E-Cafe – Base de Dados de um Café

Universidade de Aveiro

Licenciatura em Engenharia Informática

UC 42532 - Bases de Dados

Corpo Docente:

Professor Carlos Costa

Professor Sousa Pinto

Trabalho Realizado por:

Diogo Moreira - 93127

Eduardo Santos - 93107

Índice

1. Introdução
2. Análise de Requisitos
3. DER - Diagrama Entidade-Relação
4. ER - Esquema Relacional
5. Linguagem SQL - DDL
6. Linguagem SQL - DML
    6.1. Stored Procedures
    6.2. UDFs
    6.3. Triggers
7. Features adicionadas depois da Apresentação Final
8. Trabalho Futuro
9. Conclusão

1. Introdução

Este relatório tem como objetivo descrever em detalhe todo o trabalho que foi realizado para a realização do projeto da unidade curricular de Bases de Dados da Universidade de Aveiro.

O tema escolhido para a realização deste projeto foi criar um **Sistema de Gestão de um Café**, com especial importância no desenvolvimento da base de dados para o sistema.

As imagens dos diagramas estão disponíveis no diretório **Diagrams**.

Os scripts contendo o esquema da base de dados e os dados está disponível no diretório **Script**.

O código relativo à programação da interface em C# encontra-se disponível no diretório **ProjetoBD**.

Para alterar o utilizador da base de dados para poder testar o sistema é necessário alterar a SqlConnection dentro da função **getBDConnection()** em todas as forms (Form1,Form2,FormAdmin).

Os admins já criados para se poder aceder às funções de admin do sistema estão no ficheiro **InsertCafes.sql**

2. Análise de Requisitos

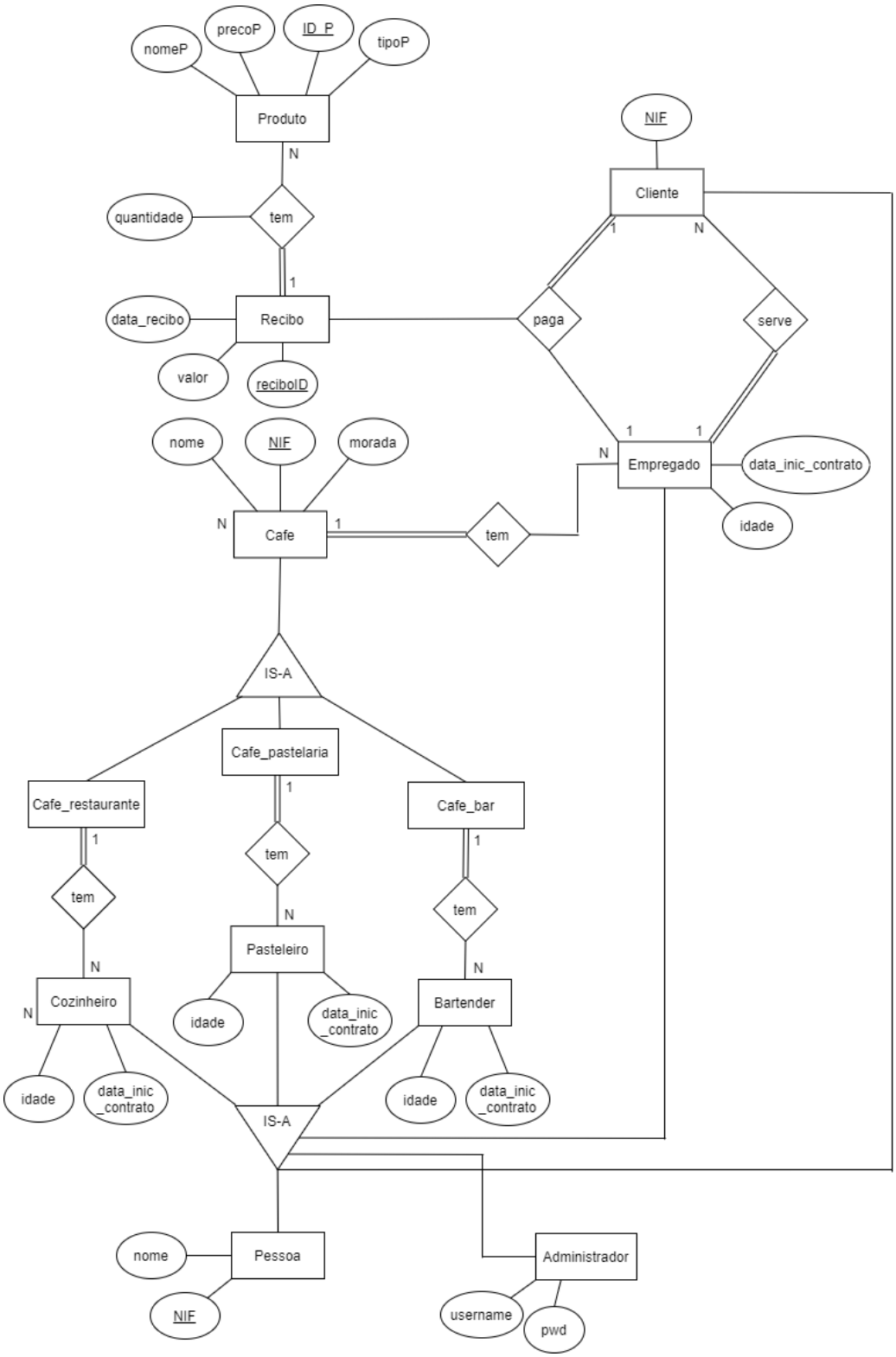
Após o tema do projeto ter sido escolhido, foi necessário realizar uma análise de requisitos com vista a desenvolver um desenho conceptual da nossa base de dados. Nesta análise, foram identificadas as diferentes entidades do sistema e as relações que estabelecem entre si. É de realçar que esta análise foi sofrendo alterações ao longo do trabalho realizado, consoante a necessidades. Agora apresentamos a lista de requisitos.

- Um **Café** é identificado pelo NIF, nome e morada.
- Uma **Pessoa** é identificada pelo NIF e nome.
- Um **Empregado** é identificado pelo NIF, NIF do café, nome, idade, data de início do contrato.
- Um **Cliente** é identificado pelo NIF e nome.
- Um **Café_Restaurante** é identificado pelo NIF, nome e morada. Um Café_Restaurante tem Cozinheiros e serve Almoços.
- Um **Cozinheiro** é identificado pelo NIF, NIF do café, nome, idade e data de início de contrato.
- Um **Café_Pastelaria** é identificado pelo NIF, nome e morada. Um Café_Pastelaria tem Pasteleiros e serve Pasteis.
- Um **Pasteleiro** é identificado pelo NIF, NIF do café, nome, idade e data de início de contrato.
- Um **Café_Bar** é identificado pelo NIF, nome e morada. Um Café_Bar tem Bartenders e serve Bebidas Alcoólicas.
- Um **Cozinheiro** é identificado pelo NIF, NIF do café, nome, idade e data de início de contrato.
- Um **Recibo** é emitido na relação de pagamento entre o Empregado e Cliente e é identificado pelo ID,NIF do Cliente, NIF do Empregado, Data que foi emitido e valor.

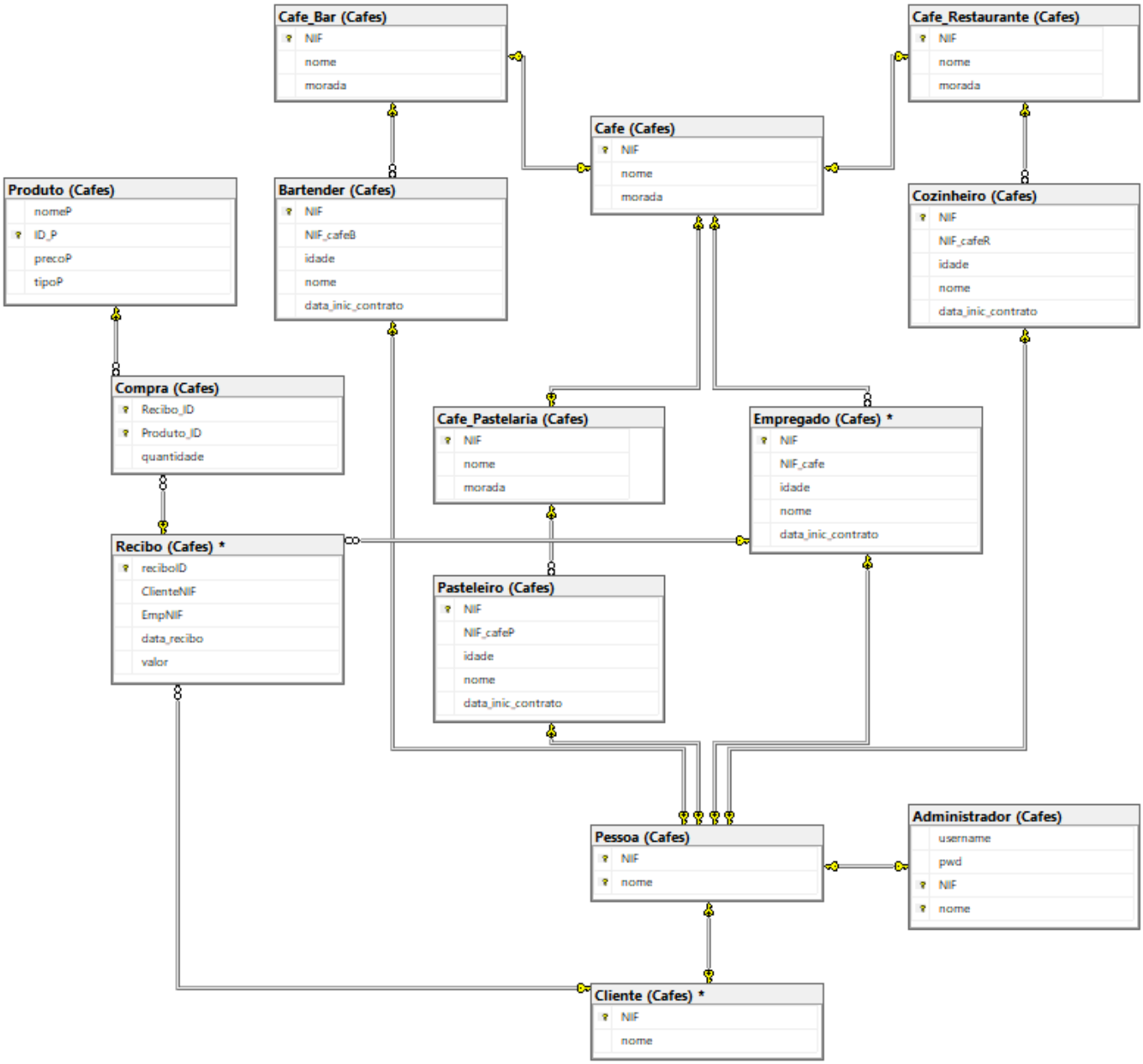
· Um **Produto** é identificado por nome, ID, preço e tipo (Bebidas, Álcool, Pasteis ou Almoços). O Produto só pode ter um único tipo.

· Um **Recibo** tem vários **Produtos** associados, esta relação está presente numa entidade **Compra** que também guarda a quantidade de **Produtos únicos** associados a um **Recibo**

3. DER - Diagrama Entidade-Relação



4. ER – Esquema Relacional



5. Linguagem SQL - DDL

A primeira tarefa após a realização do desenho conceptual da nossa base de dados foi a **criação das tabelas** e a imposição de restrições de integridade sobre os dados das mesmas. Para isto foi utilizada a linguagem **SQL Data Definition Language**, em primeiro lugar numa **base de dados loca**l onde se foram corrigindo erros e melhoramentos e só depois é que foi replicado todo o trabalho no **servidor das aulas**. Também é importante mencionar que todas as tabelas criadas pertencem ao schema **Cafes**.

Há uma tabela que convém realçar que é a tabela **Compra** esta tabela representa a relação do Recibo e do Produto no qual o Recibo tem uma certa quantidade de um Produto. Esta quantidade está guardada na tabela Compra.

6. Linguagem SQL - DML

Para a realização do nosso projeto utilizámos também a linguagem **SQL DML (Data Manipulation Language)**, a qual, tal como o próprio nome indica, nos permite manipular todos os dados. Esta inclui a maior parte das **SQL statements** mais utilizadas tais como **SELECT, INSERT, UPDATE, DELETE**, entre outras.

Esta linguagem foi, sem dúvida, bastante utilizada no desenvolvimento do projeto pois é ela que nos permite fazer todas as operações pretendidas, sendo estas operações utilizadas por parte do **cliente**, na interface do programa.

6.1 Stored Procedures

Stored procedure é um conjunto de comandos armazenados em SQL que podem ser executados de uma só vez, aceitando parâmetros de entrada, sendo que estes dependem da tarefa que se pretende executar, tal como uma função.

Estes devem ser criados quando:

- temos várias aplicações escritas em diferentes linguagens, pretendendo estas executar a **mesma** função.
- queremos manter um certo nível de **consistência** e **segurança** na solução final, pois utilizando **stored procedures** os utilizadores **não** têm acesso às tabelas da base de dados de forma direta.

Foram criados os seguintes procedures consoante as necessidades do sistema:

Insert/Add:

- insertAdministrador
- insertEmpregado
- insertCliente
- insertRecibo
- insertCompra
- addProduto

Remove:

- removeEmpregado
- removeCliente
- removeRecibo
- removeProduto

Get:

- getLastReciboID
- getLastProdutoID
- getBebidas
- getAlcool
- getAlmocos
- getPasteis

- getProdutosInRecibo
- getProdutoQ
- getEmps
- getClientes

Edit/Search/Verify Login:

- editProduto
- searchRecibo
- verifyLogin

Destas procedures, as que achámos mais **importantes** foram as seguintes:

- insertAdministrador: ao adicionar um Admin ao sistema, não guardamos a password do mesmo no sistema diretamente, esta passa primeiro pela procedure insertAdministrador, onde aplicamos um **hashing** utilizando a função *HASHBYTES* e o algoritmo MD5.

- verifyLogin: nesta procedure fazemos a verificação do input do User nos campos username e pwd, verificando se o username está na tabela e se a respetiva pwd na tabela corresponde à pwd introduzida pelo User, pós aplicado o **hashing** anteriormente referido.

6.2 UDFs

UDF (User-Defined Function) é uma função que pode ser utilizada em praticamente todos os comandos **SQL**, sejam este **DML** ou **DDL**.

As UDFs que criámos foram as seguintes:

- checkPessoa
- checkCl
- checkEmp
- checkQuantidadeProduto
- getProdutoQuantidade
- checkReciboInCompra
- checkReciboInRecibo
- checkEmpregadoByNIF
- checkClienteByNIF

A maior parte destas **UDFs** foram criadas com o intuito de serem chamadas nos **triggers**, sendo estes os que vamos referir a seguir.

6.3 Triggers

Um **trigger** está relacionado a uma tabela, sendo este ativado quando certas ações são **realizadas** sobre a mesma: **INSERT, DELETE ou UPDATE**.

Tivemos a necessidade de criar os seguintes triggers:

INSTEAD OF INSERT:

- checkEmpregado & checkCliente: o procedimento para ambas é semelhante, sendo que o objetivo de ambos os triggers é, ao verificar se a pessoa introduzida **existe**, caso exista então prosseguimos a introduzi-la nas respetivas tabelas (Cafes.Empregado ou Cafes.Cliente), e, caso não exista, **primeiro** adicioná-la à tabela Cafes.Pessoa e só **depois** às respetivas tabelas.

- checkInsertProduto: este trigger verifica antes de adicionar o produto à tabela Cafes.Compra se este já foi adicionado com o ID do recibo introduzido, se **não**, então é introduzido com a **quantidade = 1**, caso já tenha sido inserido, é utilizada a statement **UPDATE** para atualizar a coluna quantidade do mesmo, **somando 1 ao valor anterior da mesma.**

- checkNIFs: trigger criado para verificar se os NIFs introduzidos para serem adicionados juntamente com outros parâmetros ao recibo existem, se não existiram então a inserção **não** é permitida.

INSTEAD OF DELETE:

- checkRemoveRecibo: este trigger é acionado quando tentamos remover um recibo da tabela Cafes.Recibo, caso este exista tanto na tabela Cafes.Compra ou Cafes.Recibo, é **primeiro** removido da tabela Cafes.Compra e só **depois** é removido da tabela Cafes.Recibo (devido às dependências entre as tabelas).

7. Features adicionadas depois da Apresentação Final

Depois da apresentação final decidimos que seria importante criar uma feature para **pesquisar** por um **Recibo** através do seu **ID**, sendo que, no futuro, para um enorme número de recibos, um cliente que quer encontrar um dos seus recibos iria ter dificuldade sem uma ferramenta de pesquisa.

8. Trabalho Futuro

Algo que não chegou a ser implementado no sistema foi a opção de implementar cozinheiros, pasteleiros e bartenders. Esta opção não chegou a ser implementada pois só faria sentido se tivéssemos um sistema individual para o Cafe_Restaurante, Cafe_Pastelaria e Cafe_Bar onde estes (cozinheiros,pasteleiros e bartenders) só apareceriam no seu café específico e para implementar esta opção precisaríamos de mais tempo.

9. Conclusão

A realização deste trabalho foi de grande importância para o desenvolvimento dos conhecimentos e capacidades que devem ser adquiridos na cadeira de Base de Dados.

Neste trabalho o principal objetivo foi criar um sistema que permitisse realizar maior parte das funções que um café teria tanto da parte dos empregados (adicionar/remover recibos) como da parte do dono do café (adicionar/remover empregados e clientes e adicionar/editar/remover produtos), pelo que o foco no desenho de uma interface gráfica apelativa a todos os utilizadores não foi tido em conta.

Para concluir acreditamos que a maioria dos objetivos propostos foram alcançados e que foi possível criar uma base de dados de acordo com os objetivos iniciais.