Faculdade de Engenharia da Universidade do Porto

CouchSurfing

BDAD 2015/2016 – Mestrado Integrado em Engenharia Informática e Computação

(Carlos Manuel Milheiro de Oliveira Pinto Soares)

Autoras:

Andreia Rodrigues up201404691@fe.up.pt

Inês Gomes up201405778@fe.up.pt

Catarina Ramos up201406219@fe.up.pt

Resumo

Este relatório, elaborado no âmbito da unidade curricular “Base de Dados” do 2º ano e 2º semestre do Mestrado Integrado de Engenharia Informática e Computação, tem como objetivo, a apresentação dos conceitos e conhecimentos adquiridos ao longo do semestre lecionados nesta cadeira.

Nesta terceira fase do projeto foi nos proposto a elaboração de dez instruções tipo SELECT tendo em conta o nosso tema e o desenvolvimento numa fase posterior. De forma a prevenir alguma alteração em dados anteriormente tais como: instruções LDD e diagrama de classes, esses mesmos serão apresentados novamente neste relatório.

Índice

1. Introdução

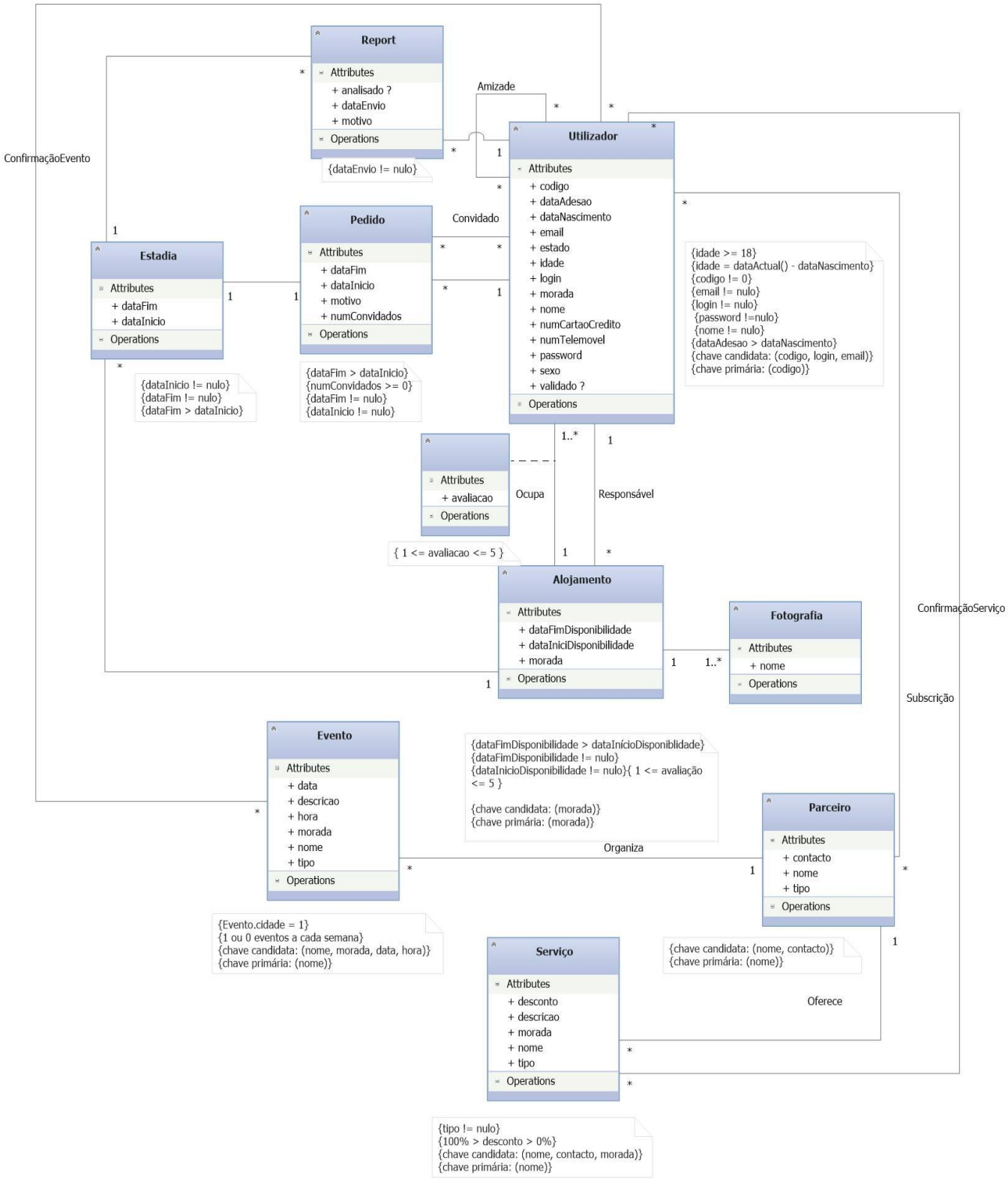
O *CouchSurfing* é uma rede de hospedagem gratuita para viajantes que procuram uma experiência diferente e inovadora. A ideia foi criada em 2003 e foi implementada por uma organização sem fins lucrativos.

Para usufruir deste tipo de alojamento, existe um site oficial destinado a este tipo de serviço. Dentro deste, os utilizadores após a sua inscrição, têm os meios necessários á pesquisa de um alojamento ou prestação do mesmo numa certa cidade de um dado país. A pesquisa inclui também as datas entre as quais o viajante deseja um alojamento e o número de convidados que pretendem participar nesta viagem.

Aquando a viagem, os utilizadores podem usufruir de diversos serviços e eventos semanais, divulgados pelos parceiros ligados à organização, que promovem novas experiências e o contacto com os locais e outros *couchSurfers.*

Um dos problemas que mais se opõe a este conceito, é garantir a segurança dos que estão envolvidos, tanto como a sua legitimidade e a das residências.

2. Diagrama UML



3. Esquema Relacional

Utilizador (codigoUtilizador, dataAdesao, dataNascimento, email, login, morada, estado, idade, nome, numeroCartaoCredito, numeroTelemovel, password, sexo, validado)

Amizade (codigoUtilizador1->Utilizador, codigoUtilizador2->Utilizador)

Alojamento (idAlojamento, morada, dataInicioDisponibilidade, dataFimDisponibilidade, codigoResponsável -> Utilizador, idFotografia -> Fotografia)

Fotografia (idFotografia, nome, idAlojamento->Alojamento)

Ocupa (idOcupacao, codigoUtilizador -> Utilizador, idAlojamento->Alojamento, avaliacao)

Pedido (idPedido, dataInicioEstadia, dataFimEstadia, motivo, numeroConvidados, codigoAlojamento->Alojamento, codigoUtilizador->Utilizador)

Convidado (codigoUtilizador->Utilizador, idPedido ->Pedido)

Estadia (idEstadia, codigoUtilizador->Utilizador, idPedido->Pedido)

Reclamacao (idReclamacao, motivo, dataEnvio, analisado, idEstadia->Estadia, codigoUtilizador -> Utilizador)

Parceiro (idParceiro, nome, tipo, contacto)

Subscrição (codigoUtilizador->Utilizador, idParceiro->Parceiro)

Evento (idEvento, morada, nome, descrição, data, tipo, hora, idParceiro->Parceiro)

Serviço (idServiço, desconto, morada, nome, descrição, tipo, idParceiro->Parceiro)

Oferece (idOferta, idParceiro->Parceiro, idServico->Servico)

Organiza (idOrganizacao, idParceiro->Parceiro, idEvento ->Evento)

ConfirmaçãoEvento (idEvento->Evento, codigoUtilizador ->Utilizador)

ConfirmaçãoServiço (idServico->Servico, codigoUtilizador ->Utilizador)

4. Instruções LDD (Linguagem de Definição de dados)

```
CREATE TABLE Utilizador (
    codigoUtilizador        INTEGER NOT NULL UNIQUE,
    nome                    TEXT NOT NULL,
    idade                   INTEGER NOT NULL,
    email                   TEXT NOT NULL UNIQUE,
    morada                  TEXT,
    dataAdesao              NUMERIC NOT NULL,
    dataNascimento          NUMERIC NOT NULL,
    sexo                    TEXT,
    login                   TEXT NOT NULL UNIQUE,
    password                TEXT NOT NULL,
    estado                  NUMERIC,
    numCartaoCredito        INTEGER,
    numTelemovel            INTEGER,
    validado                NUMERIC CHECK( numCartaoCredito != NULL),
    PRIMARY KEY(codigo),
    CHECK( dataAdesao >= dataNascimento + 18)
);

CREATE TABLE Alojamento (
    idAlojamento                INTEGER NOT NULL UNIQUE,
    dataInicioDisponibilidade   NUMERIC NOT NULL,
    dataFimDisponibilidade      NUMERIC NOT NULL,
    morada                      TEXT NOT NULL UNIQUE,
    codigoResponsavel           INTEGER NOT NULL UNIQUE,
    idFotografia                INTEGER NOT NULL UNIQUE,
    PRIMARY KEY(idAlojamento,morada),
    FOREIGN KEY(codigoResponsavel) REFERENCES Utilizador(codigo),
```

```
    FOREIGN KEY(idFotografia) REFERENCES Fotografia(idFotografia) ON DELETE SET NULL,
    CHECK (dataFimDisponibilidade > dataInicioDisponibilidade)
);

CREATE TABLE Ocupa (
    idOcupacao              INTEGER NOT NULL UNIQUE,
    codigoUtilizador        INTEGER NOT NULL UNIQUE,
    avaliação               INTEGER NOT NULL CHECK (1 <= avaliação <=5),
    idAlojamento            INTEGER NOT NULL UNIQUE,
    PRIMARY KEY(idOcupacao),
    FOREIGN KEY(codigoUtilizador) REFERENCES Utilizador(codigo),
    FOREIGN KEY(idAlojamento) REFERENCES Alojamento(idAlojamento)
);

CREATE TABLE Fotografia (
    idFotografia            INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
    nome                    TEXT NOT NULL UNIQUE,
    idAlojamento            INTEGER NOT NULL UNIQUE,
    FOREIGN KEY(idAlojamento) REFERENCES Alojamento(idAlojamento)
);

CREATE TABLE Amizade (
        codigoUtilizador1   INTEGER NOT NULL UNIQUE,
        codigoUtilizador2   INTEGER NOT NULL UNIQUE,
        PRIMARY KEY(codigoUtilizador1, codigoUtilizador2),
        FOREIGN KEY(codigoUtilizador1) REFERENCES Utilizador(codigo),
        FOREIGN KEY(codigoUtilizador2) REFERENCES Utilizador(codigo)
);

CREATE TABLE Pedido (
    idPedido                INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
    dataInicioEstadia       NUMERIC NOT NULL,
    dataFimEstadia          NUMERIC NOT NULL,
```

```
    numeroConvidados        INTEGER CHECK (numeroConvidados >= 0),
    motivo                  TEXT,
    codigoAlojamento        INTEGER NOT NULL,
    codigoUtilizador        INTEGER NOT NULL,
    FOREIGN KEY(codAlojamento) REFERENCES Alojamento(idAlojamento),
    FOREIGN KEY(codUtilizador) REFERENCES Utilizador(codigo),
    CHECK (dataFimEstadia > dataInicioEstadia)
);

CREATE TABLE Convidado (
    codigoUtilizador    INTEGER NOT NULL UNIQUE,
    idPedido            INTEGER NOT NULL UNIQUE,
    FOREIGN KEY(codigoUtilizador) REFERENCES Utilizador(codigo),
    FOREIGN KEY(idPedido) REFERENCES Pedido(idPedido)
);

CREATE TABLE Estadia (
    idEstadia           INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
    codigoUtilizador    INTEGER NOT NULL UNIQUE,
    idPedido            INTEGER NOT NULL UNIQUE,
    FOREIGN KEY(codigoUtilizador) REFERENCES Utilizador(codigo),
    FOREIGN KEY(idPedido) REFERENCES Pedido(idPedido)
);

CREATE TABLE Reclamação (
    idReclamacao        INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
    dataEnvio           NUMERIC NOT NULL,
    motivo              TEXT NOT NULL,
    analisado           NUMERIC NOT NULL,
    idEstadia           INTEGER NOT NULL UNIQUE,
    codigoUtilizador    INTEGER NOT NULL UNIQUE,
    FOREIGN KEY(idEstadia) REFERENCES Estadia(idEstadia)
    FOREIGN KEY(codigoUtilizador) REFERENCES Utilizador(codigoUtilizador)
```

```
);

CREATE TABLE Parceiro (
    idParceiro          INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
    nome                INTEGER NOT NULL UNIQUE,
    tipo                TEXT NOT NULL,
    contacto            INTEGER NOT NULL
);

CREATE TABLE Evento (
    idEvento            INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
    nome                TEXT NOT NULL UNIQUE,
    data                NUMERIC NOT NULL UNIQUE,
    hora                NUMERIC NOT NULL UNIQUE,
    morada              TEXT NOT NULL UNIQUE,
    tipo                TEXT,
    descricao           TEXT NOT NULL,
    idParceiro          INTEGER NOT NULL UNIQUE,
    FOREIGN KEY(idParceiro) REFERENCES Parceiro(idParceiro)
);

CREATE TABLE Servico (
    idServico           INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
    nome                TEXT NOT NULL UNIQUE,
    tipo                TEXT,
    morada              TEXT NOT NULL UNIQUE,
    desconto            REAL NOT NULL CHECK( 0% < desconto < 100%),
    descricao           TEXT NOT NULL,
    idParceiro          INTEGER NOT NULL UNIQUE,
    FOREIGN KEY(idParceiro) REFERENCES Parceiro(idParceiro)
);

CREATE TABLE Subscricao (
```

```
    codigoUtilizador        INTEGER NOT NULL UNIQUE,
    idParceiro              INTEGER NOT NULL UNIQUE,
    FOREIGN KEY(codigoUtilizador) REFERENCES Utilizador(codigo),
    FOREIGN KEY(idParceiro) REFERENCES Parceiro(idParceiro)
);

CREATE TABLE Oferece (
    idOferta        INTEGER NOT NULL UNIQUE,
    idParceiro      INTEGER NOT NULL UNIQUE,
    idServico       INTEGER NOT NULL UNIQUE,
    PRIMARY KEY(idOferta),
    FOREIGN KEY(idParceiro) REFERENCES Parceiro(idParceiro),
    FOREIGN KEY(idServico) REFERENCES Servico(idServico)
);

CREATE TABLE Organiza (
    idOrganizacao           INTEGER NOT NULL UNIQUE,
    idParceiro              INTEGER NOT NULL UNIQUE,
    idEvento                INTEGER NOT NULL UNIQUE,
    PRIMARY KEY(idOrganizacao),
    FOREIGN KEY(idParceiro) REFERENCES Parceiro(idParceiro),
    FOREIGN KEY(idEvento) REFERENCES Evento(idEvento)
    );

    CREATE TABLE ConfirmacaoEvento (
    idEvento                INTEGER NOT NULL UNIQUE,
    codigoUtilizador        INTEGER NOT NULL UNIQUE,
    PRIMARY KEY(idConfirmacaoEvento),
    FOREIGN KEY(idEvento) REFERENCES Evento(idEvento),
    FOREIGN KEY(codigoUtilizador) REFERENCES Utilizador(codigoUtilizador)
    );

    CREATE TABLE ConfirmacaoServico (
```

```
idServico                   INTEGER NOT NULL UNIQUE,
codigoUtilizador            INTEGER NOT NULL UNIQUE,
PRIMARY KEY(idConfirmacaoServico),
FOREIGN KEY(idServico) REFERENCES Servico(idServico),
FOREIGN KEY(codigoUtilizador) REFERENCES Utilizador(codigoUtilizador)
);
```

5. Tabelas

Utilizador

código	nome	idade	email	morada	dataAdesao	dataNascimento
1	Andreia	20	andreia@mail.com	Rua da Andreia, Porto, Portugal	22/04/2016	13/03/1996
2	Inês	19	ines@mail.com	Rua da Inês, Madrid, Espanha	22/04/2016	08/08/1996
3	Catarina	20	catarina@mail.com	Rua da Catarina, São Miguel, Açores, Portugal	22/04/2016	24/02/2016

sexo	login	password	estado	numCC	numTel	validado
F	andreia	123	null	123456789	123456789	1
F	ines	123	0	null	123456780	0
F	catarina	123	0	null	123456781	0

Alojamento

idAlojamento	dataInicioDisp	dataFimDisp	morada	codigoResponsavel	idFotografia
1	01/05/2016	01/06/2016	Rua da Andreia, Porto, Portugal	1	1

Responsável

idResponsavel	codigoUtilizador	idAlojamento
1	1	1

Ocupa

idOcupacao	codigoUtilizador	idAlojamento	avaliação
1	2	1	4
2	3	1	4

Fotografia

idFotografia	nome	idAlojamento
1	Fotografia.png	1

Amizade

codigoUtilizador1	codigoUtilizador2
1	2
2	3

Pedido

idPedido	dataInicioEstadia	dataFimEstadia	numeroConvidados	motivo	codAlojamento	codUtilizador
1	02/05/2016	08/05/2016	1	"Eu quero ir à queima"	1	2

Convidado

codigoConvidado	codigoUtilizador	idPedido
3	2	1

Estadia

idEstadia	codigoUtilizador	idPedido
1	2	1
1	3	1

Reclamação

idReclamacao	dataEnvio	analisado	motivo	idEstadia	codUtilizador
1	09/05/2016	0	"A água era fria"	1	3

Parceiro

idParceiro	nome	tipo	contacto
1	Federação Académica do Porto	Estudantil	123456781
2	Senhor Guia Turístico	Guia Turistico	123456782

Evento

idEvento	nome	data	hora	morada	tipo	descrição	idParceiro
1	Queima	01/05/2016	23:00	Queimódromo, Porto	null	Festa Académica	1

Organiza

idOrganizacao	idParceiro	idEvento
1	1	1

Serviço

idServico	nome	tipo	morada	descrição	desconto	idParceiro
1	Roteiro pela cidade do Porto	Turístico	Aliados, Porto, Portugal	Rota para estrangeiros	20%	2

Oferece

idOferta	idParceiro	idServico
1	2	1

Subscrição

idSubscricao	codigoUtilizador	idParceiro
1	2	2

Confirmação Evento

idConfirmacaoEvento	idEvento	codUtilizador
1	1	2

Confirmação Serviço

idConfirmacaoServico	idServico	codUtilizador
1	1	3

6. Instruções LMD (Linguagem de Manipulação de Dados)

➔ **Selecionar todos os "hosts" validados da cidade do Porto: só nos interessa os "hosts" cujo estado seja "a não viajar", e que as casas pelas quais são responsáveis tenham a sua morada na cidade do Porto**

```
SELECT DISTINCT nome AS Host
FROM UTILIZADOR JOIN ALOJAMENTO
ON ALOJAMENTO.idResponsavel = UTILIZADOR.codigoUtilizador
WHERE ALOJAMENTO.morada LIKE '%Porto%' AND estado = 0 AND validado = 1
```

➔ **Selecionar todos os "travellers" da cidade do Porto no dia 01-05-2016: utilizadores que se encontram hospedados e com estado "a viajar", neste dia, na cidade do Porto.**

```
SELECT codigoUtilizador
FROM UTILIZADOR JOIN OCUPA USING(codigoUtilizador)
WHERE estado = 1 AND 01-05-2016 >= dataInicioPedido AND 01-05-2016 <= dataFimPedido
AND morada LIKE '%Porto%'
```

➔ **Selecionar a média de dias que os "travellers" usualmente ficam na cidade do Porto: de todos os pedidos que têm uma ocupação correspondente, verificar quais se localizam na cidade do Porto e fazer a média dos dias da estadia**

```
SELECT avg(numDias)
FROM UTILIZADOR JOIN PEDIDO USING(codigoUtilizador)
WHERE morada LIKE '%Porto%'
        AND numDias = (SELECT numDias AS (dataFimEstadia – dataInicioEstadia)
                        FROM PEDIDO JOIN ESTADIA USING(idPedido))
```

➔ **Melhor "host" da cidade do Porto: é considerado o melhor "host" aquele cuja avaliação geral (média das avaliações de todos os seus alojamentos) é superior.**

```
SELECT utilizador.nome, MAX(avalMedUtil)
FROM UTILIZADOR
WHERE utilizador.codUtilizador =  responsavel AND avalMedUtil IN (
        SELECT utilizador.codUtilizador AS responsavel, AVG(avalMedCasa) AS
        avalMedUtil
        FROM UTILIZADOR JOIN ALOJAMENTO
        ON Alojamento.codigoResponsavel = responsavel
        WHERE avalMedCasa IN (
                SELECT idAlojamento, avalMedCasa AS AVG(avaliação)
                FROM ALOJAMENTO JOIN OCUPA USING(idAlojamento)
                WHERE morada LIKE '%Porto%'
                GROUP BY idAlojamento
        )
GROUP BY idResponsavel
```

➔ **Conjunto de residências disponíveis na cidade do Porto entre os dias '3-05-2016' e '6-05-2016' ordenadas por ordem decrescente de avaliação.**

```
SELECT idAlojamento
FROM ALOJAMENTO
WHERE ALOJAMENTO.morada LIKE '%Porto%' AND
        (SELECT idAlojamento, avaliacaoMed AS AVG(avaliacao)
        FROM ALOJAMENTO JOIN OCUPA USING(idAlojamento)
        WHERE OCUPA.dataInicio >= "3-05-2016" AND OCUPA.dataFim <= "6-05-2016"
        GROUP by idAlojamento)
ORDER by OCUPA.avalicaoMed DESC
```

➔ **Quantos amigos da utilizadora "Andreia" vão usufruir de algum evento ou serviço do parceiro "Federação Académica do Porto".**

```
SELECT DISTINCT count(*)
FROM (
        SELECT DISTINCT utilizador.nome
        FROM UTILIZADOR, AMIZADE, CONFIRMACAOEVENTO
        WHERE amizade.utilizador1 = 'Andreia'
                AND amizade.utilizador2 = confirmação.codigoUtilizador
                AND confirmacao.idEvento  IN (
                        SELECT evento.idEvento
                        FROM EVENTO, PARCEIRO
                        WHERE evento.idParceiro = parceiro.idParceiro
                                AND parceiro.nome = "Federação Académica do Porto"
                        )
        UNION
        SELECT DISTINCT utilizador.nome
        FROM UTILIZADOR, AMIZADE, CONFIRMACAOSERVICO
        WHERE amizade.utilizador1 = 'Andreia'
                AND amizade.utilizador2 = confirmacaoServico.codigoUtilizador
                AND confirmacaoServico.idServico IN (
                        SELECT servico.idServico
                        FROM SERVICO, PARCEIRO
                        WHERE servivo.idParceiro = parceiro.idParceiro
                                AND parceiro.nome = "Federação Académica do Porto" )
                )
```

➔ **Ordenar os eventos de todos os serviços por ordem cronológica desde '1-05-2016' até uma certa até dia '08-05-2016' na cidade do Porto.**

```
SELECT idEVENTO
FROM EVENTO
WHERE EVENTO.data >= dataHoje AND EVENTO.data < datalimite AND EVENTO.morada LIKE "%Porto%"
ORDER by EVENTO.data CRESC
```

➔ **Lista utilizadores que não tem amigos na sua conta.**

```
(SELECT * DISTINCT nome
FROM UTILIZADOR)
MINUS
(SELECT * DISTINCT nome
FROM UTILIZADOR, AMIZADE
WHERE
        (utilizador.idUtilizador = amizade.idUtilizador1)
         OR
        (utilizador.idUtilizador = amizade.idUtilizador2))
```

➔ **Selecionar eventos organizador pelo parceiro 'Senhor Guia Turístico' do tipo 'Turistico', que o Utilizador 'Inês' tenha subscrito.**

```
SELECT nomeParceiro
FROM SUBSCRIÇÃO
WHERE
        (subscrição.idParceiro IN
                (SELECT parceiro.idParceiro AS idParceiro, parceiro.nome AS nomeParceiro
                FROM PARCEIRO JOIN SUBSCRIÇÃO USING (idParceiro)
                WHERE (parceiro.nome = 'Senhor Guia Turístico' AND parceiro.tipo =
'Turistico')))
```

```
        AND
        (subscrição.idUtilizador IN
                (SELECT utilizador.idUtilizador AS idUtilizador
                FROM UTILIZADOR JOIN SUBSCRIÇÃO USING (idUtilizador)
                WHERE (utilizador.nome = 'Inês')))
```

➔ **Selecionar alojamentos situados em Lisboa ou no Porto, cuja data de ocupação seja entre '03-05-16' e '06-05-16', cujo nome do proprietário comece por 'A' e a avaliação seja >= 4.**

```
(SELECT idAlojamento
FROM ALOJAMENTO
WHERE (alojamento.morada in ('Lisboa', 'Porto')
        AND
        alojamento.dataInicioEstadia <= '03-05-16'
        AND
        alojamento.dataFimEstadia >= '06-05-16'
        AND
        codigoResponsavel IN
                (SELECT idResponsavel
                FROM Utilizador
                WHERE utilizador.nome = '%A%')
        )
)
UNION
(SELECT idAlojamento
FROM ALOJAMENTO
WHERE(avaliacao IN
                (SELECT AVG(ocupa.avaliação) AS avaliacao
                FROM ALOJAMENTO JOIN OCUPA(idAlojamento)) >= 4))
```

➔ **Selecionar todas as reclamações enviadas entre o dia '03-05-16' e '16-05-16' cujo alojamento em questão tem avaliação >= 3 e a reclamação ainda não foi analisada.**

```
SELECT idReclamação
FROM RECLAMACAO
WHERE(reclamação.dataEnvio >= '03-05-16'
     AND
    reclamação.dataEnvio <= '16-05-16'
    AND
    reclamação.analisado = 0
    AND
    reclamação.idEstadia IN
                    (SELECT estadia.idEstadia
                    FROM ESTADIA JOIN PEDIDO USING (idPedido)
                     WHERE idAlojamento IN
                              (SELECT idAlojamento
                               WHERE avaliacao IN
                                   (SELECT AVG(ocupa.avaliação) AS avaliacao
                                     FROM ALOJAMENTO JOIN OCUPA(idAlojamento)) >= 3)
                              )
  )
```

7. Triggers

➔ **Trigger que é acionado quando alguém coloca o número de cartão de crédito. Neste caso, o utilizador passa para o estado de "validado".**

```
CREATE TRIGGER InsertCartaoCredito
AFTER INSERT Utilizador.numeroCC
BEGIN
        Utilizador.validado = 1;
END
```

➔ **Trigger acionado quando dois utilizadores se tornam amigos. Neste caso, se o utilizador1 é amigo do utilizador2, o utilizador 2 também deve ser amigo do utilizador1.**

```
CREATE TRIGGER UpdateAmizade
AFTER INSERT Amizade
(SELECT Amizade.codUtilizador1 AS utilizador1, Amizade.codUtilizador2 AS utilizador2)
BEGIN
        INSERT INTO Amizade (codigoUtilizador1, codigoUtilizador2)
        VALUES (utilizador2, utilizador1)
END
```

➔ **Trigger acionado quando adicionamos um convidado a um pedido de alojamento. No caso de adicionar um novo convidado, o parêmetro responsável pelo numero de convidados na tabela do Pedido deve ser incrementado.**

```
CREATE TRIGGER UpdateConvidados
AFTER INSERT Convidado
BEGIN
        Pedido.numeroConvidados++
END
```

➔ **Se eliminar um utilizador, todas as suas amizades devem ser eliminadas.**

```
CREATE TRIGGER eliminaAmizades
AFTER DELETE ON UTILIZADOR
WHEN (amizade.utilizador1 = utilizador.codUtilizador
        OR amizade.utilizador2 = utilizador.codUtilizador)
BEGIN
        DELETE AMIZADE
END;
```

➔ **Acionar o Trigger quando um cartão de Crédito é retirado do sistema, neste caso deixa de ser um utilizador "validado"**

```
CREATE TRIGGER UpdateCartaoCredito
AFTER UPDATE Utilizador.numeroCC
WHEN utilizador.numeroCC = NULL
BEGIN
        Utilizador.validado = 0;
END
```

8. Conclusão

Com a finalização deste trabalho, deparamo-nos com algumas dificuldades. Em que a que mais se destacou foi: a forma mais direta de obter determinada informação sob a forma de instruções LDD e de forma a fazer um uso mais alargado possível do "vocabulário" que as instruções SELECT permitem.

As instruções do tipo SELECT são importantes e muito úteis porque é através delas que podemos ter acesso à informação contida numa base de dados e apresenta-la da forma que nos dá mais jeito, tendo em conta o contexto e caso isso seja possível.

9. Bibliografia

https://www.couchsurfing.com/

https://en.wikipedia.org/wiki/CouchSurfing

http://www.flyertalk.com/forum/budget-travel/1560406-hostels-vs-couchsurfing.html

http://www.fodors.com/community/europe/travel-safety-sexual-assault-couchsurfin-gorg.cfm