deti departamento de eletrónica, telecomunicações e informática

universidade de aveiro

Relatório

Desenvolvimento de uma base de dados para gestão de uma cadeia de Hotéis

Base de Dados

P3G4

M.I. em Engenharia de Computadores e Telemática

2º semestre - 2015/2016

Marcos Pires nmec: 73192

Tiago Henriques nmec: 73046

Índice

Introdução

O trabalho que nos propusemos a fazer para o projeto final da unidade curricular de Base de Dados é uma plataforma para gestão de uma base de dados de uma cadeia de Hóteis.
O nosso projeto está dividido em duas partes: a primeira parte é referente ao cliente e permite a reserva de um quarto num determinado Hotel abrangido na nossa cadeia, a segunda parte é referente a um qualquer administrador de um Hotel e permite a gestão de Hotéis, Tipos de Quartos, Quartos, Camas, Clientes, Funcionários, Reservas, Serviços, Pagamentos, Temporadas e Pensões. O nosso sistema destina-se a Turistas, pessoas em viagens de negócio e Famílias que procurem um Hotel para ficarem instalados e relativamente à parte de gestão, destina-se a qualquer administrador de um Hotel.
O relatório tem como finalidade tornar a perceção e compreensão mais clara por parte do leitor e vai conter também todos os passos, dificuldades e soluções que encontrámos e implementámos durante a execução deste projeto.

Análise de requisitos

Sistema de Gestão de uma cadeia de Hotéis com as seguintes características:

-Pessoa é caracterizada por: id, data nascimento, e-mail, primeiro e ultimo nome, endereço, sexo,nº telefone, código postal
-Cliente e Funcionários são Pessoas, um funcionário pode ser um cliente e vice versa.
-Um Funcionário é caracterizado pelo salário e trabalha num hotel.
-Um cliente pode fazer uma ou várias reservas.
-Uma reserva é caracterizada por: id, número de pessoas, datas de início e fim da estadia.
-Uma reserva só permite a requisição de um quarto.
-Um Cliente pode escolher um tipo de Pensão que vai estar associado à sua reserva.
-O tipo de pensão assume determinados valores com um significado local (SA(Apenas Alojamento), APA(Alojamento e pequeno-almoço), PC(Pensão completa, alojamento, pequeno-almoço, almoço e jantar).
-Uma reserva é registada pelo recepcionista que estiver a trabalhar no momento da efetuação da mesma.
-Um pagamento está associado a uma reserva, que é caracterizado por: id, custo total a pagar, método de pagamento e data.
-O Pagamento só é efetuado no momento de checkout do cliente.

-O pagamento é registado pelo recepcionista que estiver a trabalhar no momento do pagamento por parte do cliente.
-Os preços da reserva dependem da temporada em que incide a estadia.
-As temporadas são caracterizadas por: id, data de começo e termino, razão e os preços de todos os tipos de quartos que são possíveis reservar (simples, duplo, etc.)
-Uma reserva pode ter um conjunto de serviços associados.
-Os Serviços são caracterizados por: id, custo e data de requisição.
-Existem diversos tipos de Serviços: RoomService, Restaurante/Bar, Estacionamento, Aluguer de Video e Serviços externos.
-RoomService caracteriza-se por: hora de consumo e id do produto.
-Restaurante/Bar caracteriza-se por: tipo e descrição.
-Estacionamento caracteriza-se por: lugar de estacionamento.
-Vídeo caracteriza-se por: hora de requisição e id do filme.
-Serviço externo caracteriza-se por: descrição do serviço externo e tipo de serviço externo(aluguer de carros, excursões, serviços de baby-sitting, espetáculos)
-A gestão dos Serviços é gerida pelo funcionário que estiver de serviço no momento da requisição. Como um cliente pode usufruir de vários serviços, o funcionário a gerir o serviço não é necessariamente sempre o mesmo.
-Uma reserva solicita a requisição de um quarto disponível no hotel para o tempo de estadia do cliente.
-Um Quarto pode estar associado a uma ou mais Reservas em tempos diferentes da estadia dos clientes.
-Tipo de quarto é caracterizado por um tipo que assume um dos valores seguintes com significado local (single, double, twin, mini-suite, suite) e numero máximo de camas extra.
-Um Quarto é caracterizado por id, se permite fumadores, estado atual (ocupado ou não), nº telefone interno no sistema do hotel.
-Um Quarto está associado a um tipo de quarto.
-Uma Cama é caracterizada por um tipo que assume um dos valores seguintes: (single, double, twin, queen, king) e está associada a uma reserva.
-Podem ser adicionadas Camas extra a uma reserva específica se assim for desejado/necessário.
-Uma cadeia de Hotéis é composta por mais do que um Hotel, em que cada um é caracterizado por código postal, nome, classificação, localização e id
-Um hotel tem vários quartos.
-Existem diversos tipo de Funcionario: Recepcionista, Empregado, Supervisor e Gerente.
-Um Recepcionista pode ter outro Recepcionista (Advisor) que lhe dá conselhos sobre linhas de trabalho.
-O Gerente gere o hotel onde trabalha e cada hotel só pode ter um gerente.
-Nenhum funcionário pode ter mais do que uma função(ou é gerente, ou é recepcionista ou é empregado)
-Um Empregado também pode ter outro Empregado (Supervisor) que lhe dá conselhos sobre linhas de trabalho e o supervisiona.

Diagrama de Relações Entidades (DER)

O diagrama encontra-se na pasta de anexos.

Esquema relacional

O esquema relacional encontra-se na pasta de anexos.

SQL DDL

```
use p3g4;

-- DROP SCHEMA gestaoHotel;
-- CREATE SCHEMA gestaoHotel;

CREATE TABLE gestaoHotel.Temporada (
    idTemporada         INT             PRIMARY KEY,
    dataComeco          DATE            NOT NULL,
    dataTermino         DATE            NOT NULL,
    razao               VARCHAR(30),
    precoSimples        MONEY           NOT NULL,
    precoDouble         MONEY           NOT NULL,
    precoTwin           MONEY           NOT NULL,
    precoMiniSuite      MONEY           NOT NULL,
    precoSuite          MONEY           NOT NULL,
    CHECK(dataTermino>dataComeco));

CREATE TABLE gestaoHotel.Pensao(
    tipo            VARCHAR(3)          PRIMARY KEY,
    descricao       VARCHAR(50),
    CHECK (tipo IN ('SA', 'APA', 'PC')));

CREATE TABLE gestaoHotel.Pessoa(
    idPessoa        INT                 PRIMARY KEY,
    email           VARCHAR(50)         UNIQUE  NOT NULL,
    Pnome           VARCHAR(15)         NOT NULL,
    Unome           VARCHAR(30)         NOT NULL,
    dataNasc        DATE,
    endereco        VARCHAR(50),
    sexo            CHAR(1),
    nrTelefone      CHAR(9),
    codigoPostal    VARCHAR(10));
```

```
CREATE TABLE gestaoHotel.Cliente(
    id                  INT                 PRIMARY KEY,
    FOREIGN KEY (id)    REFERENCES gestaoHotel.Pessoa(idPessoa));

CREATE TABLE gestaoHotel.Funcionario(
    id                  INT             PRIMARY KEY,
    salario             INT             NOT NULL,
    hotel               INT             NOT NULL,
    FOREIGN KEY (id)    REFERENCES gestaoHotel.Pessoa(idPessoa));

CREATE TABLE gestaoHotel.Gerente (
    nrFuncionario               INT             PRIMARY KEY,
    FOREIGN KEY(nrFuncionario)  REFERENCES gestaoHotel.Funcionario(id));

CREATE TABLE gestaoHotel.Hotel (
    idHotel                 INT         PRIMARY KEY,
    nome                    VARCHAR(30) NOT NULL,
    classificacao           INT         NOT NULL,
    localizacao             VARCHAR(50) NOT NULL,
    codigoPostal            VARCHAR(10),
    gerente                 INT         NOT NULL,
    FOREIGN KEY(gerente)    REFERENCES gestaoHotel.Gerente(nrFuncionario));

--ALTER TABLE gestaoHotel.Hotel ALTER COLUMN gerente INT;


CREATE TABLE gestaoHotel.Recepcionista (
    nrFuncionario               INT             PRIMARY KEY,
    supervisor                  INT,
    FOREIGN KEY (supervisor)    REFERENCES gestaoHotel.Recepcionista (nrFuncionario),
    FOREIGN KEY(nrFuncionario)  REFERENCES gestaoHotel.Funcionario (id));

CREATE TABLE gestaoHotel.Empregado(
    nrFuncionario               INT             PRIMARY KEY,
    supervisor                  INT,
    FOREIGN KEY(supervisor)     REFERENCES gestaoHotel.Empregado(nrFuncionario),
    FOREIGN KEY(nrFuncionario)  REFERENCES gestaoHotel.Funcionario (id));


CREATE TABLE gestaoHotel.TipoQuarto(
    tipo                VARCHAR(10)     PRIMARY KEY,
    descricao           VARCHAR(50),
    numCamasExtraDisp   INT             NOT NULL,
    CHECK (tipo IN ('single', 'double', 'twin', 'mini-suite', 'suite')));

CREATE TABLE gestaoHotel.Quarto(
    idQuarto            INT         PRIMARY KEY,
    fumador             BINARY(1)   NOT NULL,
    estado              BINARY(1)   NOT NULL,
    telefone            INT         NOT NULL,
    tipoQuarto          VARCHAR(10) NOT NULL,
    hotel               INT         NOT NULL,
    FOREIGN KEY(hotel)  REFERENCES gestaoHotel.Hotel(idHotel),
    FOREIGN KEY(tipoQuarto) REFERENCES gestaoHotel.TipoQuarto(tipo),
    CHECK (fumador IN (0,1)),
    CHECK (tipoQuarto IN ('single', 'double', 'twin', 'mini-suite', 'suite')),
    CHECK (estado IN (0,1)));


CREATE TABLE gestaoHotel.Pagamento (
    idPagamento                 INT         PRIMARY KEY,
    metodo                      VARCHAR(20),
    dataPagamento               DATE        NOT NULL,
    custoTotal                  MONEY       NOT NULL,
    recepcionista               INT         NOT NULL,
    FOREIGN KEY(recepcionista)  REFERENCES gestaoHotel.Recepcionista(nrFuncionario));
```

```
CREATE TABLE gestaoHotel.Reserva (
    idReserva                   INT         PRIMARY KEY,
    numPessoas                  INT         NOT NULL,
    tipoPensao                  VARCHAR(3)  DEFAULT 'SA',
    dataInicio                  DATE        NOT NULL,
    dataFim                     DATE        NOT NULL,
    pagamento                   INT,
    quarto                      INT         NOT NULL,
    recepcionista               INT         NOT NULL,
    cliente                     INT         NOT NULL,
    FOREIGN KEY(tipoPensao)     REFERENCES gestaoHotel.Pensao(tipo),
    FOREIGN KEY(pagamento)      REFERENCES gestaoHotel.Pagamento(idPagamento),
    FOREIGN KEY(quarto)         REFERENCES gestaoHotel.Quarto(idQuarto),
    FOREIGN KEY(recepcionista)  REFERENCES gestaoHotel.Recepcionista(nrFuncionario),
    FOREIGN KEY(cliente)        REFERENCES gestaoHotel.Cliente(id),
    CHECK (tipoPensao IN ('SA', 'APA', 'PC')),
    CHECK (DataFim > DataInicio));

CREATE TABLE gestaoHotel.Depende(
    idTemporada                 INT     NOT NULL,
    idReserva                   INT     NOT NULL,
    PRIMARY KEY(idTemporada, idReserva),
    FOREIGN KEY(idTemporada)    REFERENCES gestaoHotel.Temporada(idTemporada),
    FOREIGN KEY(idReserva)  REFERENCES gestaoHotel.Reserva(idReserva));

CREATE TABLE gestaoHotel.Cama(
    tipo                    VARCHAR(10) PRIMARY KEY,
    preco                   MONEY       NOT NULL,
    CHECK (tipo IN ('single', 'double', 'twin', 'queen', 'king')));
```

```
CREATE TABLE gestaoHotel.Requere(
    tipo                    VARCHAR(10) NOT NULL,
    idReserva               INT         NOT NULL,
    quantidade              INT         NOT NULL,
    PRIMARY KEY(tipo, idReserva),
    FOREIGN KEY(tipo)   REFERENCES gestaoHotel.Cama(tipo),
    FOREIGN KEY(idReserva)  REFERENCES gestaoHotel.Reserva(idReserva),
    CHECK (quantidade IN (0,1,2)),
    CHECK (tipo IN ('single', 'double', 'twin', 'queen', 'king')));


CREATE TABLE gestaoHotel.Servico(
    idServico                   INT         PRIMARY KEY,
    custo                       MONEY       NOT NULL,
    data                        DATE        NOT NULL,
    reserva                     INT         NOT NULL,
    funcionario                 INT         NOT NULL,
    FOREIGN KEY(reserva)        REFERENCES gestaoHotel.Reserva(idReserva),
    FOREIGN KEY(funcionario)    REFERENCES gestaoHotel.Funcionario(id));

CREATE TABLE gestaoHotel.RoomService(
    idServico               INT         PRIMARY KEY,
    idProduto               INT         NOT NULL,
    hora                    TIME,
    FOREIGN KEY(idServico)  REFERENCES gestaoHotel.Servico(idServico));

CREATE TABLE gestaoHotel.Estacionamento(
    idServico               INT         PRIMARY KEY,
    lugar                   VARCHAR(4),
    FOREIGN KEY(idServico)  REFERENCES gestaoHotel.Servico(idServico));

CREATE TABLE gestaoHotel.Video(
    idServico               INT         PRIMARY KEY,
    idFilme                 INT         NOT NULL,
    hora                    TIME,
    FOREIGN KEY(idServico)  REFERENCES gestaoHotel.Servico(idServico));


CREATE TABLE gestaoHotel.ServicoExterno(
    idServico               INT         PRIMARY KEY,
    tipoServicoExt          VARCHAR(20) NOT NULL,
    descricao               VARCHAR(50),
    FOREIGN KEY(idServico)  REFERENCES gestaoHotel.Servico(idServico),
    CHECK (tipoServicoExt IN ('aluguer', 'excursao', 'babySitting', 'espetaculo')));

CREATE TABLE gestaoHotel.RestauranteBar(
    idServico               INT         PRIMARY KEY,
    tipo                    VARCHAR(20) NOT NULL,
    descricao               VARCHAR(50),
    FOREIGN KEY(idServico)  REFERENCES gestaoHotel.Servico(idServico),
    CHECK (tipo IN ('bar','restaurante')));

-- Funcionario
ALTER TABLE gestaoHotel.Funcionario ADD CONSTRAINT HOFKIH FOREIGN KEY(hotel) REFERENCES gestaoHotel.Hotel(idHotel)
    ON UPDATE CASCADE;
```

SQL DML

Encontra-se na pasta de anexos.

Normalização

A normalização tem com objetivo reduzir a redundância e garantir que existe integridade referencial entre relações.
Após a análise das nossas tabelas, concluímos que não existem dependências parciais nem transitivas, pois todos os atributos não chave dependem de toda a chave da relação.

Temporada

idTemporada	dataComeco	dataTermino	razao	precoSimples	precoDuplo	precoTwin	precoMiniSuite	precoSuite

Índices

Índices são estruturas de dados que oferecem uma segunda forma (rápida) de acesso aos dados.
Com o intuito de otimizar os tempos de consulta, utilizámos apenas non-clustered índices em dados não primários que aparecem muitas vezes em pesquisas na cláusula WHERE (como o salário). Também implementámos este tipo de índices em atributos chave estrangeira que aparecem frequentemente em relações de JOIN (como o atributo hotel)
Tendo como objetivo minimizar os page split, usámos um fill factor entre 65-85% dado que não se tratam de inserções ordenadas mas sim no meio da B-Tree.

Índices criados:

- idxSalaryFunc;

- idxHotelFunc;
- idxDataReserva;

```
use p3g4;

GO
CREATE NONCLUSTERED INDEX idxSalaryFunc ON gestaoHotel.Funcionario(salario)
WITH (FILLFACTOR=75,pad_index=ON);

GO
CREATE NONCLUSTERED INDEX idxHotelFunc ON gestaoHotel.Funcionario(hotel)
WITH (FILLFACTOR=75,pad_index=ON);
```

Triggers

Triggers são um tipo especial de stored procedure que é executado em determinadas circunstâncias (eventos) associadas à manipulação de dados. São usados como garantia na consistência do modelo de dados a quando acesso direto aos dados .
Desenvolvemos três triggers mas apenas implementámos um. A razão para que decidimos não colocar os outros dois encontra-se em comentário no ficheiro do trigger.
Foram implementados apenas AFTER triggers.

Trigger implementado:

- trigger_gerente_hotel;

Ex: Trigger que não permite que nenhum Gerente seja Gestor de mais do que um Hotel
Trigger que dada a operação de insert ou update, verifica se o gerente em questão já é gestor de um Hotel. Em caso afirmativo, impossibilita estas operações através da transação de rollback e apresenta uma mensagem de erro.

Stored Procedures

Foram usadas Stored Procedures pois os procedimentos são guardados em memória cache na primeira vez em que são executados, logo o SQL Server não tem de recompilar o código cada vez que o procedimento é invocado o que resulta numa execução bastante mais rápida.
As Stored Procedures foram usadas na criação de métodos de criar/editar ou remover dados.

Ex: Inserção de um Recepcionista

Especificamos os dados que queremos receber na sp e o tipo e fazemos as validações e a robustez necessárias. Como o id é gerado automaticamente, necessitamos de saber qual é o id máximo da Pessoa que está na BD e só depois, procedemos à introdução da Pessoa, Funcionário e Recepcionista.

```
GO
--Criação de um rececionista
CREATE PROCEDURE sp_AddRec
    @Pnome              VARCHAR(15),
    @Unome              VARCHAR(30),
    @email              VARCHAR(50),
    @dataNasc           DATE,
    @endereco           VARCHAR(50),
    @codigoPostal       VARCHAR(10),
    @sexo               CHAR(1),
    @nrTelefone         CHAR(9),
    @salario            INT,
    @hotel              INT,
    @supervisor         INT
    WITH ENCRYPTION
    AS
        IF @Pnome is NULL OR @Unome is NULL OR @email is NULL OR @dataNasc is NULL OR @endereco is NULL OR @codigoPostal is NULL OR @sexo is NULL OR @nrTelefone is NULL
            OR @salario is NULL OR @hotel is NULL
        BEGIN
            PRINT 'O primeiro nome, último nome, email, data nascimento, endereco, codigo-postal, sexo, telefone, salário ou hotel não podem estar por preencher!!'
            RETURN
        END
        ----------------------------------------------------------------------
        -- check if the client already exists so the addition could not be completed
        DECLARE @COUNT INT
        SELECT @COUNT = COUNT(*) FROM gestaoHotel.Pessoa JOIN gestaoHotel.Funcionario ON idPessoa=id WHERE Pnome=@Pnome AND Unome=@Unome AND email=@email
            AND dataNasc=@dataNasc AND endereco=@endereco AND codigoPostal=@codigoPostal AND sexo=@sexo AND nrTelefone=@nrTelefone;
        IF @COUNT != 0
            BEGIN
                RAISERROR ('O Funcionario que pretende adicionar já existe!', 14, 1)
                RETURN
            END
        ----------------------------------------------------------------------
        -- check the max person id
        DECLARE @MaxPerson INT
        SELECT @MaxPerson = MAX(idPessoa) FROM gestaoHotel.Pessoa;
        ----------------------------------------------------------------------
        BEGIN TRY
            INSERT INTO GestaoHotel.Pessoa([idPessoa],[Pnome],[Unome],[email],[dataNasc],[endereco],[codigoPostal],[sexo],[nrTelefone])
                VALUES (@MaxPerson+1,@Pnome,@Unome,@email,@dataNasc,@endereco,@codigoPostal,@sexo,@nrTelefone)
            INSERT INTO GestaoHotel.Funcionario([id],[salario],[hotel])
                VALUES (@MaxPerson+1,@salario,@hotel)
            INSERT INTO GestaoHotel.Recepcionista([nrFuncionario],[supervisor])
                VALUES (@MaxPerson+1,@supervisor)
        END TRY
        ----------------------------------------------------------------------
        BEGIN CATCH
            RAISERROR ('Ocorreu um erro durante a criação do recepcionista!', 14, 1)
        END CATCH;
```

Ex: Remoção de um Empregado

Apenas necessitamos do id da Pessoa como parâmetro de entrada pois como é chave primária, não existem ids duplicados/repetidos. Consideramos impossível eliminar um empregado que tenha serviços associados, portanto, só procedemos à remoção de empregados que não tenham participado em serviços. Como o empregado pode ser Supervisor de outros funcionários, antes de passar à remoção, temos de verificar se o empregado é supervisor de outros empregados e em caso positivo, temos de colocar esse atributo a *null* . Só neste momento é possível proceder à remoção do empregado e consequente funcionário e pessoa da Base de Dados.

```
GO
CREATE PROCEDURE sp_DeleteEmp
    @idPessoa   INT
    WITH ENCRYPTION
    AS
    IF @idPessoa is NULL
    BEGIN
        PRINT 'O id da Pessoa não pode ser estar vazio!'
        RETURN
    END
    ------------------------------------------------------------------------
    -- check if employee has services
    DECLARE @COUNT2 INT
    SELECT @COUNT2 = COUNT(*) FROM gestaoHotel.Servico WHERE funcionario=@idPessoa;
    IF @COUNT2 != 0
        BEGIN
            RAISERROR ('Impossível eliminar empregado pois participou em serviços!', 14, 1)
            RETURN
        END;
    ------------------------------------------------------------------------
    -- check if employee is another employee supervisor
    DECLARE @COUNT INT
    SELECT @COUNT = COUNT(*) FROM gestaoHotel.Empregado WHERE supervisor = @idPessoa;
    IF @COUNT != 0
        BEGIN
            UPDATE gestaoHotel.Empregado SET supervisor = NULL WHERE supervisor=@idPessoa;
        END;
    ------------------------------------------------------------------------
    BEGIN TRY
        DELETE FROM gestaoHotel.Empregado WHERE nrFuncionario=@idPessoa;
        DELETE FROM gestaoHotel.Funcionario WHERE id = @idPessoa;
        DELETE FROM gestaoHotel.Pessoa WHERE idPessoa = @idPessoa;
    END TRY
    ------------------------------------------------------------------------
    BEGIN CATCH
        RAISERROR ('Ocorreu um erro durante a remoção do empregado!', 14, 1)
    END CATCH;
```

UDFs

Utilizámos praticamente sempre UDF Multi-Statement Table-Valued com Schema Binding de modo a prevenir a alteração ou eliminição dos objetos utilizados pela função e cifrámos ainda o conteúdo do udf.
A razão de usármos praticamente sempre UDF Multi-Statement Table-Valued prende-se com questões de segurança na gestão da base de dados, pois deste modo são usadas ferramentas de prevenção de SQL Injection através da adição do parâmentro à query em questão.

Ex: **"cmd_hotel.Parameters.AddWithValue("@nome", Nome_Hoteis.Text);"**

Ainda nas funções definidas pelo utilizador, foi usada sempre a mesma lógica para as diferentes funções implementadas:

- Apresentamos **todos** os resultados quando o(s) parâmetro(s) de entrada é(são) NULL;
- Apresentamos resultados **específicos** quando o(s) parâmetro(s) de entrada não é(são) NULL;

```
use p3g4;

GO
--DROP FUNCTION udf_Reserva_DataGrid;
CREATE FUNCTION udf_Reserva_DataGrid(@idReserva INT=null)
RETURNS @table TABLE (idReserva INT, numPessoas INT, tipoPensao VARCHAR(3), dataInicio DATE,
    dataFim DATE, pagamento INT, quarto INT, recepcionista INT, cliente INT, tipo VARCHAR(10),quantidade INT)
WITH SCHEMABINDING, ENCRYPTION
AS
BEGIN
    IF (@idReserva is NULL)
        BEGIN
            INSERT @table SELECT Reserva.idReserva, numPessoas, tipoPensao, dataInicio,
    dataFim, pagamento, quarto, recepcionista, cliente, tipo, quantidade
    FROM gestaoHotel.Requere RIGHT OUTER JOIN gestaoHotel.Reserva ON Requere.idReserva=Reserva.idReserva;
        END;

    ELSE
        BEGIN
            INSERT @table SELECT Reserva.idReserva, numPessoas, tipoPensao, dataInicio,
    dataFim, pagamento, quarto, recepcionista, cliente, tipo, quantidade
    FROM gestaoHotel.Requere RIGHT OUTER JOIN gestaoHotel.Reserva ON Requere.idReserva=Reserva.idReserva
        WHERE gestaoHotel.Reserva.idReserva = @idReserva
        END;
    RETURN;
END;
```

7. Conclusão

A base de dados foi integrada num ambiente de uma aplicação de uma cadeia de hotéis, e numa análise detalhada parece ser capaz de suportar as várias funcionalidades planeadas, uma vez que os principais problemas foram encontrados e corrigidos.

O desenvolvimento deste projeto foi bastante interessante, e permitiu-nos perceber as maiores dificuldades durante o planeamento e desenvolvimento de uma base de dados que pudesse ser integrada numa aplicação real.

8. Referências

Nenhuma referência a referir.