Base de Dados

p5g8

Gestão de uma Empresa de Vinhos

Docentes:

Joaquim Sousa Pinto (carlos.costa@ua.pt)

Carlos Costa (jsp@ua.pt)

Pedro Sobral, 98491

Daniel Figueiredo, 98498

24 de junho de 2021

Índice

1 - Introdução 3
2 - Análise de Requisitos 4
2.1 - Entidades 4
3 - Conceptualização da base de dados 5
3.1 - Diagrama ER 6
3.2 - Esquema Entidade - Relação 7
3.3 - Diagrama da Base de Dados 8
4 - Construção da Base de Dados 8
4.1 - Criação das Tabelas 9
4.2 - Inserção de dados nas tabelas 10
5 - SQL Programming 11
5.1 - Stored Procedures 11
5.2 - UDF's 12
5.3 - Triggers 13
5.4 - Views 16
5.5 - Cursor 17
5.6 - Indexes 18
6 - Interface Gráfica 19
7 - Segurança 21
8 - Vídeo 21
6 - Conclusão 22
7 - Bibliografia 23

1 - Introdução

No âmbito da unidade curricular de Base de Dados, foi-nos proposta a realização deste trabalho prático, cujo tema recai sobre a gestão de uma empresa de vinhos.

A base de dados criada incide, portanto, na gestão das adegas, cubas, terrenos, funcionários, e maioritariamente mais entendidas importantes e que se interligam num processo deste tipo.

O desenvolvimento deste trabalho prático tem como objetivo a aplicação prática dos conhecimentos adquiridos nas aulas teóricas e práticas desta unidade curricular em toda a sua plenitude, desde o pensamento e desenho do modelo da base de dados à sua gestão e manipulação por sistemas de software.

2 - Análise de Requisitos

Considerando um sistema de gestão de uma empresa de vinhos, tendo em consideração o processo de produção, armazenamento e venda dos mesmos.

2.1 - Entidades

Pessoa - Existem vários tipos de pessoas na nossa plataforma, sendo elas: Funcionário, Gerente, Operador de Adega, Operador Agrícola, Cliente.
A pessoa é caracterizada por um nome, um NIF, data de nascimento, morada, número de telemóvel e género.

Cliente - Um cliente, pode comprar vinho, e é apenas caracterizado pelo seu NIF.

Funcionário - Um funcionário representa um trabalhador no sistema. Está encarregue de efetuar todas as operações relativas à empresa de vinhos que trabalha, podendo este funcionário ser gerente de adega, operador de adega e operador agrícola, sendo estes caracterizados por um IBAN, número de segurança social e a data de início de atividade na empresa.

Gerente - Um gerente trabalha fora dos trabalhos mais pesados e tal como o nome indica, este gere todo o sistema, sendo este definido com um número de funcionário.

Operador de Adega - Um funcionário que como o nome indica trabalha nas adegas e tem associado um número de funcionário e está associado, também, à adega onde trabalha.

Operador Agrícola - Um funcionário que trabalha nos terrenos da empresa, e que possui também um número de funcionário.

Adega - A Adega contém um nome, um endereço, e um responsável (sendo este um funcionário da empresa), possui também uma capacidade máxima de litros de vinho que pode armazenar dependendo do número de cubas que tem.

Vinho - O vinho é caraterizado por um ID, um nome, o ano de produção e um DOC. O vinho será armazenado numa cuba e tem associado uma casta.

Casta - A casta do vinho tem associada um ID e um nome

Cuba - Uma cuba é uma estrutura que armazena o vinho, tendo uma capacidade máxima (litros), um ID, o tipo de cuba que pode ser - metal refrigerado, metal não refrigerado, madeira, etc.

Terreno - Um terreno, é uma parcela de terra que tem vinha plantada, sendo essa parcela caracterizada pelo ano de plantação e os hectares da vinha, também possui uma localização. A este terreno encontra-se também associada a casta das uvas. Destes terrenos saem as uvas que irão ser usadas para produzir os vinhos na adega.

Armazém - Existe também um armazém, que é caracterizado por um nome, uma localização. O armazém recebe os vinhos engarrafados da adega.

Tipo de Trabalho - O tipo de trabalho é referente ao operador agrícola e refere-se aos tipos de trabalho que este pode exercer nos terrenos que trabalha.

Armazenado - A relação "Armazenado", é referente ao sítio onde os vinhos se encontram guardados (armazéns).

3 - Conceptualização da base de dados

Durante o processo de conceptualização da base de dados o desenho da mesma sofreu algumas alterações, quer por particularidades que nos apercebemos que fariam mais sentido de certa maneira, quer também para tornar todo o sistema mais robusto e mais próximo da realidade. Sendo este um relatório final, vamos apresentar os diagramas finais da nossa base de dados.

NOTA: dado o tamanho de alguns diagramas que serão mostrados nas próximas páginas, poderá não ser fácil entender algumas entidades, assim sendo, as figuras com os diagramas encontram-se na pasta "Diagrams".

3.1 - Diagrama ER

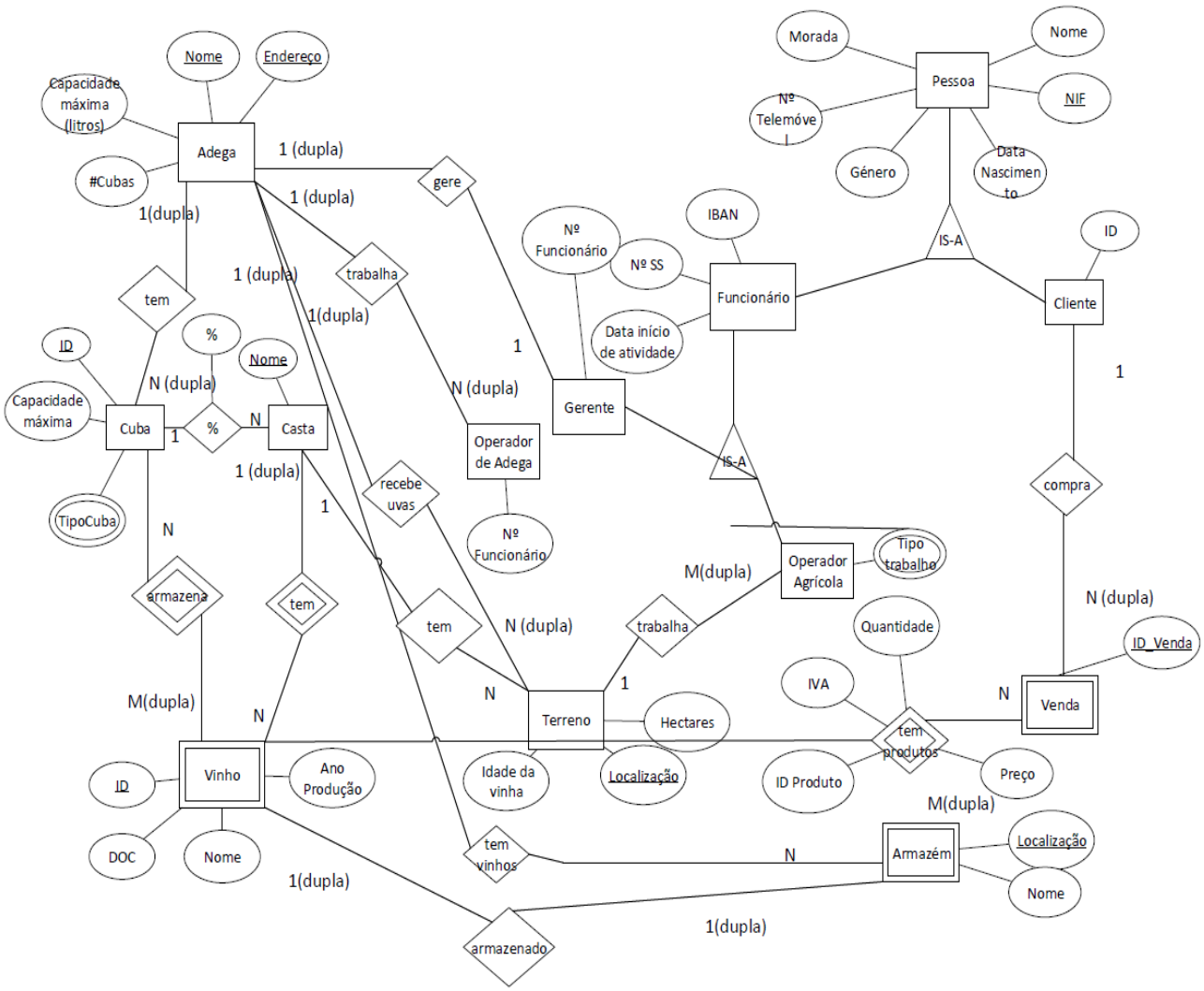

Figura 1 - Diagrama ER

3.2 - Esquema Entidade - Relação

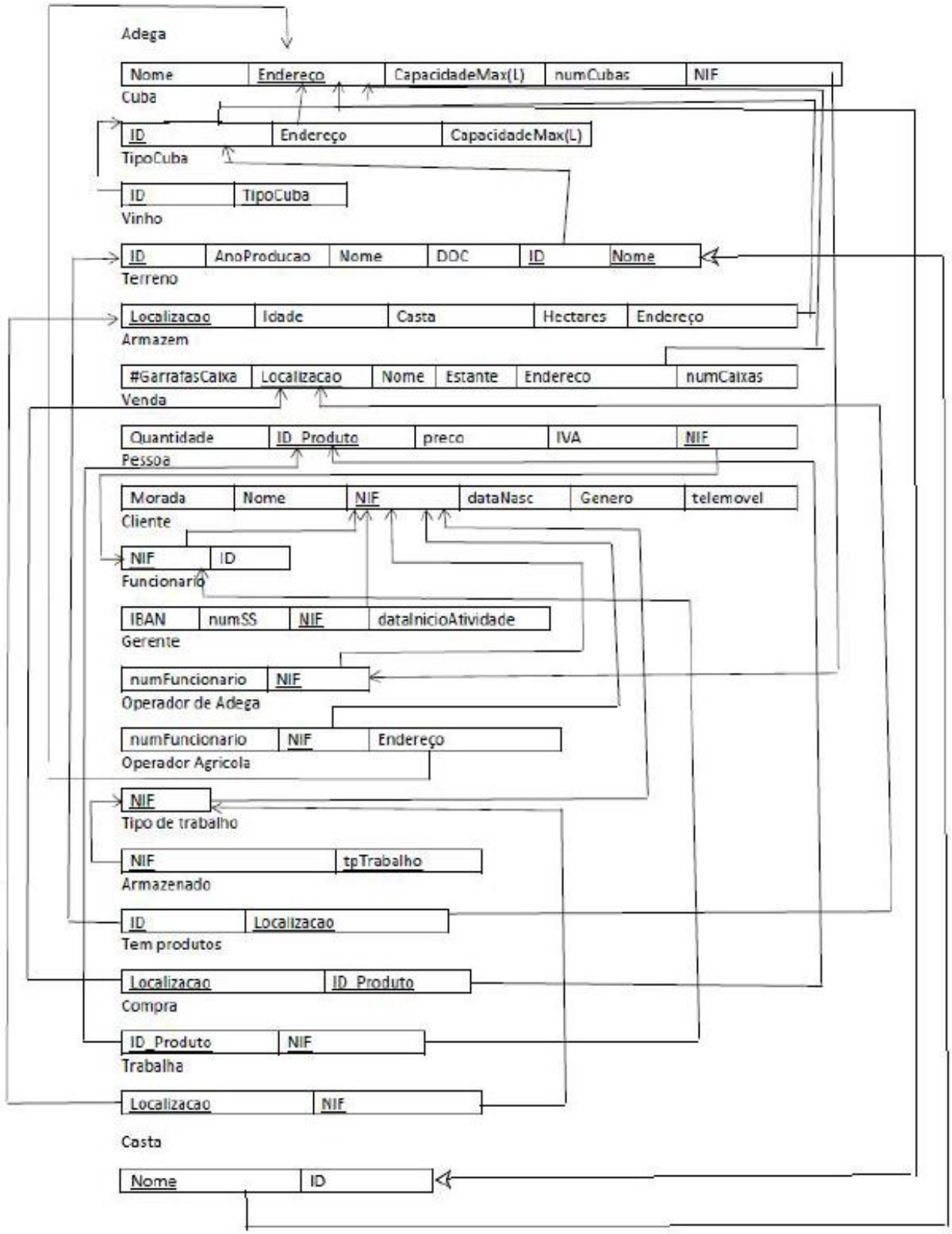

Figura 2 - Modelo Entidade - Relação

3.3 - Diagrama da Base de Dados

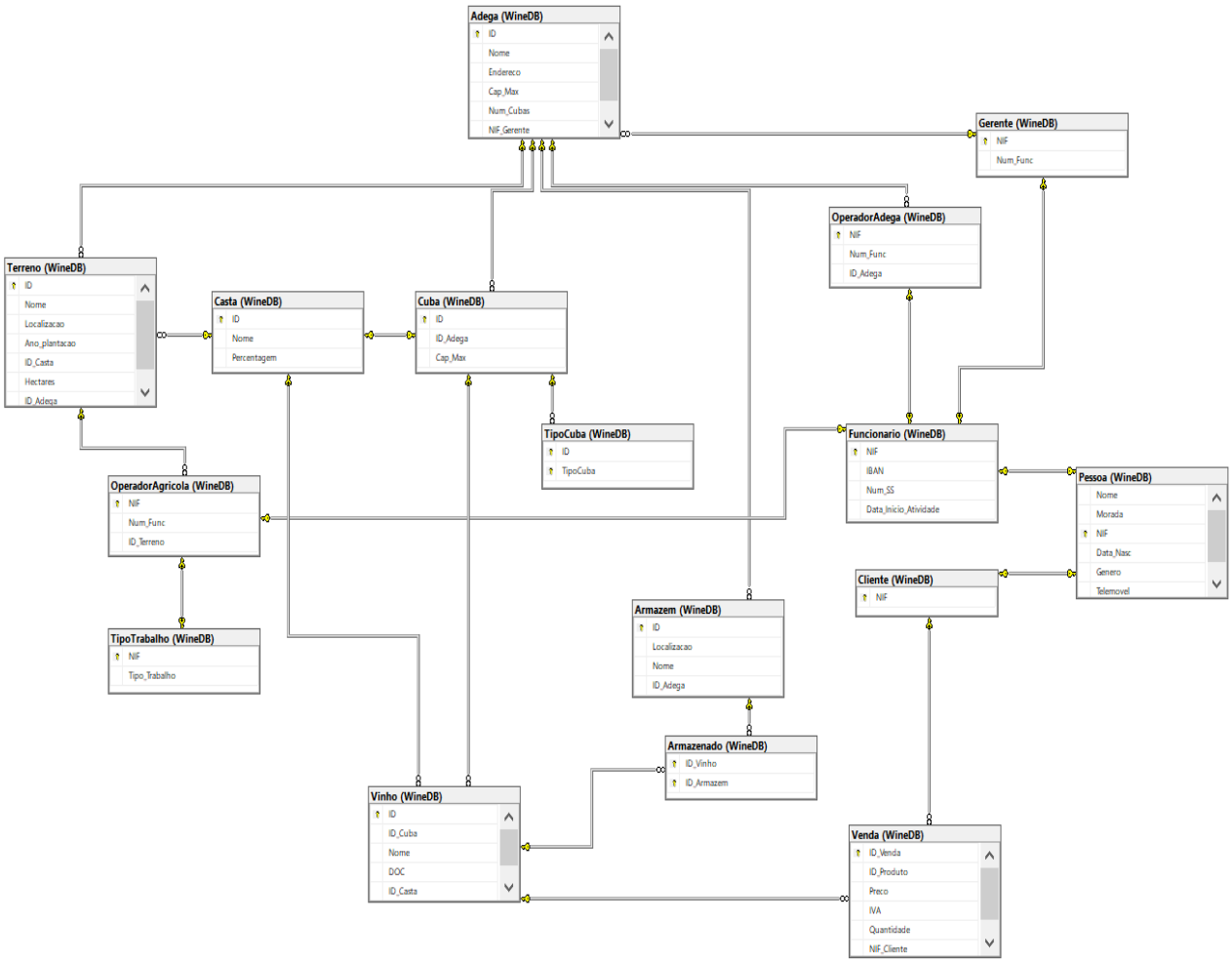

Figura 3 - Diagrama da Base de Dados

Este diagrama representa as ligações entre as entidades, sendo o mesmo gerado pelo SQL Server, após a criação das tabelas na base de dados.

4 - Construção da Base de Dados

A base de dados foi implementada em SQL, linguagem que já nos era familiar pois foi utilizada durante as aulas práticas e lecionada nas aulas teóricas. No que toca à criação da base de dados podemos separar esta em 2 fases distintas, criação das tabelas (Data Definition Language), as tabelas podem ser consultadas no ficheiro DDL.sql, e inserção de

dados nas tabelas (Data Manipulation Language), as inserções podem ser consultadas no ficheiro DML.sql.

4.1 - Criação das Tabelas

De modo a definirmos as entidades referidas no ponto 2 deste relatório, usamos DDL para criar as tabelas com os atributos correspondentes a cada uma. Durante este processo, fomos definindo algumas restrições de certos atributos de modo a que determinadas necessidades da nossa implementação fossem cumpridas.

As seguintes figuras, são exemplificativas de algumas tabelas implementadas.

```
CREATE TABLE WineDB.Armazem (
    ID                  VARCHAR(5)      NOT NULL,
    Localizacao         VARCHAR(256)    NOT NULL,
    Nome                VARCHAR(256)    NOT NULL,
    ID_Adega            VARCHAR(5)      NOT NULL,

    PRIMARY KEY (ID),
    FOREIGN KEY (ID_Adega) REFERENCES WineDB.Adega(ID) ON UPDATE CASCADE,
);
GO
```

Figura 4 - Criação da Tabela "Armazem"

```
CREATE TABLE WineDB.Pessoa (
    Nome        VARCHAR(256)    NOT NULL,
    Morada      VARCHAR(256)    NOT NULL,
    NIF         INT             NOT NULL,
    Data_Nasc   VARCHAR(10)     NOT NULL,
    Genero      VARCHAR(1),
    Telemovel   VARCHAR(9),

    PRIMARY KEY (NIF)
);
GO
```

Figura 5 - Criação da Tabela "Pessoa"

```
CREATE TABLE WineDB.Terreno (
    ID              VARCHAR(5)      NOT NULL,
    Nome            VARCHAR(256)    NOT NULL,
    Localizacao     VARCHAR(256)    NOT NULL,
    Ano_plantacao   VARCHAR(4),
    ID_Casta        INT,
    Hectares        FLOAT           NOT NULL,
    ID_Adega        VARCHAR(5)      NOT NULL,

    PRIMARY KEY (ID),
    FOREIGN KEY (ID_Adega) REFERENCES WineDB.Adega(ID),
    FOREIGN KEY (ID_Casta) REFERENCES WineDB.Casta(ID),
);
GO
```

Figura 6 - Criação da Tabela "Terreno"

4.2 - Inserção de dados nas tabelas

Para a inserção de dados na base de dados recorremos à DDL, deste modo conseguimos preencher as tabelas criadas no ponto anterior. À medida que íamos inserindo os dados executamos algumas *queries* simples para perceber como a inserção dos dados nas tabelas estava a correr.

```
INSERT INTO WineDB.Adega(ID,Nome,Endereco,Cap_Max,Num_Cubas, NIF_Gerente) VALUES
('6A3E4','Solar Dona Maria','Rua de Cima, 234, Ervedosa do Douro',350000,38, '237489547'),
('54W3T','Casa dos Vinhos','Estrada Municipal 501, 12, Soutelo do Douro',1250000,80, '233765487'),
('03ED5','Boa Uva','Travessa da Laranja, 12, Anadia',250000,32, '257483675'),
('6S4U3','Adega La Rose','Estrada Nacional 10, 123, Azeit◆o',900000,75, '374985902');
```

Figura 7 - Inserção de dados na Tabela "Adega"

```
INSERT INTO WineDB.Vinho(ID, ID_Cuba, Nome, DOC, ID_Casta) VALUES
('AS3FR', 12452,'Vinho Velho', 'Douro', 12452),
('SF231', 23411,'Vinho Dão', 'DãO', 16556),
('FGDF3', 52143,'Vinho Alegre', 'Alenquer', 23411),
('GDDF3', 47581,'Vale Dona Maria', 'Douro',52143 ),
('AG434', 17543,'Vinho Velho - 2017', 'Tejo',47581 ),
('GFH54', 54432,'VZ', 'Douro', 19876),
('H6G34', 34234,'VZ', 'Douro', 96755),
('G5325', 43633,'Bajancas', 'Douro', 86565),
('A34R2', 46423,'Bajancas', 'Tejo', 17543),
```

Figura 8 - Inserção de dados na Tabela "Vinho"

5 - SQL Programming

5.1 - Stored Procedures

As Stored Procedures tratam-se de conjuntos de instruções batch, compiladas pelo SQL-Server, que têm um nome associado.

Dado que não precisam de ser recompiladas cada vez que são invocadas (isto é, quando são invocadas pela primeira vez, são carregadas e guardadas em memória cache), apresentam um conjunto de mais valias bastante atrativas para o nosso projeto.

Apresenta-se, de seguida, um exemplo de Stored Procedures.

```
CREATE PROCEDURE WineDB.AdicionarCliente (@Nome VARCHAR(256), @Morada VARCHAR(256), @NIF INT, @Data_Nasc DATE, @Genero VARCHAR(1), @Telemovel VARCHAR(9))
AS
	BEGIN
		DECLARE @count INT;
		DECLARE @erro VARCHAR(100);
		SET @count = (SELECT WineDB.checkIfNIFExists(@NIF))
		IF(@count>=1)
			RAISERROR ('O NIF introduzido já existe, não é possível adicionar o Cliente', 16,1);
		ELSE
			BEGIN
				BEGIN TRY
					BEGIN TRAN
						INSERT INTO WineDB.Pessoa (Nome, Morada, NIF, Data_Nasc, Genero, Telemovel) VALUES (@Nome, @Morada, @NIF, @Data
						INSERT INTO WineDB.Cliente (NIF) VALUES (@NIF)
					COMMIT TRAN
				END TRY
				BEGIN CATCH
					Rollback TRAN
					SELECT @erro = ERROR_MESSAGE();
					SET @erro =  'O Cliente não foi inserido, algum valor inserido incorretamento'
					RAISERROR (@erro, 16,1);
				END CATCH
			END
	End
GO
```

Figura 9 - Stored Procedure para Adicionar um Cliente

5.2 - UDF's

As User Defined Functions apresentam os mesmos benefícios que os Stored Procedures, em termos de compilação e otimização de execução.

De um modo geral, utilizámos as UDF's sempre que queríamos verificar se algum atributo já se encontrava definido em alguma tabela, ou para retornar um valor de uma entidade específica.

Os exemplos seguintes são referentes à aplicação de cada um dos tipos de UDF que foram aplicados no projeto.

```
CREATE FUNCTION WineDB.checkIfNIFExists (@NIF INT) RETURNS INT
AS
        BEGIN
                DECLARE @counter INT
                SELECT @counter=COUNT(1) FROM WineDB.Pessoa WHERE NIF=@NIF
                RETURN @counter
        END
GO
```

Figura 10 - UDF para verificar se um NIF existe

```
CREATE FUNCTION WineDB.getNIFfromNome (@Nome VARCHAR(256)) RETURNS INT
AS
        BEGIN
                DECLARE @nif INT
                SELECT @nif = NIF FROM WineDB.Pessoa WHERE Nome = @nome
                RETURN @nif
        END
GO
```

Figura 11 - UDF que retorna o NIF pelo nome de uma Pessoa

```
CREATE FUNCTION WineDB.checkIfNum_FuncExists (@Num_Func INT) RETURNS INT
AS
	BEGIN
		DECLARE @NIFPessoa INT
		DECLARE @counter INT
		DECLARE @counterPessoa INT
		DECLARE @counterGerente INT
		DECLARE @counterOpAdega INT
		DECLARE @counterOpAgricola INT

		--Check if Pessoa exists
		SELECT @NIFPessoa = WineDB.getNIFFuncfromNum_Func(@Num_Func)

		IF @NIFPessoa IS NOT NULL
			SET @counterPessoa=(SELECT WineDB.checkIfNIFExists(@NIFPessoa))

		IF(@counterPessoa IS NOT NULL)
			--Check if gerente exists
			SELECT @counterGerente = COUNT(1) FROM WineDB.Gerente WHERE Num_Func = @Num_Func
			--Check if Operador Adega exists
			SELECT @counterOpAdega = COUNT(1) FROM WineDB.OperadorAdega WHERE Num_Func = @Num_Func
			--Check if Operador Agricola exists
			SELECT @counterOpAgricola = COUNT(1) FROM WineDB.OperadorAgricola WHERE Num_Func = @Num_Func

		IF(@counterGerente = 1 OR @counterOpAdega = 1 OR @counterOpAgricola = 1)
			SET @counter = 1
		ELSE
			SET @counter = 0

		RETURN @counter
	END
GO
```

Figura 12 - UDF que verifica se um Nº de Funcionário já existe

5.3 - Triggers

Os Triggers são representativos de um tipo especial de Stored Procedure, que são apenas executados em determinados eventos associados à manipulação de dados – isto é, quando uma das ações previstas ocorre, os Triggers são ativados.

Na nossa base de dados, apenas utilizámos triggers do tipo “after insert, update”, ou seja, eram ativadas, caso ocorresse algum “insert” ou “update” na tabela associada ao trigger.

Os exemplos apresentados a seguir, são referentes à inserção ou atualização de dados nas tabelas Pessoa e Funcionário.

```
CREATE TRIGGER WineDB.addPersona ON WineDB.Pessoa
AFTER INSERT, UPDATE
AS
        SET NOCOUNT ON;
        DECLARE @total AS int;
        DECLARE @totalTelemovel AS int;
        DECLARE @NIFPerson AS int;
        DECLARE @numTele AS int;
        SELECT @NIFPerson = NIF FROM INSERTED;
        SELECT @numTele = Telemovel FROM INSERTED;

        IF LEN(@NIFPerson) <> 9
        BEGIN
                RAISERROR('NIF tem de ter 9 números!' , 16, 1);
                ROLLBACK TRAN;
        END

        IF LEN(@numTele) <> 9
        BEGIN
                RAISERROR('Número de Telemovel tem de ter 9 números!' , 16, 1);
                ROLLBACK TRAN;
        END

        SELECT @totalTelemovel = count(*) FROM WineDB.Pessoa WHERE Telemovel = @numTele
        IF @totalTelemovel > 1
        BEGIN
                RAISERROR('Número de telemovel repetido na base de dados!' , 16, 1);
                ROLLBACK TRAN;
        END

        SELECT @total = count(*) FROM WineDB.Pessoa where NIF = @NIFPerson;
        IF @total > 1
        BEGIN
                RAISERROR('NIF repetido na base de dados!' , 16, 1);
                ROLLBACK TRAN;
        END
GO
```

Figura 13 - Trigger que valida na adição de uma pessoa à Base de Dados

```
CREATE TRIGGER WineDB.addFunc ON WineDB.Funcionario
AFTER INSERT, UPDATE
AS
        SET NOCOUNT ON;
        DECLARE @IBANFunc AS VARCHAR(25);
        DECLARE @SSFunc AS int;
        DECLARE @IBANTotal AS int;
        DECLARE @SSTotal AS int;
        SELECT @IBANFunc = IBAN FROM INSERTED;
        SELECT @SSFunc = Num_SS FROM INSERTED;

        IF NOT @IBANFunc LIKE 'PT%'
        BEGIN
                RAISERROR('IBAN mal inserido!' , 16, 1);
                ROLLBACK TRAN;
        END

        IF LEN(@IBANFunc) <> 25
        BEGIN
                RAISERROR('IBAN mal inserido!' , 16, 1);
                ROLLBACK TRAN;
        END

        SELECT @IBANTotal = count(*) FROM WineDB.Funcionario WHERE IBAN = @IBANFunc
        IF @IBANTotal > 1
        BEGIN
                RAISERROR('IBAN repetido na base de dados!' , 16, 1);
                ROLLBACK TRAN;
        END

        SELECT @SSTotal = count(*) FROM WineDB.Funcionario WHERE Num_SS = @SSFunc
        IF @SSTotal > 1
        BEGIN
                RAISERROR('Número de SS repetido na base de dados!' , 16, 1);
                ROLLBACK TRAN;
        END

        IF LEN(@SSFunc) <> 8
        BEGIN
                RAISERROR('Número de SS mal inserido!' , 16, 1);
                ROLLBACK TRAN;
        END

GO
```

Figura 14 - Trigger que valida a inserção de um novo Funcionario

5.4 - Views

Uma view pode ser utilizada como fonte de dados (similar a uma tabela normal) num conjunto de operações SQL.

Para facilitar a consulta por parte das interfaces decidimos implementar várias views de forma a que a interface não tivesse de executar uma query tão extensa.

Em seguida, seguem-se exemplos das views criadas.

```
CREATE VIEW WineDB.ViewOperadorAgricola AS
SELECT P.Nome, P.Morada, P.NIF, P.Data_Nasc, P.Genero, P.Telemovel, F.IBAN, F.Num_SS, F.Data_Inicio_Atividade, OA.Num_Func, OA.ID_Terreno, TT.Tipo_Trabalho
FROM WineDB.Pessoa AS P JOIN WineDB.Funcionario AS F ON P.NIF = F.NIF
JOIN WineDB.OperadorAgricola AS OA ON F.NIF = OA.NIF
JOIN WineDB.TipoTrabalho AS TT ON TT.NIF = OA.NIF
```

Figura 15 - View de Operador Agricola

```
CREATE VIEW WineDB.ViewOperadorAdega AS
SELECT P.Nome, P.Morada, P.NIF, P.Data_Nasc, P.Genero, P.Telemovel, F.IBAN, F.Num_SS, F.Data_Inicio_Atividade, OA.Num_Func, A.Nome AS Nome_Adega
FROM WineDB.Pessoa AS P JOIN WineDB.Funcionario AS F ON P.NIF = F.NIF
JOIN WineDB.OperadorAdega AS OA ON F.NIF = OA.NIF
JOIN WineDB.Adega AS A ON A.ID = OA.ID_Adega
```

Figura 16 - View de Operador da Adega

5.5 - Cursor

Os cursores servem para iterar por todos os tuplos referentes a uma tabela e fazer as operações necessárias.

Deste modo, decidimos utilizar um cursor, para cada vez que fosse adicionada uma nova venda, iria alterar a quantidade de vinho em stock e caso não houvesse a quantidade desejada seria informado que não era possível fazer essa venda.

Segue-se o exemplo da utilização deste cursor.

```
CREATE PROCEDURE WineDB.AdicionarVenda (@ID_Produto VARCHAR(5), @Preco FLOAT , @IVA INT = 23, @Quantidade INT, @Nome VARCHAR(256))
AS
	BEGIN

		DECLARE @countVinho INT;
		DECLARE @nif INT;
		DECLARE @countNIF INT;
		DECLARE @erro VARCHAR(100);
		DECLARE @quantity INT;
		DECLARE @IDVinho VARCHAR(5);
		SET @nif = (SELECT WineDB.getNIFfromNome(@Nome))
		SET @countNIF = (SELECT WineDB.checkIfNIFExists(@nif))
		SET @countVinho = (SELECT WineDB.checkIfWineExists(@ID_Produto))

		IF (@countVinho < 1)
			RAISERROR ('N@o existe esse vinho em armaz@m.', 16,1);
		ELSE IF (@countNIF < 1)
			RAISERROR ('N@o existe esse cliente na base de dados.', 16,1);
		ELSE
			BEGIN
				DECLARE WineCursor CURSOR FAST_FORWARD
					FOR SELECT ID , Quantidade FROM WineDB.Vinho WHERE ID = @ID_Produto;
				OPEN WineCursor;
				FETCH WineCursor INTO @IDVinho , @quantity;
					WHILE @@FETCH_STATUS = 0
						BEGIN
							IF @quantity >= @Quantidade
								BEGIN
									UPDATE WineDB.Vinho SET Quantidade = Quantidade - @Quantidade WHERE ID = @ID_Produto
									INSERT INTO WineDB.Venda(ID_Produto, Preco, IVA, Quantidade, NIF_Cliente) VALUES (@ID_Produto, @Preco, @IVA, @
								END
							ELSE
								BEGIN
									RAISERROR ('Vinho nao disponivel', 16,1);
								END
							FETCH WineCursor INTO @IDVinho , @quantity
						END
				CLOSE WineCursor;
				DEALLOCATE WineCursor;
				RETURN ;

			END
	End
GO
```

Figura 17 - Cursor implementado num SP que verifica se é possível fazer Venda

5.6 - Indexes

Apesar de se tratar de uma base de dados relativamente pequena decidimos fazer uso da utilização de indexes. Deste modo, fizemos uso da sua implementação na tabela Pessoa e Vinho, quando tentamos pesquisar pelo nome e NIF da primeira e pelo nome e ID da segunda, visto estas serem as nossas tabelas de maior extensão.

```
CREATE INDEX searchNomePessoa
ON WineDB.Pessoa (Nome)
GO

CREATE INDEX searchNIFPessoa
ON WineDB.Pessoa (NIF)
GO

CREATE INDEX searchNomeVinho
ON WineDB.Vinho (Nome)
GO

CREATE INDEX searchIDVinho
ON WineDB.Vinho (ID)
GO
```

Figura 18 - Indexes implementados nas tabelas Pessoa e Vinho

6 - Interface Gráfica

A interface gráfica foi desenvolvida no Visual Studio, recorrendo ao Windows Forms, sendo o código desenvolvido inteiramente em C#.
De modo a mostrar a informação presente na base de dados de forma simples e bastante percetível, usamos *ListView*, conseguimos assim organizar a informação por colunas, sendo que esta pode ser ordenada por atributo (coluna), o que se torna uma mais valia para agrupar dados, ou pesquisar por eles por uma ordem.

O *layout* de cada formulário é bastante semelhante, há uma *ListView* para mostrar a informação referente ao mesmo, no campo superior do formulário há uma pesquisa implementada, com recurso a uma *comboBox* temos implementada a possibilidade de pesquisar por categoria, e na parte mais inferior do *form*, os campos que serão usados para a inserção e eliminação de informação na base de dados. Há também um botão no canto superior esquerdo que permite voltar ao menu principal. Os formulários não variam muito desse *layout* base, no entanto há alguns com pequenas alterações.

Durante a inserção, atualização, e eliminação de dados na interface gráfica, são verificados erros, que serão mostrados com recurso a uma *MessageBox,* dando assim a informação ao utilizador que algo anómalo ocorreu. Em contrapartida, quando algo ocorre conforme o esperado, por exemplo, foi inserido com sucesso uma *Adega* à base de dados, o utilizador também recebe uma notificação via *MessageBox* de que a sua ação foi executada com sucesso.

Seguem-se algumas imagens demonstrativas da interface gráfica.

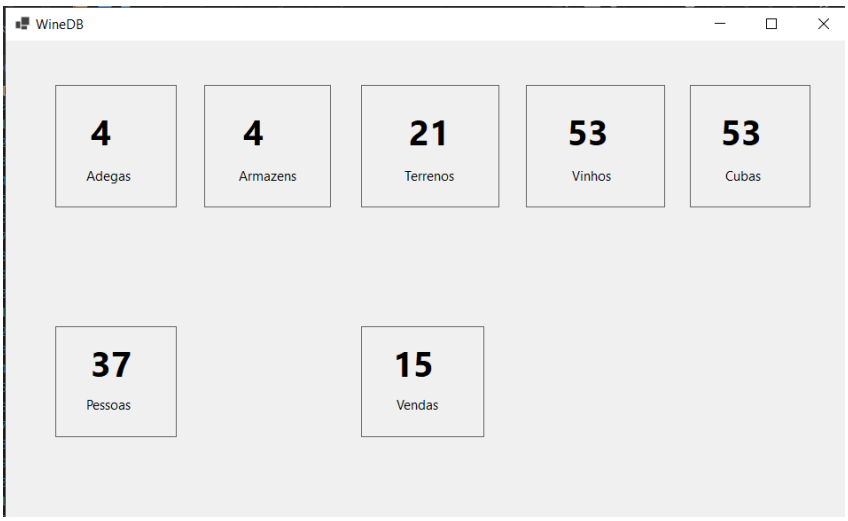

Figura 19 - Menu inicial da Interface gráfica

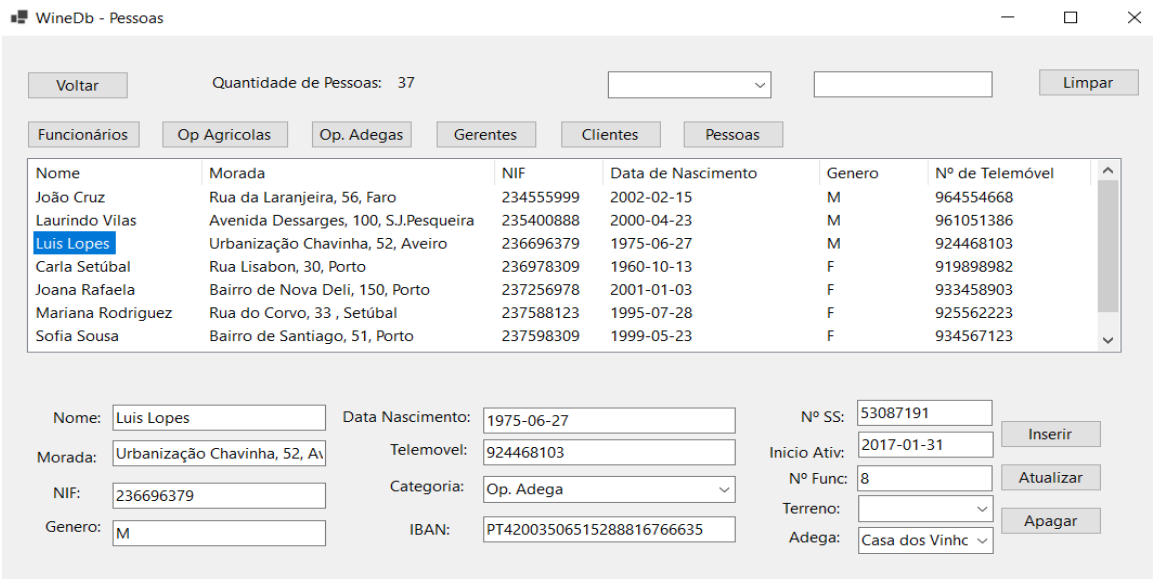

Figura 20 - Form Pessoas

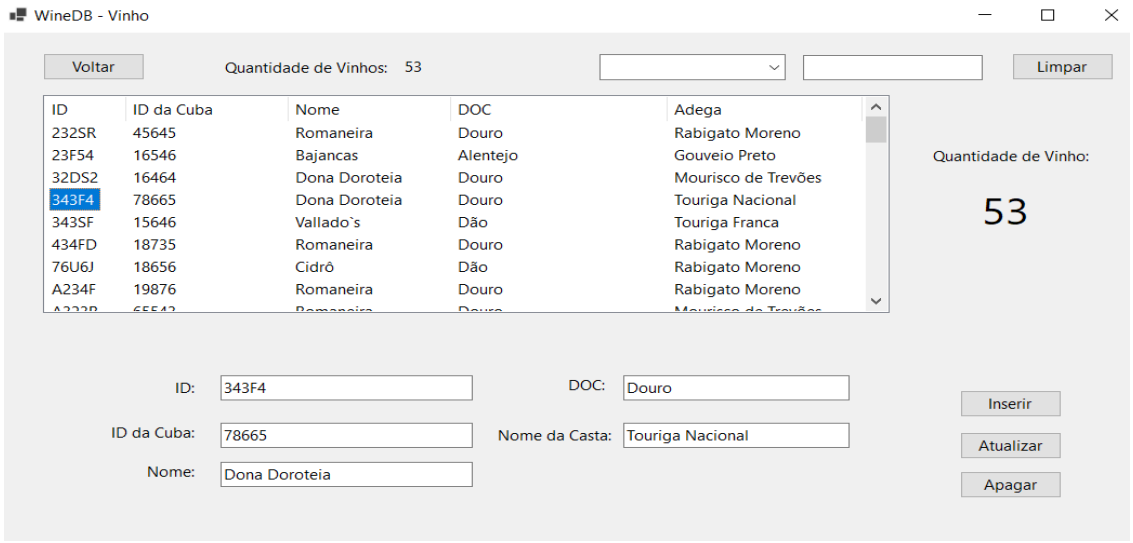

Figura 21 - Form Vinho

7 - Segurança

De modo a tentar minimizar possíveis vulnerabilidades da base de dados a ataques de *SQL Injection*. De modo a prevenir estes problemas, na implementação da base de dados foram tidos em conta os seguintes pontos:

- Verificações dos dados inseridos pelos utilizadores, (através de triggers)
- Tentamos ao máximo utilizar *SQL Parametrizado* ou *Stored Procedures* , em vez de recorrermos ao *SQL Dinâmico.*
- São apresentadas mensagens de erro, ao utilizador de modo a que este perceba que introduzir valores ou informações erradas.

8 - Vídeo

No link que se segue, é possível observar um exemplo demonstrativo das principais funcionalidades da interface gráfica, que foram descritas anteriormente.

- https://youtu.be/zzGJN0dj9f4

6 - Conclusão

Sendo neste ponto notório que a base de dados se encontra em funcionamento, podemos considerar que a mesma foi implementada de forma correta e a mais próxima da realidade possível, sendo que em algumas situações tivemos de simplificar algumas complexidades pois só estariam a complicar toda a implementação, sendo que não eram relevantes o suficiente para o contexto do tema por nós proposto.

Com a implementação da interface gráfica, conseguimos de uma maneira visual e apelativa ver toda a estrutura a funcionar corretamente, sendo então concretizado o objetivo inicial de uma implementação correta de um sistema de gestão de adegas e todos os seus subsequentes.

7 - Bibliografia

Slides disponibilizados na página do eLearning@UA da Unidade Curricular.
https://docs.microsoft.com/en-us/dotnet/csharp/