR('Pontos Gastos fornecidos inválidos
(Ultrapassam o preço da compra).', 16, 1);
				RETURN;
			END
```

```
                        SET @desconto = dbo.PontosParaDinheiro(@pontosGastos);

                        IF @desconto > @total
                        BEGIN
                                RAISERROR('Pontos Gastos fornecidos inválidos
(Ultrapassam o preço da compra).', 16, 1);
                                RETURN;
                        END

                        SET @pontosGanhos = dbo.DinheiroParaPontos(@total);
                END
                ELSE IF @pontosGastos > 0 OR @pontosGastos < 0
                BEGIN
                        RAISERROR('Não pode usar pontos.', 16, 1);
                        RETURN;
                END

                UPDATE [dbo].[Fatura]
                   SET [codCliente] = @codCliente
                  ,[nifCliente] = @nifCliente
                  ,[pontosGastos] = @pontosGastos
                  ,[pontosGanhos] = @pontosGanhos
                  ,[subTotal] = @total
                  ,[desconto] = @desconto
                  ,[numerario] = @numerario
                  ,[codTipoPagamento] = @codTipoPagamento
                 WHERE codFatura = @codFatura;

                COMMIT TRANSACTION;
        END TRY
        BEGIN CATCH
                IF @@TRANCOUNT > 0
                        ROLLBACK TRANSACTION;

                DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();
                DECLARE @ErrorSeverity INT = ERROR_SEVERITY();
                DECLARE @ErrorState INT = ERROR_STATE();

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

- Processo para reportar defeito

```
CREATE PROCEDURE [dbo].[ReportarDefeito]
	@codDefeito codDefeito,
	@codFuncionario  nif,
	@codPeca codPeca,
	@codCor codCor,
	@codTamanho codTamanho,
	@codTipoDefeito codTipoDefeito,
	@foto anexo,
	@quantidadeLoja quantidade  = 0,
	@quantidadeArmazem quantidade = 0,
	@descricao descricao
AS
BEGIN
	SET NOCOUNT ON;
	SET XACT_ABORT ON;

	BEGIN TRY
    BEGIN TRANSACTION;
		IF @quantidadeLoja <= 0 and @quantidadeArmazem <= 0
		BEGIN
			RAISERROR('Quantidades fornecidas inválidas.', 16, 1)
			RETURN;
		END

		DECLARE @stockLoja quantidade
		DECLARE @stockArmazem quantidade

		SELECT TOP 1 @stockLoja = stockLoja, @stockArmazem =
stockArmazem
		FROM [dbo].[Peca/Cor/Tamanho]
		WHERE codPeca = @codPeca AND codCor = @codCor AND codTamanho =
@codTamanho

		IF (@@ROWCOUNT = 0)
		BEGIN
			RAISERROR('Combinação de Peca, Cor e Tamanho não
existe.', 16, 1)
			RETURN;
		END

		IF (@quantidadeLoja > @stockLoja)
		BEGIN
			RAISERROR('Quantidade loja fornecida inválida.', 16, 1)
			RETURN;
		END

		IF (@quantidadeArmazem > @stockArmazem)
		BEGIN
			RAISERROR('Quantidade armazem fornecida inválida.', 16,
1)
			RETURN;
		END
```

```
		INSERT INTO [dbo].[Defeito]
		 ([codDefeito], [codFuncionario], [codPeca], [codCor],
[codTamanho], [codTipoDefeito], [foto], [quantidade], [descricao])
		VALUES
		  (
				@codDefeito,
				@codFuncionario,
				@codPeca,
				@codCor,
				@codTamanho,
				@codTipoDefeito,
				@foto,
				@quantidadeLoja + @quantidadeArmazem,
				@descricao
		   )

		EXEC [dbo].[RemoverPecasDosEscaparates] @codPeca = @codPeca,

	@codCor = @codCor,

	@codTamanho = @codTamanho,

	@quantidade = @quantidadeLoja


		UPDATE [dbo].[Peca/Cor/Tamanho]
		SET [stockArmazem] = [stockArmazem] - @quantidadeArmazem,
			[stockLoja] = [stockLoja] - @quantidadeLoja,
			[quantDefeitos] = [quantDefeitos] + @quantidadeLoja +
@quantidadeArmazem
		WHERE codPeca = @codPeca AND codCor = @codCor AND codTamanho =
@codTamanho;

		COMMIT TRANSACTION;
	END TRY
	BEGIN CATCH
		IF @@TRANCOUNT > 0
			ROLLBACK TRANSACTION;

		DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();
		DECLARE @ErrorSeverity INT = ERROR_SEVERITY();
		DECLARE @ErrorState INT = ERROR_STATE();

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

- Processo para criar reposição

```
CREATE PROCEDURE [dbo].[CriarReposicao]
@codReposicao codReposicao,
@codFuncionario nif
AS
BEGIN
	SET NOCOUNT ON;
	SET XACT_ABORT ON;

	BEGIN TRY
	BEGIN TRANSACTION;
		INSERT INTO [dbo].[Reposicao]
		   ([codReposicao]
		   ,[codFuncionario])
		 VALUES
		   (@codReposicao, @codFuncionario);

		COMMIT TRANSACTION;
	END TRY
	BEGIN CATCH
		IF @@TRANCOUNT > 0
			ROLLBACK TRANSACTION;

		DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();
		DECLARE @ErrorSeverity INT = ERROR_SEVERITY();
		DECLARE @ErrorState INT = ERROR_STATE();

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

- Processo para reposição de stock

```
CREATE PROCEDURE [dbo].[ReporPecaALoja]
@codReposicao codReposicao,
@codPeca codPeca,
@codCor codCor,
@codTamanho codTamanho,
@quantidade quantidade
AS
BEGIN
	SET NOCOUNT ON;
	SET XACT_ABORT ON;

	BEGIN TRY
	BEGIN TRANSACTION;
		DECLARE @stockArmazem quantidade;
		SELECT TOP 1 @stockArmazem = [stockArmazem]
		FROM [dbo].[Peca/Cor/Tamanho]
		WHERE codPeca = @codPeca AND codCor = @codCor AND codTamanho =
@codTamanho;

		IF @quantidade <= 0 OR @quantidade > @stockArmazem
		BEGIN
			RAISERROR('Quantidade fornecida inválida.', 16, 1);
			RETURN;
		END
```

```
            INSERT INTO [dbo].[StockReposicao]
               ([codReposicao]
               ,[codPeca]
               ,[codCor]
               ,[codTamanho]
               ,[quantidadeAdicionar])
             VALUES
               (@codReposicao
               ,@codPeca
               ,@codCor
               ,@codTamanho
               ,@quantidade);

            UPDATE [dbo].[Peca/Cor/Tamanho]
            SET [stockArmazem] = [stockArmazem] - @quantidade,
            [stockLoja] = [stockLoja] + @quantidade
            WHERE codPeca = @codPeca AND codCor = @codCor AND codTamanho =
@codTamanho;

            UPDATE [dbo].[Reposicao]
               SET [totalAdicionadas] = [totalAdicionadas] + @quantidade
             WHERE [codReposicao] = @codReposicao;

            COMMIT TRANSACTION;
      END TRY
      BEGIN CATCH
            IF @@TRANCOUNT > 0
                  ROLLBACK TRANSACTION;

            DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();
            DECLARE @ErrorSeverity INT = ERROR_SEVERITY();
            DECLARE @ErrorState INT = ERROR_STATE();

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

- Processo para retirar peças da loja para o armazém

```
CREATE PROCEDURE [dbo].[RemoverPecaALoja]
      @codReposicao codReposicao,
      @codPeca codPeca,
      @codCor codCor,
      @codTamanho codTamanho,
      @quantidade quantidade
AS
BEGIN
      SET NOCOUNT ON;
      SET XACT_ABORT ON;

      BEGIN TRY
    BEGIN TRANSACTION;
            DECLARE @stockLoja quantidade;
            SELECT TOP 1 @stockLoja = [stockLoja]
            FROM [dbo].[Peca/Cor/Tamanho]
            WHERE codPeca = @codPeca AND codCor = @codCor AND codTamanho =
@codTamanho;
```

```
		IF @quantidade <= 0 OR @quantidade > @stockLoja
		BEGIN
			RAISERROR('Quantidade fornecida inválida.', 16, 1);
			RETURN;
		END

		INSERT INTO [dbo].[StockReposicao]
			   ([codReposicao]
			   ,[codPeca]
			   ,[codCor]
			   ,[codTamanho]
			   ,[quantidadeDeduzir])
		 VALUES
			   (@codReposicao
			   ,@codPeca
			   ,@codCor
			   ,@codTamanho
			   ,@quantidade)

		EXEC [dbo].[RemoverPecasDosEscaparates] @codPeca = @codPeca,

	@codCor = @codCor,

	@codTamanho = @codTamanho,

	@quantidade = @quantidade

		UPDATE [dbo].[Peca/Cor/Tamanho]
		SET [stockArmazem] = [stockArmazem] + @quantidade,
			[stockLoja] = [stockLoja] - @quantidade
		WHERE codPeca = @codPeca AND codCor = @codCor AND codTamanho =
@codTamanho

		UPDATE [dbo].[Reposicao]
		   SET [totalAdicionadas] = [totalAdicionadas]+@quantidade
		 WHERE [codReposicao] = @codReposicao

		COMMIT TRANSACTION;
	END TRY
	BEGIN CATCH
		IF @@TRANCOUNT > 0
			ROLLBACK TRANSACTION;

		DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();
		DECLARE @ErrorSeverity INT = ERROR_SEVERITY();
		DECLARE @ErrorState INT = ERROR_STATE();

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

Triggers

Para realizar processos de calculo ou validar a informação quando inserida da alterada ou apagada decidimos criar *Triggers* para assim manter a integridade dos dados da BD. Então criamos os seguintes Triggers:

- Trigger para calcular o troco.

```
CREATE   TRIGGER [dbo].[calcularTroco]
ON [dbo].[Fatura]
AFTER INSERT, UPDATE
AS
BEGIN
    UPDATE F
    SET troco = IIF(F.numerario IS NOT NULL AND F.total IS NOT NULL,
F.numerario - F.total, NULL)
    FROM Fatura AS F
    INNER JOIN inserted AS I ON F.codFatura = I.codFatura;
END;
```

- Trigger para validar se quantidade de peças nos escaparates excede a quantidade de peças em loja.

```
CREATE TRIGGER [dbo].[StockEscaparateVerirficarQuantidade]
ON [dbo].[StockEscaparate]
AFTER INSERT, UPDATE
AS
BEGIN
    IF EXISTS (
        SELECT 1
        FROM [Grupo_206].[dbo].[Peca/Cor/Tamanho] PCT
        INNER JOIN inserted I
            ON I.[codPeca] = PCT.[codPeca]
            AND I.[codCor] = PCT.[codCor]
            AND I.[codTamanho] = PCT.[codTamanho]
        WHERE PCT.[stockLoja] < I.[quantidade]
    )
    BEGIN
        ROLLBACK
        RAISERROR ('Quantidades nos escaparates maiores do que as do stock em
loja.', 16, 1);
    END
END;
```

Questões pedidas pelo Cliente

Para resposder as questoes pedidas pelo cliente, o cliente tem de executar os seguintes comandos:

1. Qual o total faturado no dia anterior;

```
SELECT * FROM [Grupo_206].[dbo].[NumeroFaturasDiaAnterior]
```

2. Qual o total faturado no mes anterior;

```
SELECT * FROM [Grupo_206].[dbo].[NumeroFaturasMesAnterior]
```

3. Qual o total faturado no ano anterior;

```
SELECT * FROM [Grupo_206].[dbo].[NumeroFaturasAnoAnterior]
```

4. Lista das peças vendidas no dia anterior;

```
SELECT * FROM [Grupo_206].[dbo].[NumeroFaturasDiaAnterior]
```

5. Lista dos funcionários por ordem decrescente do valor de vendas do mês anterior;

```
SELECT * FROM [Grupo_206].[dbo].[MelhoresFuncionariosMesAnterior]
```

6. Qual a peça mais vendida;

```
SELECT * FROM [Grupo_206].[dbo].[PecaMaisVendida]
```

7. Qual o valor de peças defeituosas identificadas numa determinada semana;

```
SELECT [dbo].[NumeroPecasDefeituosasIdentificadasNaSemana] (CAST('2023-06-12'
AS DATE))
```

8. Lista das peças vendidas no dia anterior;

```
SELECT * FROM [Grupo_206].[dbo].[ PecasVendidasDiaAnterior]
```

3. Conclusões e Trabalho Futuro

Com base no enunciado, acreditamos que conseguimos atingir os objetivos propostos, utilizando os conhecimentos adquiridos durante as aulas. Durante o desenvolvimento do projeto, também pudemos aprender e aprimorar nossas competências técnicas, e, com base nisso, consideramos que obtivemos sucesso na conclusão do trabalho.

Pontos fortes:

- Utilização de Views e Stored Procedures para facilitar pesquisas e inserções para os utilizadores;
- Implementação de Triggers e restrições de integridade para manter a consistência dos dados na BD.

Pontos fracos:

- Possíveis bugs;
- Restrições não pensadas;
- Falta de automatização do pagamento.

Trabalho futuro:

- Validar o stock do produto com o quando adicionado um novo pedido;
- Automatizar pagamento.

Tendo em consideração todo o trabalho realizado e as melhorias propostas, acreditamos que merecemos uma nota de 15 para o projeto.

Bibliografia

[01] Connolly, Thomas M. and Carolyn E. Begg, "11", in Database Systems: A Practical Approach to Design, Person, Inglaterra, 2015, pp. 375-404

[02] Connolly, Thomas M. and Carolyn E. Begg, "12", in Database Systems: A Practical Approach to Design, Person, Inglaterra, 2015, pp. 405-432

[03] Connolly, Thomas M. and Carolyn E. Begg, "16", in Database Systems: A Practical Approach to Design, Person, Inglaterra, 2015, pp. 503-526

[04] Connolly, Thomas M. and Carolyn E. Begg, "17", in Database Systems: A Practical Approach to Design, Person, Inglaterra, 2015, pp. 527-560

Referências WWW

[01] **www. ieeeauthorcenter.ieee.org/wp-content/uploads/IEEE-Reference-Guide.pdf**
Página que contem as normas de com realizar bibliografias.

[02] **www.stackoverflow.com**
Forum duvidas e repostas de programação, BD, etc ...

[03] **www.moodle2.estg.ipp.pt/course/view.php?id=214**
Página da disciplina no moodle.

Lista de Siglas e Acrónimos

BD Base de Dados

SGBD Sistema de Gestão de Base de Dados

UNF Forma Não Normalizada

1FN Primeira Forma Normal

2FN Segunda Forma Normal

3FN Terceira Forma Normal