

Universidade do Minho
Departamento de Informática

Green Distribution
Inteligência Artificial
Grupo 24

2 de Dezembro, 2021



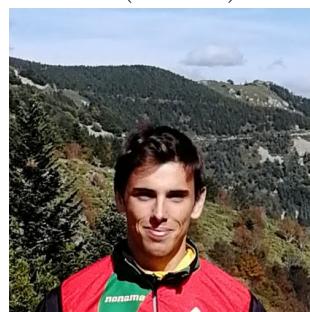
Beatriz Rodrigues
(a93230)



Francisco Neves
(a93202)



Henrique Costa
(a93325)



José Pedro
(a93163)

Índice

1	Introdução	3
2	Estrutura de conhecimento	4
2.1	Transporte Mais Ecológico	4
2.2	Encomenda	4
2.3	Estafeta	5
2.4	Cliente	5
2.5	Penalizado	5
2.6	Preço da Encomenda	5
2.7	Transporte	5
3	Funcionalidades	6
3.1	Consultas	6
3.1.1	Query 1	6
3.1.2	Query 2	6
3.1.3	Query 3	7
3.1.4	Query 4	7
3.1.5	Query 5	7
3.1.6	Query 6	7
3.1.7	Query 7	7
3.1.8	Query 8	7
3.1.9	Query 9	8
3.1.10	Query 10	8
3.2	Penalização	8
3.3	Inserção de conhecimento	9
4	Recursos extras	10
4.1	Invariante s	10
4.1.1	Estafeta	10
4.1.2	Cliente	10
4.1.3	Encomenda	11
4.1.4	Penalização	11
4.2	Consulta de ficheiros	11
4.3	Interface Gráfica	12
5	Conclusão	14

1. Introdução

Este trabalho foi realizado no âmbito da unidade curricular de Inteligência Artificial, do terceiro ano, da Licenciatura em Engenharia Informática da Universidade do Minho, e teve como objectivo a implementação de uma base de conhecimento para a empresa fictícia *Green Distributions*. Neste trabalho foram implementados mecanismos de consulta e inserção de conhecimento, de modo a ser possível a gestão de um sistema na área de logística de distribuição de encomendas.

2. Estrutura de conhecimento

De forma a estruturar o conhecimento necessário para ser possível responder às possíveis queries e funcionalidades do programa, foi decidido dividi-lo em diversos factos, sendo que, cada um destes seria responsável por uma parte específica e clara do programa.

Desta maneira, serão enumerados os factos que consideramos necessários implementar.

2.1 Transporte Mais Ecológico

Indica o veículo que se apresenta como sendo o mais ecológico da *Green Distribution*. Se futuramente for indicado que o veículo mais ecológico mudou, apenas se terá de alterar este facto para as queries acerca do veículo mais ecológico continuarem a apresentar uma execução correta.

Este facto tem aridade 1 e é dado da seguinte forma:

tMaisEcologico(*bicicleta*).

Sendo que, neste caso, indica que o veículo mais ecológico é a bicicleta.

2.2 Encomenda

Define como é indicada uma encomenda no programa. Esta deverá possuir diversos parâmetros, sendo que, para o caso de já ter sido entregue terá uns e, para o caso de apenas ter sido criada (e possivelmente não entregue) terá outros. Desta forma, foi decidido que uma encomenda criada teria aridade 10 e que uma encomenda entregue teria também um facto de aridade 3. Os campos de datas possuem a sintaxe: dia/mês/ano/hora de forma a permitir a precisão ao nível das "horas" relativa a tempos de entregas.

A representação deste facto para o caso de uma encomenda criada é dada da seguinte forma:

encomenda(*código encomenda, tempo máximo entrega, código cliente, código estafeta, peso, volume, veículo, preço base, data criação, zona entrega*).

Por outro lado, para o caso de uma encomenda entregue, a representação é dada da seguinte forma:

encomenda(*código encomenda, data entrega, classificação*).

2.3 Estafeta

Define como é indicado um estafeta, associando-se um código ao seu nome próprio, permitindo que, desta forma, existam vários estafetas com o mesmo nome próprio, no entanto, o seu código será único visto ser o seu identificador.

Este facto tem aridade 2 e é dado da seguinte forma:

estafeta(*código estafeta, nome estafeta*).

2.4 Cliente

Define como é indicado um cliente, associando-se um código ao seu nome próprio, permitindo que, desta forma, existam vários clientes com o mesmo nome próprio, no entanto, o seu código será único visto ser o seu identificador.

Este facto tem aridade 2 e é dado da seguinte forma:

cliente(*código cliente, nome cliente*).

2.5 Penalizado

Define uma penalização a um estafeta, de forma a poder impôr-lhe restrições caso este não cumpra a data de entrega máxima estipulada. A penalização consiste na proibição da realização de entregas num determinado intervalo de tempo por parte de um estafeta.

Este facto tem aridade 3 e é dado da seguinte forma:

penalizado(*código estafeta,data de inicio,data de fim*).

2.6 Preço da Encomenda

Define o aumento do preço de uma encomenda, tendo por base o seu preço base. Este aumento, depende do veículo utilizado e do tempo máximo para a entrega da encomenda.

Este facto tem aridade 4 e é dado da seguinte forma:

precoEncomenda(*preço base,tempo máximo entrega,veículo,preço final*).

2.7 Transporte

Define os tipos de transporte possíveis associando-lhe o peso máximo que estes poderão transportar e a sua velocidade média.

Este facto tem aridade 3 e é dado da seguinte forma:

transporte(*nome veículo,peso máximo,velocidade média*).

3. Funcionalidades

Neste capítulo serão enumeradas as diversas funcionalidades do trabalho. Para experimentá-las basta executar no terminal:

```
swipl queries.pl
```

As consultas podem ser realizadas diretamente pela sua invocação ou através do termo **menu/0**, que disponibiliza todas as funcionalidades enumeradas. Também é possível a utilização de uma interface gráfica que será descrita no capítulo de **Recursos Extras**.

3.1 Consultas

3.1.1 Query 1

Para identificar o estafeta que utilizou mais vezes um meio de transporte mais ecológico, foi decidido fazer proveito do termo *tMaisEcologico/1*. Assim, esta consulta identificará o(s) estafeta(s) que fizeram mais entregas utilizando o transporte indicado em **tMaisEcologico**.

O procedimento usado, inicialmente, filtra todos os códigos dos estafetas que estão referidos em encomendas registadas como entregues que utilizaram o transporte mais ecológico. Este filtro foi executado com *findall/3*. De seguida, foi identificado quantas vezes cada estafeta surgia na lista anteriormente filtrada. Com este passo, foi possível saber quantas vezes cada estafeta utilizou o meio de transporte mais ecológico. Restou então encontrar aquele que mais o utilizou. Porém, foi decidido que, caso existisse mais que um estafeta na solução, o programa deveria identificar cada um deles. Por esta razão foi utilizado *maxOcor/3* para, primeiramente, identificar a maior ocorrência de utilização do transporte mais ecológico e, em seguida, *getEstafetasMax/3* de forma a identificar os estafetas com o número de ocorrências encontrado no passo anterior.

Para ser possível identificar na resposta os códigos dos estafetas e os seus nomes, foi utilizado *getEstafetasNome/2* para a recolha dos nomes.

3.1.2 Query 2

De modo a identificar os estafetas que entregaram encomendas a um determinado cliente, filtrou-se, inicialmente, todos os códigos de estafetas que estavam registados em encomendas entregues para o cliente em questão. Como poderia existir estafetas que fizeram entregas ao mesmo cliente mais que uma vez, em seguida, foram removidos os elementos repetidos da lista anteriormente obtida.

Pela mesma razão da *Query 1*, recorreu-se à utilização de *getEstafetasNome/1* apenas para a recolha dos nomes dos estafetas.

3.1.3 Query 3

A *Query 3* funciona de maneira muito similar à *Query 2*, diferindo apenas que, desta vez, filtraram-se os códigos dos clientes ao invés dos estafetas. A remoção dos elementos repetidos surge pela mesma razão apontada na *Query* anterior.

3.1.4 Query 4

A *Query 4* permite a obtenção do preço total faturado num determinado dia de entregas.

Para tal filtraram-se todas as encomendas entregues no dia recebido como argumento, resultando numa lista de códigos de encomenda. De seguida, foi criada uma lista com os preços base de cada encomenda filtrada. Por fim, foi calculado o somatório dessa lista.

3.1.5 Query 5

A *Query 5* tinha como objetivo determinar o número de zonas com maior volume de encomendas. Decidiu-se que se iria permitir ao utilizador indicar quantos resultados queria receber. Assim, utilizou-se *findall* para recolher uma lista com todas as ocorrências das zonas. De seguida, converteu-se a lista anterior numa lista de pares (*Zona, Número de ocorrências*), em que o número de ocorrências corresponde ao número de entregas realizado nessa zona. Por fim, ordenou-se a lista de pares por ordem decrescente do número de ocorrências e filtra-se os primeiros *N* elementos, em que *N* é o input do utilizador, referido anteriormente.

3.1.6 Query 6

A *Query 6* permite o cálculo da classificação média de um estafeta. Para cumprir este requisito, recolheram-se para uma lista todas as classificações dadas ao estafeta pretendido. Para o cálculo da média são necessárias a soma total das avaliações e o número de avaliações. Para o primeiro, utilizou-se *foldl* de forma a acumular numa variável o valor da soma. Para o segundo, bastou calcular o comprimento da lista. De seguida, desde que o número de avaliações recebidas seja superior a 0, basta dividir o valor total pelo número de avaliações.

Para ser possível identificar na resposta os códigos dos estafetas e os seus nomes, foi utilizado *getEstafetasNome/2* para a recolha dos nomes.

3.1.7 Query 7

Nesta query, é calculado o número de entregas totais feitas por cada transporte num intervalo de tempo. Para isto, recolheu-se para uma lista o transporte utilizado em cada encomenda que pertença ao intervalo de tempo indicado. De seguida, formou-se uma nova lista de pares (*Transporte, Número de ocorrências*), em que o número de ocorrências corresponde ao número de utilizações do transporte correspondente.

3.1.8 Query 8

O objetivo desta query é o cálculo do número total de entregas feitas por cada estafeta num dado intervalo de tempo. Para isto, recolheu-se para uma lista todos

os códigos dos estafetas que realizaram entregas num intervalo de tempo indicado. De seguida, formou-se uma nova lista de pares (*Código do estafeta, Número de ocorrências*), em que o número de ocorrências corresponde ao número de entregas que o estafeta identificado pelo código realizou nesse período. Por fim, por questões de legibilidade durante a impressão no ecrã, foi decidida a criação de uma nova lista semelhante, na qual em vez dos estafetas estarem identificados pelo código passam a estar identificados pelo seu nome.

3.1.9 Query 9

A *Query 9* tem como objetivo calcular o número de encomendas entregues e não entregues, com base num intervalo de tempo. No entanto a definição de uma encomenda não entregue ou entregue pode ter várias visões. A abordagem escolhida foi a de considerar apenas as encomendas criadas entre o intervalo de tempo dado, e, para verificar se foram entregues verificou-se se houve alguma entrada de encomendas entregues nesse intervalo. Todas as outras encomendas foram consideradas como não entregues. Por exemplo poderá haver uma encomenda criada no intervalo dado e haver uma encomenda entregue para essa mesma encomenda, no entanto, se a data de entrega não pertencer ao intervalo é considerada como não entregue.

Para tal, encontraram-se todas as encomendas criadas que pertenciam ao intervalo dado, resultando numa lista com códigos de encomendas. De seguida encontraram-se as encomendas entregues dentro do intervalo dado e que pertenciam à lista calculada anteriormente. Esta lista é também uma lista com códigos de encomendas que foram entregues. Por fim calculou-se o tamanho da segunda lista e a diferença entre o tamanho da primeira lista e o tamanho da segunda lista, resultando no número de encomendas entregues e não entregues, respetivamente.

3.1.10 Query 10

A *Query 10* tem como objetivo calcular para cada estafeta o peso total transportado/entregue num determinado dia.

Para tal, recolheram-se todos as encomendas entregues na data dada como argumento. O resultado é uma lista de tuplos (*Código de Encomenda, Peso*) relativas à encomenda entregue. Esta lista tem várias entradas com o mesmo código de estafetas. De seguida foi necessário juntar os pesos de cada estafeta. Resultando numa lista com o mesmo tipo de tuplo mas com o *pesoTotal* transportado por cada técnico.

3.2 Penalização

A penalização de um estafeta é representada através do seguinte termo:

$$\text{penalizado}(\text{CodEstafeta}, \text{DataInicio}, \text{DataFim})$$

A funcionalidade desta representação de conhecimento encontra-se no momento de inserção de uma **encomenda**. Isto porque, não é permitido a atribuição de encomendas a estafetas penalizados, ou seja, durante o intervalo de tempo entre **DataInicio** e **DataFim**, o estafeta com o código indicado estará isento de trabalho. A forma decidida para a implementação deste mecanismo será abordada no capítulo de Invariantes. Numa fase inicial do trabalho optou-se que a inserção das penalizações seria feita manualmente, a partir da utilização da regra **evolucao**, que permite a expansão sólida e consistente do conhecimento.

3.3 Inserção de conhecimento

Para além das consultas mencionadas, também é possível inserir novos conhecimentos no sistema. Fazendo uso do termo *evolucao/1*, consegue-se expandir a base de conhecimento de uma forma sólida e consistente. Com isto, torna-se possível a adição de novas encomendas, clientes, estafetas e penalizações. A forma decidida para a implementação desta funcionalidade será abordada no capítulo de **Invariante**s.

4. Recursos extras

4.1 Invariantes

Para além das funcionalidades descritas no enunciado, foram implementados mecanismos de expansão de conhecimento. Por conseguinte, foi necessário o recurso ao conceito de Invariantes. Ou seja, a inserção de conhecimento na base de conhecimento terá de respeitar algumas regras. O objetivo desta preocupação é garantir informação consistente e fidedigna na base de conhecimento. Para ser possível a passagem desta ideia para *Prolog*, utilizou-se um dos conceitos lecionados nas aulas práticas, definindo-se o seguinte operador:

```
: - op(900,xfy,':::')
```

A partir deste operador, existiam agora condições para a criação da regra de evolução de conhecimento:

```
evolucao(Termo) :-  
    findall( Invariante,+Termo:::Invariante,Lista),insercao(Termo),teste(Lista)
```

Assim sendo, a inserção dos termos dependerá destes passarem em todos os testes de validação a si destinados, isto é, a respeitarem os Invariantes. De seguida serão enumerados os Invariantes utilizados.

4.1.1 Estafeta

A inserção de um novo **Estafeta** só é permitida no caso de não existir nenhum estafeta previamente registado com o mesmo código. Ou seja, foi utilizado o conceito de identificação única de estafetas (mencionada no capítulo 2) de forma a excluir possíveis ambiguidades de registo.

4.1.2 Cliente

De modo similar aos estafetas, a inserção de um **Cliente** depende da não existência prévia de um outro cliente registado com o mesmo código de identificação.

4.1.3 Encomenda

Os invariantes da Encomenda garantem a consistência dos seguintes dados:

- Não é permitida a inserção de um registo de criação de encomenda com um código de encomenda que já esteja registado no sistema;
- Não é permitida a inserção de um registo de criação de encomenda atribuída a um **Estafeta** que não esteja registado no sistema;
- Não é permitida a inserção de um registo de criação de encomenda atribuída a um **Cliente** que não esteja registado no sistema;
- Não é permitida a inserção de um registo de criação de encomenda atribuída a um **Estafeta** que esteja penalizado;
- Não é permitida a inserção de um registo de criação de encomenda associada a um meio de **Transporte** inexistente no sistema, ou que não possua capacidade de levá-la (devido ao seu peso máximo de carregamento);
- Não é permitida a inserção de um registo de entrega de encomenda se essa mesma encomenda não tiver um registo de criação;
- Não é permitida a inserção de um registo de entrega de encomenda mais de uma vez;
- Não é permitida a inserção de um registo de entrega de com data anterior ao registo de criação.

4.1.4 Penalização

Os invariantes das penalizações garantem que:

- Não é permitida a inserção de uma penalização a um estafeta que não esteja registado no sistema;
- Não é permitida a inserção de uma penalização com intervalos de tempos negativos;

4.2 Consulta de ficheiros

Como crédito extra para o projeto, foi também implementada a funcionalidade de leitura de conhecimento a partir de um ficheiro *.txt*. A função *carregaFicheiro* recebe o nome do ficheiro a ler e abre-o com permissões de leitura, gerando uma *stream*, que permite à função *lerFicheiro* efetuar a leitura do ficheiro linha a linha, guardando-as numa lista de linhas. Como cada linha corresponde a um termo , faz-se a *evolucao* deste termo para poder ir construindo a base de conhecimento de forma consistente (para situações em que os ficheiros venham com informações inconsistentes, como por exemplo códigos de encomendas repetidas, entre outros).

Para efeitos de teste e como forma de enriquecer o conhecimento, foi criado um *script* em *Python* que é capaz de gerar ficheiros *.txt* que contêm factos gerados de forma parcialmente aleatória. Desta forma, é possível escolher o tipo (estafeta, cliente ou encomenda) e a quantidade de factos pretendidos.

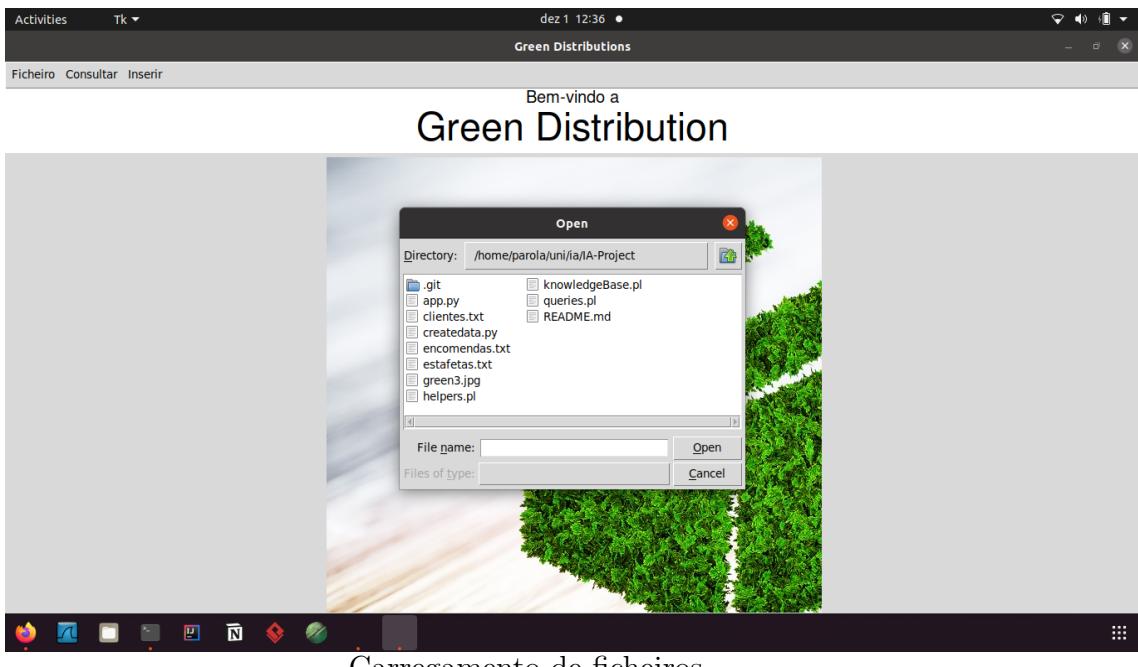
4.3 Interface Gráfica

Para além do **menu** disponibilizado na linha de comando, foi também criada uma interface gráfica com recurso à linguagem de programação **Python** de forma a obter uma melhor visualização das funcionalidades da aplicação. Com isso, o código implementado em **Prolog** acaba por funcionar como *Back-End* do sistema. Tal mecanismo é possível graça a biblioteca *pyswip* que permite a consulta de bases de conhecimento construídas em Prolog, assim como a invocação de *Queries*.

Para a interface, é preciso como dependências apenas o *pyswip* e a biblioteca *pillow*. Para executá-la basta correr o comando:

```
python3 app.py
```

Já na aplicação, é possível a consulta de ficheiros, realização das consultas propostas pelo enunciado e inserção de conhecimento. Todas as funcionalidades extras estão identificadas como *EXTRA*. Segue então algumas imagens da interface implementada:



Carregamento de ficheiros

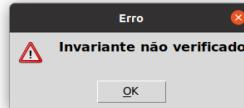
A utilização dos invariantes é notável quando se tenta inserir conhecimento, mas este de alguma forma não é consistente com a base de conhecimento, conseguindo assim aproveitar o valor **false** do predicado *evolucao* para a exibição de mensagens de erro ao utilizador:

Activities Tk • dez 1 12:35 •

Green Distributions

Ficheiro Consultar Inserir

Código Cliente
1
Nome Cliente
eren
<input type="button" value="Adicionar"/>



Invariante não verificado

Para além das consultas do enunciado é também possível visualizar os registo de estafetas, clientes, encomendas entregues e não entregues:

Activities TkDialog • dez 1 12:34 •

Green Distributions

Ficheiro Consultar Inserir

Encomendas Criadas

Código Encomenda	Tempo Max	Código Cliente	Código Estafeta	Peso	Volume	Transporte	Preço Base	Data Criação	Zona de Entrega
1	40	4	7	34	23	moto	19.95	21/11/2021 às 0 h	gothamCity
2	20	2	2	34	23	bicicleta	20.1	21/11/2021 às 0 h	centralCity
3	50	3	3	34	23	carro	23	1/6/2021 às 0 h	gothamCity
4	10	2	4	34	23	bicicleta	32	1/6/2021 às 0 h	centralCity
5	10	2	5	34	23	bicicleta	43	11/9/2021 às 0 h	centralCity
6	30	6	6	34	23	moto	10	23/2/2018 às 0 h	wonderland
7	40	2	4	34	23	bicicleta	2	12/2/2019 às 0 h	narnia
8	50	8	8	34	23	carro	2	21/11/2021 às 0 h	centralCity
9	60	2	2	34	23	bicicleta	28	24/12/2020 às 0 h	narnia
10	10	5	1	34	23	moto	40	26/12/2020 às 0 h	narnia
11	20	3	2	34	23	bicicleta	42	1/1/2021 às 0 h	centralCity
12	70	2	4	34	23	bicicleta	24.5	21/11/2021 às 0 h	narnia
0	841	54	90	4	118.79	bicicleta	28.76	9/2/2018 às 9 h	springfield
13	317	11	24	4	64.49	bicicleta	26.48	3/11/2012 às 10 h	centralCity
14	758	91	33	3	91.25	bicicleta	2.67	26/8/2008 às 7 h	neverland
15	894	76	25	13	38.31	moto	133.99	10/1/2019 às 3 h	centralCity
16	113	57	84	3	37.12	bicicleta	41.61	10/7/2021 às 23 h	bedrock
17	359	34	73	2	144.14	bicicleta	103.88	5/10/2014 às 14 h	springfield
18	180	77	15	8	84.88	moto	39.88	11/10/2001 às 19 h	asgard
19	437	68	58	4	123.49	bicicleta	129.81	21/10/2014 às 22 h	mordor
20	327	98	3	14	16.21	moto	23.69	9/3/2004 às 15 h	wonderland
21	86	84	38	70	138.4	carro	124.87	8/12/2009 às 19 h	tatooine
22	158	37	23	11	86.7	moto	7.87	11/3/2005 às 20 h	wonderland
23	661	78	66	15	75.52	carro	103.93	22/4/2004 às 12 h	westEgg
24	7	78	15	10	43.18	carro	26.23	22/3/2015 às 12 h	westEgg
25	338	81	21	14	89.43	moto	50.47	7/9/2000 às 15 h	tatooine
26	335	95	25	18	51.84	moto	71.17	6/6/2005 às 23 h	gravityFalls
27	853	35	11	4	139.88	bicicleta	105.27	30/8/2016 às 22 h	rivendell
28	76	69	89	15	14.38	moto	10.98	27/1/2003 às 2 h	centralCity

Consulta das encomendas registadas no sistema

5. Conclusão

Neste trabalho foram implementadas todas as funcionalidades descritas no enunciado, sendo que, também foram acrescentadas diversas novas utilidades extraordinárias. A sua realização revelou-se importante para a compreensão das facilidades que o *Prolog* dispõe no âmbito do manuseamento de dados. Isto porque, tal linguagem fornece um nível de abstração suficientemente grande, permitindo, assim, uma preocupação, quase exclusiva, com o raciocínio por trás de cada implementação.