Universidade do Minho

Departamento de Informática

Inteligência Artificial - Projeto - Fase 1 - Grupo Nº46

- Francisco Luís Rodrigues Claudino, A89493, LEI

- Luís Tiago Batoca Fernandes, A76712, LEI

- Hugo Lima Fernandes, A89481, LEI

- Miguel Ângelo de Sousa Martins, A89584, MIEI

2 de Dezembro de 2021

Resumo

O Presente Relatório serve de Apoio à exposição da Primeira Fase do Trabalho Prático de Inteligência Artificial. Divide-se em Cinco Partes:

Na Primeira faz-se uma Introdução do Problema, e do Pedido pelo Enunciado.

Em seguida, é feita uma Apresentação da Base de Conhecimento, bem como uma explicação de todas as Entidades contidas nesta.

Numa Terceira Fase são explicitadas as queries necessárias ao Projeto, e é descrito o seu Processo de Execução com o apoio do código-fonte.

É também feita uma Apresentação da Interface Gráfica.

Finalmente, são partilhadas algumas Conclusões, bem como alguns pareceres quanto a possível Trabalho Futuro.

Índice

Introdução

No âmbito da unidade curricular de Inteligência Artificial, foi proposta a resolução de um problema, utilizando a linguagem de programação *PROLOG*, com o intuito de representar conhecimento referente a uma empresa de distribuições, capaz de fazer entregas a diversas freguesias e moradas, bem como ser capaz de obter informação sobre esse conhecimento.

Nesse sentido, foi desenvolvido o seguinte sistema, capaz de cumprir os requisitos estabelecidos.

Descrição da Solução e Análise de Resultados

Base de Conhecimento

No sentido de retratar o conhecimento necessário ao problema foi analisada a situação. Desta análise, surgiram quatro Entidades relevantes. Estas são o **Cliente**, o **Estafeta**, a **Encomenda** e a **Entrega**.

Cliente

```
Cliente(idCliente, Morada, Freguesia) -> {V,F}
```

O **Cliente** é identificado por um **ID** único. A sua representação contém, para além disso, a sua **Morada** e a sua **Freguesia**.

Estafeta

```
Estafeta(idEstafeta,[idEncomenda]) -> {V,F}
```

O **Estafeta** é, tal como o cliente, identificado por um **ID** único. Contém também uma **Lista de ID's de Encomenda**.

Encomenda

```
Encomenda(idEncomenda, idCliente, Peso, Volume, Estado, Preco, "DataDePedido", "DataPrevistaDeEntrega") -> {V,F}
```

A **Encomenda** possui igualmente um **ID** único. Contém referências ao **Cliente** através do seu ID, bem como indicações do **Peso** e **Volume** do pacote, **Estado** da encomenda (i.e. "Entregue", "A Ser Entregue", ou seja, em viagem, e "Efetuada", ou seja, o pedido foi registado, no entanto ainda não está em viagem), **Preço** base da Encomenda, e às **Datas** do **Pedido** e de **Entrega Prevista**.

Entrega

```
Entrega(idEstafeta, Classificacao, idEncomenda, "DataDeEntregaEfetiva", "MeioDeTransporte") -> {V,F}
```

Finalmente, a **Entrega** representa uma encomenda entregue, e possui também um **ID** único. Contém referências ao **Cliente**, e à **Encomenda**, através dos IDs, tal como referências à **Data** em que a encomenda foi entregue e o meio de **Transporte** utilizado.

Povoamento da Base de Conhecimento

Definidos os formatos para as Entidades, passamos a povoar a Base de Conhecimento do seguinte modo:

- **Povoação dos Clientes :**

```
cliente(c1,m1_1,f1).
cliente(c2,m1_2,f1).
cliente(c3,m1_3,f1).
cliente(c4,m1_4,f1).
cliente(c5,m1_5,f1).
cliente(c6,m1_6,f1).
cliente(c7,m1_7,f1).
cliente(c8,m1_8,f1).
cliente(c9,m1_9,f1).
cliente(c10,m1_10,f1).
cliente(c11,m2_1,f2).
cliente(c12,m2_2,f2).
cliente(c13,m2_3,f2).
cliente(c14,m2_4,f2).
cliente(c15,m2_5,f2).
cliente(c16,m2_6,f2).
cliente(c17,m2_7,f2).
cliente(c18,m2_8,f2).
cliente(c19,m2_9,f2).
cliente(c20,m2_10,f2).
```

- **Povoação dos Estafetas :**

```
estafeta(es1,[enc1,enc11,enc21,enc31,enc41,enc51,enc52,enc53]).
estafeta(es2,[enc2,enc12,enc22,enc32,enc42]).
estafeta(es3,[enc3,enc13,enc15,enc17,enc10]).
estafeta(es4,[enc4,enc14,enc24,enc34,enc44,enc54]).
estafeta(es5,[enc5,enc15,enc25,enc35,enc45]).
estafeta(es6,[enc6,enc16,enc26,enc36,enc46]).
estafeta(es7,[enc7,enc17,enc27,enc37,enc47]).
estafeta(es8,[enc8,enc18,enc28,enc38,enc48]).
estafeta(es9,[enc9,enc19,enc29,enc39,enc49]).
estafeta(es10,[enc10,enc20,enc30,enc40,enc50]).
```

- **Povoação das Encomendas :**

```
encomenda(enc1, c1, 2, 5, "Efetuada", 5, "2021-11-14", "2021-11-18").
encomenda(enc2, c2, 10, 20, "Efetuada", 10, "2021-11-15", "2021-11-19").
encomenda(enc3, c3, 5, 10, "Efetuada", 12, "2021-11-16", "2021-11-21").
encomenda(enc4, c4, 12, 25, "Efetuada", 20, "2021-11-16", "2021-11-19").
encomenda(enc5, c5, 4, 8, "Efetuada", 34, "2021-11-17", "2021-11-22").
encomenda(enc6, c6, 7, 11, "Efetuada", 49, "2021-11-17", "2021-11-27").
encomenda(enc7, c7, 20, 12, "Efetuada", 32, "2021-11-18", "2021-11-24").
encomenda(enc8, c8, 40, 25, "Efetuada", 20, "2021-11-18", "2021-11-26").
encomenda(enc9, c9, 35, 20, "Efetuada", 35, "2021-11-19", "2021-11-22").
encomenda(enc10, c10, 22, 12, "Efetuada", 45, "2021-11-19", "2021-11-28").
encomenda(enc11, c11, 20, 15, "Efetuada", 30, "2021-11-20", "2021-11-25").
encomenda(enc12, c12, 13, 23, "Efetuada", 20, "2021-11-20", "2021-11-23").
encomenda(enc13, c13, 1, 3, "Efetuada", 2, "2021-11-21", "2021-11-29").
encomenda(enc14, c14, 15, 34, "Efetuada", 20, "2021-11-21", "2021-11-25").
encomenda(enc15, c15, 34, 15, "Efetuada", 25, "2021-11-22", "2021-11-27").
encomenda(enc16, c16, 44, 32, "Efetuada", 35, "2021-11-22", "2021-11-26").
encomenda(enc17, c17, 43, 23, "Efetuada", 45, "2021-11-23", "2021-11-28").
encomenda(enc18, c18, 34, 43, "Efetuada", 54, "2021-11-23", "2021-11-29").
encomenda(enc19, c19, 22, 12, "Efetuada", 65, "2021-11-24", "2021-11-27").
encomenda(enc20, c20, 12, 22, "Efetuada", 22, "2021-11-24", "2021-11-30").
encomenda(enc21, c20, 4, 15, "A ser Entregue", 27, "2021-11-10", "2021-11-16").
encomenda(enc22, c19, 9, 12, "A ser Entregue", 23, "2021-11-10", "2021-11-14").
encomenda(enc23, c18, 19, 34, "A ser Entregue", 24, "2021-11-11", "2021-11-17").
encomenda(enc24, c17, 23, 44, "A ser Entregue", 27, "2021-11-11", "2021-11-18").
encomenda(enc25, c16, 27, 50, "A ser Entregue", 30, "2021-11-12", "2021-11-17").
encomenda(enc26, c15, 15, 34, "A ser Entregue", 28, "2021-11-12", "2021-11-15").
encomenda(enc27, c14, 18, 40, "A ser Entregue", 31, "2021-11-13", "2021-11-19").
encomenda(enc28, c13, 35, 21, "A ser Entregue", 32, "2021-11-13", "2021-11-17").
encomenda(enc29, c12, 39, 32, "A ser Entregue", 20, "2021-11-14", "2021-11-16").
encomenda(enc30, c11, 43, 41, "A ser Entregue", 27, "2021-11-14", "2021-11-20").
encomenda(enc31, c10, 24, 10, "Entregue", 23, "2021-11-01", "2021-11-07").
encomenda(enc32, c9, 11, 5, "Entregue", 34, "2021-11-01", "2021-11-05").
encomenda(enc33, c8, 7, 2, "Entregue", 18, "2021-11-02", "2021-11-06").
encomenda(enc34, c7, 5, 30, "Entregue", 15, "2021-11-02", "2021-11-10").
encomenda(enc35, c6, 38, 32, "Entregue",20, "2021-11-03", "2021-11-08").
encomenda(enc36, c5, 50, 38, "Entregue", 23, "2021-11-03", "2021-11-05").
encomenda(enc37, c4, 34, 39, "Entregue", 45, "2021-11-04", "2021-11-09").
encomenda(enc38, c3, 25, 12, "Entregue", 30, "2021-11-04", "2021-11-10").
encomenda(enc39, c2, 20, 15, "Entregue", 12, "2021-11-05", "2021-11-11").
encomenda(enc40, c1, 10, 18, "Entregue", 7, "2021-11-05", "2021-11-09").
encomenda(enc41, c1, 22, 21, "Entregue", 16, "2021-11-06", "2021-11-12").
encomenda(enc42, c2, 26, 16, "Entregue", 9, "2021-11-06", "2021-11-15").
encomenda(enc43, c3, 34, 27, "Entregue", 24, "2021-11-07", "2021-11-10").
encomenda(enc44, c4, 53, 38, "Entregue", 34, "2021-11-07", "2021-11-12").
encomenda(enc45, c5, 32, 28, "Entregue", 2, "2021-11-08", "2021-11-14").
encomenda(enc46, c6, 27, 43, "Entregue", 29, "2021-11-08", "2021-11-15").
encomenda(enc47, c7, 16, 6, "Entregue", 75, "2021-11-09", "2021-11-16").
encomenda(enc48, c8, 18, 15, "Entregue", 20, "2021-11-09", "2021-11-14").
encomenda(enc49, c9, 5, 23, "Entregue", 13, "2021-11-10", "2021-11-15").
encomenda(enc50, c10, 20, 39, "Entregue", 100, "2021-11-10", "2021-11-17").
encomenda(enc51, c10, 1, 10, "Entregue", 23, "2021-11-01", "2021-11-07").
encomenda(enc52, c10, 2, 10, "Entregue", 23, "2021-11-01", "2021-11-07").
encomenda(enc53, c10, 3, 10, "Entregue", 23, "2021-11-01", "2021-11-07").
encomenda(enc54, c7, 4, 12, "Entregue", 25, "2021-12-01", "2021-14-07").
```

- **Povoação das Entregas :**

```
entrega(es1, 5, enc31, "2021-11-07", "Carro").
entrega(es1, 4, enc41, "2021-11-13", "Carro").
entrega(es2, 5, enc32, "2021-11-05", "Mota").
entrega(es2, 4, enc42, "2021-11-16", "Carro").
entrega(es3, 4, enc33, "2021-11-07", "Mota").
entrega(es3, 3, enc43, "2021-11-18", "Carro").
entrega(es4, 5, enc34, "2021-11-10", "Bicicleta").
entrega(es4, 5, enc44, "2021-11-11", "Carro").
entrega(es5, 5, enc35, "2021-11-08", "Carro").
entrega(es5, 5, enc45, "2021-11-12", "Carro").
entrega(es6, 3, enc36, "2021-11-10", "Carro").
entrega(es6, 4, enc46, "2021-11-17", "Carro").
entrega(es7, 4, enc37, "2021-11-10", "Carro").
entrega(es7, 4, enc47, "2021-11-18", "Mota").
entrega(es8, 5, enc38, "2021-11-10", "Carro").
entrega(es8, 5, enc48, "2021-11-12", "Mota").
entrega(es9, 3, enc39, "2021-11-15", "Mota").
entrega(es9, 4, enc49, "2021-11-17", "Bicicleta").
entrega(es10, 5, enc40, "2021-11-09", "Mota").
entrega(es10, 5, enc50, "2021-11-15", "Carro").
entrega(es1, 5, enc51, "2021-11-07", "Carro").
entrega(es1, 5, enc52, "2021-11-07", "Carro").
entrega(es1, 5, enc53, "2021-11-07", "Carro").
entrega(es4, 5, enc54, "2021-12-07", "Bicicleta").
```

Queries

No sentido de cumprir os Requisitos Propostos no Enunciado, foram realizados os seguintes Predicados.

Query 1 - Identificar o Estafeta que utilizou mais vezes um Meio de Transporte mais Ecológico

Para a resolução desta primeira Query foram utilizados dois predicados.

```
%--------------------------------- - - - - - - - - - - - - - - - - - -
% Query 1 - Identificar o Estafeta que utilizou mais vezes um Meio de Transporte mais Ecológico.

query1(Estafeta) :-
    getBicicletas(Estafetas),
    sort(Estafetas, Uniq),
    findall([Freq, X], (
        member(X, Uniq),
        include( =(X), Estafetas, XX),
        length(XX, Freq)
    ), Freqs),
    sort(Freqs, SFreqs),
    last(SFreqs, [Freq, Estafeta]).
```

O Primeiro, e Principal, consiste numa Ordenação de todas as Utilizações da **Bicicleta** que é guardada na lista **Uniq**. A Lista a ordenar é obtida, através do Predicado auxiliar **getBicicletas**.

```
% Função auxiliar da função 'query1' que encontra todos os Estafetas que usam Bicicletas.
getBicicletas(Estafetas) :-
    findall(IdEstafeta,(entrega(IdEstafeta, _, _, _, "Bicicleta")),Estafetas).
```

Em seguida, é criada uma Segunda Lista que contém uma Única Instância de cada **Entrega**, que é depois ordenada com base na Frequência de cada **Estafeta**. Finalmente, o Predicado **last** retorna o Último Elemento desta Lista, que representa a Resposta ao Problema.

Eis a query 1, executada:

```
?- query1(X).
X = es4.

?-
```

Query 2 - Identificar que Estafetas entregaram determinada(s) Encomenda(s) a um determinado Cliente

Na segunda query, foram necessários, mais uma vez, dois Predicados.

Um Predicado Auxiliar, chamado **estafetaParaCliente** é Utilizado para encontrar o **Estafeta** que Entregou uma determinada **Encomenda**.

```
% Função auxiliar da função 'query2' que encontra todas os Id's das Encomendas entregues por um determinado Estafeta.
estafetaParaCliente(IdEstafeta,IdEncomenda) :-
    entrega(IdEstafeta,_, IdEncomenda,_,_).
```

O Predicado Principal é uma Operação sobre Listas, que percorre uma Lista de **Encomendas** recursivamente, e que vai aplicando o Predicado Auxiliar para descobrir o **Estafeta** referente a cada **Encomenda**.

```
%------------------------------ - - - - - - - - - - - - - - - - - -
% Query 2 - Identificar que Estafetas entregaram determinadas Encomendas a um determinado Cliente.

query2([],[]).
query2([IdEncomenda|IdEncomendas], IdEstafetas) :-
    estafetaParaCliente(Estafeta,IdEncomenda),
    query2(IdEncomendas,Y),
    append([[Estafeta],Y],IdEstafetas).
```

Eis a query 2, executada:

```
?- query2([enc31,enc39,enc111],R).
R = [es1, es9, es1].

?-
```

Query 3 - Identificar os Clientes servidos por um determinado Estafeta

Este Predicado simplesmente aplica um *findall* que encontra todas as Ocorrências do **Estafeta** fornecido, e através da **Encomenda**, retorna os **Clientes** que serviu.

```
query3(Estafeta,ClientesOrdenados) :-
    findall(IdCliente,(entrega(Estafeta, _, IdEncomenda, _, _),encomenda(IdEncomenda, IdCliente, _, _, _, _, _, _)),Clientes),
    removeClientesDuplicados(Clientes,ClientesUnicos),
    sort(ClientesUnicos,ClientesOrdenados).
```

É, em seguida, usada a Função **removeClientesDuplicados**, para remover as possíveis Ocorrências repetidas do **Estafeta**, e tornar o resultado mais Esteticamente Agradável.

```
removeClientesDuplicados([],[]).
removeClientesDuplicados([H | T], Clientes) :- member(H, T), removeClientesDuplicados( T, Clientes).
removeClientesDuplicados([H | T], [H|T1]) :- \+member(H, T), removeClientesDuplicados( T, T1).
```

Eis a query 3, executada:

```
?- query3(es1,R).
R = [c1, c10] .

?-
```

Query 4- Calcular o Valor Faturado pela Green Distribution num determinado Dia

Nesta query, é calculado o Valor Faturado num dia, através de um *findall* que devolve os Valores de cada **Entrega** nesse dia.

Esses Valores são calculados com base em Parâmetros como o Meio de Transporte utilizado na **Entrega**, e o cumprimento da **Data Prevista** de **Entrega.** Finalmente, o predicado **sum_List** soma todos os Valores dessa Lista, resultando no Valor Final.

```
%------------------------------ - - - - - - - - - - - - - - - - - -
% Query 4 - Calcular o Valor Faturado pela Green Distribution num determinado dia.

query4(Data,TotalFaturado) :-
    findall(ValorFinal,
            ((entrega(_,_,IdEncomenda,Data,MeioDeTransporte),encomenda(IdEncomenda,_,_,_,_,Preco,_,DataPrevistaDeEntrega)),
            parse_time(DataPrevistaDeEntrega,I),
            parse_time(Data,F),
            (((MeioDeTransporte = "Bicicleta"),F > I, ValorFinal is Preco * 0.85 * 0.75);
            ((MeioDeTransporte = "Carro"),F > I, ValorFinal is Preco * 1 * 0.75);
            ((MeioDeTransporte = "Mota"),F > I, ValorFinal is Preco * 1.1 * 0.75);
            ((MeioDeTransporte = "Bicicleta"),F =< I, ValorFinal is Preco * 0.85);
            ((MeioDeTransporte = "Carro"),F =< I, ValorFinal is Preco * 1);
            ((MeioDeTransporte = "Mota"),F =< I, ValorFinal is Preco * 1.1)))
            ,Valores),
    sum_list(Valores,TotalFaturado).
```

Eis a query 4, executada:

```
?- query4("2021-11-17",R).
R = 30.0375.

?-
```

Query 5 - Identificar quais as zonas (e.g., rua ou freguesia) com maior volume de entregas por parte da Green Distribution

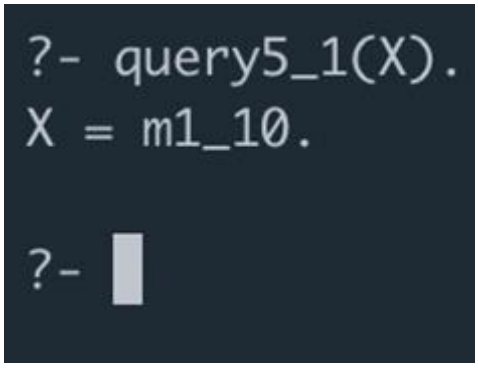
A query 5 segue a mesma Linha de Pensamento da query 1, sendo que, em vez de contar os **Estafetas** com mais Utilizações da Bicicleta, conta as Moradas com mais **Encomendas** efetuadas.

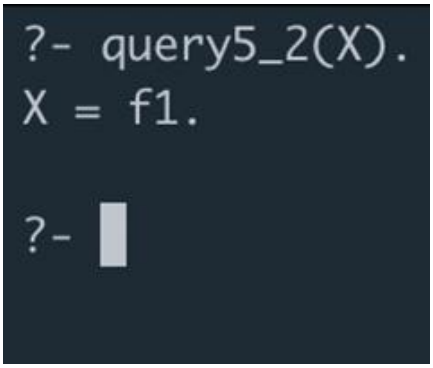
```
%------------------------------ - - - - - - - - - - - - - - - - - -
% Query 5 - Identificar quais as zonas (e.g., Morada ou Freguesia) com maior Volume de Entregas por parte da Green Distribution.

% Função que devolve a Morada com maior Volume de Entregas.
query5_1(MoradaMaisEntregas) :-
    moradasTodasEntregas(Moradas),
    sort(Moradas, Uniq),
    findall([Freq, X], (
        member(X, Uniq),
        include( =(X), Moradas, XX),
        length(XX, Freq)
    ), Freqs),
    sort(Freqs, SFreqs),
    last(SFreqs, [Freq, MoradaMaisEntregas]).

% Função auxiliar da função 'query5_1' que devolve todas as Moradas onde ocorre pelo menos uma Entregas de uma Encomenda.
moradasTodasEntregas(Moradas) :-
    findall(Morada,(cliente(IdCliente, Morada, _),encomenda(IdEncomenda, IdCliente, _, _, _, _, _),entrega(_, _, IdEncomenda, _, _)),Moradas).
```

Um Processo Idêntico é seguido para as Freguesias.

```
% Função que devolve a Freguesia com maior Volume de Entregas.
query5_2(FreguesiaMaisEntregas) :-
    freguesiasTodasEntregas(Freguesias),
    sort(Freguesias, Uniq),
    findall([Freq, X], (
        member(X, Uniq),
        include( =(X), Freguesias, XX),
        length(XX, Freq)
    ), Freqs),
    sort(Freqs, SFreqs),
    last(SFreqs, [Freq, FreguesiaMaisEntregas]).

% Função auxiliar da função 'query5_2' que devolve todas as Freguesias onde ocorre pelo menos uma Entregas de uma Encomenda.
freguesiasTodasEntregas(Freguesias) :-
    findall(Freguesia,(cliente(IdCliente, _, Freguesia),encomenda(IdEncomenda, IdCliente, _, _, _, _, _),entrega(_, _, IdEncomenda, _, _)),Freguesias).
```

Eis a query 5, executada:





Query 6 - Calcular a Classificação Média de Satisfação de Cliente para um determinado Estafeta

A query 6 pode-se dividir em duas partes :

```
query6(Estafeta,Media) :-
    classificacoesEstafeta(Estafeta,Classificacoes),
    mediaLista(Classificacoes,Media).
```

Na Primeira é gerada uma Lista de todas as **Classificações** desse **Estafeta**, através de um *findall*.

```
classificacoesEstafeta(Estafeta,Classificacoes) :-
    findall(Classificacao,entrega(Estafeta,Classificacao,_,_,_),Classificacoes).
```

Na Segunda, é calculada a Média Aritmética das **Classificações**, fazendo uso dos Predicados Auxiliares length2 e sum.

```
% Função auxiliar da função 'mediaLista' que calcula o tamanho de uma Lista.
length2([], 0).
length2([H|T], N) :- length(T, N1), N is N1 + 1.

% Função auxiliar da função 'mediaLista' que soma todos os elementos de uma Lista.
sum([],0).
sum([X|List],Sum) :-
    sum(List,Sum1),
    Sum = X + Sum1.

% Função auxiliar da função 'query6' que calcula a média dos elementos de uma Lista.
mediaLista(Lista,Media) :-
    length2(Lista,N), sum(Lista,Soma),
    Media is Soma/N.
```

Eis a query 6, executada:

```
?- query6(es1,R).
R = 4.8.

?-
```

Query 7 - Identificar o Número Total de Entregas pelos diferentes Meios de Transporte, num determinado Intervalo de Tempo

A query 7 serve para Contar quantas vezes cada **Meio de Transporte** ocorre no Intervalo de Tempo dado. Pode ser dividida em Três Partes:

```
query7(Initial_Time,Final_Time,X,Y,Z):-
    get_all_filter_time(Initial_Time,Final_Time,Time),
    filter_transports(Time,T),
    count(T,"Bicicleta",X),
    count(T,"Mota",Y),
    count(T,"Carro",Z).
```

A Primeira Parte do Predicado tem como Objetivo filtrar as **Entregas** para que restem apenas as que ocorreram dentro do Intervalo fornecido.

```
get_all_filter_time(Initial_Time,Final_Time,R):-
    findall(Entrega,filter_time(Initial_Time,Final_Time,Entrega),R).
```

```
filter_time(Initial_Time,Final_Time,Entrega):-
    get_time(X,Entrega),
    parse_time(Initial_Time,I),
    parse_time(Final_Time,F),
    parse_time(X,Y),
    Y >= I,
    Y =< F.
```

Em Seguida, é feita uma Conversão da **Entrega** para o **Meio de Transporte**.

```
filter_transports([Entrega],R):- get_transport(Entrega,X), R = [X].
filter_transports([Entrega|Entregas],R):- get_transport(Entrega,X), filter_transports(Entregas,Y), R = [X|Y].
```

Por fim, são feitas Contagens para cada Tipo de **Transporte**.

```
count([],X,0).
count([X|T],X,Y):- count(T,X,Z), Y is 1 + Z.
count([X1|T],X,Z):- X1 \= X,count(T,X,Z).
```

Eis a query 7, executada:

```
?- query7("2021-11-10","2021-11-17",X,Y,Z).
X = Y, Y = 2,
Z = 9 .

?-
```

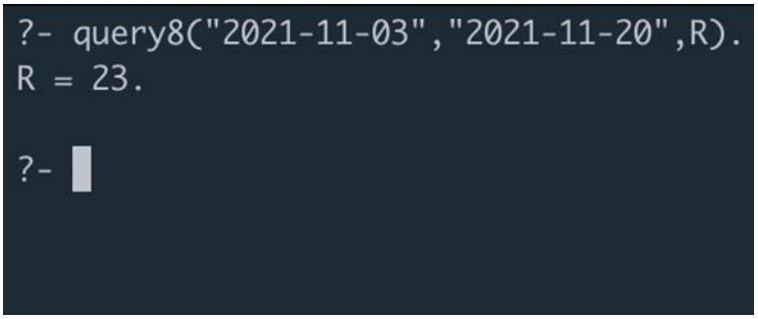
Query 8 - Identificar o Número Total de Entregas pelos Estafetas, num determinado Intervalo de Tempo

Nesta query, é mais uma vez aplicado o Predicado Auxiliar da última query. Este serve para Filtrar as **Entregas** dentro de um dado Intervalo de Tempo.

```
%------------------------------ - - - - - - - - - - - - - - - - - -
% Query 8 - Identificar o Número Total de Entregas pelos Estafetas, num determinado Intervalo de Tempo.

query8(Initial_Time,Final_Time,R):-
    get_all_filter_time(Initial_Time,Final_Time,X),
    length(X,R).
```

Em Seguida, é calculado o Comprimento dessa Lista de **Entregas**, servindo esse Valor de Resultado.

Eis a query 8, executada:



Query 9 - Calcular o Número de Encomendas Entregues e não Entregues pela Green Distribution, num determinado Período de Tempo

A query 9 pode ser dividida em Duas Partes: a parte das **Encomendas** Entregues, e as não Entregues.

```
%------------------------------ - - - - - - - - - - - - - - - - - -
% Query 9 - Calcular o Número de Encomendas Entregues e não Entregues pela Green Distribution, num determinado Período de Tempo.

query9(Initial_Time,Final_Time,X,Z):-
    get_all_filter_time_encomenda(Initial_Time,Final_Time,Time),
    filter_estado(Time,T),
    count(T,"Entregue",A),
    query7(Initial_Time,Final_Time,B,C,D),
    Z is B + C + D,
    length(T,L),
    X is L - A.
```

As **Encomendas** entregues, são calculadas através de um *findall,* que que filtra de acordo com o intervalo de tempo.

```
% Função auxiliar da função 'query9' que recolhe em R todas as Encomendas pedidas durante o Intervalo de Tempo indicado.
get_all_filter_time_encomenda(Initial_Time,Final_Time,R):-
    findall(Encomenda,filter_time_encomenda(Initial_Time,Final_Time,Encomenda),R).

% Função auxiliar da função 'get_all_filter_time_encomenda' verifica se uma Encomenda foi pedida no Intervalo de Tempo indicado.
filter_time_encomenda(Initial_Time,Final_Time,Encomenda):-
    get_time_encomenda(X,Encomenda),
    parse_time(Initial_Time,I),
    parse_time(Final_Time,F),
    parse_time(X,Y),
    Y >= I,
    Y =< F.

% Função auxiliar da função 'filter_time_encomenda' usada para obter a Data na qual uma Encomenda foi pedida.
get_time_encomenda(X,Encomenda):-
    encomenda(A,B,C,D,E,F,X,G),
    Encomenda = encomenda(A,B,C,D,E,F,X,G).
```

A Lista de Encomendas é, de seguida, convertida numa Lista de **Estados**.

```
% Função auxiliar da função 'query9' que filtra as Encomendas guardando o Estado de cada uma delas.
filter_estado([Encomenda],R):- get_estado(Encomenda,X), R = [X].
filter_estado([Encomenda|Encomendas],R):- get_estado(Encomenda,X), filter_estado(Encomendas,Y), R = [X|Y].

% Função auxiliar da função 'filter_estado' usada para obter o Estado duma encomenda.
get_estado(encomenda(_,_,_,_,S,_,_,_),S).
```

E, por fim, são Contadas as Ocorrências do **Estado** “Entregue”, e é usada a query 7 (que devolve as **Entregas** feitas por cada **Meio de Transporte**, sendo que somando esses Três Valores obtemos o Valor Total de Entregas). Subtraindo as **Encomendas** Entregues chegamos ao valor das **Encomendas** não Entregues.

Eis a query 9, executada:

```
?- query9("2021-11-15","2021-11-28",X,Y).
X = 19,
Y = 7 .

?-
```

Query 10 - Identificar o Estafeta que Utilizou mais vezes um Meio de Transporte mais Ecológico

Num Raciocínio muito Homólogo ao da query 4, é, mais uma vez, utilizado um *findall* que devolve as **Entregas** referentes ao Dia escolhido. No entanto, em vez do **Preço**, é o **Peso**. A Lista de **Pesos** é, de novo, somada, saindo daqui o Resultado Final.

```
query10(Estafeta,Data,TotalPesos) :-
    findall(Peso,(entrega(Estafeta, _,IDencomenda, Data, _),encomenda(IDencomenda,_,Peso,_,_,_,_,_)),ListaPesos),
    sum_list(ListaPesos,TotalPesos).
```

Eis a query 10, executada:

```
?- query10(es3,"2021-11-07",R).
R = 7.

?-
```

Interface Gráfica

No Sentido de Valorizar o Trabalho, e de tornar a Utilização do Programa mais digerível, foi implementada uma Interface Gráfica.

Esta interface foi Implementa recorrendo, sobretudo, às funções *write/writeln e read/readln*.

É, também, fundamental, a Implementação de uma Função *run*, que dada a query escolhida, a corre.

Eis a implementação do Menu:

```
main() :- nl,
    write('------------------------------------------------ Centro de Estatisticas da Green Distribution ------------------------------------------------'), nl, nl,
    write('-    1 - Identificar o Estafeta que utilizou mais vezes um Meio de Transporte mais Ecologico.'), nl,
    write('-    2 - Identificar que Estafetas entregaram determinadas Encomendas a um determinado Cliente.'), nl,
    write('-    3 - Identificar os Clientes servidos por um determinado Estafeta.'), nl,
    write('-    4 - Calcular o Valor Faturado pela Green Distribution num determinado dia.'), nl,
    write('-    5 - Identificar quais as zonas (e.g., Morada ou Freguesia) com maior Volume de Entregas por parte da Green Distribution.'), nl,
    write('-    6 - Calcular a Classificacao Media de Satisfacao de Cliente para um determinado Estafeta.'), nl,
    write('-    7 - Identificar o Numero Total de Entregas pelos diferentes Meios de Transporte, num determinado Intervalo de Tempo.'), nl,
    write('-    8 - Identificar o Numero Total de Entregas pelos Estafetas, num determinado Intervalo de Tempo.'), nl,
    write('-    9 - Calcular o Numero de Encomendas Entregues e nao Entregues pela Green Distribution, num determinado Periodo de Tempo.'), nl,
    write('-    10 - Calcular o Peso Total transportado por Estafeta num determinado dia.'), nl,
    write('-    11 - Sair.'), nl, nl,
    write('-------------------------------------------------------------------------------------------------------------------------------------------'), nl, nl,
    read(Choice), run_query(Choice), main.
```

```
% Função que faz com que os Inputs e os Outputs da 'query1' ocorram, quando chamados pela função 'main'.
run_query(1) :- query1(IdEstafeta),
    write('O Estafeta que utilizou mais vezes o Meio de Transporte mais Ecologico foi o '),  write(IdEstafeta), writeln('.').

% Função que faz com que os Inputs e os Outputs da 'query2' ocorram, quando chamados pela função 'main'.
run_query(2) :- write('Indique qual/quais a(s) Encomenda(s) que pretende analisar : '), read(IdEncomenda),
    query2(IdEncomenda, IdEstafeta), write('O Estafeta que entregou a(s) encomenda(s) '),
    write(IdEncomenda), write(' foi o '), write(IdEstafeta), writeln('.').

% Função que faz com que os Inputs e os Outputs da 'query3' ocorram, quando chamados pela função 'main'.
run_query(3) :- write('Indique qual o Estafeta que pretende analisar : '), read(Estafeta),
    query3(Estafeta,Clientes), write('Os Clientes servidos por um determinado Estafeta foram : '), write(Clientes), writeln('.').

% Função que faz com que os Inputs e os Outputs da 'query4' ocorram, quando chamados pela função 'main'.
run_query(4) :- write('Indique qual a Dia que pretende analisar, no formato "YYYY-MM-DD" : '), read(Data),
    query4(Data,TotalFaturado), write('O Valor Faturado pela Green Distribution no dia '), write(Data),
    write(' foi '), write(TotalFaturado), writeln('.').

% Função que faz com que os Inputs e os Outputs da 'query5' ocorram, quando chamados pela função 'main'.
run_query(5) :- query5_1(MoradaMaisEntregas), query5_2(FreguesiaMaisEntregas),
    write('A Freguesia com maior volume de Entregas e a : '),  write(FreguesiaMaisEntregas), writeln('.'),
    write('A Morada com maior volume de Entregas e a : '), write(MoradaMaisEntregas), writeln('.').

% Função que faz com que os Inputs e os Outputs da 'query6' ocorram, quando chamados pela função 'main'.
run_query(6) :- write('Indique qual o Estafeta que pretende analisar : '), read(Estafeta),
    query6(Estafeta,Media), write('A Classificacao do Estafeta '), write(Estafeta), write(' e '), write(Media), writeln('.').

% Função que faz com que os Inputs e os Outputs da 'query7' ocorram, quando chamados pela função 'main'.
run_query(7) :- write('Indique qual a Data Inicial que pretende analisar, no formato "YYYY-MM-DD" : '), read(Initial_Time),
    write('Indique qual a Data Final que pretende analisar, no formato "YYYY-MM-DD" : '), read(Final_Time),
    query7(Initial_Time,Final_Time,X,Y,Z),
    write('O Numero Total de Entregas realizadas por Bicicleta foi : '),
    write(X), write(', por Mota foi : '), write(Y), write(' e por Carro foi : '), write(Z), writeln('.').

% Função que faz com que os Inputs e os Outputs da 'query8' ocorram, quando chamados pela função 'main'.
run_query(8) :- write('Indique qual a Data Inicial que pretende analisar, no formato "YYYY-MM-DD" : '), read(Initial_Time),
    write('Indique qual a Data Final que pretende analisar, no formato "YYYY-MM-DD" : '), read(Final_Time),
    query8(Initial_Time,Final_Time,R), write('O Numero Total de Entregas realizadas entre as Datas escolhidas foram : '),  write(R), writeln('.').

% Função que faz com que os Inputs e os Outputs da 'query9' ocorram, quando chamados pela função 'main'.
run_query(9) :- write('Indique qual a Data Inicial que pretende analisar, no formato "YYYY-MM-DD" : '), read(Initial_Time),
    write('Indique qual a Data Final que pretende analisar, no formato "YYYY-MM-DD" : '), read(Final_Time),
    query9(Initial_Time,Final_Time,X,Y),
    write('O Numero de Encomendas Entregues pela Green Distribution e : '), write(Y),
    write(' e o Numero de Encomendas nao Entregues pela Green Distribution e : '), write(X), writeln('.').

% Função que faz com que os Inputs e os Outputs da 'query10' ocorram, quando chamados pela função 'main'.
run_query(10) :- write('Indique qual o Estafeta que pretende analisar : '), read(Estafeta),
    write('Indique qual a Dia que pretende analisar, no formato "YYYY-MM-DD" : '), read(Data),
    query10(Estafeta,Data,TotalPesos), write('O Estafeta '), write(Estafeta), write(' transportou '),
    write(TotalPesos), write('Kg no dia '), write(Data), writeln('.').

% Função que faz com que a função 'main' acabe de executar.
run_query(11) :- nl, halt.

% Função que invalida qualquer opção que não esteja na 'main'.
run_query(_) :- writeln("Escolha uma Opcao Valida!").
```

Conclusão

Nesta Primeira Fase do Projeto, construímos a Base de Conhecimentos, povoamos a mesma, e elaboramos um conjunto de funções para responder às funcionalidades pretendidas pelo Enunciado do Projeto. Também desenvolvemos uma Interface Gráfica, onde estas funcionalidades se tornam mais interativas, com um Sistema de IO Funcional e Simples.

Trabalho Futuro

No entanto, por falta de tempo, não conseguimos implementar duas funcionalidades ao nosso projeto, pelo que pretendemos implementá-las para a Fase 2 do Projeto.

A primeira delas seria tornar a nossa Base de Conhecimentos expansível, ou seja, criar Invariantes e um Sistema de Inferência que nos permitisse adicionar informação à Base de Conhecimentos. Necessitávamos de alterar a main, para conter um Menu Inicial que funcionasse como uma aplicação, onde Clientes, Estafetas e Gestores pudessem registar-se e entrar na aplicação. O Cliente registava-se, sendo adicionado à Base de Conhecimentos, e poderia realizar várias Encomendas, sendo estas também adicionadas à Base de Conhecimentos. O Estafeta registava-se, sendo adicionado à Base de Conhecimentos e ser-lhe ia atribuída uma lista de Encomendas para entregar, ficando encarregue de realizar as Entregas de tais Encomendas. Por fim, o Gestor, registava-se com Username e Password, sendo adicionado à Base de Conhecimentos como uma nova "Entidade", e apenas os Gestores que se encontrassem na Base de Conhecimentos teriam acesso à main atual, ao "Centro de Estatísticas da Green Distribution".

A segunda consistiria em construir a Base de Conhecimentos e povoar a mesma, tendo por base uma Base de Dados Relacional. Em vez de termos ficheiros .pl como a nossa Base de Conhecimento, ela estaria armazenada numa Base de Dados Relacionais, sendo as queries feitas à mesma. Além da falta de tempo, deparamo-nos com escassa Documentação Disponível para perceber como o fazer.