universidade de aveiro deti departamento de electrónica, telecomunicações e informática

Ferramenta de Gestão de uma Mercearia Convencional

Bases de Dados

-- Desenvolvimento de uma aplicação interativa --

P5G9 - Prof. Carlos Costa

João Santos - 73761

Diogo Reis - 64231

Índice

Introdução

No âmbito da unidade curricular de Bases de Dados foi proposto aos alunos que desenvolvessem uma aplicação interativa. Tendo em conta um processo de aprendizagem de diferentes conteúdos esta deveria ser implementada gradualmente consoante os conhecimentos adquiridos ao longo do semestre.

O tema escolhido para a posterior implementação foi a Gestão de uma Mercearia Convencional.

Hoje em dia não é credível que tendo em conta os avanços tecnológicos uma mercearia continue a trabalhar com papel e caneta, todo este processo infere erros de cálculos e dificuldade na pesquisa de informações pois o que acontece na maioria das vezes é que se forma um "amontoado" de papéis contendo informações relevantes de Produtos, Fornecedores e Clientes.

Como tal, a aplicação desenvolvida vai de encontro a facilitar e organizar todas as tarefas inerentes a uma Mercearia.

Aplicação - Windows Forms/C#

Base de Dados - SQL Server

Este relatório tem como objetivo explicar todos os passos na execução do projeto:

- ❑ Análise de Requisitos
- ❑ Objetivos de Usabilidade
- ❑ Diagrama Entidade-Relação
- ❑ Esquema Relacional
- ❑ SQL DDL
- ❑ Views
- ❑ Stored Procedures
- ❑ User-Defined Functions
- ❑ Triggers
- ❑ Conclusões e Referências

Análise de Requisitos

Tendo em conta o objetivo principal da aplicação esta destina-se aos proprietários e funcionários de uma Mercearia. Foram reunidas um conjunto de normas que pensamos ser necessárias para tornar o sistema credível e funcional.

- Uma Mercearia convencional pouco informatizada necessita de melhorar/automatizar o sistema de encomendas ao fornecedor.
- Um dos maiores problemas é a gestão do stock existente.
- Cada produto tem um nome, marca, IVA, preço, stock atual, stock mínimo e percentagens de lucro associadas.
- O IVA pode ser de taxa reduzida (6%), taxa intermédia (13%) ou taxa normal (23%).
- Nessa mercearia há produtos que convém estar sempre em stock (leite, farinha, açúcar, ...) e produtos que nem sempre estão em stock que podem ser pedidos pelos clientes (roupa, produtos de marca específica, ...).
- Nessa mercearia há necessidade de informatizar a forma como são geradas as encomendas de produtos em falta.
- Cada produto deve ter um stock atual e um stock mínimo.
- Quando o stock de um determinado produto alcança o valor do stock mínimo é possível receber essa informação na altura de realizar uma nova encomenda.
- O Stock atual é atualizado à medida que são efetuadas vendas na mercearia.
- Na altura em que é inserido um novo cliente no sistema este fica associado a uma conta corrente.
- Esta conta corrente vem ajudar num dos grandes problemas das mercearias, os fiados.
- Depois de recebida a encomenda, é necessário atualizar as quantidades de produtos recebidas

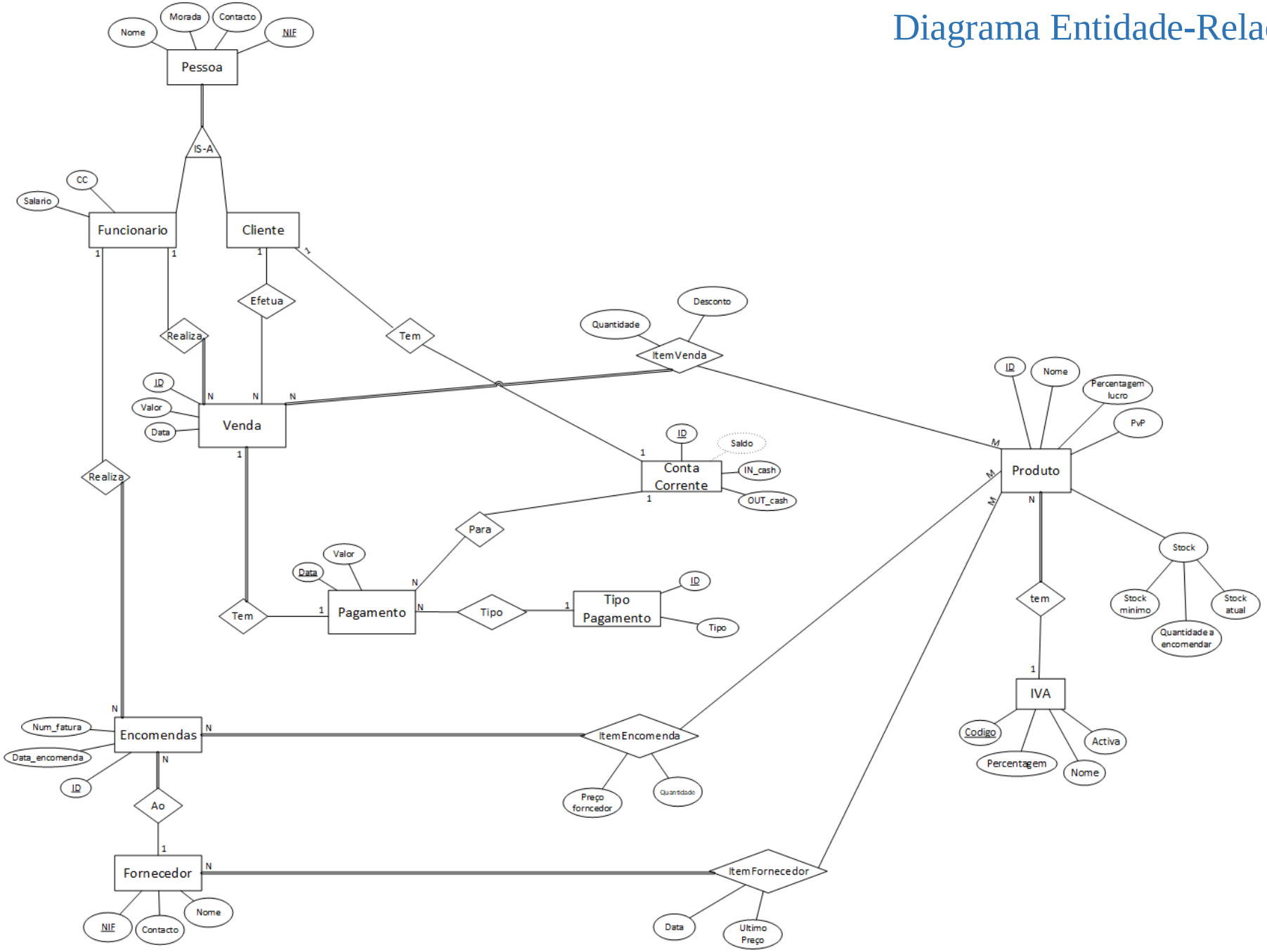
Objetivos de Usabilidade

A aplicação deve permitir efetuar tarefas e ações de modo a facilitar a introdução de dados e a sua consequente manipulação.

1. Introdução e Edição de Novos Clientes (Informações Básicas).
2. Introdução e Edição de Novos Funcionários (Informações Básicas e Salário).
3. Introdução e Edição de Novos Fornecedores (Informações Básicas).
4. Introdução e Edição de Novos Produtos (Informações Básicas, Percentagem de Lucro, PVP, Código de IVA, Stock Mínimo e Stock Atual).
5. Realizar Novas Vendas associadas a Produtos e Clientes.
6. Gestão de Contas Corrente associadas ao Cliente.
7. Efetuar e Receber Encomendas associando Fornecedores e Encomendas.
8. Gerir todo o processo de Stock de Produtos relacionando Novas Vendas e Encomendas.

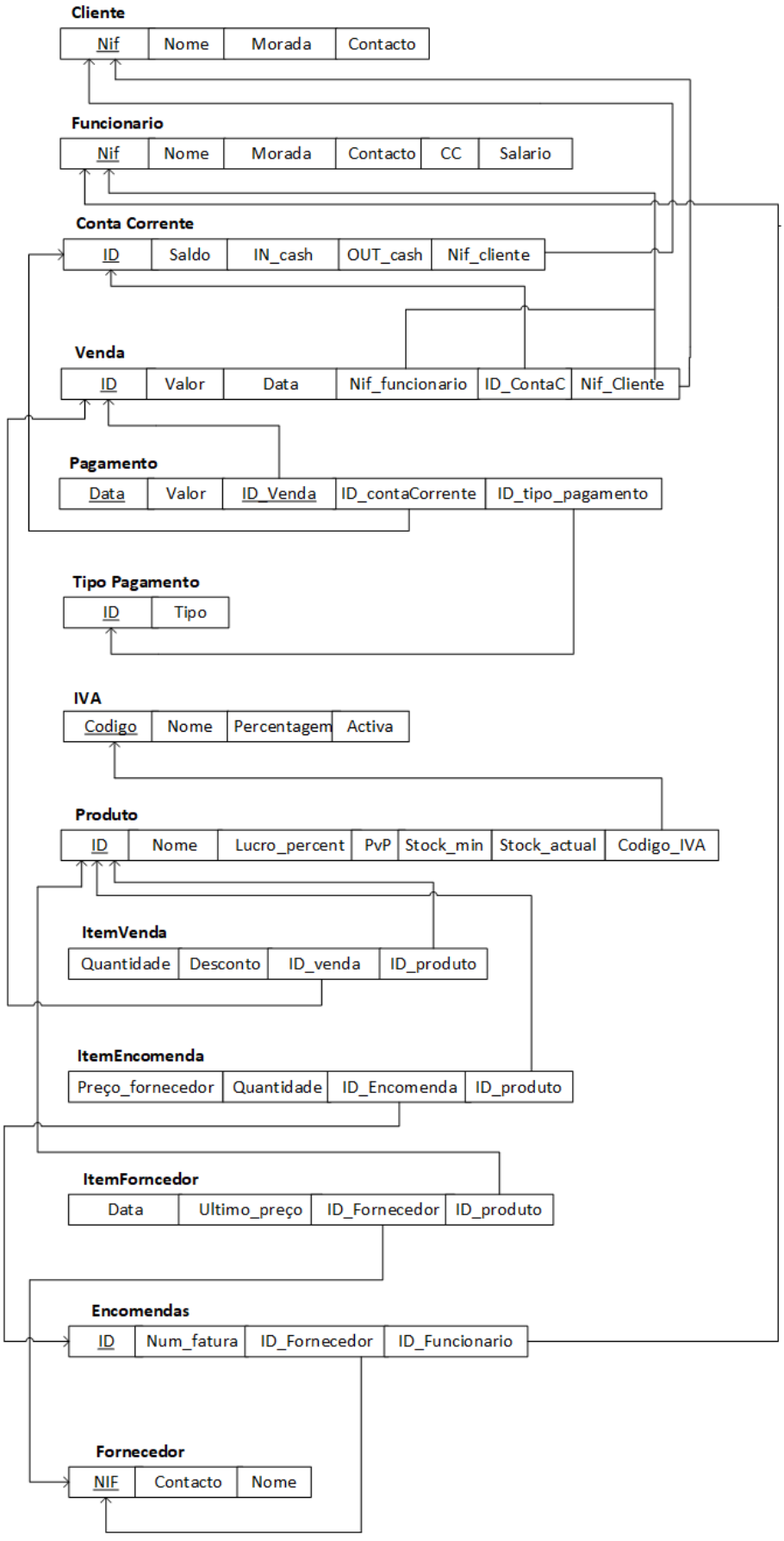
A maioria dos proprietários de Mercearias são pessoas mais velhas e com mais dificuldade em adaptar-se a novas realidades. Por isso a aplicação desenvolvida deve ser focada na simplicidade dos processos e tornar o mais intuitiva possível a interface.

Diagrama Entidade-Relação



Esquema Relacional

- 13 Tabelas.
- 3 Forma Normal.
- Atributos Atómicos.
- Não existem Dependências Parciais.
- Não existem Dependências Transitivas.



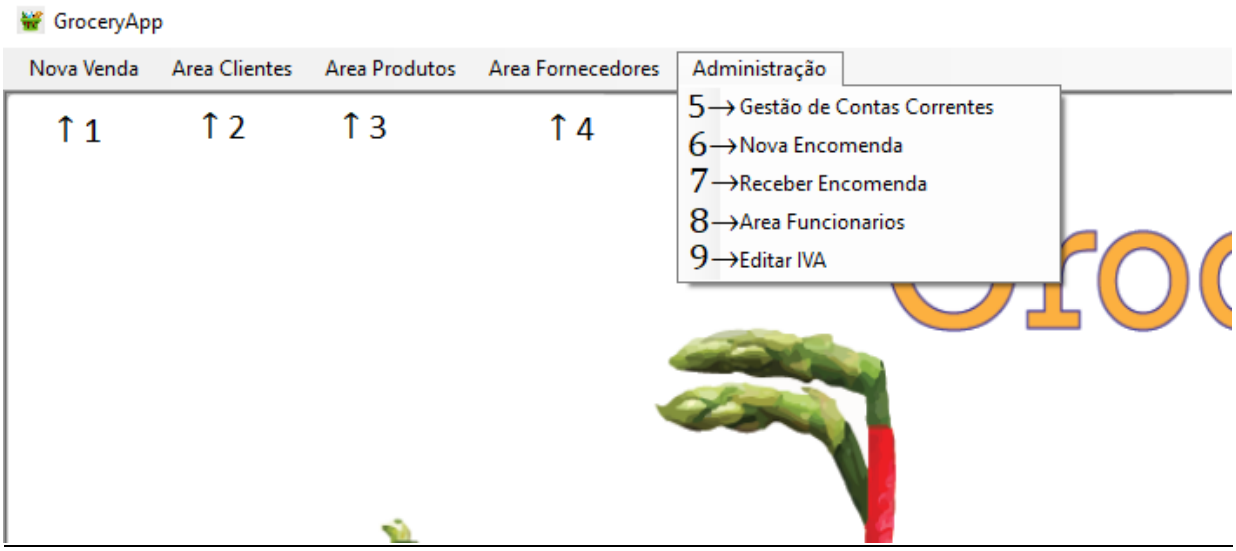
SQL DDL

```
CREATE TABLE grocery.CLIENTE(
  Nif          INT PRIMARY KEY,
  Nome         VARCHAR(50) NOT NULL,
  Morada       VARCHAR(50) NOT NULL,
  Contacto     VARCHAR(50) NOT NULL,
);
GO

CREATE TABLE grocery.FUNCIONARIO(
  Nif          INT PRIMARY KEY,
  Nome         VARCHAR(50) NOT NULL,
  Morada       VARCHAR(50) NOT NULL,
  Contacto     VARCHAR(50) NOT NULL,
  CC           INT NOT NULL,
  Salario      DECIMAL(10,2) NOT NULL,
);
GO

CREATE TABLE grocery.VENDA(
  ID           INT IDENTITY(1,1) PRIMARY KEY,
  Valor        DECIMAL(10,2),
  Nif_func     INT NOT NULL,
  Nif_cliente  INT NOT NULL,
  DataVenda    DATE NOT NULL DEFAULT GETDATE(),
  FOREIGN KEY (Nif_func) REFERENCES grocery.FUNCIONARIO(Nif),
  FOREIGN KEY (Nif_cliente) REFERENCES grocery.CLIENTE(Nif)
);
GO

CREATE TABLE grocery.CONTA_CORRENTE(
  ID           INT IDENTITY(1,1) PRIMARY KEY,
  IN_cash      DECIMAL(10,2),
  OUT_cash     DECIMAL(10,2),
  Nif_cliente  INT NOT NULL,
  Saldo        AS (IN_cash - OUT_cash),
  FOREIGN KEY (Nif_cliente) REFERENCES grocery.CLIENTE(Nif)
);
GO

CREATE TABLE grocery.TIPO_PAGAMENTO(
  ID           INT IDENTITY(1,1) PRIMARY KEY,
  Tipo         VARCHAR(30) NOT NULL,
);
GO
```

```
CREATE TABLE grocery.PAGAMENTO(
  DataPagm       DATE NOT NULL DEFAULT GETDATE(),
  ID_venda       INT NOT NULL,
  Valor          DECIMAL(10,2) NOT NULL,
  ID_conta_corr  INT,
  ID_tipo_pag    INT,
  PRIMARY KEY(DataPagm, ID_venda),
  FOREIGN KEY (ID_venda) REFERENCES grocery.VENDA(ID),
  FOREIGN KEY (ID_conta_corr) REFERENCES grocery.CONTA_CORRENTE(ID),
  FOREIGN KEY (ID_tipo_pag) REFERENCES grocery.TIPO_PAGAMENTO(ID)
);
GO

CREATE TABLE grocery.IVA(
  Codigo         INT IDENTITY(1,1) PRIMARY KEY,
  Nome           VARCHAR(30) NOT NULL,
  Percentagem    INT NOT NULL,
  Estado         BIT NOT NULL
);
GO

CREATE TABLE grocery.PRODUTO(
  ID             INT IDENTITY(1,1) PRIMARY KEY,
  Nome           VARCHAR(50) NOT NULL,
  Percent_lucro  INT NOT NULL,
  PVP            DECIMAL(10,2) NOT NULL,
  Stock_min      INT,
  Stock_atual    INT,
  Codigo_IVA     INT NOT NULL,
  FOREIGN KEY (Codigo_IVA) REFERENCES grocery.IVA(Codigo)
);
GO

CREATE TABLE grocery.ITEM_VENDA(
  ID_venda       INT NOT NULL,
  ID_produto     INT NOT NULL,
  Quantidade     INT NOT NULL,
  Desconto       FLOAT,
  FOREIGN KEY (ID_venda) REFERENCES grocery.VENDA(ID),
  FOREIGN KEY (ID_produto) REFERENCES grocery.PRODUTO(ID)
);
GO

CREATE TABLE grocery.FORNECEDOR(
  Nif            INT PRIMARY KEY,
  Contacto       VARCHAR(50) NOT NULL,
  Nome           VARCHAR(50) NOT NULL
);
GO
```

```
CREATE TABLE grocery.ENCOMENDAS(
  ID            INT IDENTITY(1,1) PRIMARY KEY,
  NumFatura     VARCHAR(50),
  ID_forn       INT NOT NULL,
  ID_func       INT NOT NULL,
  DataEncomenda  DATE NOT NULL DEFAULT GETDATE(),
  FOREIGN KEY (ID_forn) REFERENCES grocery.FORNECEDOR(Nif),
  FOREIGN KEY (ID_func) REFERENCES grocery.FUNCIONARIO(Nif)
);
GO

CREATE TABLE grocery.ITEM_ENCOMENDA(
  ID_encomenda   INT NOT NULL,
  ID_produto     INT NOT NULL,
  Quantidade     INT NOT NULL,
  Preco_forn     DECIMAL(10,2),
  FOREIGN KEY (ID_encomenda) REFERENCES grocery.ENCOMENDAS(ID),
  FOREIGN KEY (ID_produto) REFERENCES grocery.PRODUTO(ID)
);
GO

CREATE TABLE grocery.ITEM_FORNECEDOR(
  ID_fornecedor  INT NOT NULL,
  ID_produto     INT NOT NULL,
  Preco_forn     DECIMAL(10,2) NOT NULL,
  Data_rec       DATE NOT NULL DEFAULT GETDATE(),
  FOREIGN KEY (ID_fornecedor) REFERENCES grocery.FORNECEDOR(Nif),
  FOREIGN KEY (ID_produto) REFERENCES grocery.PRODUTO(ID)
);
GO
```

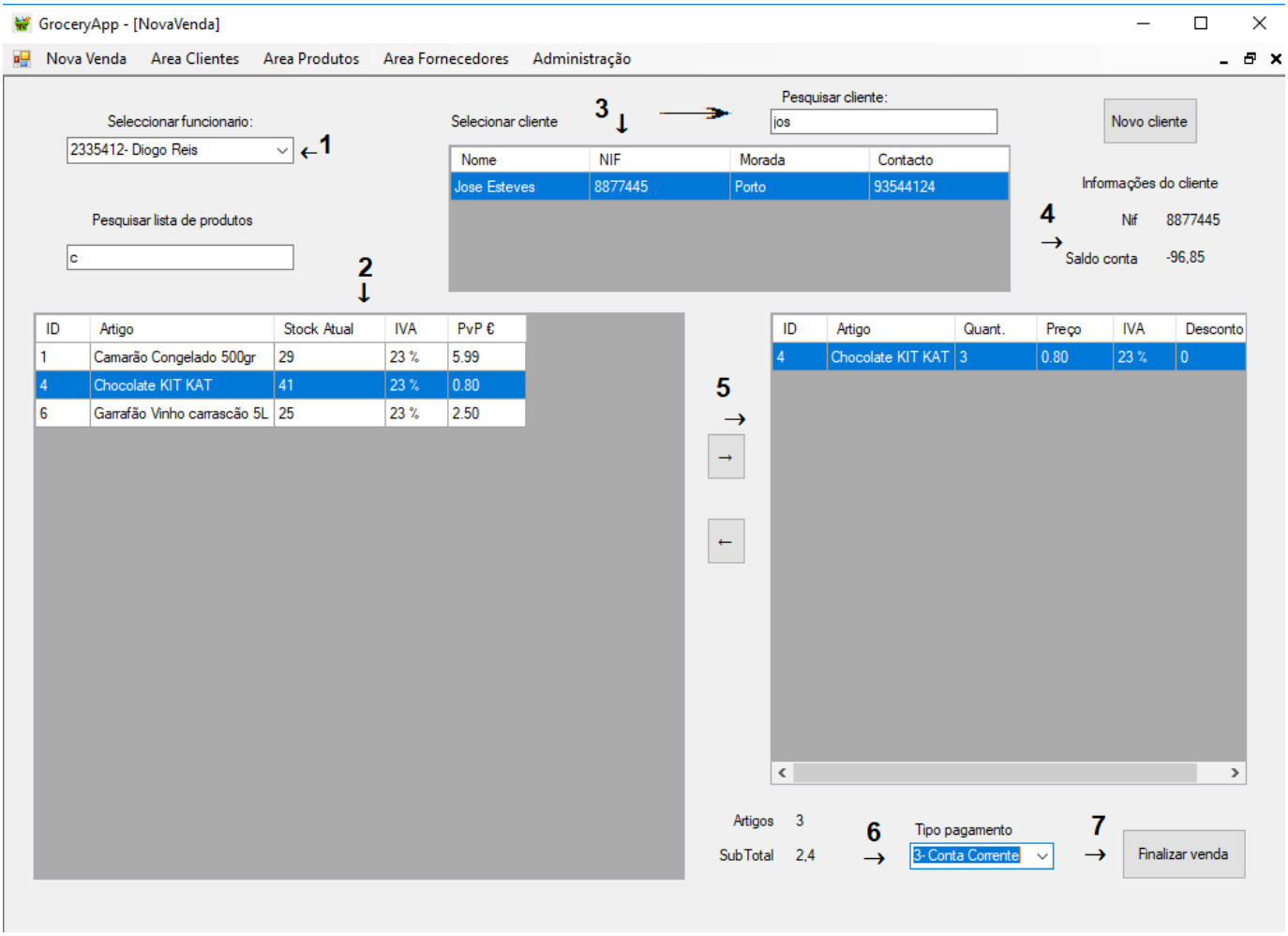
Views, Stored Procedures, UDF's e Triggers



Nas próximas paginas vamos explicar as Views, os Stored Procedures, e as UDF's utilizadas em cada um dos seguintes pontos:

1- Nova venda;
2- Área clientes;
3- Área de produtos;
4- Área de fornecedores:
5- Gestão de conta corrente;
6- Nova encomenda;
7- Receber encomenda;
8- Área de funcionários;
9- Editar IVAs

1 - Nova venda



1-

```
--View para obter funcionarios com nome e nif
CREATE VIEW grocery.VIEW_FUNCIONARIOS_NIFnome
AS
   SELECT CONVERT(NVARCHAR(10),func.Nif)+'- '+CONVERT(NVARCHAR(50),func.Nome) AS
Funcionario
   FROM grocery.FUNCIONARIO as func
GO
```

2-

```
--View para devolver apenas informações necessarias dos produtos para as vendas
CREATE VIEW grocery.VIEW_PRODUTO_NAVENDA
AS
   SELECT  prod.ID AS ID, prod.Nome AS Artigo, prod.Stock_atual AS 'Stock Atual',
CONVERT(NVARCHAR(5),iva.Percentagem)+' %' AS IVA, CONVERT(NVARCHAR(30),prod.PVP) AS
'PvP €'
   FROM grocery.PRODUTO AS prod JOIN grocery.IVA AS iva ON prod.Codigo_IVA=iva.Codigo
GO
```

3-

```
--View para devolver todos os clientes da loja
CREATE VIEW grocery.VIEW_CLIENTES

AS
   SELECT CLIENTE.Nome AS Nome, CLIENTE.Nif AS NIF, CLIENTE.Morada AS Morada,
CLIENTE.Contacto AS Contacto
   FROM grocery.CLIENTE
GO


--UDF para pesquisar clientes pelo nome
CREATE FUNCTION grocery.UDF_PESQUISACLIENTESNOME
   (@nome VARCHAR(50))
RETURNS TABLE
AS
RETURN(
   SELECT CLIENTE.Nome AS Nome,CLIENTE.Nif AS NIF, CLIENTE.Morada AS Morada,
CLIENTE.Contacto AS Contacto
   FROM grocery.CLIENTE
   WHERE Cliente.Nome LIKE '%' + @nome + '%'
   )
GO
```

4-

```
--UDF pesquisar saldo da conta corrente por nif cliente
CREATE FUNCTION grocery.UDF_PESQUISACONTACORRENTENIF
   (@nif INT)
RETURNS DECIMAL(10,2)
AS
BEGIN
   RETURN(
   SELECT Saldo
   FROM grocery.CONTA_CORRENTE
   WHERE Nif_cliente=@nif
   )
END
GO
```

5-

Nestes botões os dados foram manipulados na aplicação;

6-

```
--View para obter tipos de pagamento
CREATE VIEW grocery.VIEW_TIPO_PAGAMENTO
AS
   SELECT CONVERT(nvarchar(5),tipoPagm.ID)+'- '+CONVERT(NVARCHAR(20),tipoPagm.Tipo) AS
TipoPagam
   FROM grocery.TIPO_PAGAMENTO AS tipoPagm
GO
```

7-

1º

```
--Criar nova venda

CREATE PROC grocery.sp_CRIARNOVAVENDA
   @nifCliente INT,
   @nifFunc INT
AS
   INSERT INTO grocery.VENDA
   (Nif_func,Nif_cliente)
   VALUES (@nifFunc,@nifCliente);
GO
```

2º

```
--Obter ID da ultima encomenda criada(Funciona porque nao há nada concorrencial na tabela)
CREATE VIEW grocery.VIEW_VENDA_LAST_ID
AS
   SELECT IDENT_CURRENT('grocery.VENDA') AS lastID
GO
```

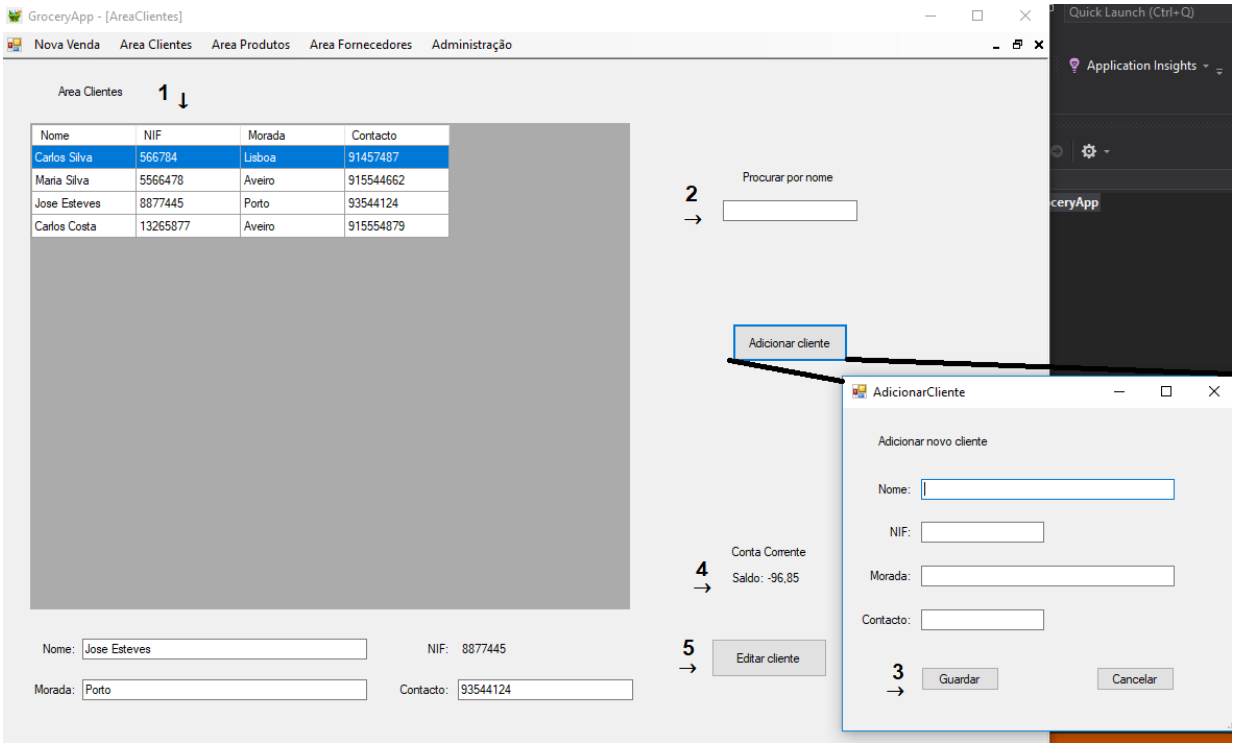
3º

```
--Adicionar produtos à tabela ITEMVENDA com o id da venda
CREATE PROC grocery.sp_AdicionarArtigoITEMvenda
   @idVenda INT,
   @idProduto INT,
   @quantidade INT,
   @desconto INT
AS
BEGIN
   INSERT INTO grocery.ITEM_VENDA
   VALUES (@idVenda,@idProduto,@quantidade,@desconto);
END
BEGIN
   UPDATE grocery.PRODUTO SET Stock_atual = Stock_atual - @quantidade
   WHERE ID=@idProduto
END
GO
```

4º

```
--multiplica o preço de cada produto e soma os resultado para obter o total da venda
--faz update ao total da venda
--verifica qual é o tipo de pagamento e cria-o
--caso seja para a conta corrente cria o pagamento e executa o SP sp_InsertOUTContaCorrente
CREATE PROC grocery.sp_PagamentoTotalValorVenda
   @idVenda INT,
   @tipoPagamento INT
AS
BEGIN
   DECLARE @total DECIMAL(10,2)
   DECLARE @idContaCorr INT
   DECLARE @idCliente INT

   SELECT @total=SUM(PROD.PVP * IV.Quantidade)
   FROM grocery.ITEM_VENDA AS IV JOIN grocery.PRODUTO AS PROD ON IV.ID_produto = PROD.ID
   WHERE IV.ID_venda=@idVenda

   SELECT @idContaCorr=CONTCORR.ID,@idCliente=VEN.Nif_cliente
   FROM grocery.VENDA AS VEN JOIN grocery.CONTA_CORRENTE AS CONTCORR ON
VEN.Nif_cliente=CONTCORR.Nif_cliente

   UPDATE grocery.VENDA
   SET Valor=@total
   WHERE ID=@idVenda

   IF(@tipoPagamento=3)
      BEGIN
         INSERT INTO grocery.PAGAMENTO
         (ID_venda,Valor,ID_conta_corr,ID_tipo_pag)
         VALUES
         (@idVenda,@total,@idContaCorr,@tipoPagamento);
         EXEC grocery.sp_InsertOUTContaCorrente @idContaCorr, @total
      END
   ELSE
      BEGIN
         INSERT INTO grocery.PAGAMENTO
         (ID_venda,Valor,ID_tipo_pag)
         VALUES
         (@idVenda,@total,@tipoPagamento)
      END
END
GO
```

2 - Área Clientes



1-

```
--View select clientes
CREATE VIEW grocery.VIEW_CLIENTES
AS
   SELECT CLIENTE.Nome AS Nome, CLIENTE.Nif AS NIF, CLIENTE.Morada AS Morada,
CLIENTE.Contacto AS Contacto
   FROM grocery.CLIENTE
GO
```

2-

```
--UDF para pesquisar clientes pelo nome
CREATE FUNCTION grocery.UDF_PESQUISACLIENTESNOME
   (@nome VARCHAR(50))
RETURNS TABLE
AS
RETURN(
   SELECT CLIENTE.Nome AS Nome,CLIENTE.Nif AS NIF, CLIENTE.Morada AS Morada,
CLIENTE.Contacto AS Contacto
   FROM grocery.CLIENTE
   WHERE Cliente.Nome LIKE '%' + @nome + '%'
   )
GO
```

3-

Numa form nova:

```
--Criar cliente com conta corrente
CREATE PROC grocery.sp_CriarCliente
   @nif INT,
   @nome VARCHAR(50),
   @morada VARCHAR(50),
   @contacto VARCHAR(50)
AS
BEGIN
   INSERT INTO grocery.CLIENTE
   VALUES (@nif,@nome,@morada,@contacto)
END
BEGIN
   INSERT INTO grocery.CONTA_CORRENTE
   (IN_cash,OUT_cash,Nif_cliente)
   VALUES (0.0,0.0,@nif)
END
GO
```

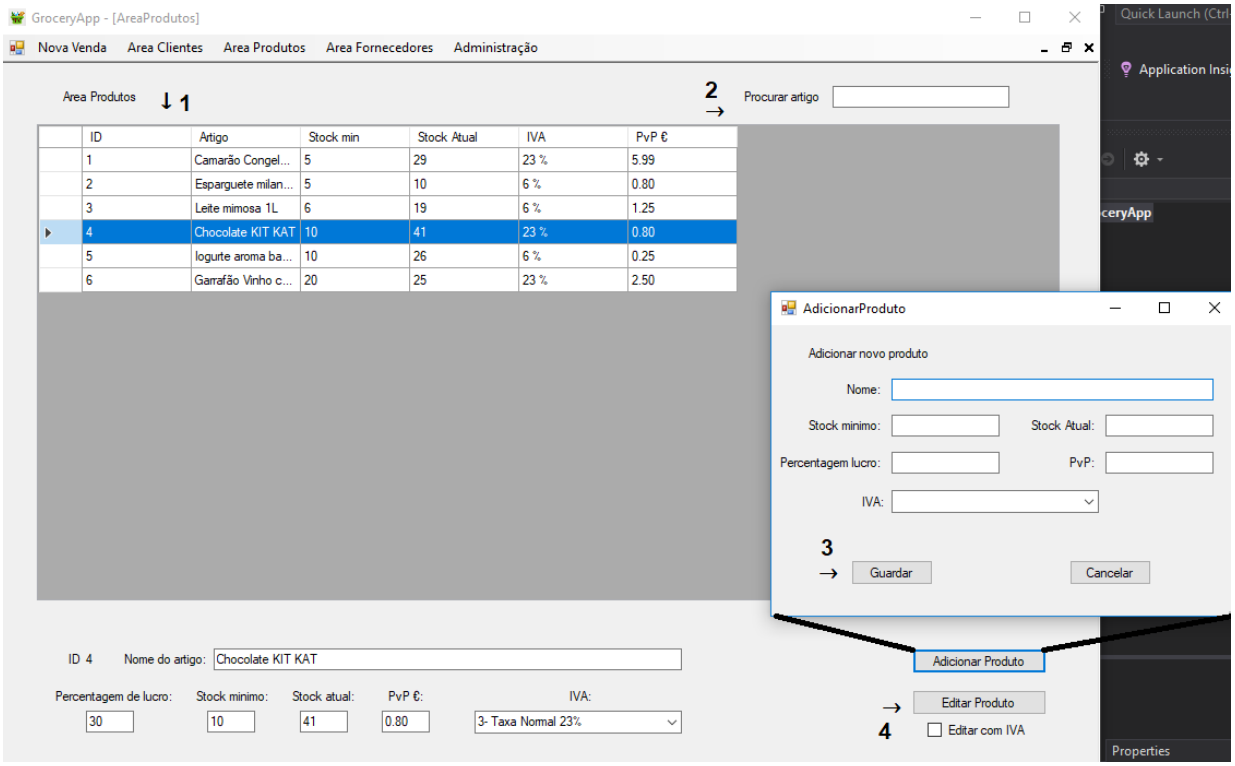
4-

```
--UDF pesquisar saldo da conta corrente por nif cliente
CREATE FUNCTION grocery.UDF_PESQUISACONTACORRENTENIF
   (@nif INT)
RETURNS DECIMAL(10,2)
AS
BEGIN
   RETURN(
   SELECT Saldo
   FROM grocery.CONTA_CORRENTE
   WHERE Nif_cliente=@nif
   )
END
GO
```

5-

```
--Update cliente SP
CREATE PROC grocery.sp_EditarCliente
    @nome VARCHAR(50),
    @nif INT,
    @morada VARCHAR(50),
    @contacto VARCHAR(50)
AS
BEGIN
    UPDATE grocery.CLIENTE SET Nome = @nome, Morada = @morada, Contacto = @contacto
    WHERE Nif = @nif
END
GO
```

3 - Área de Produtos



1-

```
--View para devolver os produtos com todas as informações e o valor do iva
CREATE VIEW grocery.VIEW_PRODUTO_IVA
AS
   SELECT  prod.ID AS ID, prod.Nome AS Artigo, prod.Stock_min AS 'Stock min', prod.Stock_atual AS 'Stock
Atual', CONVERT(NVARCHAR(5),iva.Percentagem)+' %' AS IVA, CONVERT(NVARCHAR(30),prod.PVP)
AS 'PvP €'
   FROM grocery.PRODUTO AS prod JOIN grocery.IVA AS iva ON prod.Codigo_IVA=iva.Codigo
GO
```

2-

```
--UDF para pesquisar artigos pelo nome

CREATE FUNCTION grocery.UDF_PESQUISARARTIGOSNOME
   (@artigo VARCHAR(50))
RETURNS TABLE
AS
RETURN(
   SELECT *
   FROM grocery.VIEW_PRODUTO_IVA
   WHERE VIEW_PRODUTO_IVA.Artigo LIKE '%' + @artigo + '%'
   )
GO
```

3-

Numa form nova:

```
--Criar produtos
CREATE PROC grocery.sp_CriarProduto
   @nome VARCHAR(50),
   @percLucro INT,
   @pvp DECIMAL(10,2),
   @stockMin INT,
   @stockAtual INT,
   @codIva INT
AS
BEGIN
   INSERT INTO grocery.PRODUTO
   VALUES (@nome,@percLucro,@pvp,@stockMin,@stockAtual,@codIva)
END
GO
```

4-

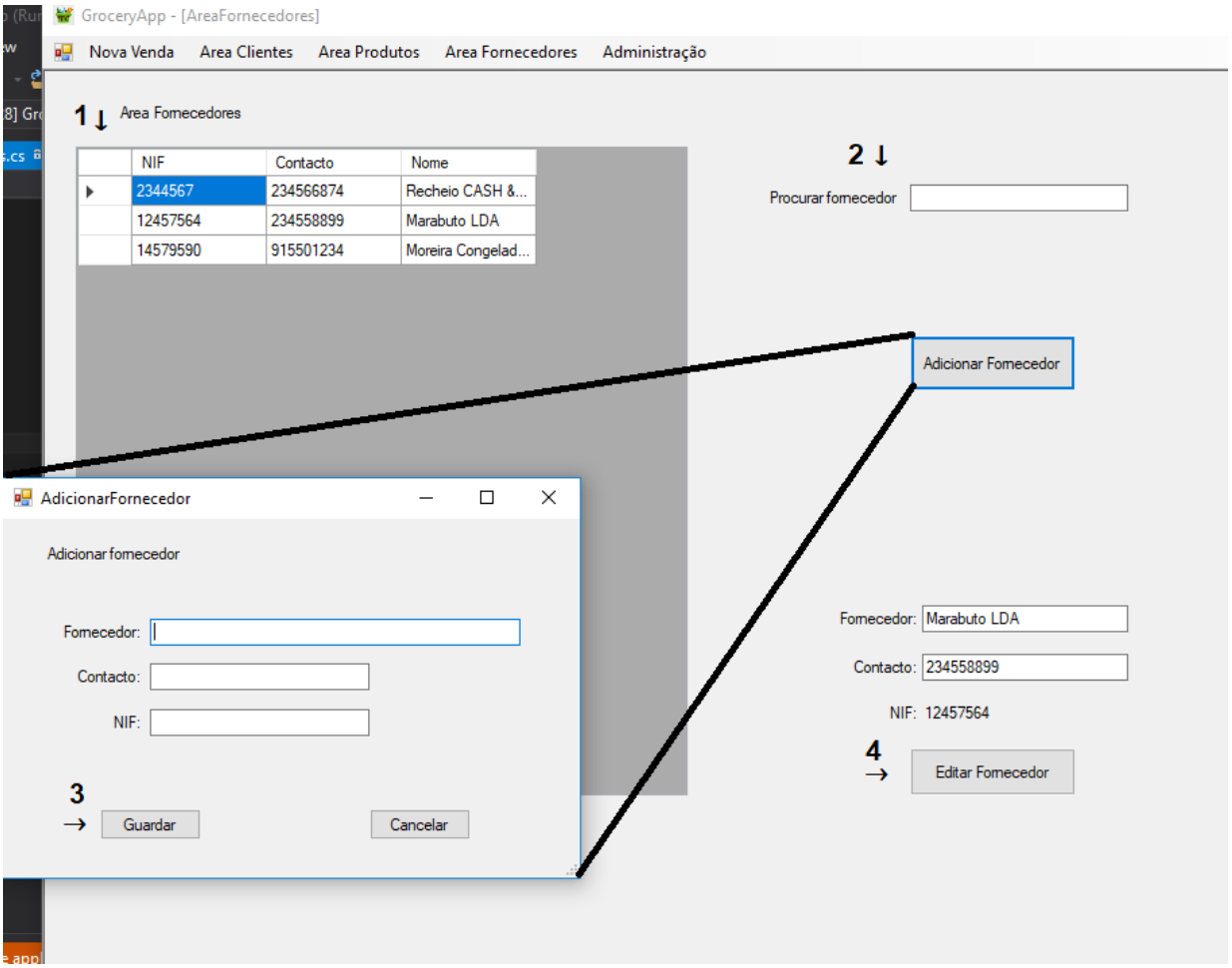
comboBox dos IVAs ativos:

```
--View para IVAS activos
CREATE VIEW grocery.VIEW_IVAS_ACTIVOS
AS
   SELECT CONVERT(NVARCHAR(10),iva.Codigo)+'- '+CONVERT(NVARCHAR(30),iva.Nome)+'
'+CONVERT(NVARCHAR(10),iva.Percentagem)+'%' AS IVA
   FROM grocery.IVA as iva
   WHERE iva.Estado=1
GO
```

Ao editar um produto há a possibilidade de editar o seu IVA ou não, bastando selecionar a *checkBox*, para tal temos dois Stored Procedures:

```
--Editar produto (com IVA)
CREATE PROC grocery.sp_EditarProdutoComIVA
    @nome VARCHAR(50),
    @percLucro INT,
    @pvp DECIMAL(10,2),
    @stockMin INT,
    @stockAtual INT,
    @codIva INT,
    @id INT
AS
BEGIN
    UPDATE grocery.PRODUTO SET Nome = @nome, Percent_lucro = @percLucro, PVP = @pvp,
Stock_min=@stockMin, Stock_atual=@stockAtual, Codigo_IVA=@codIva
    WHERE ID = @id
END
GO

--Editar produto (sem IVA)
CREATE PROC grocery.sp_EditarProdutoSemIVA
    @nome VARCHAR(50),
    @percLucro INT,
    @pvp DECIMAL(10,2),
    @stockMin INT,
    @stockAtual INT,
    @id INT
AS
BEGIN
    UPDATE grocery.PRODUTO SET Nome = @nome, Percent_lucro = @percLucro, PVP = @pvp,
Stock_min=@stockMin, Stock_atual=@stockAtual
    WHERE ID = @id
END
GO
```

4 – Área de fornecedores



1-

```
--View select fornecedores
CREATE VIEW grocery.VIEW_FORNECEDORES
AS
   Select FORNECEDOR.Nif AS NIF, FORNECEDOR.Contacto AS Contacto, FORNECEDOR.Nome AS Nome
   FROM grocery.FORNECEDOR
GO
```

2-

```
--UDF para pesquisar fornecedores pelo nome
CREATE FUNCTION grocery.UDF_PESQUISAFORNECEDORESNOME
   (@nome VARCHAR(50))
RETURNS TABLE
AS
   RETURN(
   SELECT  FORNECEDOR.Nif AS NIF, FORNECEDOR.Contacto AS Contacto, FORNECEDOR.Nome AS
Nome
   FROM grocery.FORNECEDOR
   WHERE FORNECEDOR.Nome LIKE '%' + @nome + '%'
   )
GO
```

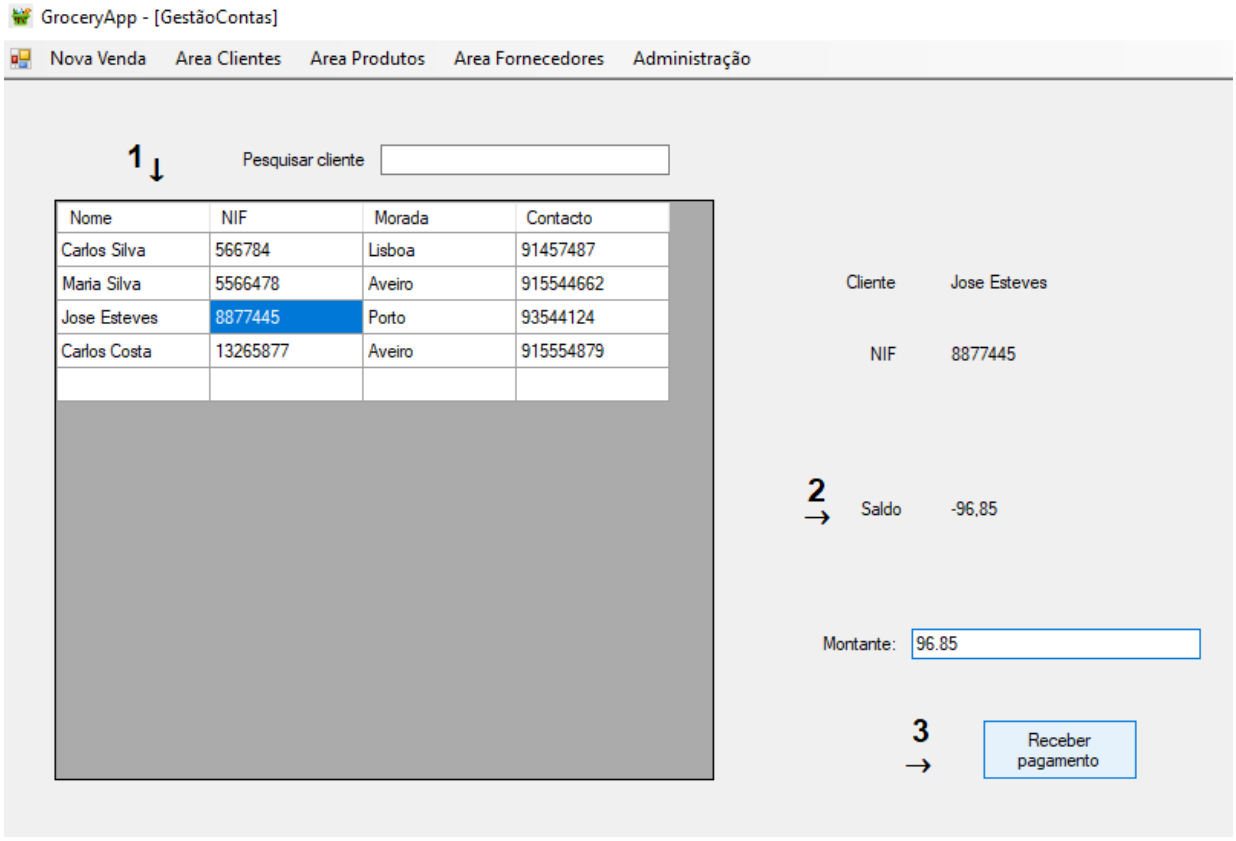
3-

```
--Criar fornecedor
CREATE PROCEDURE grocery.sp_AdicionarFornecedores
   @Nif int,
   @Contacto varchar(50),
   @Nome varchar(50)
AS
BEGIN
   INSERT INTO grocery.FORNECEDOR
   VALUES (@nif,@contacto,@nome)
END
GO
```

4-

```
--Update Fornecedor SP
CREATE PROC grocery.sp_EditarFornecedor
    @nome VARCHAR(50),
    @contacto VARCHAR(50),
    @nif int
AS
BEGIN
    UPDATE grocery.FORNECEDOR SET Nome = @nome, Contacto = @contacto
    WHERE Nif = @nif
END
GO
```

5 - Gestão de Conta Corrente



1-

```
--View select clientes
CREATE VIEW grocery.VIEW_CLIENTES
AS
   SELECT CLIENTE.Nome AS Nome, CLIENTE.Nif AS NIF, CLIENTE.Morada AS Morada,
CLIENTE.Contacto AS Contacto
   FROM grocery.CLIENTE
GO
--UDF para pesquisar clientes pelo nome
CREATE FUNCTION grocery.UDF_PESQUISACLIENTESNOME
   (@nome VARCHAR(50))
RETURNS TABLE
AS
RETURN(
   SELECT CLIENTE.Nome AS Nome,CLIENTE.Nif AS NIF, CLIENTE.Morada AS Morada,
CLIENTE.Contacto AS Contacto
   FROM grocery.CLIENTE
   WHERE Cliente.Nome LIKE '%' + @nome + '%'
   )
GO
```

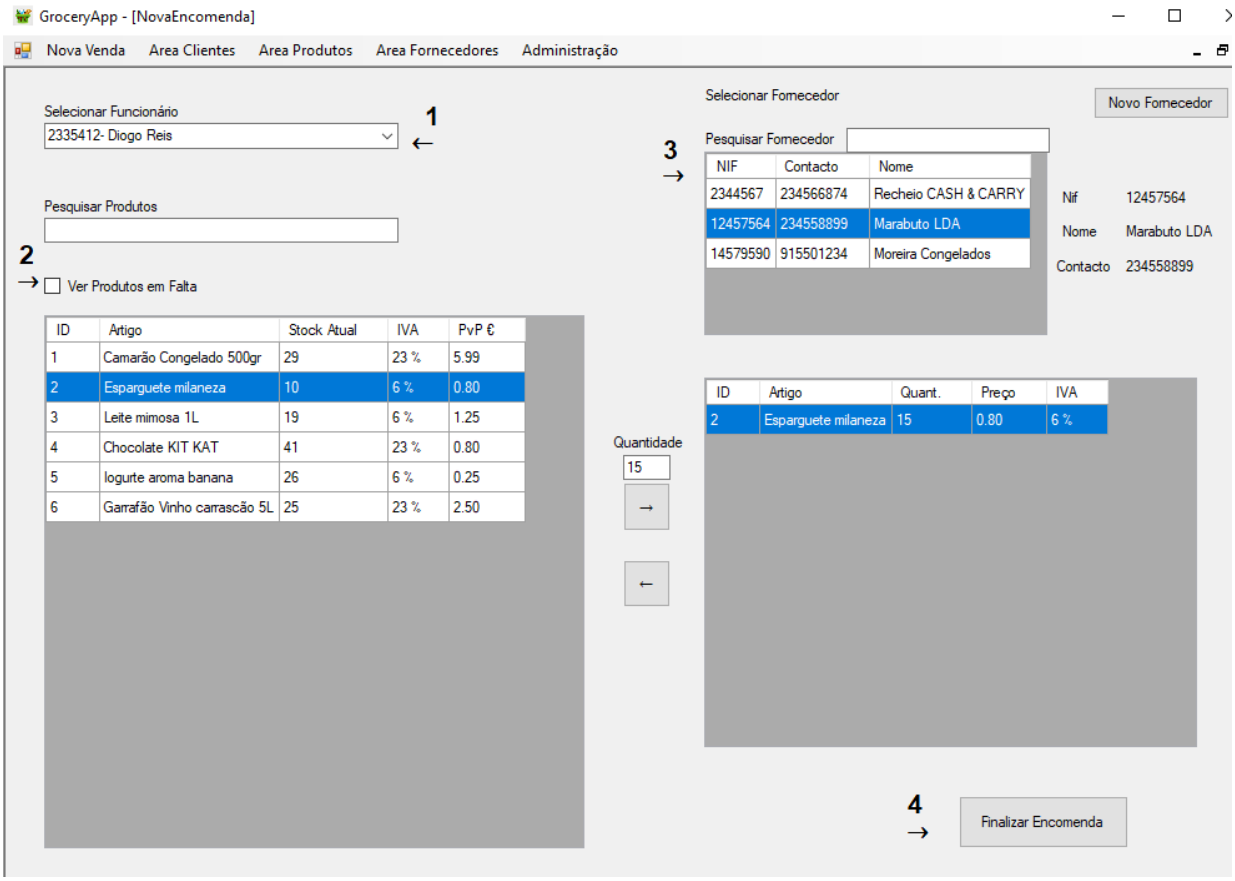
2-

```
--UDF pesquisar conta corrente por nif cliente
CREATE FUNCTION grocery.UDF_PESQUISACONTACORRENTENIF
   (@nif INT)
RETURNS DECIMAL(10,2)
AS
BEGIN
   RETURN(
   SELECT Saldo
   FROM grocery.CONTA_CORRENTE
   WHERE Nif_cliente=@nif
   )
END
GO
```

3-

```
--receber pagamento para a conta corrente
CREATE PROC grocery.sp_InsertINContaCorrente
   @idCliente INT,
   @valor DECIMAL(10,2)
AS
   UPDATE grocery.CONTA_CORRENTE
   SET IN_cash = IN_cash + @valor
   WHERE Nif_cliente=@idCliente
GO
```

6 – Nova Encomenda



1-

```
--View para obter funcionarios com nome e nif
CREATE VIEW grocery.VIEW_FUNCIONARIOS_NIFnome
AS
   SELECT CONVERT(NVARCHAR(10),func.Nif)+'- '+CONVERT(NVARCHAR(50),func.Nome) AS
Funcionario
   FROM grocery.FUNCIONARIO as func
GO
```

2-

```
--View para devolver apenas informações necessarias dos produtos para as vendas/encomendas
CREATE VIEW grocery.VIEW_PRODUTO_NAVENDA
AS
   SELECT  prod.ID AS ID, prod.Nome AS Artigo, prod.Stock_atual AS 'Stock Atual',
CONVERT(NVARCHAR(5),iva.Percentagem)+' %' AS IVA, CONVERT(NVARCHAR(30),prod.PVP) AS
'PvP €'
   FROM grocery.PRODUTO AS prod JOIN grocery.IVA AS iva ON prod.Codigo_IVA=iva.Codigo
GO
--view para devolver informaçoes de produtos com stock minimo atingido
CREATE VIEW grocery.VIEW_PRODUTO_STOCKMINIMO
AS
   SELECT  prod.ID AS ID, prod.Nome AS Artigo, prod.Stock_atual AS 'Stock Atual',
CONVERT(NVARCHAR(5),iva.Percentagem)+' %' AS IVA, CONVERT(NVARCHAR(30),prod.PVP) AS
'PvP €'
   FROM grocery.PRODUTO AS prod JOIN grocery.IVA AS iva ON prod.Codigo_IVA=iva.Codigo
   WHERE prod.Stock_atual < prod.Stock_min
GO
```

```
--UDF para pesquisar artigos pelo nome
CREATE FUNCTION grocery.UDF_PESQUISARARTIGOSNOME
   (@artigo VARCHAR(50))
RETURNS TABLE
AS
RETURN(
   SELECT *
   FROM grocery.VIEW_PRODUTO_IVA
   WHERE VIEW_PRODUTO_IVA.Artigo LIKE '%' + @artigo + '%'
   )
GO
```

3-

```
--View select fornecedores
CREATE VIEW grocery.VIEW_FORNECEDORES
AS
   Select FORNECEDOR.Nif AS NIF, FORNECEDOR.Contacto AS Contacto, FORNECEDOR.Nome AS Nome
   FROM grocery.FORNECEDOR
GO
--UDF para pesquisar fornecedores pelo nome
CREATE FUNCTION grocery.UDF_PESQUISAFORNECEDORESNOME
   (@nome VARCHAR(50))
RETURNS TABLE
AS
   RETURN(
   SELECT  FORNECEDOR.Nif AS NIF, FORNECEDOR.Contacto AS Contacto, FORNECEDOR.Nome AS
Nome
   FROM grocery.FORNECEDOR
   WHERE FORNECEDOR.Nome LIKE '%' + @nome + '%'
   )
GO
```

4-

1º

```
--Criar Encomenda
CREATE PROC grocery.sp_CriarEncomenda
   @idForn int,
   @idFunc int
AS
   INSERT INTO grocery.ENCOMENDAS (ID_forn,ID_func) VALUES (@idForn,@idFunc);
GO
```

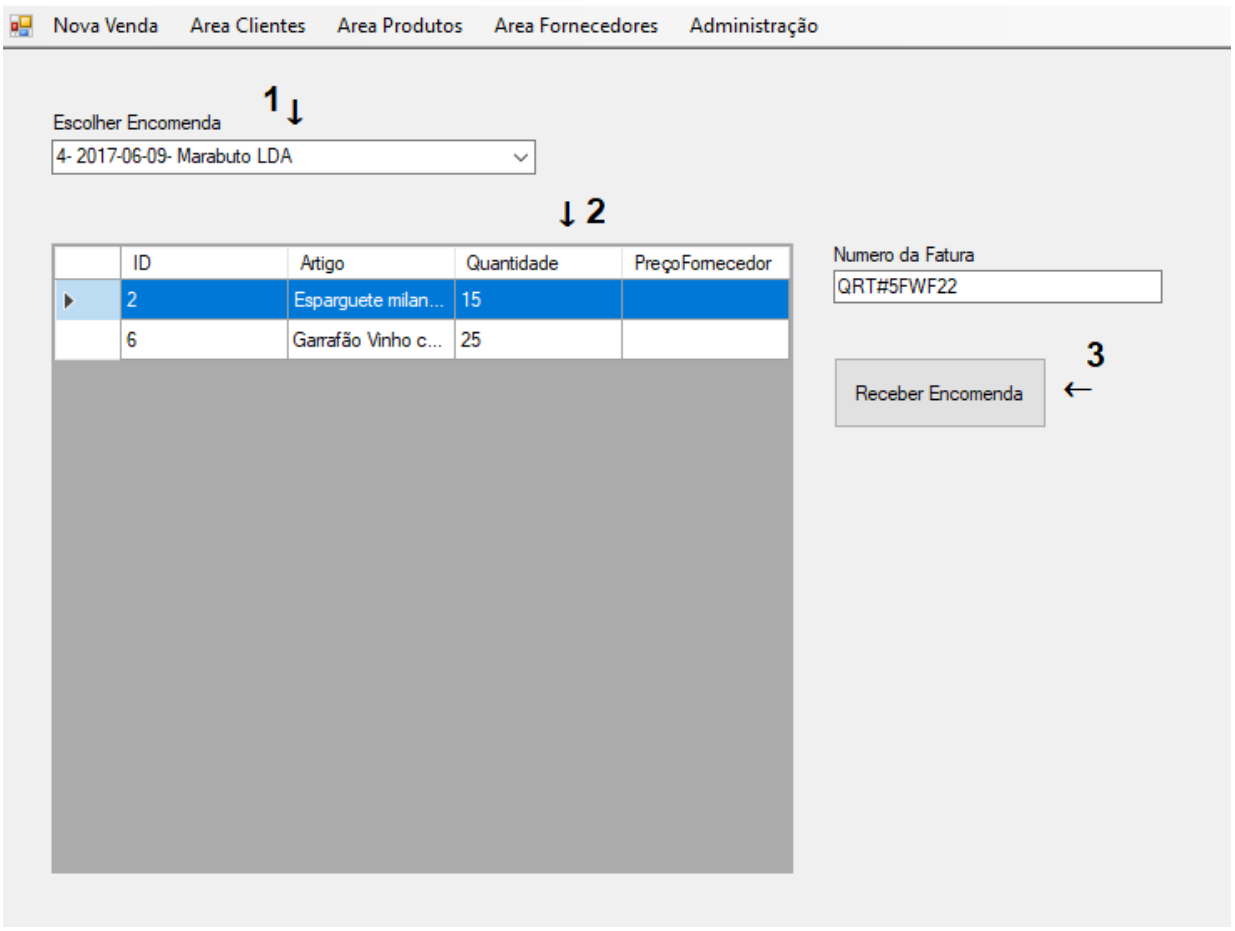
2º

```
-- View para ultimo id da encomenda
CREATE VIEW grocery.VIEW_ENCOMENDA_LAST_ID
AS
   SELECT IDENT_CURRENT('grocery.ENCOMENDAS') AS lastID
GO
```

3º

```
--Adicionar produtos a encomenda
CREATE PROC grocery.sp_AdicionarProdutosEncomenda
   @idEncomenda INT,
   @idProduto INT,
   @quantidade INT
AS
   INSERT INTO grocery.ITEM_ENCOMENDA
   (ID_encomenda,ID_produto,Quantidade)
   VALUES(@idEncomenda,@idProduto,@quantidade);
GO
```

7 – Receber Encomenda



1-

```
--View para preencher combobox de receber encomendas, Encomendas com o numero de encomenda a NULL
CREATE VIEW grocery.VIEW_ID_ENCOMENDA_FORNECEDOR
AS
   SELECT CONVERT(NVARCHAR(10),enc.ID)+'- '+CONVERT(NVARCHAR(30),enc.DataEncomenda)+'-
'+CONVERT(NVARCHAR(50),forn.Nome) AS Encomenda_fornecedor
   FROM grocery.ENCOMENDAS AS enc JOIN grocery.FORNECEDOR AS forn ON enc.ID_forn=forn.Nif
   WHERE enc.NumFatura is null
GO
```

2-

```
--UDF para preencher datagrid com produtos de uma encomenda
CREATE FUNCTION grocery.UDF_PRODUTOS_DAENCOMENDA
   (@idEncomenda int)
RETURNS TABLE
AS
RETURN(
   SELECT prod.ID AS ID, prod.Nome AS Artigo, item.Quantidade AS Quantidade, item.Preco_forn AS
PreçoFornecedor
   FROM grocery.ENCOMENDAS AS enc JOIN grocery.ITEM_ENCOMENDA AS item ON
enc.ID=item.ID_encomenda JOIN grocery.PRODUTO AS prod ON prod.ID=item.ID_produto
   WHERE enc.ID = @idEncomenda
   )
GO
```

3-

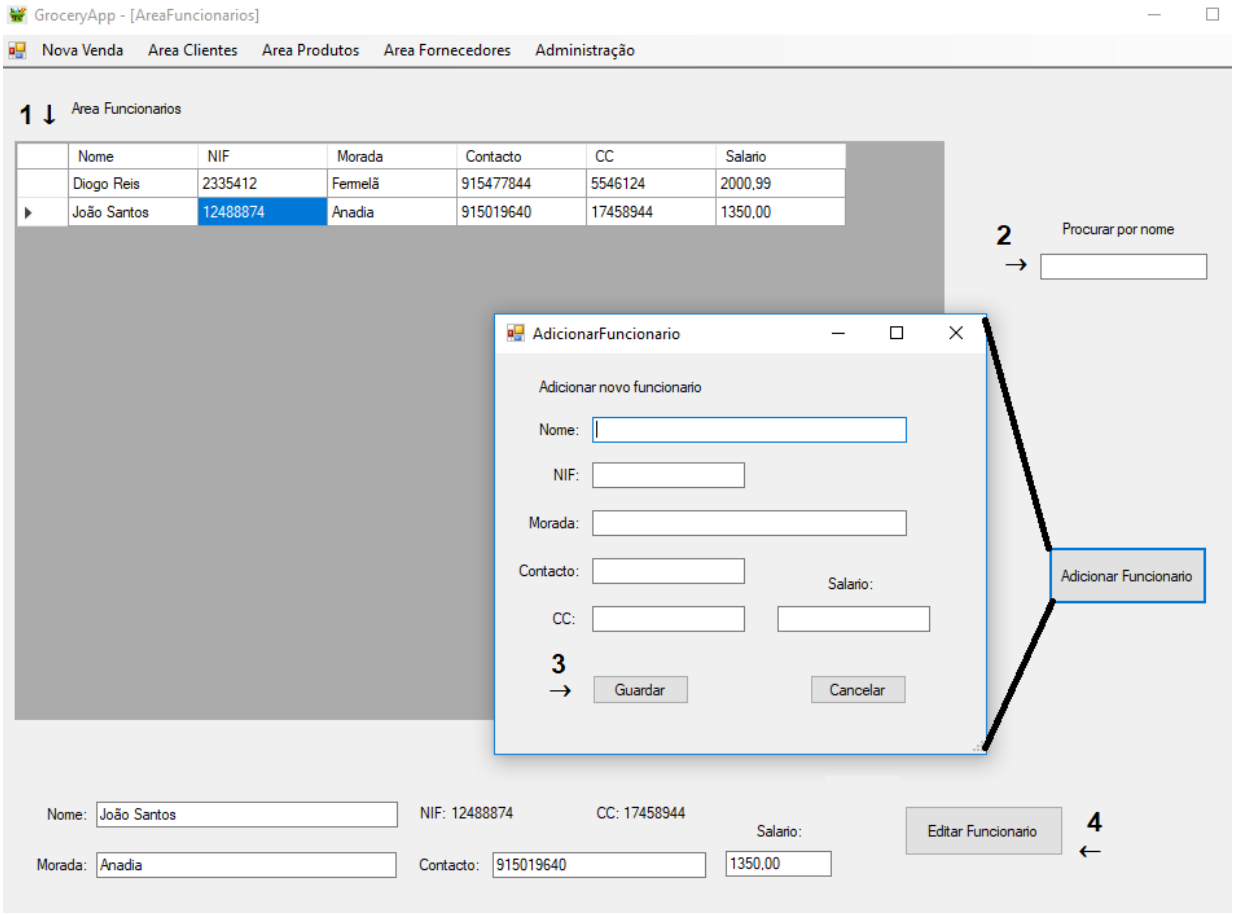
1º

```
-- Adicionar numero de fatura
CREATE PROC grocery.sp_AdicionarNumeroFatura
   @idEnc int,
   @numFat VARCHAR(50)
AS
   UPDATE grocery.ENCOMENDAS
   SET NumFatura = @numFat
   WHERE ID = @idEnc
GO
```

2º

```
--atualiza o stock dos produtos apos receber encomenda
CREATE PROC grocery.sp_UpdateStockEncomendas
   @idProduto INT,
   @quantidade INT
AS
BEGIN
   UPDATE grocery.PRODUTO SET Stock_atual = Stock_atual + @quantidade
   WHERE ID=@idProduto
END
GO
```

8 – Área de Funcionários



1-

```
--View para obter funcionarios com nome e nif
CREATE VIEW grocery.VIEW_FUNCIONARIOS_NIFnome
AS
   SELECT CONVERT(NVARCHAR(10),func.Nif)+'- '+CONVERT(NVARCHAR(50),func.Nome) AS
Funcionario
   FROM grocery.FUNCIONARIO as func
GO
```

2-

```
-- UDF para pesquisar fornecedores pelo nome
CREATE FUNCTION grocery.UDF_PESQUISAFUNCIONARIOSNOME
   (@nome VARCHAR(50))
RETURNS TABLE
AS
   RETURN(
   Select FUNCIONARIO.Nome AS Nome, FUNCIONARIO.Nif AS NIF, FUNCIONARIO.Morada AS Morada,
FUNCIONARIO.Contacto AS Contacto, FUNCIONARIO.CC AS CC, FUNCIONARIO.Salario AS Salario
   FROM grocery.FUNCIONARIO
   WHERE FUNCIONARIO.Nome LIKE '%' + @nome + '%'
   )
GO
```

3-

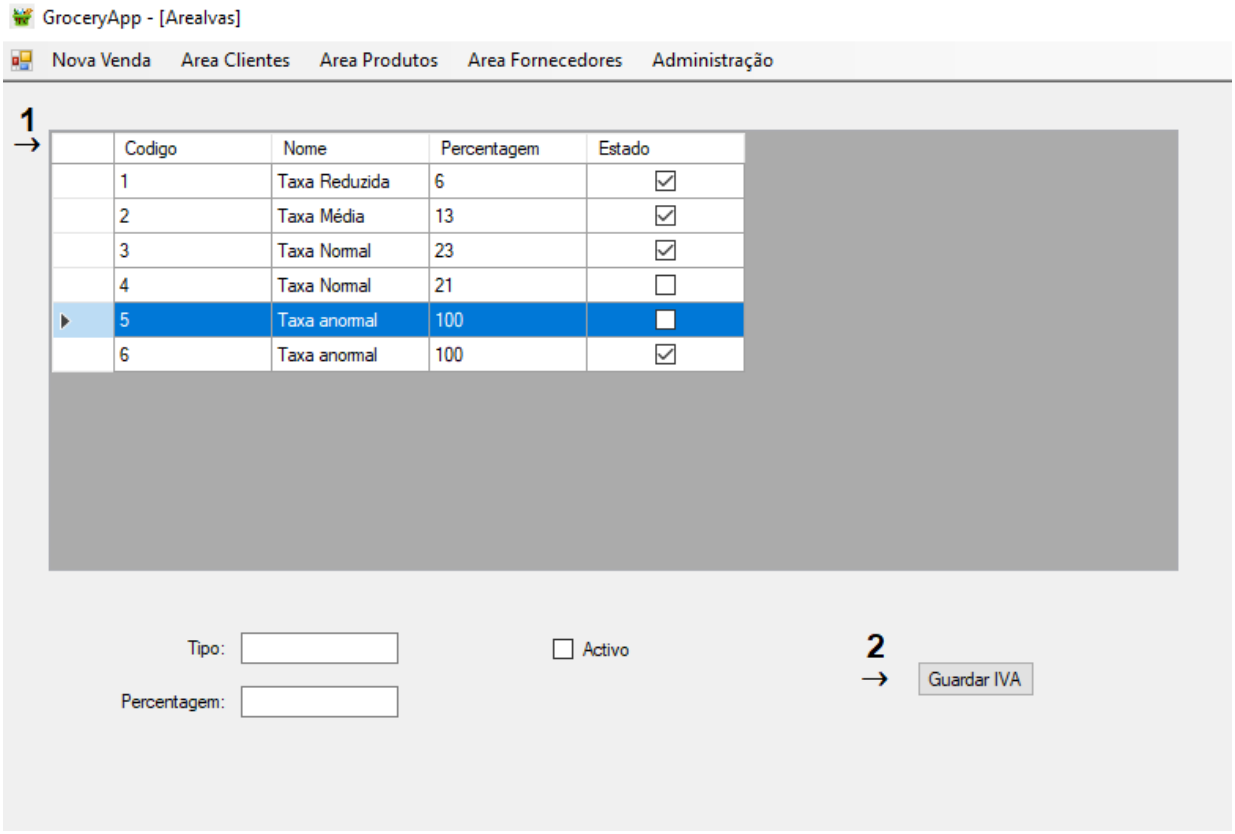
Numa form nova:

```
--Criar funcionario
CREATE PROC grocery.sp_CriarFuncionario
   @nif INT,
   @nome VARCHAR(50),
   @morada VARCHAR(50),
   @contacto VARCHAR(50),
   @cc INT,
   @salario DECIMAL(10, 2)
AS
BEGIN
   INSERT INTO grocery.FUNCIONARIO
   VALUES (@nif,@nome,@morada,@contacto,@cc,@salario)
END
GO
```

4-

```
--Update Funcionario SP
CREATE PROC grocery.sp_EditarFuncionario
    @nif INT,
    @nome VARCHAR(50),
    @morada VARCHAR(50),
    @contacto VARCHAR(50),
    @salario DECIMAL(10,2)
AS
BEGIN
    UPDATE grocery.FUNCIONARIO SET Nome = @nome, Morada = @morada, Contacto = @contacto, Salario
= @salario
    WHERE Nif = @nif
END
GO
```

9 – Área de IVAS's



1-

```
--View para IVAS activos
CREATE VIEW grocery.VIEW_IVAS_ACTIVOS
AS
   SELECT CONVERT(NVARCHAR(10),iva.Codigo)+'- '+CONVERT(NVARCHAR(30),iva.Nome)+'
'+CONVERT(NVARCHAR(10),iva.Percentagem)+'%' AS IVA
   FROM grocery.IVA as iva
   WHERE iva.Estado=1
GO
```

2-

```
--criar IVA
CREATE PROCEDURE grocery.sp_AreaIvas(
   @Nome varchar(30),
   @Percentagem int,
   @Estado bit
   )
AS
BEGIN
   INSERT INTO grocery.IVA
   VALUES(@Nome,@Percentagem,@Estado)
END
GO
```

Conclusão

A execução deste projeto permitiu a ambos os elementos uma grande evolução relativamente ao desenvolvimento de uma aplicação interativa e consequente criação, construção e manipulação da base de dados.

Através das análises de requisitos e dos diagramas realizados foi possível recolher feedback importante por parte do docente para realizar melhorias na aplicação e por isso percebemos a grande importância de todos os passos na execução do projeto. As aulas práticas foram uma excelente ferramenta para a implementação da base de dados do nosso projeto, contudo sentimos alguma dificuldade inicial relativamente à sintaxe da linguagem C#.

Em suma, a realização do projeto foi bastante importante para o desenvolvimento pessoal e como grupo pois permitiu uma grande evolução técnica nas diferentes fases do projeto e uma maior noção de trabalho em grupo e divisão de tarefas.