



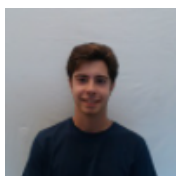
**Universidade do Minho**  
Escola de Engenharia

UNIVERSIDADE DO MINHO  
LICENCIATURA EM ENGENHARIA INFORMÁTICA

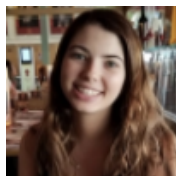
# INTELIGÊNCIA ARTIFICIAL

## Trabalho Prático

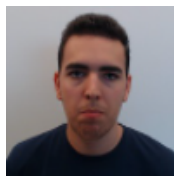
### Grupo 55



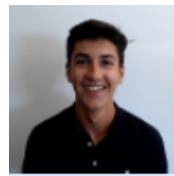
André  
Pizarro  
a93275



Isabela  
Perez  
a90723



João  
Giesteira  
a93295



Júlio  
Gonçalves  
a93243

2 de dezembro de 2021

# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>4</b>
<b>2</b>	<b>Declarações Iniciais</b>	<b>5</b>
2.1	Base de conhecimento . . . . .	5
<b>3</b>	<b>Funcionalidades</b>	<b>8</b>
3.1	Identificar o estafeta que utilizou mais vezes um meio de transporte mais ecológico	9
3.2	Identificar que estafetas entregaram determinada(s) encomenda(s) a um determinado cliente . . . . .	9
3.3	Identificar os clientes servidos por um determinado estafeta . . . . .	10
3.4	Calcular o valor faturado pela Green Distribution num determinado dia . . . . .	10
3.5	Identificar quais as zonas (e.g., rua ou freguesia) com maior volume de entregas por parte da Green Distribution . . . . .	11
3.6	Calcular a classificação média de satisfação de cliente para um determinado estafeta	11
3.7	Identificar o número total de entregas pelos diferentes meios de transporte,num determinado intervalo de tempo . . . . .	12
3.8	Identificar o número total de entregas pelos estafetas, num determinado intervalo de tempo . . . . .	12
3.9	Calcular o número de encomendas entregues e não entregues pela Green Distribution, num determinado período de tempo . . . . .	13
3.10	Calcular o peso total transportado por estafeta num determinado dia . . . . .	14
<b>4</b>	<b>Predicados Auxiliares</b>	<b>15</b>
4.1	listarEstafetas . . . . .	15
4.2	listarClientes . . . . .	15
4.3	listarEncomendas . . . . .	15
4.4	listarEntregas . . . . .	15
4.5	finalizaEncomenda . . . . .	15
4.6	eConcluida . . . . .	16
4.7	diferentes . . . . .	16
4.8	removerElemento . . . . .	16
4.9	somaC . . . . .	16
4.10	concatenar . . . . .	16
4.11	comprimento . . . . .	16
4.12	teste . . . . .	17
4.13	naoNegativoLista . . . . .	17
4.14	naoNegativo . . . . .	17
4.15	contem . . . . .	17
4.16	media . . . . .	17
4.17	removeAll . . . . .	17
4.18	count . . . . .	18
4.19	calculaValorEcologico . . . . .	18
4.20	devolveListaVeiculos . . . . .	18
4.21	valorEcologico . . . . .	18
4.22	velocidade . . . . .	18
4.23	testaTransporte . . . . .	18
4.24	testaData . . . . .	19
4.25	currentDate . . . . .	19
4.26	getPeso . . . . .	19
4.27	testaPesoTransporte . . . . .	19
4.28	testaClassificacao . . . . .	19
4.29	Datas . . . . .	19

<b>5</b>	<b>Invariantes</b>	<b>20</b>
5.1	Encomenda . . . . .	20
5.2	Entrega . . . . .	20
5.3	Estafeta . . . . .	20
5.4	Cliente . . . . .	21
<b>6</b>	<b>Conclusão</b>	<b>22</b>

# 1 Introdução

Para o primeiro exercício prático foi-nos pedido para desenvolver um projeto de programação em lógica PROLOG com capacidade para caracterizar um universo de discurso na área da logística de distribuição de encomendas, entre outros objetos.

Através deste será nos possível mostrar o conhecimento adquirido durante esta fase, destacando-se a representação de conhecimento, desenvolvimento de predicados e invariantes recorrendo à linguagem PROLOG.

O projeto será desenvolvido tendo como identidade uma empresa de nome 'Green Distribution' que apela à vertente ecológica.

## 2 Declarações Iniciais

Para começarmos a resolução dos problemas, iniciamos a fazer algumas declarações iniciais e definir a quantidade de argumentos relativos a cada conhecimento. Teremos então os seguintes factos na base de conhecimento, **encomenda(6)**, **entrega(5)**, **estafeta(2)**, **cliente(3)** e **concluido(2)**, mais um predicado **solucoes(T,Q,S)** que servirá de alias para o **findall(T,Q,S)**.

---

```
1  % SICStus PROLOG: Declarações iniciais.
2  :- dynamic encomenda/6.
3  :- dynamic entrega/5.
4  :- dynamic estafeta/2.
5  :- dynamic cliente/3.
6  :- dynamic concluido/3.
7  %-----
8  % SICStus PROLOG: Definições iniciais.
9  :- op( 900,xfy,'::' ).
10 solucoes(T,Q,S) :- findall(T,Q,S).
11 %-----
12 evolucao( Termo ) :-
13     solucoes( Invariante,+Termo::Invariante,Lista ),
14     insercao( Termo ),
15     teste( Lista ).
16
17 insercao( Termo ) :-
18     assert( Termo ).
19 insercao( Termo ) :-
20     retract( Termo ),!,fail.
21 %-----
22 involucao( Termo ) :-
23     solucoes( Invariante, -Termo::Invariante, Lista ),
24     remocao( Termo ),
25     teste( Lista ).
26
27 remocao( Termo ) :-
28     retract( Termo ).
29 remocao( Termo ) :-
30     assert( Termo ),!,fail.
```

---

### 2.1 Base de conhecimento

Definimos os factos da base de conhecimento que são pedidos no enunciado, que por definição serão sempre verdadeiros, sendo estes:

- Concluídos - Identificado por IdEntrega, IdEncomenda e DataConcluida.
- Encomenda - Identificado por IdEncomenda, Freguesia, Peso, Volume, Preço e Estado.
- Entrega - Identificado por IdEntrega, Data, IdEncomenda, Prazo e Transporte.
- Estafeta - Identificado por NomeEstafeta e IdEntrega.
- Cliente - Identificado por NomeCliente, IdEntrega e Classificacao.

---

```
1  %----- Base de Conhecimento -----
2
3  % Concluído - Predicado para dar as encomendas que já foram entregues;
4  % Extensão do concluído : IdEntrega, IdEncomenda, DataConcluida -> { V, F }.
5  concluido(5, 4, date(2021,11,8)).
6  concluido(7, 7, date(2021,6,9)).
7  concluido(6, 5, date(2021,6,9)).
```

```

8   concluido(3, 6, date(2021,11,12)).
9   concluido(9, 9, date(2021,6,9)).
10
11  %----- Encomenda - - - - -
12  % Extensão de encomenda : IdEncomenda, Freguesia, Peso, Volume, Preço, Estado -> { V, F }.
13  encomenda(1, prado, 15, 5, 50, pendente).
14  encomenda(2, maximinos, 3, 10, 60, caminho).
15  encomenda(3, prado, 12, 67, 250, pendente).
16  encomenda(4, lamacaes, 10, 18, 27, finalizada).
17  encomenda(5, gualtar, 2, 2, 1500, finalizada).
18  encomenda(6, lomar, 70, 4, 30, caminho).
19  encomenda(7, prado, 30, 4, 42, finalizada).
20  encomenda(8, merelim, 35, 3, 25, caminho).
21  encomenda(9, braga, 20, 2, 21, finalizada).
22
23  %----- Entrega - - - - -
24  % Extensão de entrega : IdEntrega, Data, IdEncomenda, Prazo, Transporte -> {V, F}.
25  entrega( 4, date(2021,10,2), 2,13, bicicleta).
26  entrega( 2, date(2021,10,3), 3, 1, moto).
27  entrega( 1, date(2021,11,4), 1, 2, moto).
28  entrega( 3, date(2021,11,7), 6, 3, carro).
29  entrega( 5, date(2021,11,7), 4, 2, moto).
30  entrega( 6, date(2021, 6,6), 5, 7, bicicleta).
31  entrega( 7, date(2021, 6,4), 7, 8, carro).
32  entrega( 8, date(2021, 9,10), 8, 5, carro).
33  entrega( 9, date(2021, 6,1), 9, 20, carro).
34
35  %----- Estafeta - - - - -
36  % Extensão de estafeta : NomeEstafeta, IdEntrega -> {V, F}.
37  estafeta(manuel,1).
38  estafeta(luis,3).
39  estafeta(andre,2).
40  estafeta(maria,5).
41  estafeta(andre,4).
42  estafeta(maria,7).
43  estafeta(andre,6).
44  estafeta(luis,8).
45  estafeta(andre,9).
46
47  %----- Cliente - - - - -
48  % Extensão de cliente : NomeCliente, IdEntrega, Classificacao -> {V, F}.
49  cliente(joao, 1,3).
50  cliente(joaquim, 5,3).
51  cliente(joao, 2,2).
52  cliente(martim, 3,5).
53  cliente(daniel, 4,5).
54  cliente(ze, 8, 4).
55  cliente(joao,6, 3.5).
56  cliente(martim, 7, 2).
57  cliente(ze, 9, 2).

```

---

```
?- listing(cliente).
:- dynamic cliente/3.

cliente(joao, 1, 3).
cliente(joaquim, 5, 3).
cliente(joao, 2, 2).
cliente(martim, 3, 5).
cliente(daniel, 4, 5).
cliente(ze, 8, 4).
cliente(joao, 6, 3.5).
cliente(martim, 7, 2).
cliente(ze, 9, 2).

true.
```

Figura 1: Base de Conhecimento Cliente

```
?- listing(concluido).
:- dynamic concluido/3.

concluido(5, 4, date(2021, 11, 8)).
concluido(7, 7, date(2021, 6, 9)).
concluido(6, 5, date(2021, 6, 9)).
concluido(3, 6, date(2021, 11, 12)).
concluido(9, 9, date(2021, 6, 9)).
```

Figura 2: Base de Conhecimento Concluído

```
?- listing(encomenda).
:- dynamic encomenda/6.

encomenda(1, prado, 15, 5, 50, pendente).
encomenda(2, maximinos, 3, 10, 60, caminho).
encomenda(3, prado, 12, 67, 250, pendente).
encomenda(4, lamacaes, 10, 18, 27, finalizada).
encomenda(5, gualtar, 2, 2, 1500, finalizada).
encomenda(6, lomar, 70, 4, 30, caminho).
encomenda(7, prado, 30, 4, 42, finalizada).
encomenda(8, merelim, 35, 3, 25, caminho).
encomenda(9, braga, 20, 2, 21, finalizada).
```

Figura 3: Base de Conhecimento Encomendas

```
?- listing(entrega).
:- dynamic entrega/5.

entrega(4, date(2021, 10, 2), 2, 13, bicicleta).
entrega(2, date(2021, 10, 3), 3, 1, moto).
entrega(1, date(2021, 11, 4), 1, 2, moto).
entrega(3, date(2021, 11, 7), 6, 3, carro).
entrega(5, date(2021, 11, 7), 4, 2, moto).
entrega(6, date(2021, 6, 6), 5, 7, bicicleta).
entrega(7, date(2021, 6, 4), 7, 8, carro).
entrega(8, date(2021, 9, 10), 8, 5, carro).
entrega(9, date(2021, 6, 1), 9, 20, carro).
```

Figura 4: Base de Conhecimento Entrega

```
?- listing(estafeta).
:- dynamic estafeta/2.

estafeta(manuel, 1).
estafeta(luis, 3).
estafeta(andre, 2).
estafeta(maria, 5).
estafeta(andre, 4).
estafeta(maria, 7).
estafeta(andre, 6).
estafeta(luis, 8).
estafeta(andre, 9).

true.
```

Figura 5: Base de Conhecimento Estafeta

### 3 Funcionalidades

No enunciado deste exercício prático são pedidas as seguintes funcionalidades:

- Identificar o estafeta que utilizou mais vezes um meio de transporte mais ecológico;
- Identificar que estafetas entregaram determinada(s) encomenda(s) a um determinado cliente;
- Identificar os clientes servidos por um determinado estafeta;
- Calcular o valor facturado pela Green Distribution num determinado dia;
- Identificar quais as zonas (e.g., rua ou freguesia) com maior volume de entregas por parte da Green Distribution;
- Calcular a classificação média de satisfação de cliente para um determinado estafeta;
- Identificar o número total de entregas pelos diferentes meios de transporte, num determinado intervalo de tempo;
- Identificar o número total de entregas pelos estafetas, num determinado intervalo de tempo;
- Calcular o número de encomendas entregues e não entregues pela Green Distribution, num determinado período de tempo;
- Calcular o peso total transportado por estafeta num determinado dia.



### 3.1 Identificar o estafeta que utilizou mais vezes um meio de transporte mais ecológico

O predicado **estafetaEcológico** começa por guardar a lista de todos os estafetas listados na base de dados, e chama o predicado auxiliar **estafetaEcológicoAux**.

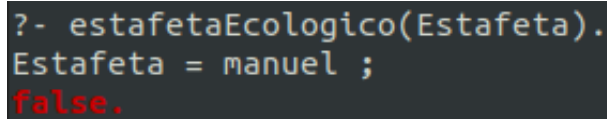
O predicado **estafetaEcológicoAux**, percorre a lista de estafetas e para cada elemento da lista calcula o valor ecológico de todas as entregas a que este estava associado. Compara o valor com o que tem guardado na variável **MIN**, caso seja menor, guarda o valor e atualiza o nome do estafeta para depois devolver, caso contrário avança na lista.

O predicado **calcula**, dado o nome de um estafeta, devolve o valor ecológico total das encomendas a ele associadas

---

```
1 % Apresenta o estafeta que utilizou mais vezes um meio de transporte ecológico.
2 % Extensão do predicado estafetaEcológico: Nome cliente, Lista -> {V, F}.
3 estafetaEcológico(FINAL) :- listarEstafetas(R), estafetaEcológicoAux(R,100,_,FINAL).
4
5 % Extensão do predicado estafetaEcológicoAux: Lista, Variavel, Variavel, Variavel -> {V, F}.
6 estafetaEcológicoAux([],_,L,L).
7 estafetaEcológicoAux([R|T],MIN,L,FINAL) :- calcula(R, R1),
8                                         (R1 < MIN) -> estafetaEcológicoAux(T, R1, R,FINAL) ; estafetaEcológicoAux(T, MIN, L, FINAL).
9
10 % Extensão do predicado estafetaEcológicoAux: Variavel, Variavel -> {V, F}.
11 calcula(R, L) :- solucoes(Id, estafeta(R, Id), R1), devolveListaVeiculos(R1, R2), calculaValorEcológico(R2, L).
```

---



```
?- estafetaEcológico(Estafeta).
Estafeta = manuel ;
false.
```

Figura 6: Resultado da Estafeta que Utilizou mais vezes Transporte Ecológico.

### 3.2 Identificar que estafetas entregaram determinada(s) encomenda(s) a um determinado cliente

O predicado **estafetaCliente**, dado o nome de um cliente, calcula uma lista de **IdsEntrega** associado a esse cliente e chama um predicado auxiliar passando-lhe essa lista.

O predicado **estafetaClienteAux**, dado uma lista de **IdsEntregas** recebida, retorna uma lista de estafetas (não repetidos) associados a essas entregas.

---

```
1 % Apresenta uma lista dos estafetas que entregaram uma encomenda a um determinado cliente
2 % Extensão do predicado estafetaCliente: Nome cliente, Lista -> {V,F}
3 estafetaCliente(NomeCliente, L) :- solucoes(ID, cliente(NomeCliente, ID, _), S),
4                                     diferentes(S, S1),
5                                     estafetaClienteAux(S1, L).
6
7 % Dado os Ids das Encomenda faz uma lista dos estafetas a que estavam associadas as encomendas
8 estafetaClienteAux([], []).
9 estafetaClienteAux([ID|T], L) :- solucoes(Nome, estafeta(Nome, ID), R1),
10                                estafetaClienteAux(T, R2),
11                                concatenar(R1, R2, R3),
12                                diferentes(R3,L).
```

---

```
?- estafetaCliente(martim,L).
L = [luis, maria].
```

Figura 7: Resultado ao Exemplo do Cliente Martim.

### 3.3 Identificar os clientes servidos por um determinado estafeta

O predicado **clienteEstafeta**, dado o nome de um estafeta, calcula uma lista de IdsEntrega associado a esse estafeta e chama um predicado auxiliar passando-lhe essa lista.

O predicado **clienteEstafetaAux**, dado uma lista de IdsEntregas recebida, retorna uma lista de clientes (não repetidos) associados a essas entregas.

---

```
1  % Apresenta uma lista dos clientes servidos por um determinado estafeta.
2  % Extensão do predicado estafetaCliente: Nome cliente, Lista -> {V, F}.
3  clienteEstafeta(NomeEstafeta, L) :- solucoes(ID, estafeta(NomeEstafeta, ID), S),
4                                     diferentes(S, S1),
5                                     clienteEstafetaAux(S1, L).
6  % Dados os Ids das encomendas faz uma lista dos clientes a que estavam associadas as encomendas.
7  clienteEstafetaAux([], []).
8  clienteEstafetaAux([ID|T], L) :- solucoes(Nome, cliente(Nome, ID, _), R1),
9                                     clienteEstafetaAux(T, R2),
10                                    concatenar(R1, R2, S),
11                                    diferentes(S, L).
```

---

```
?- clienteEstafeta(andre,L).
L = [joao, daniel, ze] ;
false.
```

Figura 8: Resultado ao Exemplo do Estafeta André.

### 3.4 Calcular o valor faturado pela Green Distribution num determinado dia

O predicado **fatura**, dado uma data, calcula uma lista com todos os IdsEncomendas que foram registados na data dada, chamando a auxiliar com esta lista, retornando a soma das faturas.

O predicado **faturaAux**, dado uma lista de IdsEncomenda, retorna uma lista com os preços associados às encomendas identificadas pelos Ids dados.

---

```
1  % Apresenta a faturação da empresa num determinado dia.
2  % Extensão do predicado estafetaCliente: Data, Variável -> {V, F}.
3  fatura(Data, L) :- solucoes(IdEncomenda, entrega(_, Data, IdEncomenda, _, _), S),
4                                     faturaAux(S, L).
5
6  % Dados os Ids das encomendas de um determinado dia calcula a faturação.
7  faturaAux([], 0).
8  faturaAux([ID|T], L) :- encomenda(ID, _, _, _, Preco, _),
9                          faturaAux(T, OldL), L is Preco + OldL.
```

---

```
?- fatura(date(2021,11,7),F).
F = 57.
```

Figura 9: Resultado do valor Facturado pela Empresa.

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

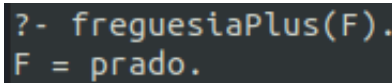
O predicado **freguesiaPlus**, calcula uma lista de freguesias e chama um predicado auxiliar passando-lhe esta lista.

O predicado **freguesiaPlusAux**, para cada elemento da lista de freguesias, calcula o seu volume de encomendas comparando o número de encomendas que cada freguesia tem registadas. Caso seja maior, atualiza a variável de controlo e guarda o nome da freguesia, caso contrário chama a recursiva. Em ambos os casos, são eliminadas as repetições da lista por motivos óbvios.

---

```
1 % Apresenta a freguesia com maior volume de encomendas.
2 % Extensão do predicado freguesiaPlus: Variável -> {V, F}.
3 freguesiaPlus(Final) :- solucoes(Freguesia, encomenda(_,Freguesia,_,_,_), R), freguesiaPlusAux(R,0,_, Final).
4
5
6 freguesiaPlusAux([], _, L, L).
7 freguesiaPlusAux([R|T],N, L, Final) :- count(R, [R|T], N1),
8                                     N1 > N -> removeAll(R, [R|T], S), freguesiaPlusAux(S, N1, R, Final)
9                                     ; removeAll(R, [R|T], S), freguesiaPlusAux(S, N, L, Final).
```

---



```
?- freguesiaPlus(F).
F = prado.
```

Figura 10: Resultado da Zona Mais Frequentada.

### 3.6 Calcular a classificação média de satisfação de cliente para um determinado estafeta

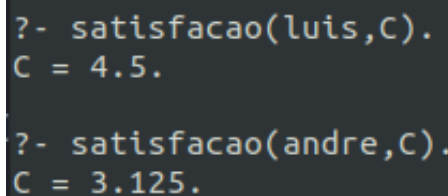
O predicado **satisfacao**, dado o nome de um estafeta, calcula uma lista de IdsEntregas associados a este estafeta e chama um predicado auxiliar passando-lhe esta lista. Por mim, faz a media da lista de classificações retornada pelo predicado auxiliar.

O predicado **satisfacaoAux**, percorre a lista de IdsEntrega passada, e para cada Id, calcula uma lista de classificações retornando esta.

---

```
1 % Apresenta a classificação media de satisfação de um estafeta.
2 % Extensão do predicado satisfacao: Nome Estafeta, Variável -> {V, F}.
3 satisfacao(Nome, L) :- solucoes(ID, estafeta(Nome, ID), S),
4                       satisfacaoAux(S, L1),length(S, X),
5                       L is L1/X.
6
7 % Dado uma lista de Ids de encomendas calcula as classificações dadas pelos clientes.
8 satisfacaoAux([],0).
9 satisfacaoAux([ID|T], Res) :- cliente(_, ID, Classificacao),!,
10                             satisfacaoAux(T, Y) , Res is Classificacao+Y.
```

---



```
?- satisfacao(luis,C).
C = 4.5.

?- satisfacao(andre,C).
C = 3.125.
```

Figura 11: Resultado da Classificação Média.

### 3.7 Identificar o número total de entregas pelos diferentes meios de transporte,num determinado intervalo de tempo

Apresenta o número total de entregas pelo diferentes meios de transporte num determinado intervalo de tempo. O predicado **totalEntregasIntervaloTempo** filtra todas as Entregas conhecidas que se encontram dentro do intervalo de tempo pedido. De seguida temos três predicados auxiliares que percorrem esta lista e contam quantas entregas correspondem a determinado veículo Bicicleta/Moto/Carro.

```
1  % Apresenta o número total de entregas pelo diferentes meios de transporte num determinado intervalo de tempo.
2  % Extensão do predicado totalEntregasVeiculo: DataI, DataF, Bicicleta/Moto/Carro -> {V, F}.
3
4  totalEntregasVeiculo(D1,D2,B/M/C) :- totalEntregasIntervaloTempo(D1,D2,L),
5      totalEntregasBic(L,B),
6      totalEntregasMoto(L,M),
7      totalEntregasCarro(L,C).
8
9  totalEntregasIntervaloTempo(D1,D2,L) :- listarEntregas( Entregas ),
10     totalEntregasIntervaloTempoAux(D1,D2,Entregas,L).
11
12 totalEntregasIntervaloTempoAux(_,_,[],[]).
13 totalEntregasIntervaloTempoAux(D1,D2,[(IdEntrega, D3, IdEncomenda, Prazo, Transporte)|T],
14 [(IdEntrega, D3, IdEncomenda, Prazo, Transporte)|T1]) :-
15     antesDe(D3,D2),
16     depoisDe(D3,D1),
17     !,
18     totalEntregasIntervaloTempoAux(D1,D2,T,T1).
19 totalEntregasIntervaloTempoAux(D1,D2,[_|T],T1) :-
20     totalEntregasIntervaloTempoAux(D1,D2,T,T1).
21
22 totalEntregasCarro([],0).
23 totalEntregasCarro([( _,_,_,_,carro)|T],X) :- totalEntregasCarro(T,Y),!, X is Y+1.
24 totalEntregasCarro([_|T],X) :- totalEntregasCarro(T,X).
25
26 totalEntregasMoto([],0).
27 totalEntregasMoto([( _,_,_,_,moto)|T],X) :- totalEntregasMoto(T,Y),!,X is Y+1.
28 totalEntregasMoto([_|T],X) :- totalEntregasMoto(T,X).
29
30 totalEntregasBic([],0).
31 totalEntregasBic([( _,_,_,_,bicicleta)|T],X) :- totalEntregasBic(T,Y),!, X is Y+1.
32 totalEntregasBic([_|T],X) :- totalEntregasBic(T,X).
```

```
?- totalEntregasVeiculo(date(2021,11,1),date(2021,11,10),L).
L = 0/2/1 ;
false.
```

Figura 12: Resultado do Número de Entregas.

### 3.8 Identificar o número total de entregas pelos estafetas, num determinado intervalo de tempo

O predicado **totalEntregasEstafeta** apresenta uma lista de pares em que o primeiro elemento é o nome dum estafeta e o segundo é o número de entregas que fez no intervalo pedido.

O predicado **totalEntregasEstafetaAux** devolve uma lista de pares em que o primeiro elemento é o nome de um estafeta e um 1, caso este tenha feito uma entrega no intervalo suposto, e o predicado **agroup** junta todos os pares contidos nesta lista para dar o resultado final.

---

```

1  % Extensão do predicado totalEntregasEstafeta: DataI, DataF, [(Estafeta,NrEntregas),...] -> {V, F}.
2
3  totalEntregasEstafeta(D1,D2,Res) :- totalEntregasIntervaloTempo(D1,D2,L), %(Entregas)
4      totalEntregasEstafetaAux(L,NewL),
5      agroup(NewL,Res).
6
7  totalEntregasEstafetaAux([],[]).
8  totalEntregasEstafetaAux([(IdEntrega, _, _, _)|T],[Nome,1|T1]) :- estafeta(Nome,IdEntrega),
9      !,
10     totalEntregasEstafetaAux(T,T1).
11
12
13  agroup([],[]).
14  agroup([(A,X)|Tail],[A,Z|Res]) :- member((A,_),Tail),
15      conta(A,Tail,Y),
16      removeParesFirst(A,Tail,Tmp),
17      agroup(Tmp,Res), Z is Y+X.
18  agroup([(A,X)|Tail],[A,X|Res]) :- not(member((A,_),Tail)), agroup(Tail,Res).
19
20  removeParesFirst(_,[],[]).
21  removeParesFirst(A,[(A,_)|Tail], Res) :- removeParesFirst(A,Tail,Res).
22  removeParesFirst(A,[(B,K)|Tail], [(B,K)|Res]) :- B\=A, removeParesFirst(A,Tail,Res).
23
24  conta(_,[],0).
25  conta(A,[(B,_)|Tail],Y) :- B\=A, conta(A,Tail,Y).
26  conta(A,[(A,X)|Tail],Y) :- conta(A,Tail,X2), Y is X2+X.

```

---

```

?- totalEntregasEstafeta(date(2021,11,1),date(2021,11,11),Res).
Res = [(manuel, 1), (luis, 1), (maria, 1)] ;
false.

```

Figura 13: Resultado ao Número de Entregas pelos Estafetas.

### 3.9 Calcular o número de encomendas entregues e não entregues pela Green Distribution, num determinado período de tempo

Mais uma vez, filtramos as entregas que **começam** dentro do período desejado com o predicado **totalEntregasIntervaloTempo**. De seguida separamos as possibilidades em três casos possíveis, onde o único válido para esta ser entregue seria estar concluída, e a data de conclusão ser dentro do período. Assim, pode haver casos em que a entrega realmente começa no período e está concluída, mas devido a só ter sido entregue já fora do prazo, a mesma vai contar como NãoEntregue.

---

```

1  totalEntreguesENaoEntregues(D1,D2,Ent/NEnt) :- totalEntregasIntervaloTempo(D1,D2,L),
2      totalEntreguesENaoEntreguesAux(D1,D2,L,Ent/NEnt).
3
4  totalEntreguesENaoEntreguesAux(_,_,[],0/0).
5  totalEntreguesENaoEntreguesAux(D1,D2,[(IdEntrega, _, _, _)|T],Ent/NEnt) :-
6      \+concluido(IdEntrega,_,_),!, totalEntreguesENaoEntreguesAux(D1,D2,T,Ent/OldNEnt), NEnt is OldNEnt + 1.
7  totalEntreguesENaoEntreguesAux(D1,D2,[(IdEntrega, _, _, _)|T],Ent/NEnt) :-
8      concluido(IdEntrega,_,D3), antesDe(D3,D2), depoisDe(D3,D1),!,
9      totalEntreguesENaoEntreguesAux(D1,D2,T, OldEnt/NEnt), Ent is OldEnt + 1.
10 totalEntreguesENaoEntreguesAux(D1,D2,[_|T],Ent/NEnt) :-
11     totalEntreguesENaoEntreguesAux(D1,D2,T,Ent/OldNEnt), NEnt is OldNEnt + 1.

```

---

```
?- totalEntreguesENaoEntregues(date(2021,11,1),date(2021,11,11),X).
X = 1/2 ;
false.
```

Figura 14: Resultado ao Número de Entregas e Não Entregas da Empresa.

### 3.10 Calcular o peso total transportado por estafeta num determinado dia

O predicado **pesoTransportado**, dado uma data X, calcula uma lista dos estafetas que fazem parte da base de conhecimento e chama uma auxiliar passando-lhe essa lista.

O predicado **procura**, dada a lista de Estafetas, para cada um destes chama o predicado auxiliar **calculaIdsEntrega**, passando o nome do estafeta e uma data.

O predicado **calculaIdsEntrega**, dada uma lista de IdsEntrega, para cada IdEntrega retorna o peso associado à encomenda.

---

```
1  % Devolve uma lista com os estafetas e o peso que cada um leva numa data em que se entregaram encomendas.
2  % Extensão do predicado peso transportado: Data, Variavel -> {V, F}.
3  pesoTransportado(X, L) :- listarEstafetas(T), procura(T, X, L).
4
5  % Dada uma lista de estafetas, calcula o peso que cada um carregava recursivamente.
6  procura([],_,[]).
7  procura([R|T], X, [R/F|L]) :- calculaIdsEntrega(R, X, F),!, procura(T, X, L).
8
9  % Dado o nome de um estafeta, calcula todos os IdEntrega associados a ele.
10 calculaIdsEntrega(R, X, L) :- solucoes(IdEntrega, estafeta(R,IdEntrega), T), calculaIdsEncomenda(T, X, L).
11
12 % Dado uma lista de IdsEntrega, verifica a data -> devolve o peso, faz a recursiva.
13 calculaIdsEncomenda([], _, 0).
14 calculaIdsEncomenda([R|T], X, L) :- \+ (concluido(R, _, X)) , calculaIdsEncomenda(T, X, L).
15 calculaIdsEncomenda([R|T], X, L) :- concluido(R, IdEncomenda, X), calculaPeso(IdEncomenda,F),
16                                     calculaIdsEncomenda(T, X, R1), L is R1 + F.
17
18 % Dado um IdEncomenda devolve o Peso associado.
19 calculaPeso(R, Peso) :- encomenda(R,_,Peso,_,_).
20 calculaPeso(_,0).
```

---

```
?- pesoTransportado(date(2021,6,9),P).
P = [manuel/0, luis/0, andre/22, maria/30] ;
false.
```

Figura 15: Resultado do Peso Total.

## 4 Predicados Auxiliares

Durante a resolução foram usados predicados que, apesar de não responderem diretamente a qualquer problema proposto, serviram de suporte para os predicados que os resolvem e para algumas funcionalidades adicionais. Em baixo são apresentados estes predicados, acompanhados de uma legenda a caracterizar a funcionalidade de cada um.

### 4.1 listarEstafetas

Devolve uma lista de todos os estafetas registados na base de conhecimento.

---

```
1 % Extensão do predicado listarEstafetas: Lista -> {V, F}.
2 listarEstafetas( L ) :- solucoes(Nome, estafeta(Nome, _), R), diferentes(R, L).
```

---

### 4.2 listarClientes

Devolve uma lista de todos os clientes registados na base de conhecimento.

---

```
1 % Extensão do predicado listarCliente: Lista -> {V, F}.
2 listarCliente( L ) :- solucoes(Nome, cliente(Nome, _, _), R),
3                       diferentes(R, L).
```

---

### 4.3 listarEncomendas

Devolve uma lista de todas as encomendas registadas na base de conhecimento.

---

```
1 % Extensão do predicado listarEncomendas: Lista -> {V, F}.
2 listarEncomendas( L ) :- solucoes((IdEncomenda, Nome, Peso, Volume, PrecoEncomenda, Estado),
3                                encomenda(IdEncomenda, Nome, Peso, Volume, PrecoEncomenda, Estado), R),
4                                diferentes(R, L).
```

---

### 4.4 listarEntregas

Devolve uma lista de todas as entregas registadas na base de conhecimento.

---

```
1 % Extensão do predicado listarEntregas: Variável -> {V, F}.
2 listarEntregas( L ) :- solucoes((IdEntrega, Data, IdEncomenda, Prazo, Transporte),
3                                entrega(IdEntrega, Data, IdEncomenda, Prazo, Transporte), R),
4                                diferentes(R, L).
```

---

### 4.5 finalizaEncomenda

Predicado que dado um IdEncomenda, altera o estado da encomenda correspondente para finalizada e insere na base de conhecimentos essa encomenda. como concluída.

---

```
1 finalizaEncomenda(L) :- solucoes((L, Freguesia, Peso, Volume, Preco, _),
2                                encomenda(L, Freguesia, Peso, Volume, Preco, _), [R|_]),
3                                finalizaEncomendaAuxI(L),
4                                finalizaEncomendaAuxE(R),
5                                eConcluida(L).
6
7 % Faz a involução da encomenda.
8 finalizaEncomendaAuxI(L) :- involucao(encomenda(L, _, _, _, _)).
9
10 % Faz a evolução da encomenda com o estado atualizado.
11 finalizaEncomendaAuxE((L, Freguesia, Peso, Volume, Preco, _)) :- evolucao(encomenda(L, Freguesia, Peso, Volume, Preco, finaliz
```

---

## 4.6 eConcluida

Após uma encomenda ser dado como finalizada, aumenta a base do conhecimento introduzindo um facto concluído associada à encomenda em questão.

---

```
1 % Extensão do predicado eConcluida: Variavel -> {V,F}
2 eConcluida(L) :- solucoes(R,entrega(R,_,L,_,_), [F|_]),
3               currentDate(D),
4               evolucao(concluido(F,L,D)).
```

---

## 4.7 diferentes

Remove os elementos repetidos de uma lista.

---

```
1 % Extensão do predicado diferentes: L1, L2 -> {V, F}.
2 diferentes( [],[] ).
3 diferentes( [X|L],[X|NL] ) :- removerElemento( L,X,TL ),
4                             diferentes( TL,NL ).
```

---

## 4.8 removerElemento

Remove um elemento de uma lista.

---

```
1 % Extensão do predicado removerElemento: L1, Y, L2 -> {V, F}.
2 removerElemento( [],_,[] ).
3 removerElemento( [X|L],X,NL ) :- removerElemento( L,X,NL ).
4 removerElemento( [X|L],Y,[X|NL] ) :- X \== Y, removerElemento( L,Y,NL ).
```

---

## 4.9 somaC

Soma os elementos de uma lista.

---

```
1 % Extensão do predicado somaC : LN,R -> {V, F}.
2 somaC([],0).
3 somaC([X],X).
4 somaC([X|L],R) :- somaC(L,RL), R is X+RL.
```

---

## 4.10 concatenar

Concatena duas listas.

---

```
1 % Extensão do predicado concatenar: L1,L2,L3 -> {V, F}.
2 concatenar( [],L,L ).
3 concatenar( [H|T],L2,[H|L] ) :- concatenar(T,L2,L).
```

---

## 4.11 comprimento

Calcula o comprimento de uma lista.

---

```
1 % Extensão do predicado comprimento: L,R -> {V, F}.
2 comprimento([],0).
3 comprimento(_|T,R) :- comprimento(T,N), R is N+1.
```

---



## 4.12 teste

Testa todos os elementos da lista.

---

```
1 % Extensão do predicado teste: [R/LR] -> {V, F}.
2 teste([]).
3 teste([I|L]) :- I, teste(L).
```

---

## 4.13 naoNegativoLista

Verifica se todos os elementos da lista são não negativos.

---

```
1 % Extensão do predicado naoNegativo: L -> {V, F}.
2 naoNegativoLista([]).
3 naoNegativoLista([H|T]) :- H >=0, naoNegativoLista(T).
```

---

## 4.14 naoNegativo

Verifica se todos os elementos da lista são não negativos.

---

```
1 % Extensão do predicado naoNegativo: L -> {V, F}.
2 naoNegativo(L) :- L >=0.
```

---

## 4.15 contem

Verifica se uma lista contém um elemento dado.

---

```
1 % Extensão do predicado contem: H, [H/T] -> {V, F}.
2 contem(H, [H|_]).
3 contem(X, [_|T]) :- contem(X, T).
```

---

## 4.16 media

Faz a media de dos elementos de uma lista.

---

```
1 % Extensão do predicado media: L,R -> {V, F}.
2 media(L,S) :- somaC(L,S1),
3             comprimento(L,S2), S is S1/S2.
```

---

## 4.17 removeAll

Remove os elementos repetidos de uma lista.

---

```
1 % Extensão do predicado removeAll: Variavel, Lista, Variavel -> {V,F}
2 removeAll(_, [], []).
3 removeAll(X, [X|T], L):- removeAll(X, T, L), !.
4 removeAll(X, [H|T], [H|L]):- removeAll(X, T, L ).
```

---

## 4.18 count

Conta o numero de elementos iguais numa lista.

---

```
% Extensão do predicado count: Variavel, Lista, Variavel -> {V,F}
count(_, [], 0).
count(X, [X | T], N) :- !, count(X, T, N1), N is N1 + 1.
count(X, [_ | T], N) :- count(X, T, N).
```

---

## 4.19 calculaValorEcologico

Calcula o valor ecológico total de uma lista de transportes.

---

```
% Extensão do predicado calculaValorEcologico: Lista, Variavel -> {V,F}
calculaValorEcologico([T], L) :- valorEcologico(T, L).
calculaValorEcologico([H|T], L) :- valorEcologico(H,R),
                                   calculaValorEcologico(T,RL), L is R+RL.
```

---

## 4.20 devolveListaVeiculos

Dada uma lista de IdsEntregas, retorna uma lista dos transportes associados a essas entregas.

---

```
% Extensão do predicado count: Lista, Variavel -> {V,F}
devolveListaVeiculos([], []).
devolveListaVeiculos([H|T], L) :- solucoes(Veiculo, entrega(H, _, _, Veiculo), R),
                                   devolveListaVeiculos(T,R1),
                                   concatenar(R,R1,L).
```

---

## 4.21 valorEcologico

Dado um transporte, devolve o valor ecológico associado.

---

```
% Extensão do predicado valorEcologico: Variavel, Variavel -> {V, F}.
valorEcologico(bicicleta, L):- L is 1.
valorEcologico(moto, L):- L is 2.
valorEcologico(carro, L):- L is 3.
```

---

## 4.22 velocidade

Dado um transporte, devolve a velocidade associada.

---

```
% Extensão do predicado velocidade: Variavel, Variavel -> {V, F}.
velocidade(bicicleta, L) :- L is 10.
velocidade(moto, L) :- L is 35.
velocidade(carro, L) :- L is 25.
```

---

## 4.23 testaTransporte

Testa se um meio de transporte é válido.

---

```
% Extensão do predicado testaTransporte: Variavel -> {V, F}.
testaTransporte(bicicleta).
testaTransporte(moto).
testaTransporte(carro).
```

---

## 4.24 testaData

Testa se uma data é valida.

---

```
% Extensão do predicado count: Variavel -> {V,F}
testaData(date(Y,M,D)) :- Y > 2000, Y < 2025, M > 0, M < 13, D > 0, D < 32.
```

---

## 4.25 currentDate

Devolve a data atual.

---

```
% Extensão do predicado count: Variavel -> {V,F}
currentDate(Today) :-
    get_time(Stamp),
    stamp_date_time(Stamp,DateTime,local),
    date_time_value(date,DateTime,Today).
```

---

## 4.26 getPeso

Dado um IdEncomenda devolve o peso desta.

---

```
% Extensão do predicado count: Variavel, Variavel -> {V,F}
getPeso(Id,Peso) :- encomenda(Id,_,Peso,_,_).
```

---

## 4.27 testaPesoTransporte

Dado um peso, testa se o meio de transporte é adequado.

---

```
% Extensão do predicado count: Variavel, Variavel -> {V,F}
testaPesoTransporte(L, carro) :- L <= 100 .
testaPesoTransporte(L, moto) :- L <= 20 .
testaPesoTransporte(L, bicicleta) :- L <= 5 .
```

---

## 4.28 testaClassificacao

Testa se uma classificação é válida.

---

```
% Extensão do predicado count: Variavel -> {V,F}
testaClassificacao(L) :- L >= 0, L <= 5.
```

---

## 4.29 Datas

---

```
antesDe(date(A1,_,_),date(A2,_,_)) :- A1 < A2,! .
antesDe(date(A1,M1,_),date(A1,M2,_,_)) :- M1 < M2,! .
antesDe(date(A1,M1,D1),date(A1,M1,D2)) :- D1 <= D2 .

depoisDe(date(A1,_,_),date(A2,_,_)) :- A1 > A2,! .
depoisDe(date(A1,M1,_,_),date(A1,M2,_,_)) :- M1 > M2,! .
depoisDe(date(A1,M1,D1),date(A1,M1,D2)) :- D1 >= D2 .
```

---

## 5 Invariantes

De seguida são apresentados os invariantes estruturais que condicionam a inserção(+) e remoção(-) de conhecimento do projeto.

### 5.1 Encomenda

---

```
1  %----- Encomenda - - - - -
2
3  % Invariante Estrutural -> não permite a inserção caso já exista uma encomenda com esse IdEncomenda.
4  +encomenda(IdEncomenda,_,_,_,_) :: (solucoes((IdEncomenda), (encomenda(IdEncomenda,_,_,_,_)), R),
5                                     comprimento(R, L), L == 1).
6
7  % Invariante Estrutural -> não permite a inserção caso seja dado um Peso negativo.
8  +encomenda(_,_,Peso,_,_) :: naoNegativo(Peso).
9
10 % Invariante Estrutural -> não permite a inserção caso seja dado um Volume negativo.
11 +encomenda(_,_,_,Volume,_) :: naoNegativo(Volume).
12
13 % Invariante Estrutural -> não permite a inserção caso seja inserido um preço negativo.
14 +encomenda(_,_,_,_,Preco,_) :: naoNegativo(Preco).
```

---

### 5.2 Entrega

---

```
1  %----- Entrega - - - - -
2
3  % Invariante Estrutural -> não permite a inserção caso já exista uma entrega associado a esse IdEntrega.
4  +entrega(IdEntrega,_,_,_) :: (solucoes((IdEntrega), (entrega(IdEntrega,_,_,_)), R),
5                                     comprimento(R, L), L == 1).
6
7  % Invariante Estrutural -> não permite a inserção caso a data seja inválida.
8  +entrega(_,date(Y,M,D),_,_) :: testaData(date(Y,M,D)).
9
10 % Invariante Estrutural -> não permite a inserção caso o prazo seja negativo.
11 +entrega(_,_,_,Prazo,_) :: naoNegativo(Prazo).
12
13 % Invariante Estrutural -> não permite a inserção caso já existe uma entrega associada a uma encomenda.
14 +entrega(_,_,IdEncomenda,_,_) :: (solucoes((IdEncomenda), (entrega(_,_,IdEncomenda,_,_)), S),
15                                     comprimento(S,L), L == 1).
16
17 % Invariante Estrutural -> não permite a inserção caso não exista uma encomenda associada a esse IdEncomenda.
18 +entrega(_,_,IdEncomenda,_,_) :: (solucoes((IdEncomenda), (encomenda(IdEncomenda,_,_,_,_)), S),
19                                     comprimento(S,L), L == 1).
20
21 % Invariante Estrutural -> não permite a inserção caso não seja inserido um meio de transporte válido.
22 +entrega(_,_,_,_,Transporte) :: testaTransporte(Transporte).
23
24 % Invariante Estrutural -> não permite a inserção caso o peso da encomenda associada seja superior ao do transporte.
25 +entrega(_,_,IdEncomenda,_,_,Transporte) :: (getPeso(IdEncomenda,L), testaPesoTransporte(L, Transporte)).
```

---

### 5.3 Estafeta

---

```
1  %----- Estafeta - - - - -
2
3  % Invariante Estrutural -> não permite a inserção de um estafeta associado a uma entrega que não existe.
4  +estafeta(_, IdEntrega) :: (solucoes((IdEntrega), (entrega(IdEntrega,_,_,_,_)), S),
5                                     comprimento(S,L), L == 1).
```

---

```

6
7  % Invariante Estrutural -> não permite a inserção de um estafeta associado a uma entrega que já estava associada a outro
8  +estafeta(_, IdEntrega) :- (solucoes((IdEntrega), (estafeta(_, IdEntrega)), S),
9                                comprimento(S,L), L == 1).

```

---

## 5.4 Cliente

```

1  %----- Cliente - - - - -
2
3  % Invariante Estrutural -> não permite a inserção caso o cliente dê uma classificação não elegível.
4  +cliente(_,_,L) :- testaClassificacao(L).
5
6  % Invariante Estrutural -> não permite a inserção caso não seja inserido um meio de transporte válido
7  +cliente(_,IdEntrega,_) :- (solucoes((IdEntrega), (entrega(IdEntrega,_,_,_)), S),
8                                comprimento(S,L), L == 1).

```

---

## 6 Conclusão

Em geral, acreditamos que o nosso trabalho foi bem sucedido, apesar das diferentes adversidades que ocorreram durante o período de tempo que dispusemos para a realização do trabalho.

Em primeiro lugar, e mais importante, conseguimos criar um sistema que suporte as características de um universo de discurso na área da logística de distribuição de encomendas, com todas as características pedidas: declarações iniciais, base de conhecimento, funcionalidades obrigatórias, predicados auxiliares e invariantes. Além disto, ainda conseguimos adicionar algumas funcionalidades extras ao projeto que enriquecem a experiência do utilizador.

Com isto, conseguimos melhorar as nossas técnicas de programação na linguagem PROLOG, ganhando um a vontade com a língua e seu raciocínio que até então não nos era familiar.