



universidade  
de aveiro

---

## RELATÓRIO DO PROJETO

*Gestor de Supermercados*

---

**Preparado por:** P2G7: Francisca Barros (nº 93102)  
& Wei Ye (nº 93442)

**Disciplina:** Base de Dados

**Docente:** Carlos Costa  
Joaquim Sousa Pinto

Aveiro, 9 de junho de 2020

# Índice

<b>Introdução .....</b>	<b>3</b>
<b>1. Análise de Requisitos .....</b>	<b>4</b>
1.1 Entidades .....	4
<b>2. Conceptualização .....</b>	<b>7</b>
2.1 Diagrama Entidade-Relação .....	7
2.2 Esquema do Modelo Relacional .....	8
<b>3. Normalização .....</b>	<b>9</b>
<b>4. Construção da Base de Dados .....</b>	<b>10</b>
4.1 Data Definition Language (DDL) .....	10
4.2 Data Manipulation Language (DML) .....	11
<b>5. Stored Procedures .....</b>	<b>12</b>
<b>6. User Defined Functions .....</b>	<b>14</b>
<b>7. Triggers .....</b>	<b>15</b>
<b>8. Aspetos de Segurança .....</b>	<b>16</b>
<b>9. Componentes Não Implementadas .....</b>	<b>17</b>
9.1 Indexes .....	17
<b>10. Utilização .....</b>	<b>18</b>
10.1 Vídeo Demonstrativo .....	19
<b>Conclusão .....</b>	<b>20</b>
<b>Material de Referência .....</b>	<b>21</b>

## Introdução

O presente relatório, inserido na unidade curricular de Base de Dados, tem como principal objetivo descrever o processo de desenvolvimento do projeto apresentado, cujo tema recai sobre a Gestão de um Supermercado.

A situação atual que o mundo atravessa, devido à pandemia causada pelo novo coronavírus, levou a uma mudança de paradigma em diversas vertentes. O comércio online de bens alimentícios e essenciais, ainda que representativo de um ramo de nicho, mas promissor, cresceu exponencialmente – o medo de sair de casa aliado às ordens de recolhimento contribuíram para este desenvolvimento. Face aos estudos efetuados até à data, espera-se que exista um crescimento de cerca de 40% neste setor.

Face à mudança, torna-se essencial assegurar a capacidade de resposta destas plataformas, por forma a manter o ritmo de crescimento expectado e responder à procura. Assim, a construção de ferramentas que facilitem todo o processo envolvido numa venda é fundamental.

Deste modo, as bases de dados são de elevada importância, essenciais ao funcionamento de forma eficiente de uma plataforma que ofereça serviços de venda online. A gestão de clientes, funcionários que operam na empresa, criação de encomendas, processo de faturação e expedição estabelece uma relação íntima com o armazenamento correto de informação.

Portanto, tendo em conta o que foi exposto anteriormente, o nosso projeto tem como propósito tornar o processo de gestão de um supermercado online o mais ágil e acessível possível.

# 1. Análise de Requisitos

Sendo que o principal objetivo da aplicação desenvolvida é gerir as funcionalidades de um supermercado online, a transação principal ocorre entre um cliente e o administrador trata-se da efetivação de uma encomenda, bem como da emissão da fatura correspondente após expedição.

De seguida, é definida sucintamente a caracterização das entidades envolvidas.

## 1.1 Entidades

### **Pessoa**

Existem vários tipos de Pessoa na plataforma: o Utilizador, o Cliente, o Condutor e o Funcionário. Uma pessoa insere-se, obrigatoriamente, num dos quatro tipos anteriores.

A Pessoa é caracterizada por: ID (único no sistema), Nome, Email (único no sistema), Telefone (único no sistema), Idade, Salário (por defeito, será -1 quando se trata de um Cliente e Utilizador) e Género.

### **Utilizador**

Uma conta do tipo Utilizador pode efetuar uma encomenda, ver as encomendas efetuadas e as faturas em seu nome. Tem acesso ao sistema fornecido – à secção dos produtos disponíveis, ao carrinho de compras, às encomendas efetuadas e às respetivas faturas.

É descrita por: ID (único no sistema), Nome e Senha (por motivos de segurança, a verdadeira password não é guardada, mas sim um hashing da mesma).

### **Cliente**

Um cliente pode fazer encomendas de produtos, efetuar o seu pagamento e receber a fatura das mesmas. Querendo aproximar o sistema o mais possível da realidade, quando o este não insere NIF, este será por defeito 999999999.

Assim, é definido por: ID (único no sistema), NIF (Número de Identificação Fiscal – utilizado para a emissão de faturas) e por Endereço (utilizado para endereçar da encomenda).

### **Condutor**

Um condutor está encarregue de entregas as encomendas, realizadas no sistema, que lhe forem atribuídas. Irá conduzir um veículo da frota disponibilizada.

O Condutor é caracterizado por: ID (único no sistema), Carta de Condução (única no sistema) e Disponível (para indicar a sua disponibilidade de entrega).

Optamos por colocar todos os condutores como 'disponíveis', por forma a facilitar parte da implementação. Pelos mesmos motivos, considerou-se a Carta de Condução como sendo uma sequência de 9 dígitos.

### **Funcionário**

Um funcionário representa um trabalhador no sistema. Está encarregue de efetivar as encomendas feitas pelo cliente e de emitir as respetivas faturas.

É descrito por um ID, que é único no sistema.

### **Gerente Armazém**

Um Gerente de Armazém está excluído da realização de encomendas. Tal como o nome sugere, este gere todo o sistema, tendo privilégios de administrador do mesmo. Deste modo, pode: ver a listagem de todos os funcionários, condutores e fornecedores com quem trabalha; contratar e/ou despedir funcionários e condutores; adicionar mais fornecedores; gerir a frota de veículos; ver as marcas que estão disponibilizadas aos clientes; gerir todas as encomendas efetuadas (efetuando a sua entrega).

O Gerente de Armazém é definido por: ID (único no sistema).

### **Armazém**

Um armazém armazena os produtos disponibilizados na plataforma.

O Armazém engloba os parâmetros: ID (único no sistema) e Endereço.

### **Armazena**

A relação Armazena permite facilitar a gestão de stock existente de um certo produto. É, portanto, estabelecida a partir do Armazém, do Produto (neste caso, do seu código) e da Quantidade deste último.

### **Iva**

A relação IVA (Imposto Valor Acrescentado) está associada a cada produto, dependendo da sua categoria.

Abrange o Código (único no sistema) e o Imposto (percentagem associada à categoria de iva).

### **Marca**

Uma marca é detentora de vários produtos, podendo estes pertencer, ou não, à mesma categoria.

É caracterizada por: Nome (único no sistema), Compra (unidades adquiridas), Venda (unidades vendidas) e Ganho (decorrente das vendas dos seus produtos).

### **Tipo**

A relação tipo pretende facilitar a categorização de cada produto disponibilizado. Por outro lado, os Fornecedores fornecem produtos de um dado tipo, estando diretamente relacionados com este.

O Tipo é inferido a partir do Nome (único no sistema).

### **Produto**

Todas as encomendas efetuadas na plataforma são compostas por produtos. Dependendo da categoria (tipo) onde se enquadram, irão ter uma certa taxa de IVA associada. O stock destes está depositado num armazém, sendo que o Gestor poderá adquirir mais contactando os Fornecedores.

O Produto é definido por: Código (único no sistema), IvaCódigo, Nome, Preço, Descrição, Disponível, Armazém, Marca e Tipo.

### **Fornecedor**

Um Fornecedor está encarregue de disponibilizar produtos de um certo tipo ao Gestor, que os poderá adquirir.

Desta forma, é caracterizado por: Fax, Email, Nome (único no sistema), NIF (Número de Identificação Fiscal – único no sistema) e Tipo (de produtos que pode oferecer ao Gestor).

### **Fornece**

Como descrito anteriormente, um Fornecedor irá disponibilizar, ao Gestor, Produtos, para que este os possa vender no seu sistema. Assim, esta relação é estabelecida à custa de Produto (neste caso, o seu código identificativo), Fornecedor, Preço de Compra e Quantidade (adquirida).

### **Transporte**

Um Transporte é conduzido por um Condutor na entrega de encomendas que os Clientes efetuaram no sistema. É caracterizado por: ID (único no sistema), Matrícula, (única no sistema), Marca (do veículo), Capacidade, Disponibilidade.

### **Conduz**

Dado que um Condutor irá utilizar um veículo para transportar as encomendas houve a necessidade de administrar quais estão ocupados. Logo, estabeleceu-se a relação entre Condutor e Transporte, que engloba o ID do condutor e o ID do respetivo transporte.

### **Encomenda**

A génese principal da proposta são as encomendas. Os clientes, após confirmarem os itens do carrinho de compras, efetivam o pedido de criação. Podem optar por efetuar o pagamento de diversas formas – Débito Direto, Cartão de Crédito, Transferência Bancária. É necessário um funcionário estar relacionado com a encomenda, por forma a efetivar a sua criação. De seguida, o Gestor atua ao delegar a entrega a um dos seus Condutores. Após esta ser entregue, o cliente poderá pedir a fatura que lhe corresponde.

Assim, uma Encomenda é descrita por: ID (único no sistema), Client (o cliente que a efetua), Funcionário (encarregue pela sua efetivação), Condutor (que a irá entregar), CondiçõesPagamento, Destino, CustoTotal (leva em conta o custo de cada unidade, e as quantidades adquiridas), Estado (entregue ou por entregar), Pago (campo utilizado para verificar se a encomenda já foi paga) e Data (representativa do dia que foi efetuada).

### **Fatura**

Após a encomenda efetuada pelo cliente ter sido entregue, este pode fazer o pedido da fatura associada ao pagamento. Um funcionário será atribuído a este processo, por forma a emitir a fatura pretendida.

Uma fatura abrange, deste modo, ID (único no sistema), Encomenda (a que está associada), Client (que efetuou o pedido), Funcionário (que a emite), CondiçõesPagamento, CustoTotal e Data.

### **Pagamento**

O processamento do pagamento é algo necessário para a emissão da fatura. Assim, o cliente efetua o pagamento de uma dada encomenda, ou parte dela, ficando associado automaticamente à fatura do valor correspondente.

É caracterizado por: ID (único no sistema), Cliente, Fatura (a que está associado), CondiçõesPagamento, CustoTotal e Data.

### **Contem**

Sendo que, por norma, uma encomenda contém mais do que um produto, e em quantidades variáveis, criou-se a entidade Contem, que permite verificar os produtos associados a uma dada encomenda.

Abrange, portanto, Quantidade (do produto), Encomenda, ProdutoCodigo e Preço (representativo da quantidade total adquirida).

## 2. Conceptualização

Na seguinte seção, dividida em dois subcapítulos, pretende-se traduzir os propósitos associados às entidades do nosso projeto para um design adequado aos conceitos relacionados com bases de dados.

No decorrer do projeto ocorreram algumas alterações, uma vez que muitas das ideias iniciais rapidamente foram descartadas dada a complexidade e/ou impossibilidade de concretização.

O diagrama entidade-relação (DER) e a esquematização do modelo relacional permitirão uma compreensão mais clara das interações pretendidas entre as entidades, que compõem as tabelas, e como estas estão interligadas através dos seus atributos.

**NOTA** – Dada a complexidade e tamanho dos diagramas, estes poderão não ser totalmente percetíveis. Para facilitar a sua análise, foram incluídos no ficheiro final de submissão.

### 2.1 Diagrama Entidade-Relação

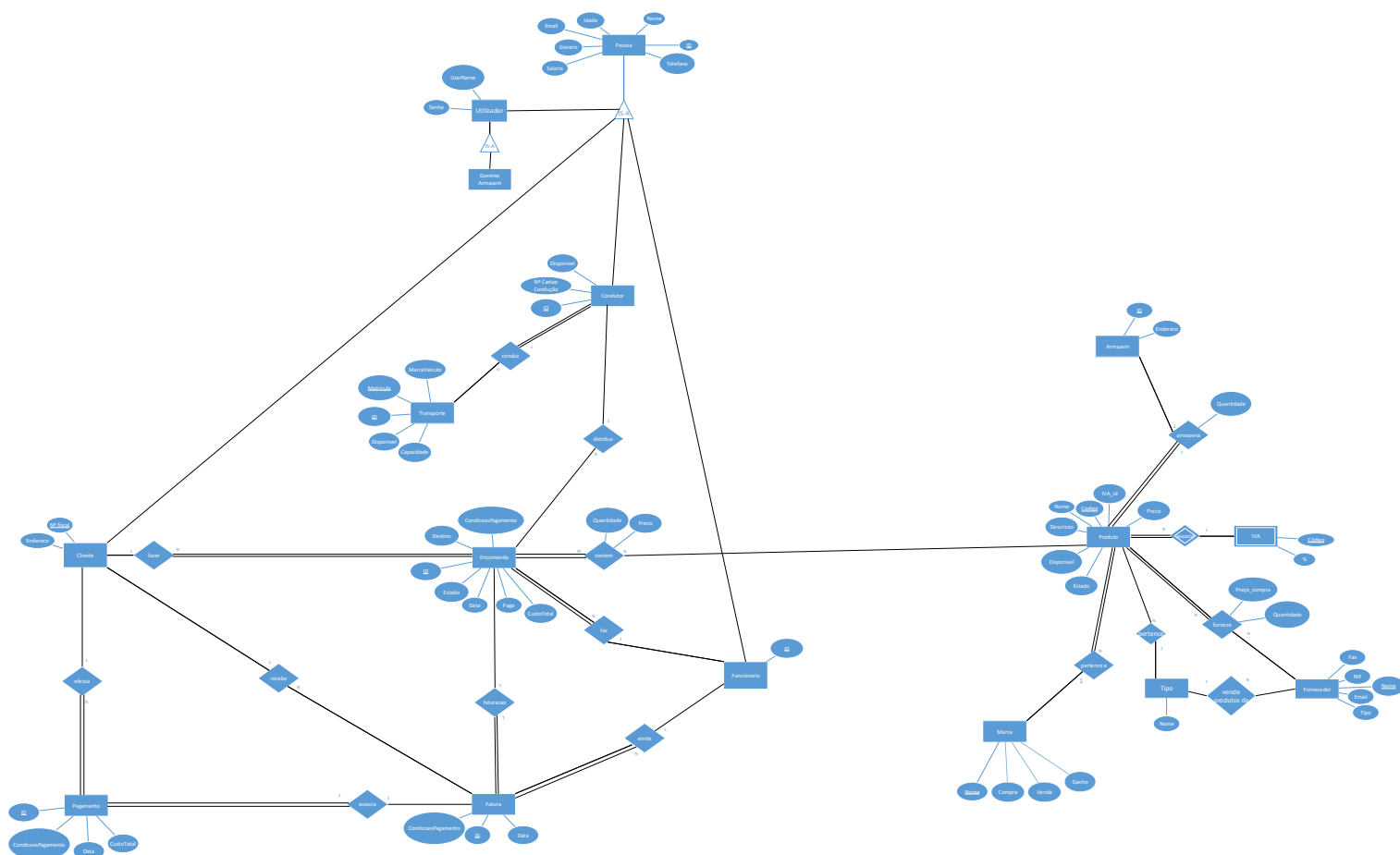


Figura 1 - Diagrama Entidade-Relação

## 2.2 Esquema do Modelo Relacional

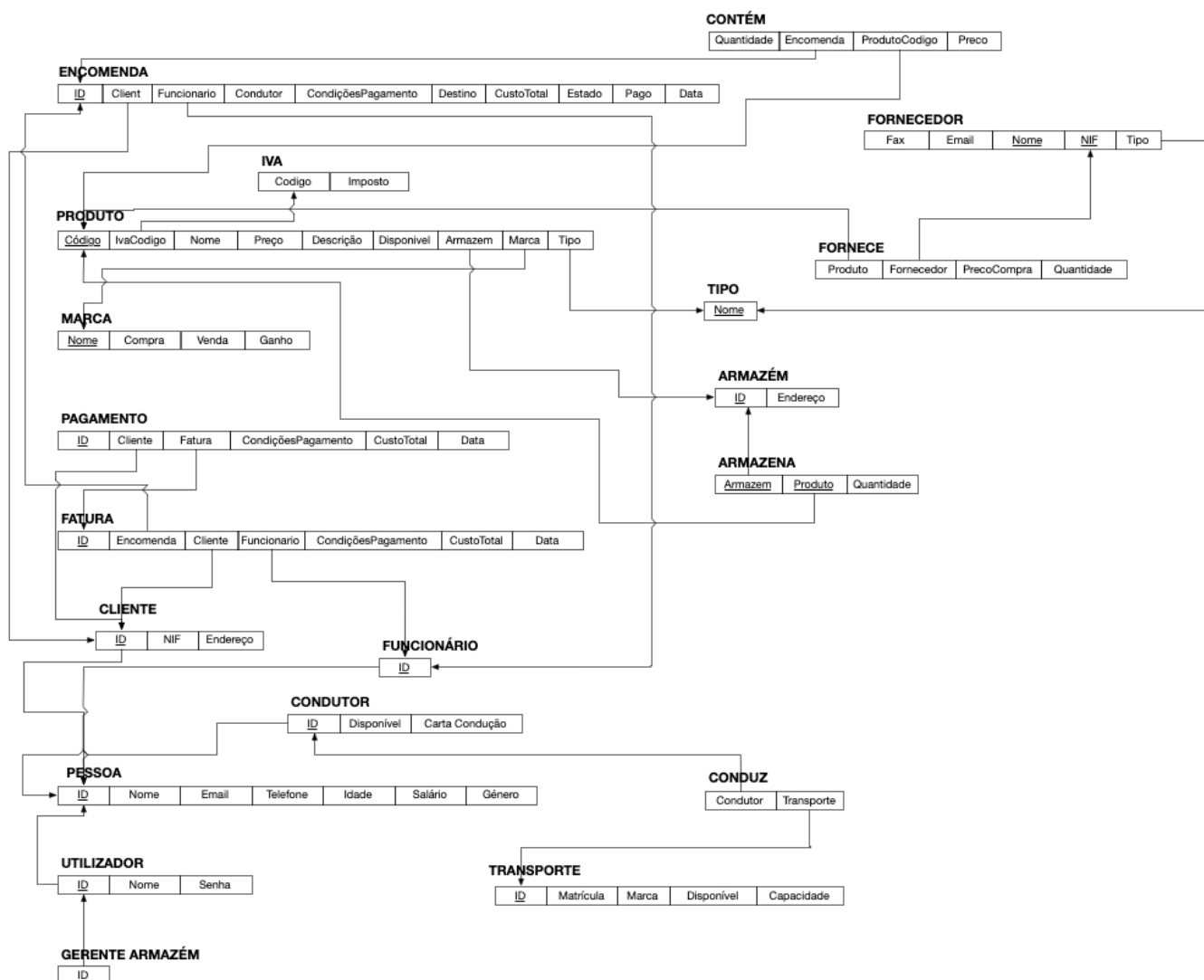


Figura 2 - Esquema do Modelo Relacional



### 3. Normalização

A aplicação do processo de normalização tem como principal objetivo reduzir a redundância de informação armazenada, melhorando o desempenho da base de dados.

Deste modo, após analisar a entrega da proposta do projeto, alguns ajustes foram feitos através da aplicação de um conjunto de testes propostos por *Boyce-Codd*, a fim de determinar qual a forma normal em que a esquematização relacional se encontrava.

Na generalidade das relações, foi possível atingir um bom grau de normalização – a terceira forma normal. Sempre que, numa relação, se está perante um ID, é possível chegar aos restantes atributos que lhe estão associados (por exemplo, na tabela Pessoa, através do ID consegue-se saber qual o Nome).

## 4. Construção da Base de Dados

Recorremos à linguagem SQL e às suas principais sub-linguagens, *Data Definition Language* (DDL) e *Data Manipulation Language* (DML) para definir, manipular e aceder à base de dados relacional do projeto.

Nas subsecções que se seguem, alguns exemplos de aplicação dos conceitos serão expostos, a fim de concretizar a esquematização anterior.

### 4.1 *Data Definition Language (DDL)*

Para definir as várias entidades da base de dados apresentadas nos capítulos anteriores, servimo-nos da sintaxe oferecida pela DDL para as exprimir sob a forma de Tabelas e especificar os atributos que lhe estão associados (representando as colunas). Definiram-se, também, as restrições de integridade necessárias para que as entidades e respetivos atributos cumprissem os requisitos que foram definidos.

Seguem-se alguns exemplos de aplicação desta sub-linguagem.

```
create table SuperMercado.Encomenda(
  ID int                                not null Identity(1,1),
  Client int                            not null,
  Funcionario int                       not null,
  Condutor int                          not null DEFAULT -1,
  CondicoesPagamento varchar(30)       not null,
  Destino varchar(50)                   not null,
  CustoTotal float                      not null DEFAULT 0,
  Estado int                            not null DEFAULT 0,
  Pago int                              not null DEFAULT 0,
  Data date                             not null,

  Primary key (ID),
  Foreign key (Client) references SuperMercado.Cliente(ID),
  Foreign key (Funcionario) references SuperMercado.Funcionario(ID)
)
```

Figura 3 - Criação da Tabela Encomenda

```
create table SuperMercado.Pessoa(
  ID int                                not null Identity(1,1),
  Nome varchar(40)                      not null,
  Email varchar(30)                     not null,
  Telefone char(9),
  Idade int                             not null,
  Salario float                          not null DEFAULT -1,
  Genero varchar(1)                     not null,

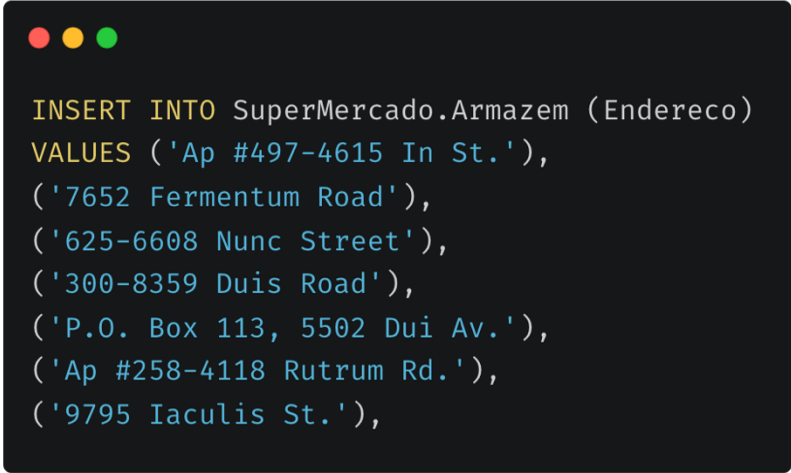
  Primary key (ID),
  Unique (Email, Telefone)
)
```

Figura 4 - Criação da Tabela Pessoa

## 4.2 Data Manipulation Language (DML)


Para popular a base de dados e efetuar algumas consultas simples sobre a mesma utilizámos a DDL. Esta foi incluída nas *Stored Procedures* e *User Defined Functions* (conceitos explorados nos capítulos seguintes do presente relatório), por forma a aceder e manipular os dados inseridos da maneira mais correta e eficaz possível.

Seguem-se alguns exemplos de aplicação desta sub-linguagem na tarefa de popular a base de dados.

A screenshot of a code editor with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The code is a SQL INSERT statement for a table named 'Armazem' in the 'SuperMercado' database. It lists seven addresses as values to be inserted.

```
INSERT INTO SuperMercado.Armazem (Endereco)
VALUES ('Ap #497-4615 In St.'),
('7652 Fermentum Road'),
('625-6608 Nunc Street'),
('300-8359 Duis Road'),
('P.O. Box 113, 5502 Dui Av.'),
('Ap #258-4118 Rutrum Rd.'),
('9795 Iaculis St.'),
```

Figura 5 - Utilização de DML para inserção de dados na tabela 'Armazem'

A screenshot of a code editor with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The code is a SQL INSERT statement for a table named 'Pessoa' in the 'SuperMercado' database. It lists six rows of personal data, including names, emails, phone numbers, ages, salaries, and genders, as values to be inserted.

```
INSERT INTO SuperMercado.Pessoa (Nome,Email,Telefone,Idade,Salario,Genero)
VALUES ('Aaron Gonzales','Proin@egestas.com','915084877',53,1774,'M'),
('Shafira Camacho','placerat.eget@Maecenasm.net','915595762',40,1789,'F'),
('Erasmus Griffin','fermentum.risus@imperdo.net','915376495',29,1606,'F'),
('Deacon Alston','aliquet.lobortis@malesuada.edu','915784625',30,1931,'M'),
('Palmer Burton','nec.ante@Aliquamgravris.com','915326458',35,1791,'F'),
```

Figura 6 - Utilização de DML para inserção de dados na tabela 'Pessoa'

## 5. 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), e permitem a implementação de lógicas elaboradas onde se altera o modelo de dados, apresentam um conjunto de mais valias bastante atrativas para o nosso projeto. Foram especialmente úteis na execução de transações.

Adicionalmente, trancar as tabelas e permitir apenas o acesso através de *Stored Procedures* é uma boa prática do desenvolvimento de base de dados, e oferece segurança complementar.

Apresentam-se exemplos de *Stored Procedures* definidas para responder às funcionalidades pretendidas com o sistema proposto.

```

Create Proc SuperMercado.createNewUser (@Nome varchar(40),@Email varchar(40),@Telefone char(9),
    @Genero varchar(1), @Idade int,@Salario float,@Username varchar(40),@Senha varchar(40))
AS
BEGIN
    DECLARE @count int;
    SELECT @count=count(1) FROM SuperMercado.Pessoa WHERE Email=@Email
    IF(@count >= 1)
        RAISERROR ('O Email ja foi usado!', 16,1);
    ELSE
        BEGIN
            BEGIN TRY
                BEGIN TRAN
                    DECLARE @InsertedID int;
                    EXEC SuperMercado.createNewPessoa
                        (@Nome,@Email,@Telefone,@Genero,@Idade,@Salario);
                    SELECT @InsertedID=@@IDENTITY
                    INSERT INTO SuperMercado.Cliente (ID,Endereco) VALUES(@InsertedID,'')
                    INSERT INTO SuperMercado.Utilizador VALUES
                        (@Username,@Senha,@InsertedID)
                COMMIT TRAN
            END TRY
            BEGIN CATCH
                Rollback TRAN
                RAISERROR ('A conta nao foi criada!', 16,1);
            END CATCH
        END
    END
END

```

Figura 7 - Stored Procedure para criar novo utilizador

```

CREATE PROC SuperMercado.payment (@Encomenda int)
AS
BEGIN
    BEGIN TRY
        BEGIN TRAN
            DECLARE @Quantidade int, @PrecoVenda float, @PrecoCompra float, @Produto int,
@Marca varchar(40)
            DECLARE C CURSOR FAST_FORWARD
            FOR SELECT Quantidade,Preco,ProdutoCodigo FROM SuperMercado.Contem WHERE
Encomenda=@Encomenda;

            OPEN C

            FETCH C INTO @Quantidade,@PrecoVenda,@Produto

            WHILE @@FETCH_STATUS = 0
            BEGIN
                SELECT @Marca=Marca FROM SuperMercado.Produto WHERE Codigo=@Produto
                SELECT @PrecoCompra=PrecoCompra FROM SuperMercado.Fornece WHERE
Produto=@Produto

                UPDATE SuperMercado.Marca SET Venda=Venda+(@Quantidade*@PrecoVenda),
Ganho=Ganho+((@PrecoVenda-@PrecoCompra)*@Quantidade)
                WHERE Nome=@Marca

                FETCH C INTO @Quantidade,@PrecoVenda,@Produto
            END
            CLOSE C
            DEALLOCATE C

            UPDATE SuperMercado.Encomenda SET Pago=1 WHERE ID = @Encomenda
            COMMIT

        END TRY
        BEGIN CATCH
            ROLLBACK TRAN
            RAISERROR ('O Pagamento nao foi realizado!Tente mais tarde!', 16,1);
        END CATCH
    END

```

Figura 8 - Stored Procedure para efetuar o pagamento de uma encomenda

**NOTA** – Dado que uma encomenda é constituída por uma lista de produtos, e não é possível passar uma lista como parâmetro em SQL, a transação que cria uma nova encomenda não foi implementada com recurso a uma *Stored Procedure*, mas sim diretamente em C# (ficheiro *Services.cs*, função *CreateEncomenda*). Após o problema ter sido apresentado ao professor Joaquim Sousa Pinto numa aula prática, este explicou que poderíamos ter recorrido ao uso de uma tabela temporária, contudo a complexidade (já elevada) associada a esta transação iria aumentar bastante, potencialmente atrasando o desenvolvimento do projeto. Desta forma, decidiu-se manter a sua implementação em C#.

## 6. User Defined Functions

As *User Defined Functions* apresentam os mesmos benefícios que os *SP*, em termos de compilação e otimização de execução, podendo ser incluídas por forma a incluir uma lógica mais complexa dentro de uma consulta (*SELECT*).

Além de oferecerem os mesmo benefícios que as *VIEWS*, estas aceitam parâmetros, tendo sido utilizadas para servir de fonte de dados numa *query*.

Recorreu-se ao uso de *UDF Escalares*, para retornar valores únicos, *UDF Inline Table-Valued*, para servirem de wrappers nas construções *SELECT*, e *UDF Multi-Statement Table-Valued*, para aceder e visualizar tabelas que são utilizadas em operações de *SELECT* de forma eficiente.

Salienta-se que todas as opções de ordenação e filtragem das listas apresentadas (por exemplo, das Marcas – lado do Gestor – ou dos Produtos – no cliente) foram conseguidas à custa de UDFs.

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

```
CREATE FUNCTION SuperMercado.getCountFaturas (@ID int) RETURNS int
AS
BEGIN
    DECLARE @res int
    SELECT @res=COUNT(1) FROM SuperMercado.Fatura
    WHERE Client = @ID
    RETURN @res
END
```

Figura 9 - User Defined Function para devolver o número de faturas de um dado cliente

```
CREATE FUNCTION SuperMercado.getPessoaByEmail (@Email varchar(30)) RETURNS TABLE
AS
RETURN (SELECT * FROM SuperMercado.Pessoa WHERE Email = @Email)
```

Figura 10 - User Defined Function para devolver a pessoa associada a um email

```
CREATE FUNCTION SuperMercado.checkIfUserExists (@Email varchar(30), @Senha varchar(40)) RETURNS
@table Table (ID int, Name varchar(50))
AS
BEGIN
    DECLARE @ID int
    Set @ID = 0
    SELECT @ID = ID FROM SuperMercado.Pessoa WHERE Email = @Email;
    IF (@ID = 0)
        INSERT @table SELECT @ID, 'unkown'
    ELSE
        INSERT @table SELECT ID, Nome FROM SuperMercado.Utilizador WHERE ID = @ID AND Senha
        = @Senha
    RETURN;
END
```

Figura 11 - User Defined Function para verificar existência de um utilizador

## 7. Triggers

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

Visto que não se deve confiar na informação inserida pelo utilizador, foram criados três *triggers* do tipo *INSTEAD OF*, de modo a verificar as entradas de dados feitas tanto pelo Cliente como pelo Gerente de Armazém.

Deste modo, no caso de algum campo estar mal preenchido, os dados não serão utilizados para, por exemplo, atualizar um atributo numa dada tabela.

```
CREATE Trigger checkInput ON SuperMercado.Cliente
INSTEAD OF UPDATE
AS
BEGIN
    IF (SELECT count(*) FROM inserted) = 1
    BEGIN
        DECLARE @NIF as char(9);

        SELECT @NIF = NIF FROM inserted;

        IF len(@NIF)≠9
            RAISERROR('O NIF tem de ser composto por 9 digitos!',16,1);
        ELSE IF PATINDEX('%[^0-9]%', @NIF) ≠0
            RAISERROR('O NIF tem de ser composto por 9 digitos!',16,1);
        ELSE
            UPDATE SuperMercado.Cliente SET NIF= @NIF WHERE ID IN (SELECT ID FROM
inserted)
    END
END
```

Figura 12 - Trigger para verificação de input do utilizador, ao inserir NIF

## 8. Aspetos de Segurança

Tendo como objetivo reduzir, ao máximo, as possíveis vulnerabilidades do projeto decorrentes de ataques de *SQL Injection*, as indicações fornecidas no decorrer nas aulas teóricas e práticas da unidade curricular foram seguidas dentro do possível.

De modo a prevenir eventuais problemas, salienta-se que:

### ***Não se confia nos dados introduzidos pelo utilizador***

Foram implementados Triggers, referidos no capítulo 7 do presente relatório, para efetuar as várias verificações necessárias a todas as entradas de dados. Assim, estas apenas são introduzidas quando são validadas.

### ***Evita-se, ao máximo, recorrer a SQL Dinâmico***

Utilizou-se SQL Parametrizado ou *Stored Procedures* no processamento de dados introduzidos pelo utilizador.

### ***Não se armazena informação sensível (por exemplo, passwords) em texto simples***

Ao criar uma conta de utilizador, sendo padrão ou administrador, a password do mesmo será guardada após ter sido efetuado o seu *hashing*. Desta forma, caso a injeção maliciosa de comandos SQL resulte na exposição da tabela que armazena os dados de login, dificilmente se conseguirá ganhar acesso a uma das contas.

### ***Apresentação de informação de erros reduzida ao mínimo***

Mensagens de erros customizadas são apresentadas, de modo a informar o utilizador que os dados inseridos não estão de acordo com o esperado. Estão presentes, por exemplo, quando um Gestor preenche o formulário de contratação de um novo funcionário.



## 9. Componentes Não Implementadas

Face os componentes presentes no projeto, explorados e explicitados nos capítulos anteriores, identificam-se os que não foram aplicados, por terem sido substituídos por outros.

### 9.1 *Indexes*

Os Indexes são estruturas complementares cujo objetivo é acelerar o processo de pesquisa, que é crítico para o desempenho da base de dados. São especialmente úteis em tabelas com muitas entradas.

Porém, podem aumentar o volume de dados armazenados e o tempo das inserções – já que, sempre que se insere um novo elemento é necessário inseri-lo também no índice.

Dada a dimensão do projeto, o uso de indexes não se mostrou necessário, optando-se por não fazer a sua implementação.

## 10. Utilização

Para que a interface do utilizador funcione com a base de dados que foi atribuída ao nosso grupo, presente no servidor da universidade, modificou-se o ficheiro *app.config* de modo a incluir a *connection string* que nos corresponde.

```
<connectionStrings>
  <clear />
  <add name="SuperMercadoConnection"
    providerName="System.Data.SqlClient"
    connectionString="server = tcp:mednat.ieeta.pt\SQLSERVER,8101;database = p2g7;user=p2g7;
    password=grupo7_weikika_2020" />
</connectionStrings>
```

Figura 13 - Connection String para aceder ao servidor da universidade

Por forma a testar as funcionalidades que um cliente terá acesso na plataforma, cria-se uma nova conta de utilizador, preenchendo o formulário apresentado.

Figura 14 - Formulário para criação de novo utilizador

De seguida, será possível aceder ao sistema, utilizando os dados que foram introduzidos.

Figura 15 - Utilização de novo utilizador criado para aceder à plataforma

De modo similar, pretendendo aceder à área atribuída ao Gerente de Armazém (ou seja, o *administrador* do sistema), os dados a introduzir na página de login apresentada ao inicializar a aplicação serão:

- Email: *admin@ua.pt*
- Password: *abcd*

De destacar que se optou por gerar, por defeito, os dados de acesso acima visto que, no sistema do projeto, assumiu-se a existência de um administrador (neste caso, Gerente) global. Numa solução presente no mercado com certeza existem vários acessos administrativos, com dados não tão simples quanto os do projeto.

## 10.1 *Vídeo Demonstrativo*

No link abaixo é possível encontrar um pequeno vídeo, que demonstra as funcionalidades da interface desenvolvida, descritas e referenciadas nos capítulos anteriores.

<https://youtu.be/7tm6lO-rehg>

## Conclusão

Tendo por base todas as explicações expostas anteriormente e todo o código desenvolvido, entregue em conjunto com este relatório, é possível concluir que os objetivos propostos por este trabalho foram atingidos na sua totalidade.

Considerámos ter cumprido o objetivo proposto na introdução – reproduzir o processo de gestão de um supermercado online de forma fiável e acessível.

Julgámos ter conseguido desenvolver uma base de dados bastante realista e próxima das plataformas encontradas no mercado. Evidentemente, recorreu-se às simplificações necessárias para evitar complexidade adicional que não seria relevante no contexto da unidade curricular.

De modo a melhorar e complementar o trabalho apresentado, uma análise do desempenho da plataforma com a implementação de *Indexes* teria sido interessante, necessitando de ter populado mais extensivamente a base de dados.

## Material de Referência

<https://www.businessinsider.com/online-grocery-report-2020>

<https://www.supermarketnews.com/online-retail/online-grocery-sales-grow-40-2020>

<https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/connection-strings-and-configuration-files>

<https://carbon.now.sh/> (Excertos de código apresentados)