



Universidade de Aveiro
Mestrado Integrado em Engenharia de Computadores e
Telemática



Café - Base de Dados

Trabalho Prático - P4G8

Gabriel Vieira 68021

Tomás Rodrigues 68129

Índice

Índice de ilustrações	3
Introdução	4
Gestão de um Café	
• Análise de Requisitos	5
• Diagrama Entidade Relação	6
• Esquema Relacional da Base de Dados	7
Base de Dados – SQL	
• SQL DDL – Data Definition Language	8
• SQL DML – Data Manipulation Language	8
• Normalização	8
• Índices	9
• UDFs	9
• Stored Procedures	9
• Triggers	10
Teste da aplicação	
• Interface gráfica	10
• Funcionalidades da Interface	
1. Adicionar linha	14
2. Editar linha	15
3. Remover linha	16
Conclusão	17
Bibliografia	18
Anexos	
• Anexo I - Tabelas	19
• Anexo II - Queries	21
• Anexo III - Lista de Indices	22
• Anexo IV - Lista de Stored Procedures	23
• Anexo V – UDFs	27
• Anexo VI - Triggers	29

Índice de Ilustrações

Diagrama Entidade Relação	6
Modelo Relacional	7
Interface Gráfica	
• Menu Principal	10
• Lista de Funcionários	10
• Lista de Clientes	11
• Lista de Fornecedores	11
• Lista de Produtos	12
• Lista de Encomendas	12
• Lista de Compras	13
• Turnos Disponíveis	13
• Lista de Reclamações	14
Funcionalidades	
• Adicionar Linha	14
• Editar Linha	15
• Remover Linha	16

Introdução

O nosso trabalho tem como objetivo a gestão de um café e tudo o que lhe diz respeito, nomeadamente, clientes, funcionários, fornecedores, encomendas e produtos.

Para cada cliente, funcionário, fornecedor, etc, o sistema permite adicionar os dados pessoais, editar e removê-los. É ainda possível também adicionar compras efetuadas por um cliente, bem como listar todos os produtos existentes no stock do café.

Inicialmente, foi implementado o diagrama entidade-relação correspondente, referenciando todas as entidades existentes e respetivos atributos e, ainda, a relação entre as diferentes entidades. Inicialmente, o processo não foi difícil, visto que a nossa base de dados final não sofreu grande alteração em relação à inicial e ao seu diagrama entidade-relação.

Depois do DER devidamente implementado, foi feita a conversão para o modelo relacional

Finalmente foi implementada a base de dados (SQL Server). Foram usados *stored procedures* para inserção, atualização, remoção e consulta e *triggers* para controlar e atualizar o *stock* de produtos. Foi ainda implementada a interface, fazendo a respetiva ligação à BD.

Análise de Requisitos

- O café é frequentado por pessoas, que podem ser Funcionários, Clientes ou Fornecedores.
- O Café emprega Funcionários caracterizados por um nome, morada, email, telefone, número e salário.
- Os funcionários têm também um horário que é caracterizado por turno e número de horas. Um funcionário pode ter também um ou mais supervisores.
- Os clientes têm um nome, NIF, email, morada e telefone. Os clientes podem efetuar uma ou mais compras.
- Um funcionário serve vários produtos que são caracterizados por um ID, preço, família e nome.
- Um funcionário regista as compras efetuadas pelos clientes que são caracterizadas por nº compra, data de compra, forma de pagamento e nome dos produtos.
- As compras podem conter um ou mais produtos, sendo registado o número de unidades.
- Os fornecedores sendo caracterizados por um código interno, fax e condições de pagamento, entregam um ou mais produtos em encomendas tendo estas últimas um número, a data da encomenda e a quantidade de produtos.
- Um cliente pode fazer reclamações que são caracterizadas por um nº de reclamação e motivo.

Diagrama Entidade-Relação

De acordo com a análise de requisitos mencionada na página anterior, obteve-se o seguinte Diagrama Entidade-Relação:

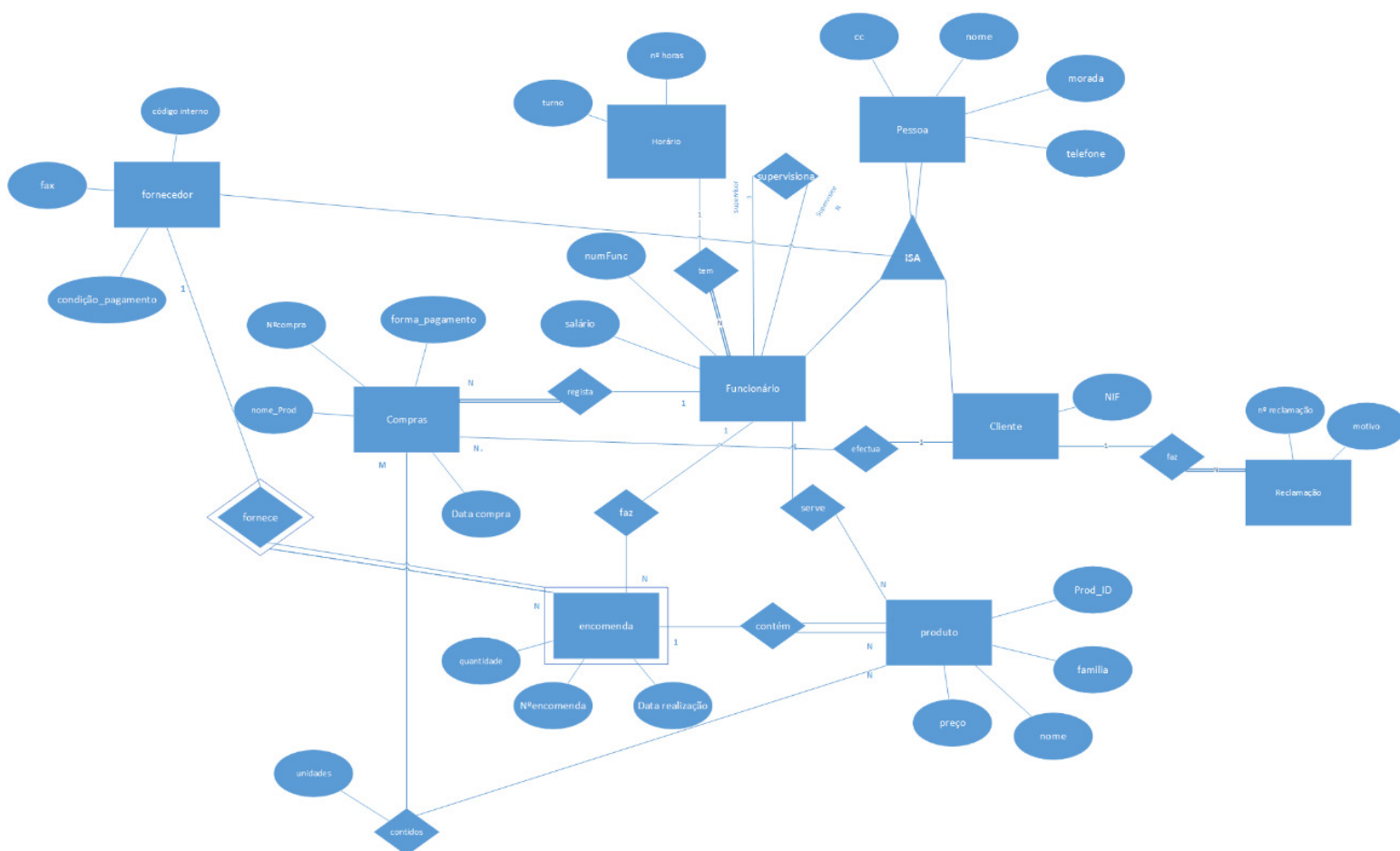


Figura 1 – Diagrama Entidade Relação

No diagrama pode-se ver as diferentes entidades que representam as classes existentes (pessoa, cliente, funcionário, fornecedor, encomenda, produto, compras, reclamação, horário(turnos), e contidos(usada para gerir o stock e vendas efectuadas), os atributos das entidades que revelam as propriedades das mesmas e ainda a relação entre diferentes entidades, o que demonstra a interação entre duas entidades, contendo a obrigatoriedade e cardinalidade.

Esquema Relacional da Base de Dados

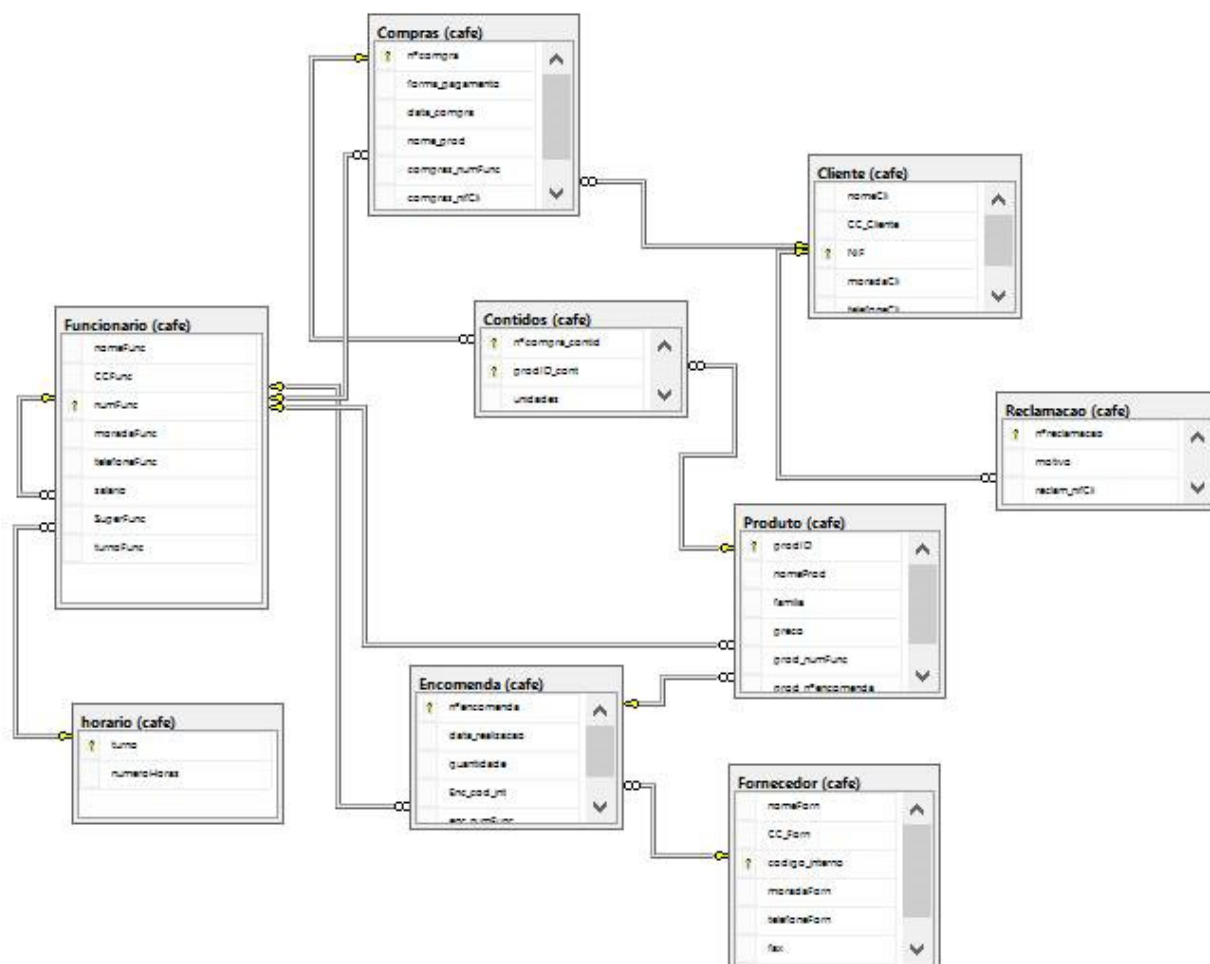


Figura 2 – Modelo Relacional

Para criação do esquema representado na Ilustração 3, tivemos em conta as regras estudadas nas aulas teóricas e práticas:

- Relação 1-1: escolher a relação com participação total e incluir nesta relação a chave primária da outra relação, como chave estrangeira;
- Relação 1-N: escolher a relação N, e incluir nessa relação a chave primária da outra relação, como chave estrangeira. Incluir ainda na relação N os atributos do relacionamento;
- Relação N-M: criar uma nova relação, e incluir nessa relação as chaves primárias das relações que nela participam, como chaves estrangeiras. Incluir ainda os atributos do relacionamento na nova relação;
- Especialização: para fazer a tradução de DER para modelo relacional, há dois métodos possíveis. Neste caso, optamos por criar uma nova relação para a entidade de maior nível, com a chave primária dessa tabela e ainda criar uma nova relação por cada

entidade de nível inferior, incluindo nestas relações a chave primária da relação anterior.

Base de Dados – SQL

SQL DDL – Data Definition Language

Linguagem usada para definição de estruturas de dados. Uma vez compilados, os parâmetros são guardados num conjunto de arquivos.

Usando a linguagem SQL DDL foram criadas tabelas auxiliares (Anexo I), baseadas no modelo relacional, incluindo as designações de chave primária e estrangeira e ainda algumas restrições.

SQL DML – Data Manipulation Language

Linguagem usada para ver, inserir, alterar e apagar tabelas/tuplos da base de dados. Principais comandos: SELECT, INSERT, UPDATE, DELETE.

Usando a linguagem SQL DDL definimos várias queries (Anexo II) utilizadas para inserir, alterar ou apagar dados da base de dados.

Normalização

Série de passos que permitem um armazenamento consistente e um eficiente acesso aos dados da base de dados. Esses passos reduzem a redundância de dados e a inconsistência dos mesmos.

Uma tabela está na Primeira Forma Normal (1FN) se não possuir atributos multivalor ou compostos nem nested relations (relações dentro de relações).

Uma relação está na Segunda Forma Normal (2FN) se estiver na 1FN e cada atributo não chave for dependente da chave primária e não de apenas parte da chave.

Uma relação está na Terceira Forma Normal (3FN) se estiver na 2FN e cada atributo não-chave da relação não possuir dependência transitiva, para cada chave candidata da relação.

De modo a estar na Forma Normal de Boyce-Codd (BCNF), uma tabela está na 3FN e todo o atributo não chave depende funcionalmente de toda a chave primária, ou seja, não há dependências entre atributos não-chave.

Após a criação das tabelas, estas já se encontravam na BCNF, logo não foram feitas mais alterações na forma da base de dados.

Índices

Os índices são estruturas de dados que oferecem uma segunda forma de acesso aos dados, melhorando o tempo de pesquisa nas tabelas.

Os campos mais pesquisados são índices clustered, temos alguns índices non-clustered, como por exemplo, a data das compras. Todos os índices podem ser visualizados no Anexo III.

Stored Procedures

Conjunto de comandos que são armazenados na base de dados, que são bastante úteis uma vez que permitem parâmetros de entrada e saída, os quais podemos utilizar para inserir, alterar, apagar, consultar tabelas da nossa BD.

Neste sistema stored procedures foram utilizados em todas as queries (Anexo IV) à base de dados, sendo um meio eficiente porque previnem SQL injections e uma melhor gestão de transações.

UDFs

São vistas que acresce o facto de aceitar parâmetros, algo impossível em vistas. Oferecem os mesmos benefícios das vistas pois podem ser utilizados como fonte de dados. Têm os mesmos benefícios dos stored procedures, pois são igualmente compilados e otimizados, podendo ser utilizadas para incorporar lógica complexa dentro de uma consulta

Existem 3 tipos de UDFs:

- **Escalar** – aceitam múltiplos parâmetros, retornam um único valor e podem ser utilizados dentro de qualquer expressão T-SQL incluindo check constraint
- **Inline table-valued** – similares a vistas, pois ambas são wrappers para construções SELECT e em relação também às vistas acresce o facto de suportar parâmetros de entrada
- **Multi-Statement table-valued** – Combinação das capacidades das UDFs escalares e inline table-valued. Cria uma tabela variable, introduz-lhe tuplos e retorna-a

Triggers

Tipo especial de procedimento armazenado, que é executado sempre que há uma tentativa de modificar os dados de uma tabela que é protegida por ele, ajudando assim a manter a consistência dos dados ou a propagar alterações num determinado dado de uma tabela para outra(s).

Neste projeto foram vários triggers (Anexo V), como por exemplo, impedir que os funcionários gestores sejam apagados da base de dados.

Teste da aplicação

A aplicação apresenta um layout simples, sendo fácil chegar à solução pretendida para uma determinada solução e prevendo o seu comportamento. Através da interação com a interface gráfica conseguimos editar, remover e adicionar elementos à Base de dados

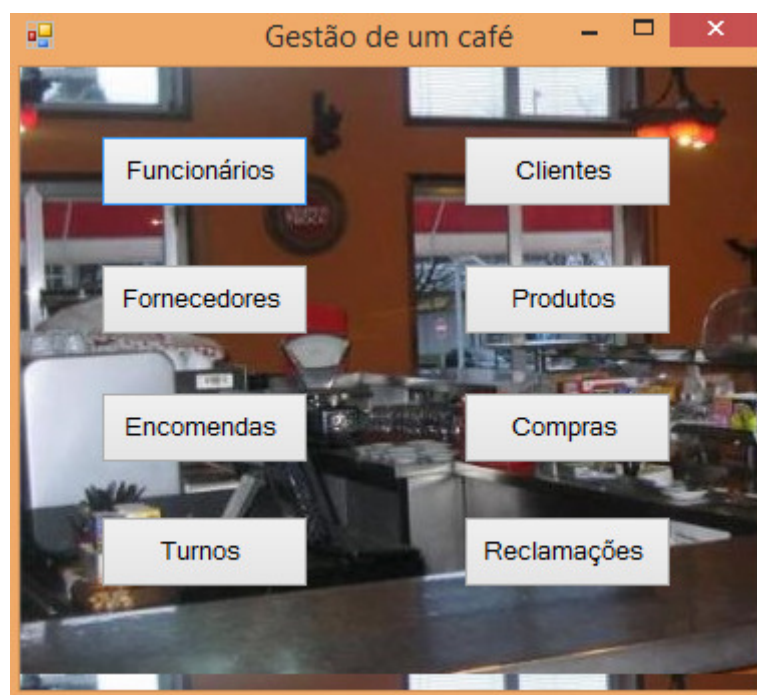


Figura 3 – Interface Gráfica – Menu Principal

Lista de Funcionários

Ficheiro

Menu Principal

Clientes

Fornecedores

Produtos

Encomendas

Compras

Tumos

Reclamações

adriana 123123
adriana 92744
André 12343424
Andreia 16742349
António 53452345
Chico 19800341
Dona Alcina 78965456
Filipa 19841305
Filipe Almeida 123456
Gabriel Vieira 14472124
Jaqueline 56898052
Joana 12342345
João 13427890
Luis Semedo 12340001
Tiago 23413212
Tomas 12345321

Nome
António

CC
53452345

Nº Funcionário
2

Morada
Rua Assada, Aveiro

Telefone
234634345

Salário
2500,00 €

Nº Supervisor
0

Tumo
manha

Adicionar

Editar

Remover


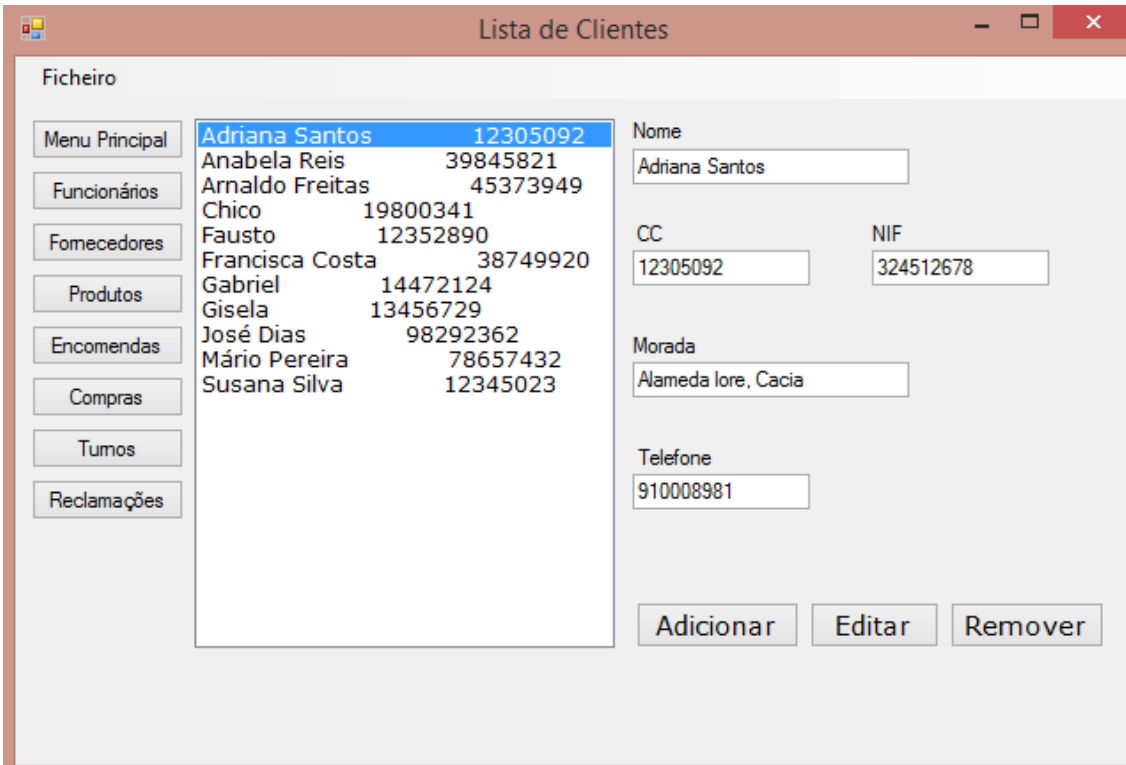
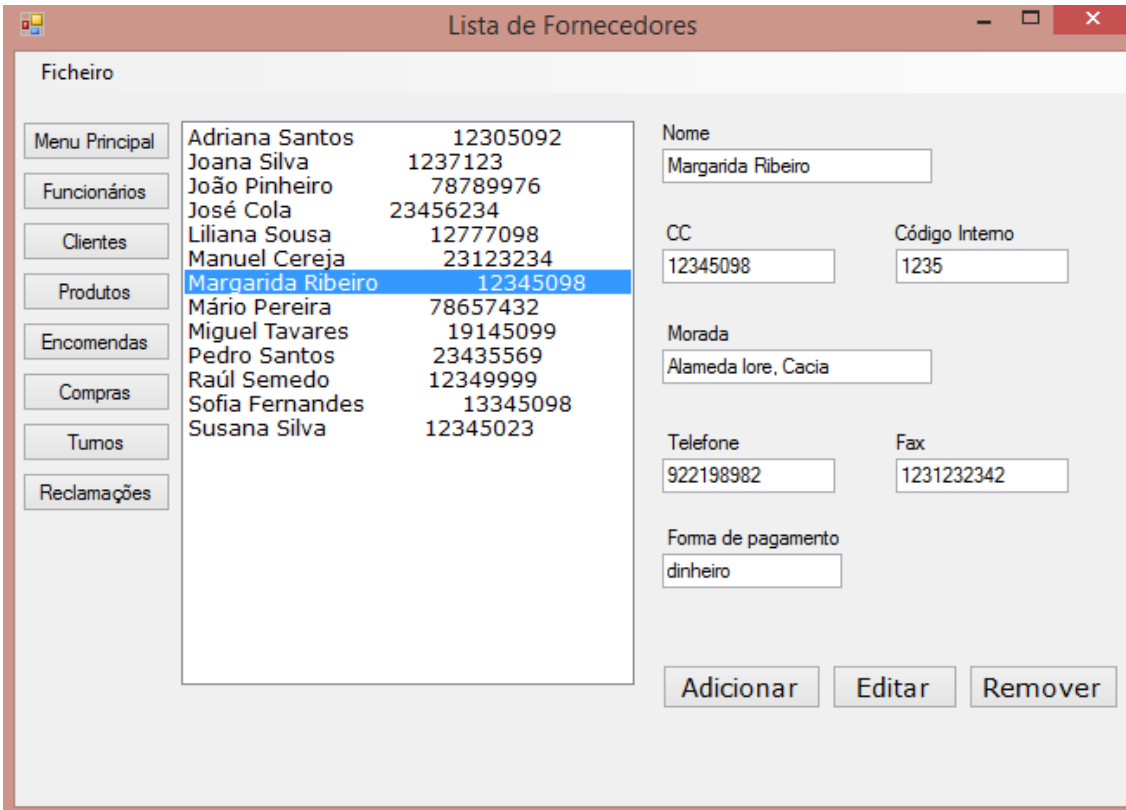


Figura 4 – Interface Gráfica – Lista de Funcionários



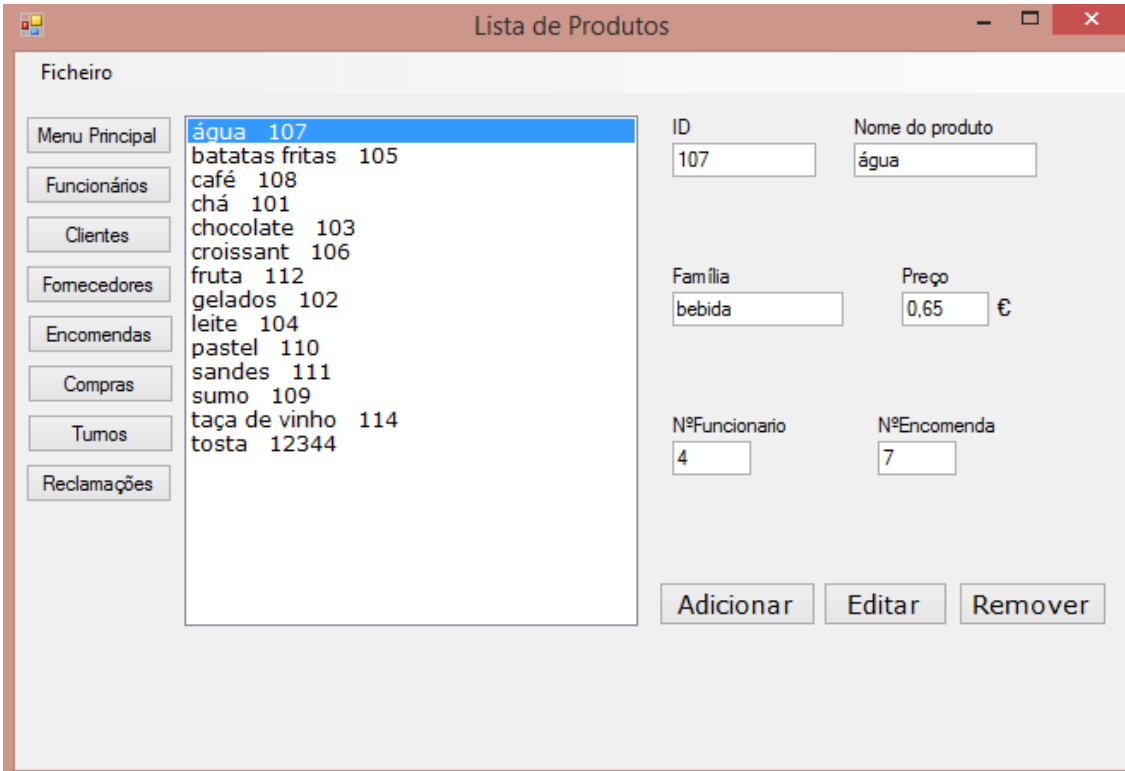
Nome	CC	NIF
Adriana Santos	12305092	324512678

Figura 5 – Interface Gráfica – Lista de Clientes



Nome	CC	Código Interno
Margarida Ribeiro	12345098	1235

Figura 6 – Interface Gráfica – Lista de Fornecedores



Lista de Produtos

Ficheiro

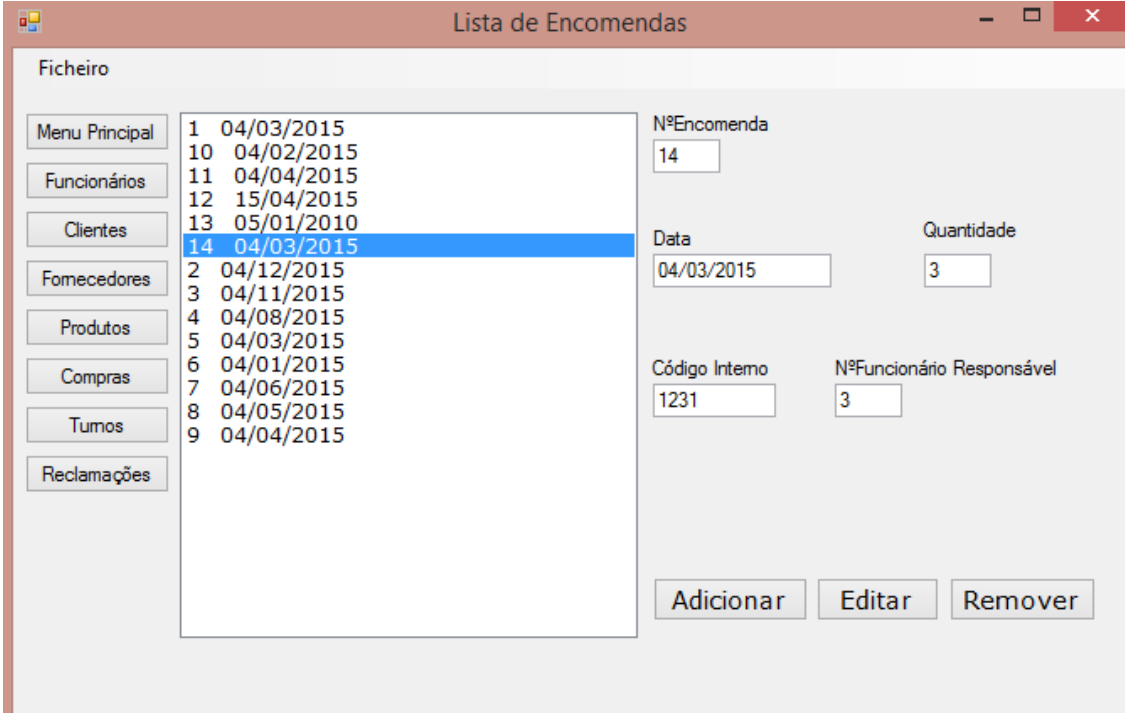
Menu Principal
Funcionários
Clientes
Fornecedores
Encomendas
Compras
Tumos
Reclamações

água 107
batatas fritas 105
café 108
chá 101
chocolate 103
croissant 106
fruta 112
gelados 102
leite 104
pastel 110
sandes 111
sumo 109
taça de vinho 114
tosta 12344

ID: 107
Nome do produto: água
Família: bebida
Preço: 0,65 €
NºFuncionario: 4
NºEncomenda: 7

Adicionar Editar Remover

Figura 7 – Interface Gráfica – Lista de Produtos



Lista de Encomendas

Ficheiro

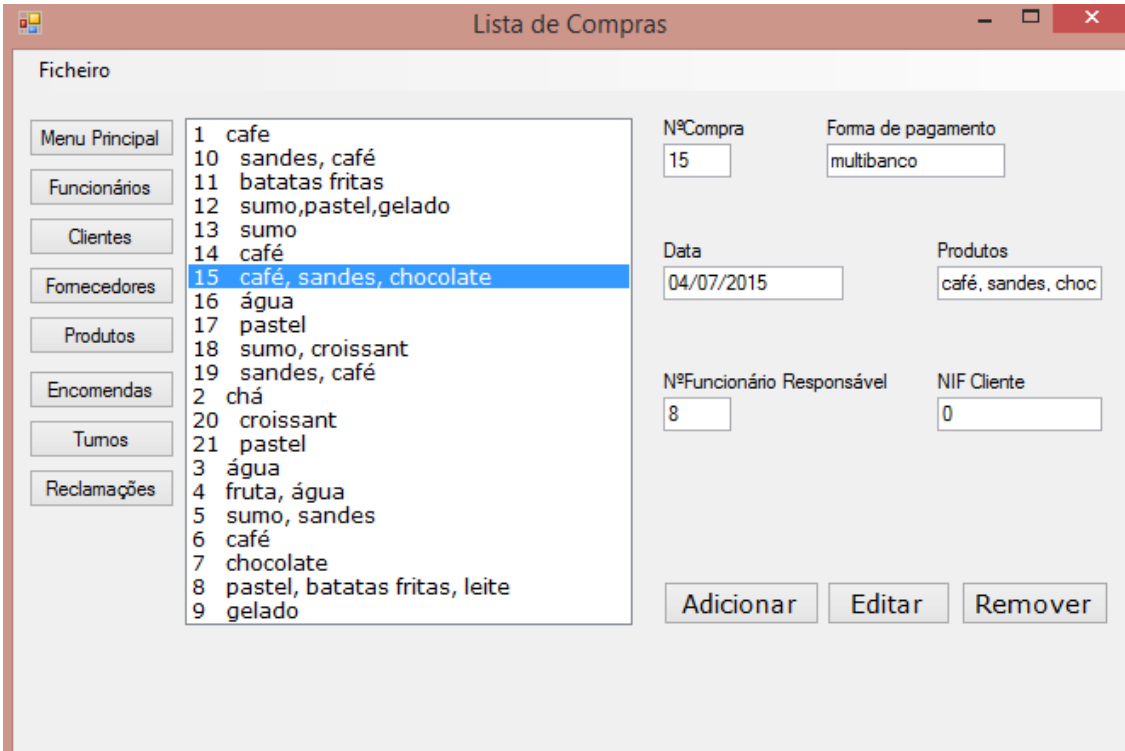
Menu Principal
Funcionários
Clientes
Fornecedores
Produtos
Compras
Tumos
Reclamações

1 04/03/2015
10 04/02/2015
11 04/04/2015
12 15/04/2015
13 05/01/2010
14 04/03/2015
2 04/12/2015
3 04/11/2015
4 04/08/2015
5 04/03/2015
6 04/01/2015
7 04/06/2015
8 04/05/2015
9 04/04/2015

NºEncomenda: 14
Data: 04/03/2015
Quantidade: 3
Código Interno: 1231
NºFuncionário Responsável: 3

Adicionar Editar Remover

Figura 8 – Interface Gráfica – Lista de Encomendas



Lista de Compras

Ficheiro

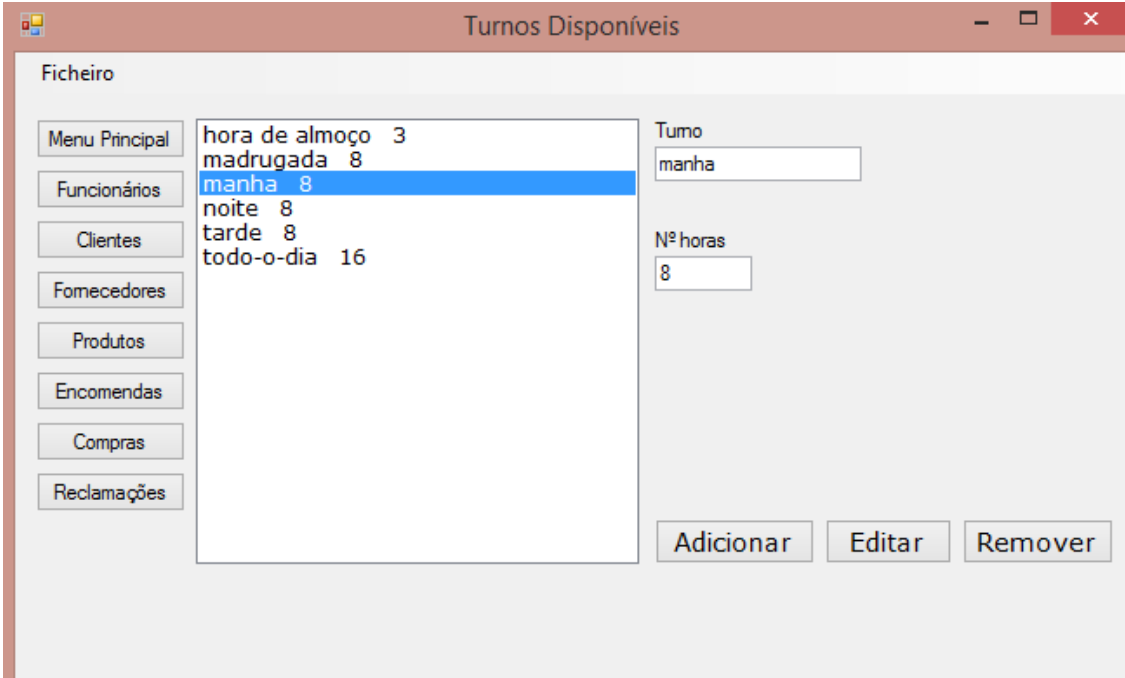
Menu Principal
Funcionários
Clientes
Fornecedores
Produtos
Encomendas
Turnos
Reclamações

1 café
10 sandes, café
11 batatas fritas
12 sumo, pastel, gelado
13 sumo
14 café
15 **café, sandes, chocolate**
16 água
17 pastel
18 sumo, croissant
19 sandes, café
2 chá
20 croissant
21 pastel
3 água
4 fruta, água
5 sumo, sandes
6 café
7 chocolate
8 pastel, batatas fritas, leite
9 gelado

NºCompra: 15
Forma de pagamento: multibanco
Data: 04/07/2015
Produtos: café, sandes, choc
NºFuncionário Responsável: 8
NIF Cliente: 0

Adicionar Editar Remover

Figura 9 – Interface Gráfica – Lista de Compras



Turnos Disponíveis

Ficheiro

Menu Principal
Funcionários
Clientes
Fornecedores
Produtos
Encomendas
Compras
Reclamações

hora de almoço 3
madrugada 8
manha 8
noite 8
tarde 8
todo-o-dia 16

Turno: manha
Nº horas: 8

Adicionar Editar Remover

Figura 10 – Interface Gráfica – Turnos Disponíveis

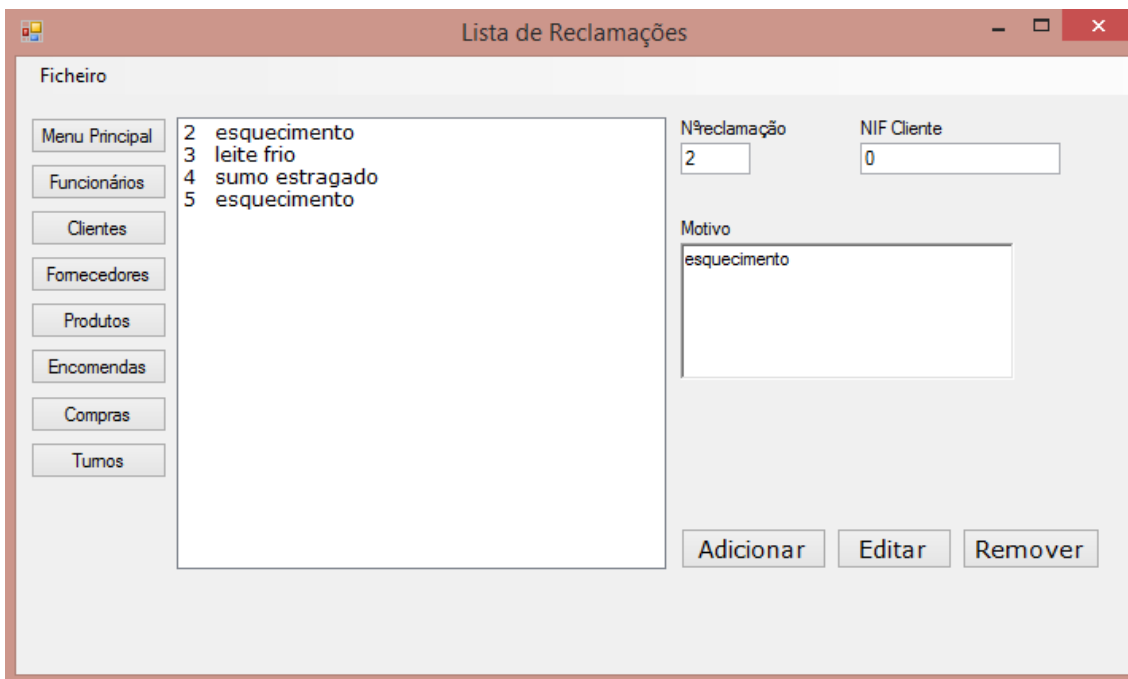


Figura 11 – Interface Gráfica – Lista de Reclamações

Funcionalidades

Adicionar cliente

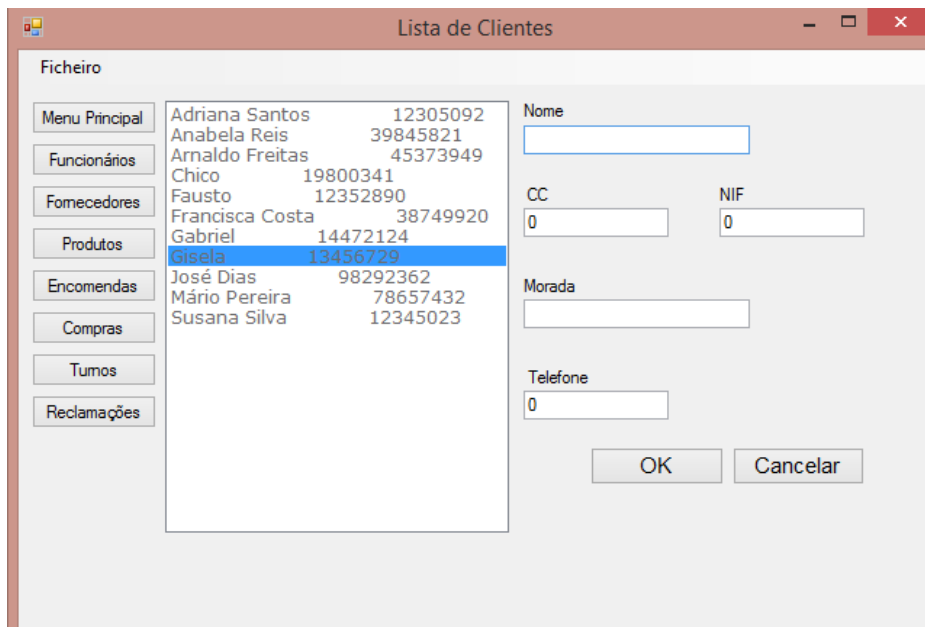


Figura 12 – Interface Gráfica – Adicionar Cliente

Como podemos ver nas imagens acima ao adicionar uma linha a “list box” fica com menos opacidade de forma a dar feedback ao utilizador e os campos a adicionar ficam vazios ou com zeros no caso de serem números.

Editar cliente

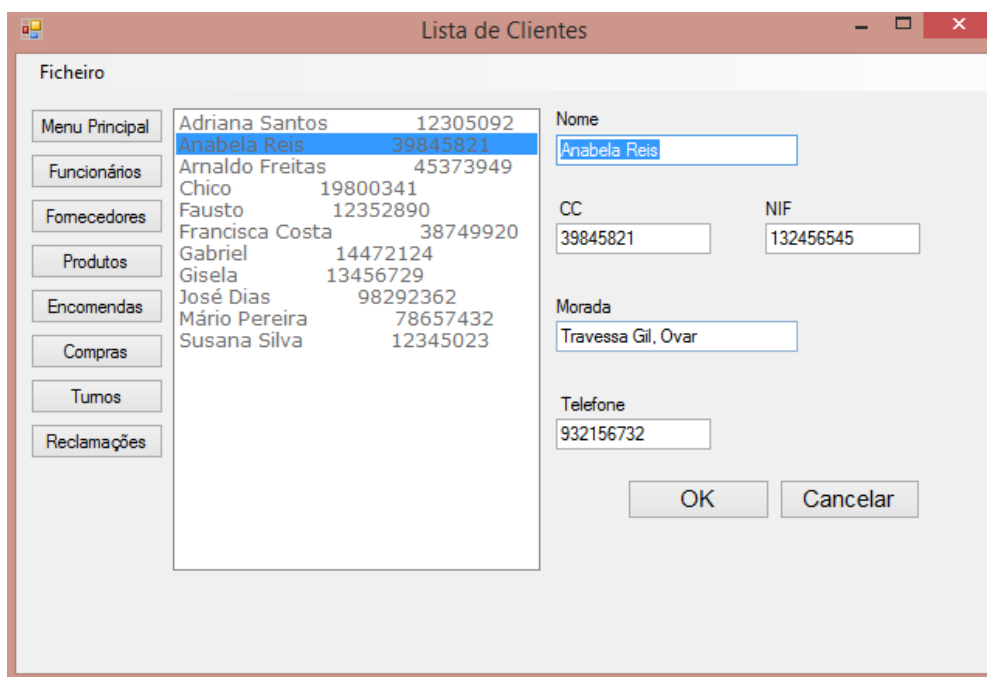


Figura 13 – Interface Gráfica – Editar Cliente

Ao editar uma linha vê-se novamente a opacidade a diminuir, o lock da “list box” e o texto a editar fica automaticamente selecionado de forma a indicar ao utilizador que vai escrever no sítio em questão.

Remover produto

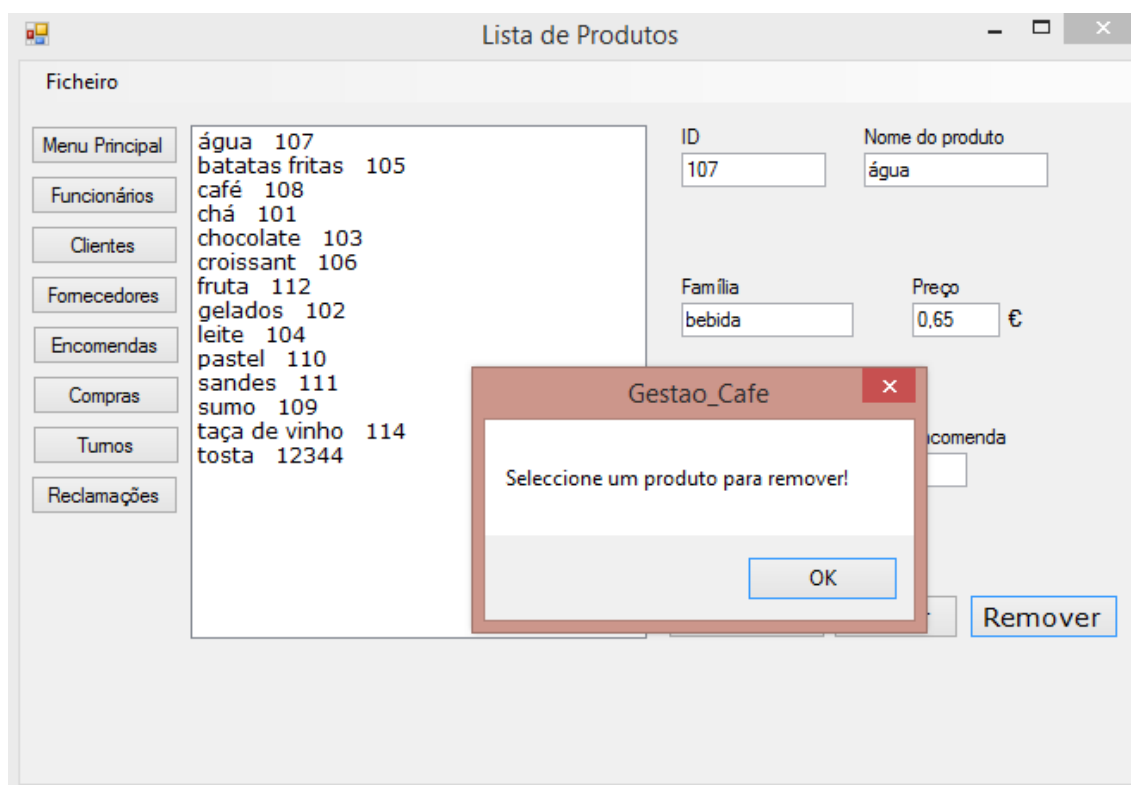


Figura 14 – Interface Gráfica – Remover Produto

Se o utilizador tentar remover uma linha sem antes ter seleccionado nada, surge uma mensagem de erro para o utilizador. Todas as remoções são feitos da base de dados tendo em conta as tabelas relacionadas com a mesma.

CONCLUSÃO

No fim deste projeto, podemos concluir que temos uma interface de fácil utilização, embora alguns aspectos tenham ficado por desenvolver (parte da pesquisa) ao qual daria uma base de dados mais completa e próxima do mundo real. A linguagem usada para o desenvolvimento da interface gráfica foi Visual Basic.

Foi-nos possível, com este trabalho, ter uma visão mais global de uma base de dados, passando pela sua implementação, até à execução final e integração com uma interface, tendo-nos facultado experiência que não se adquire lendo acerca de bases de dados.

BIBLIOGRAFIA

R. Elmasri, S. Navathe, “Fundamentals of Database Systems”, 6th Edition, 2011, Addison Wesley

Alan Dix, Janet Finlay, Gregory Abowd, Russell Beale, Human-Computer Interaction, 3rd edition, Prentice Hall, 2004

<http://msdn.microsoft.com/pt-br/sqlserver/>

<http://msdn.microsoft.com/en-us/library/k50ex0x9.aspx>

<http://moodle.ua.pt/course/view.php?id=6299>

Anexos

Anexo I - Tabelas

```
drop table cafe.Funcionario;
create table cafe.Funcionario(
    nomeFunc varchar(20) not null,
    CCFunc int,
    numFunc int,
    moradaFunc varchar(20),
    telefoneFunc int,
    salario decimal(10,2) check(salario > 0),
    SuperFunc int,
    turnoFunc varchar(20),

    primary key(numFunc),
    foreign key(turnoFunc) references cafe.horario(turno)
        on delete set null on update cascade,
);

create table cafe.Cliente(
    nomeCli varchar(20) not null,
    CC_Cliente int,
    NIF int,
    moradaCli varchar(20),
    telefoneCli int,

    primary key(NIF),
);

create table cafe.Fornecedor(
    nomeForn varchar(20) not null,
    CC_Forn int,
    codigo_interno varchar(20),
    moradaForn varchar(20),
    telefoneForn int,
    fax int,
    forma_pag varchar(20),

    primary key(codigo_interno),
);

create table cafe.horario(
    turno varchar(20),
    numeroHoras int,

    primary key(turno),
    unique(numeroHoras),
);

drop table cafe.Encomenda;
alter table cafe.Encomenda(
    nºencomenda int,
    data_realizacao date,
    quantidade int,
    Enc_cod_int varchar(20),
    enc_numFunc int,
    prods varchar(100),

    primary key(nºencomenda),
    foreign key(Enc_cod_int) references cafe.Fornecedor(codigo_interno)
```

```
        on delete set null on update cascade,

foreign key(enc_numFunc) references cafe.Funcionario(numFunc)
        on delete set null on update cascade,
);

create table cafe.Compras(
    n°compra int,
    forma_pagamento varchar(20),
    data_compra date,
    nome_prod varchar(35),
    compras_numFunc int,
    compras_nifCli int,

    primary key(n°compra),
    foreign key(compras_numFunc) references cafe.Funcionario(numFunc)
        on delete set null on update cascade,
    foreign key(compras_nifCli) references cafe.Cliente(NIF)
        on delete set null on update cascade,
);

create table cafe.Produto(
    prodID varchar(20),
    familia varchar(15),
    nomeProd varchar(20),
    preco decimal(10,2) check (preco > 0),
    prod_numFunc int,
    prod_nifCli int,
    prod_numCompra int,
    prod_n°encomenda int,

    primary key(prodID),
    foreign key(prod_numFunc) references cafe.Funcionario(numFunc),

    foreign key(prod_nifCli) references cafe.Cliente(NIF)
        on delete set null on update cascade,
    foreign key(prod_numCompra) references cafe.Compras(n°compra),
    foreign key(prod_n°encomenda) references cafe.Encomenda(n°encomenda)
        on delete set null on update cascade,
);

create table cafe.Reclamacao(
    n°reclamacao int,
    motivo varchar(15),
    reclam_nifCli int,

    primary key(n°reclamacao),
    foreign key(reclam_nifCli) references cafe.Cliente(NIF)
        on delete set null on update cascade,
);

create table cafe.Contidos(
    n°compra_contid int,
    prodID_cont varchar(20),
    unidades int,

    primary key(n°compra_contid, prodID_cont),
    foreign key(n°compra_contid) references cafe.Compras(n°compra),
    foreign key(prodID_cont) references cafe.Produto(prodID)
);
```

Anexo II – Querys

```
-- 1. Lista com todos os funcionarios
select * from cafe.Funcionario

--2. Lista com os funcionários supervisionados
select * from cafe.Funcionario
where SuperFunc is not null

--3. Lista com todos os funcionários supervisionados pela Dona Alcina
select nomeFunc as Nome_Funcionario, numFunc as Numero, SuperFunc as Supervisor
from (cafe.Funcionario Join ((select numFunc as Supervisor
                             from cafe.Funcionario
                             where nomeFunc='Dona Alcina')) as
Sp
                             on Funcionario.SuperFunc =
Sp.Supervisor);

-- 4. Lista de Clientes que efectuaram compras
select nomeCli as NomeCliente, NIF
from cafe.Cliente Join cafe.Compras on NIF=compras_nifCli

-- 5. Lista de Clientes que não efectuaram qualquer compra
select nomeCli as NomeCliente, NIF
from cafe.Cliente Left Outer Join cafe.Compras on NIF=compras_nifCli
where compras_nifCli is null

-- 6. Lista de todos os Fornecedores
select * from cafe.Fornecedor

-- 7. Lista de Fornecedores que também são clientes
select nomeForn as Nome, moradaForn as Morada
from cafe.Fornecedor, cafe.Cliente
where CC_Forn=CC_Cliente

--8. Lista de Funcionários que também são clientes
select nomeFunc as Nome_Funcionário, moradaFunc as Morada
from cafe.Funcionario, cafe.Cliente
where CCFunc=CC_Cliente

--9. Numero compras registadas pelo funcionário numero 4
select nomeFunc as Nome_Funcionário, count(nºcompra) as pedidos_atendidos
from cafe.Funcionario Join cafe.Compras on numFunc=compras_numFunc
where numFunc=4
group by nomeFunc

--10. Quantidade de compras efectuadas por dinheiro, por Cliente
select nomeCli as Nome_Cliente, NIF, count(nºcompra) as Compras_efectuadas
from cafe.Cliente Join cafe.Compras on NIF=compras_nifCli
where forma_pagamento='dinheiro'
group by nomeCli, NIF

-- 11. Numero total de vendas por produto
select nomeProd, count(unidades) as TotalProdutoVendido
from cafe.Contidos Join cafe.Produto on prodID_cont=prodID
group by nomeProd
```

Anexo III – Indices

```
--index's

drop index nameCli on cafe.Cliente

create index nameCli on cafe.Cliente(nomeCli)

select nomeCli, NIF
from cafe.Cliente
order by nomeCli

drop index nameFor on cafe.Fornecedor

create index nameFor on cafe.Fornecedor(nomeForn)
select nomeForn, CC_Forn

from cafe.Fornecedor
order by nomeForn asc

drop index nameFunc on cafe.Funcionario

create index nameFunc on cafe.Funcionario(nomeFunc)
select nomeFunc, CCFunc

from cafe.Funcionario
order by nomeFunc asc

create index nameProd on cafe.Produto(nomeProd)
create index nameReclama on cafe.Reclamacao(motivo)

-- indice para procurar os funcionarios com determinado turno
create index TurnoFuncionarios on cafe.Funcionario(turnoFunc)

select nomeFunc, CCFunc, turnoFunc
from cafe.Funcionario
where turnoFunc != 'noite'
```

Anexo IV – Stored Procedures (apenas alguns)

```
--sp
-- stored procedures para adicionar funcionarios, clientes, fornecedores, ... --
drop procedure cafe.SearchFunc

create procedure cafe.SearchFunc(@filter varchar(20))
as
begin
    select *from cafe.Funcionario
    where nomeFunc like +@filter+'%'
end

exec cafe.SearchFunc 'G'

drop proc returnFunc

go
alter proc returnFunc

    @nome varchar(20) output

as

    select distinct f.nomeFunc, f.CCFunc, f.moradaFunc, f.telefoneFunc,
f.salario, f.turnoFunc
    from cafe.funcionario f
    where f.nomeFunc = @nome

go
exec returnFunc 'Tomas'

go
create proc returnCliente

    @nome varchar(20) output

as

    select c.nomeCli, c.CC_Cliente, c.moradaCli, c.telefoneCli
    from cafe.Cliente c
    where c.nomeCli = @nome

go

go
create proc returnCompras

    @num int output

as

    select c.nºcompra, c.forma_pagamento, c.data_compra , c.nome_prod
    from cafe.Compras c
    where c.nºcompra = @num

go

exec returnCompras '1'
exec returnCompras '2'
```



```
go
create proc returnEncomenda
    @num int output
as
    select c.nºencomenda, c.data_realizacao, c.quantidade
    from cafe.Encomenda c
    where c.nºencomenda = @num

go

exec returnEncomenda '1'

go
create proc returnFornecedores
    @nome varchar(20) output
as
    select c.nomeForn, c.codigo_interno, c.moradaForn, c.telefoneForn,
c.forma_pag
    from cafe.Fornecedor c
    where c.nomeForn = @nome

go

go
create proc returnProduto
    @nome varchar(20) output
as
    select c.nomeProd, c.prodID, c.familia, c.preco
    from cafe.Produto c
    where c.nomeProd = @nome

go

exec returnProduto 'café'
exec returnProduto 'chá'

go
create proc returnReclamacao
    @num varchar(20) output
as
    select c.nºreclamacao, c.motivo, c.reclam_nifCli
    from cafe.Reclamacao c
    where c.nºreclamacao = @num

go
exec returnReclamacao '1'

-- stored procedure para eliminar funcionario --
drop proc cafe.sp_DeleteFunc
create proc cafe.sp_DeleteFunc_by_PK
    @numFuncionario int
as
    delete from cafe.Funcionario
    where numFunc = @numFuncionario
```

```
go
select *from cafe.Funcionario
exec cafe.sp_DeleteFunc @numFuncionario

-- stored procedure para eliminar fornecedor
create proc cafe.sp_DeleteForn_by_PK

    @codigoFornecedor int
as
    delete from cafe.Fornecedor
    where codigo_interno = @codigoFornecedor
go
exec cafe.sp_DeleteForn_by_PK 1234
select *from cafe.Fornecedor

-- stored procedure para eliminar cliente
create proc cafe.sp_DeleteCliente_by_CC

    @ccCliente int
as
    delete from cafe.Cliente
    where CC_Cliente = @ccCliente
go
exec cafe.sp_DeleteCliente_by_CC 12345023
select *from cafe.Cliente

-- stored procedure para eliminar Encomenda
create proc cafe.sp_DeleteEncomenda_by_PK

    @numEncomenda int
as
    delete from cafe.Encomenda
    where nºencomenda = @numEncomenda
go
exec cafe.sp_DeleteEncomenda_by_PK 1
select *from cafe.Encomenda

-- stored procedure para eliminar Horário
create proc cafe.sp_DeleteHorario_by_PK

    @turnoFunc varchar(20)
as
    delete from cafe.horario
    where turno = @turnoFunc
go
exec cafe.sp_DeleteHorario_by_PK
select *from cafe.Horario

-- stored procedure para eliminar Reclamacao
create proc cafe.sp_DeleteReclamacao_by_PK

    @numReclam int
as
    delete from cafe.Reclamacao
    where nºreclamacao = @numReclam
go
exec cafe.sp_DeleteReclamacao_by_PK 1
select *from cafe.Reclamacao

-- stored procedure para eliminar Compras
create proc cafe.sp_DeleteCompras_by_PK
```

```
        @numCompra int
as
    delete from cafe.Compras
    where n°compra = @numCompra
go
exec cafe.sp_DeleteCompras_by_PK 1
select *from cafe.Compras

-- stored procedure para eliminar Reclamacao
create proc cafe.sp_DeleteProduto_by_PK

    @IDProd int
as
    delete from cafe.Produto
    where prodID = @IDProd
go
exec cafe.sp_DeleteProduto_by_PK 19
select *from cafe.Produto

-----
-- stored procedures for update --
drop proc cafe.sp_UpdateFunc

-- update funcionario
create proc cafe.sp_UpdateFunc
    @NomeFunc varchar(20),
    @CCFunc int,
    @NumFunc int,
    @MoradaFunc varchar(35),
    @TelefoneFunc int,
    @SalarioFunc decimal(10,2),
    @SuperFunc int,
    @TurnoFunc varchar(20)
```

Anexo V – UDFs

```
-- udf ---
drop function cafe.PesquisaFuncionario
create function cafe.PesquisaFuncionario(@nomefuncionario varchar(20)) returns
table
as
    return(select nomeFunc, CCFunc
            from cafe.Funcionario
            where nomeFunc LIKE +@nomefuncionario+'%')

go

select *from cafe.PesquisaFuncionario('An')

drop function cafe.totalOrcamentoProduto

-- função para retornar o valor total ganho com determinado produto--
create function cafe.totalOrcamentoProduto(@produto varchar(20) = null)
    returns @orcamento table(nomeProduto varchar(20) null, Valor_Total
money null, Total_vendido int null)
as
    begin
        insert @orcamento select nomeProd, sum(preco) as total,
count(prodID)
                                from cafe.Produto Join cafe.Contidos on
prodID = prodID_cont
                                where nomeProd = @produto
                                group by nomeProd;

        return;
    end;
go

select *from cafe.totalOrcamentoProduto('café')

-- função para retornar o valor total ganho com todos os produtos--
drop function cafe.totalOrcamentoTodosProdutos
create function cafe.totalOrcamentoTodosProdutos()
    returns @orcamentoTodos table(nomeProduto varchar(20) null,
Valor_Total money null, Total_vendido int null)
as
    begin
        insert @orcamentoTodos select nomeProd, sum(preco) as total,
count(prodID)
                                from cafe.Produto Join cafe.Contidos on
prodID = prodID_cont
                                group by nomeProd;

        return;
    end;
go

select *from cafe.totalOrcamentoTodosProdutos()

-- UDF para retornar tabela com total de compras efectuadas, produtos comprados e
dinheiro gasto por cada cliente---
create function cafe.compraTotalCliente(@clienteNIF int = null) returns
@ClienteInfo
    table(Total_Compras_Efectuadas int null, Produtos_Comprados varchar(75)
null, Total_Gasto money null)
as
```

```

begin
    insert @ClienteInfo select count(nºcompra), nome_prod, sum(preco)
    from (((cafe.Cliente Join cafe.Compras on NIF=compras_nifCli)
Join cafe.Contidos on nºcompra = nºcompra_contid) Join cafe.Produto on
prodID_cont = prodID)
    where NIF = @clienteNIF
    group by nome_prod
    return;
end;
go

-- UDF para retornar tabela com total de compras efectuadas, produtos comprados e
dinheiro gasto por todos os clientes---
drop function cafe.compraTotalClienteTodos
create function cafe.compraTotalClienteTodos() returns @ClienteInfoToda
    table(Total_Compras_Efectuadas int null, Produtos_Comprados varchar(75)
null, Total_Gasto money null, NIF int null)
as
begin
    insert @ClienteInfoToda select count(nºcompra), nome_prod,
sum(preco), NIF
    from (((cafe.Cliente Join cafe.Compras on NIF=compras_nifCli)
Join cafe.Contidos on nºcompra = nºcompra_contid) Join cafe.Produto on
prodID_cont = prodID)
    group by nome_prod, NIF
    return;
end;
go

select *from cafe.compraTotalClienteTodos()

```

Anexo VI – Triggers

```
-- triggers-----
-- mensagem de inserção de um funcionário
create trigger trg_insertValue on cafe.Funcionario
for insert

As
    if(select count(*) from inserted) = 1
        print 'Funcionário inserido com sucesso'

go

-- trigger para impedir que funcionários sem supervisor sejam apagados
create trigger trg_deleteFunc on cafe.Funcionario
for delete

As
    if(select count(*) from deleted) = 1
    begin
        RAISERROR('Funcionários sem supervisor não podem ser removidos!',
16, 1);
        rollback tran
    End

delete from cafe.Funcionario
where SuperFunc = null
go
```