**Universidade do Minho**

Escola de Engenharia

Licenciatura em Engenharia Informática

## Unidade Curricular de Sistemas de Representação de Conhecimento e Raciocínio

Ano Letivo de 2014/2015

3º Ano 2º Semestre

# Trabalho de grupo – 2º Exercício

**Grupo nº 5**

**Daniel Caldas a67691**

**José Cortez a67716**

**Susana Mendes a63464**

**Xavier Rodrigues a67676**

Maio, 2015

# Trabalho de grupo – 1º Exercício

**Grupo nº 5**
**Daniel Caldas a67691**
**José Cortez a67716**
**Susana Mendes a63464**
**Xavier Rodrigues a67676**

Maio, 2015

# Resumo

Com o presente relatório pretendemos expor sucintamente as decisões tomadas ao longo da criação do nosso sistema de representação de conhecimento e raciocínio. Iremos apresentar o caso de estudo, seguido de uma pequena análise do problema. Posteriormente, vamos expor a solução desenvolvida com o detalhe necessário. Finalmente, será feita uma análise dos resultados e apresentaremos as conclusões deste segundo exercício.

**Área de Aplicação:** Inteligência Artificial.

# Índice

# 1 Introdução

## 1.1. Apresentação do Caso de Estudo

O sistema de representação de conhecimento deverá ser capaz de representar informação incompleta, imprecisa, interdita, entre outras; utilizando uma extensão à programação em lógica, usando a linguagem de programação em lógica PROLOG. O sistema deve poder manipular informação relativa a automóveis - matrícula, marca, modelo, cor, estado, nome do proprietário, ano de fabrico e outro tipo de conhecimento que seja relevante (funcionalidades extra).

## 1.2. Motivação e Objetivos

O objetivo será então desenvolver um sistema de representação de conhecimento e raciocínio de modo a caracterizar um universo de comércio automóvel, considerando os seguintes atributos referidos na subsecção anterior.

## 1.3. Estrutura do Relatório

O presente relatório encontra-se estruturado conforme os seguintes pontos:

- Introdução;
- Preliminares:
  - Estudo prévio;
  - Análise do problema;
- Apresentação da solução
  - Apresentação dos casos de estudo
  - Predicados implementados
  - Predicados extra
  - Invariantes
- Aplicação Stand Vitória

        - Aplicação Stand Vitória
        - Arquitetura da aplicação
        - Aspeto final da aplicação
- Solução dos casos de estudo
- Exemplo de *Use Case* com banda desenhada
- Pontos críticos;
- Análise dos resultados;
- Conclusões;

# 2. Preliminares

## 2.1 Estudo prévio

Havendo já, numa fase prévia, uma consolidação dos conceitos de programação em lógica, cabe agora fazer uma extensão desses mesmos conceitos de modo a poder ser feita uma representação mais abrangente do conhecimento. Para tal, com a resolução de problemas como o proposto, podemos cimentar a problemática da extensão à programação em lógica.

Na programação em lógica estendida é adotado o pressuposto do Mundo Aberto. De maneira a atingir tal objetivo, foi utilizada a programação em lógica estendida que, para além do verdadeiro e falso, permite também representar o desconhecido. As mudanças também foram sentidos na área do Conhecimento que, devido às alterações dos valores lógicos possíveis, passou a representar Conhecimento Imperfeito – nulo, incerto ou impreciso.

O Trabalho Prático desenvolvido tem como pilares tais fundamentos.

## 2.2 Análise do problema

O trabalho prático, desenvolvido em programação em lógica estendida, deverá tratar as enunciadas problemáticas:

- Representação de conhecimento positivo e de conhecimento negativo;
- Representação de casos de conhecimento imperfeito, utilizando os valores nulos de todos os tipos estudados;
- Manipulação de invariantes que designem restrições à inserção e remoção de conhecimento;
- Criação de procedimentos capazes de tratar a problemática da evolução de conhecimento;

- Desenvolvimento do sistema de interferência capaz de implementar os mecanismos de raciocínio inerentes a estes sistemas;
- Interagir com o sistema recorrendo à biblioteca JASPER (em Java);

# 3. Apresentação da solução

## 3.1 Apresentação dos casos de estudo

O Stand Vitória é um estabelecimento que prima pela sua qualidade e requinte. É um dos locais mais prestigiados e requisitados no Norte de Portugal. Sinónimo de boa gestão, qualquer nortenho que necessite dum automóvel procura, em primeiro lugar, este espaço. Uma das regras peculiares estabelecidas pela gerência do Stand incide na proibição de transações de automóveis de cor vermelha.

Após uma semana frenética na qual se venderam imensos carros, foi feita uma "contagem" e encontram-se os seguintes automóveis no stand:

| Proprietário | Automóvel | Matrícula | Cor | Data de Fabrico | Preço (€) | Estado |
|---|---|---|---|---|---|---|
| Maria | VW Golf | 03-02-FF | Preto | 20/08/14 | 25 000 | Bom |
| Josué | Nissan GTR | 52-42-AA | Laranja | 05/09/09 | 127 350 | Bom |
| Fernando | Renault 5 | 01-33-GH | Cinzento | 06/07/80 | 4500 | Mau |
| Cristiano | Lancia Delta Integrale | 15-64-FN | Branco | 01/02/80 | 70 000 | Médio |
| Alice | Fiat 500 Abarth | 15-MM-22 | Branco | 17/08/13 | 55 500 | Médio |
| Artur | Pagani Zonda | 07-88-KK | Cinzento | 12/10/06 | 1 500 500 | Bom |
| Joana | Peugeot 205 | 04-64-EF | Preto | 11/11/99 | 2000 | Razoável |
| Filipe | Mercedes C220 | 99-AK-47 | Azul | 11/11/14 | 25 000 | Bom |

Na última semana, o nosso grupo de trabalho observou o Stand, tomando nota de algumas situações dignas de registo:

## Caso 1

O Sr. João chegou ao Stand Vitória com um Toyota Corolla cuja matrícula e data de fabrico eram, respetivamente, “09-87-CF” e 14/09/1993. Dado o Sr. João adorar levar o carro em viagens longas no fim-de-semana, o carro já percorreu mais de 500 000 km. Por conseguinte, apresenta-se desgastado, pelo que o dono não sabe se a cor é azul ou cinza.

## Caso 2

O pai da Dona Rosa possuía um Datsun de cor verde. Infelizmente, o pai da Dona Rosa faleceu, pelo que esta ficou com o seu carro. Uma vez que o carro a fazia lembrar do seu pai, a Dona Rosa decidiu vendê-lo, recorrendo ao Stand Vitória para atingir tal efeito. No entanto, verificaram-se dois problemas:

- Como o automóvel sofreu muitos acidentes e estava extremamente desgastado (tendo inúmeras peças de outros veículos) não se conseguia averiguar o modelo. Foi chamado um especialista e a conclusão deste foi que seria impossível determinar o modelo do automóvel.
- Também como consequência do estado do automóvel, a zona na qual estava inscrita a data de fabrico na matrícula era ilegível. A Dona Rosa não conseguia encontrar o livrete, mas, segundo a própria, decerto que este se encontrava em sua casa.

## Caso 3

Sr. César tinha avistado, de relance, um Lamborghini no Stand Vitória. Ele não sabia se era um Gallardo ou um Huracán mas tinha certeza de que não era um Aventador. Mais tarde, ligou ao dono do Stand para obter mais informações sobre o carro. Este, persuadindo-o a vir ao Stand, forneceu-lhe todas as informações acerca do automóvel - cor branca, matrícula ’08-07-XP’, data de fabrico 30/6/2008, cujo preço era 345 000€ - expecto o modelo.

### Caso 4

O Júlio Mendes avistou um Audi R8, em bom estado, no Stand Vitória. Estava tão atento a todos os pormenores – reparou na matrícula (’22-23-FX’), data de fabrico (5/6/2009), no preço exorbitante de 230 000 – mas, quando chegou a casa, já não se lembrava da cor.

### Caso 5

Será apresentado posteriormente num formato diferente dos restantes casos.

## 3.2 Predicados Implementados

Foram implementados predicados para caracterizar as características de um automóvel (matricula, marca, modelo, cor, estado, nome do proprietário, ano de fabrico e preço)

Para que se possam atribuir determinadas características a uma dado automóvel, foram definidos os seguintes predicados:

**automovel( M,M,M ).**: Em que a primeira variável M representa a matrícula, a segunda a marca e a última o modelo do automóvel.
**cor( M,C ).**: Em que M é a matrícula do automóvel e C a cor.
**estado( M,E ).**: Em que M é a matrícula do automóvel e E o seu estado.
**proprietario( M,NP ).**: Em que M é a matrícula do automóvel e NP é o nome do seu proprietário.
**dataFabrico( M,D,MN,A ).**: Em que M é a matrícula do automóvel, D, MM e A dizem respeito ao dia, mês e ano de fabrico do automóvel.
**preco( M,P ).**: Em que M é a matrícula do automóvel e P o seu preço, este predicado é já um dado extra que o grupo decidiu implementar.

```
automovel('04-64-EF',peugeot,205).
automovel('09-87-CF',toyota,corolla).
```

Figura 1: Exemplo do predicado automóvel

```
cor('04-64-EF',preto).
cor('67-34-PT',verde).
cor('08-07-XP',branco).
```

Figura 2: Exemplo do predicado cor

```
estado('04-64-EF',razoavel).
estado('09-87-CF',mal).
```

Figura 3: Exemplo do predicado estado

```
proprietario('04-64-EF',fernanda).
proprietario('09-87-CF',joao).
proprietario('67-34-PT',rosa).
proprietario('08-07-XP',cesar).
```

Figura 4: Exemplo do predicado proprietario

```
dataFabrico('99-AK-47',11,11,1999).
```

Figura 5: Exemplo do predicado dataFabrico

```
preco('09-87-CF',2000).
preco('67-34-PT',2000).
preco('08-07-XP',345000).
```

Figura 6: Exemplo do predicado proprietário

## 3.3 Predicados Extra

### Demo2

Definimos um predicado baseado no predicado demo que utilizamos nas aulas da UC como uma ferramenta de extensão às funcionalidades do PROLOG por forma a conseguir aplicar a teoria da programação em lógica estendida. Neste **demo2** é feita a inferência lógica de valores de conhecimento (sendo eles verdadeiro, falso, desconhecido). Construímos uma tabela dos resultados de inferências lógicas de acordo com o nosso modelo de negócio, ou seja, a junção de valores produzirá uma única resposta que fará sentido num contexto real. **Poderemos ver um exemplo prático deste demo2 na secção 6.**

```
% Tabela de inferência de valores lógicos de conhecimento (explicados no relatorio)
% baseada na análise do conhecimento representado.

% VV->V
% VD->D
% VF->F
% DF->F
% DD->D
% FF->F
% DV->D
% FV->F
% FD->F
%-------------------------------- - - - - - - - - - -  -  -  -  -  -
% Extensao do meta-predicado demo2: Questao1, Questao2, Resposta -> {V,F}
demo2( Q1,Q2,verdadeiro ):- demo(Q1,R1), demo(Q2,R2), R1==verdadeiro, R2==verdadeiro.
demo2( Q1,Q2,desconhecido ):- demo(Q1,R1), demo(Q2,R2), R1==verdadeiro, R2==desconhecido.
demo2( Q1,Q2,desconhecido ):- demo(Q1,R1), demo(Q2,R2), R1==desconhecido, R2==desconhecido.
demo2( Q1,Q2,desconhecido ):- demo(Q1,R1), demo(Q2,R2), R1==desconhecido, R2==verdadeiro.
demo2( Q1,Q2,falso ).
```

Figura 7 - Extensão do predicado demo2

### DemoL

Num seguimento de extensão das funcionalidades do nosso stand, definimos também um predicado **demoL** que nos permite fazer **N** perguntas à base numa só query.

```
%-------------------------------- - - - - - - - - - -  -  -  -  -  -
% Extensao do meta-predicado demoL: Lista de Questoes, Lista de Respostas -> {V,F}
demoL(  [],[] ).
demoL( [Q|Q5],[X|L] ) :- demo(Q,X), demoL(Q5,L).
```

Figura 8 - Extensão do predicado demoL

## 3.4 Invariantes

As regras definidas para as inserções e remoções foram as seguintes:

- Só se pode inserir dados sobre um automóvel se este já estiver na base de conhecimento;
- Não pode haver conhecimento repetido;
- Só se pode remover um automóvel se não existirem dados sobre o mesmo na base;

De modo a espelhar estas regras, foram definidos invariantes (tanto de inserção e remoção de conhecimento) para os seguintes predicados:

- preco;
- automóvel;
- dataFabrico;
- proprietário;
- estado;
- cor;

### Predicado preco

```
% -------- Invariante Preco ------------------
%%%% Insersão
%Só pode ser atribuído preço se o automóvel já existir na base
+preco( M,C ) :: (solucoes( (MARCAS,MODS),(automovel( M,MARCAS,MODS )), S),
                  comprimento( S,N ), N==1
                  ).

% Um automóvel só pode ter um preço
+preco( M,P ) :: (solucoes( (P2),(preco( M,P2 )),S ),
                  comprimento( S,N ), N == 1
                  ).

%%%%% Remoção
%Só se pode remover o preço dum automóvel se esse mesmo existir
-preco( M,P ) :: (solucoes((M1,M2),automovel( M,M1,M2 ),S ),
                              comprimento( S,N ), N == 1
                             ).
```

Figura 9 - Invariantes para inserção e remoção de preço..

## Predicado Cor

```
% -------- Invariante Cor -------------
%%% Inserção
%Um carro tem uma e uma só cor.
+cor( M,C ) :: (solucoes( (Cs),(cor( M,Cs )),S ),
                comprimento( S,N ), N == 1
                ).

%Só pode ser atribuída cor a carro se automovel existir na base.
+cor( M,C ) :: (solucoes( (MARCAS,MODS),(automovel( M,MARCAS,MODS )), S),
                comprimento( S,N ), N==1
                ).


%%% Remoção
%Só se pode remover uma cor dum automóvel se esse mesmo existir
-cor( M,C ) :: (solucoes((M1,M2),automovel( M,M1,M2 ),S ),
                            comprimento( S,N ), N == 1
                          ).
```

Figura 10 - Invariates relativos à inserção e remoção do predicado cor

## Predicado automovel

```
% ---------- Invariante Automovel --------------
%% Inserção
%Só se pode admitir uma e so uma matricula por cada extensao(automovel,cor,dataFabrico,proprietario,preco)
+automovel( M,MARCA,MODELO ) :: (solucoes( (M1,M2),automovel( M,M1,M2 ),S ),
                            comprimento( S,N ), N == 1
                          ).


%% Remoção
%Só se pode remover um automóvel que exista na base de conhecimento
-automovel( M,MARCA,MODELO ) :: (solucoes((M1,M2),automovel( M,M1,M2 ),S ),
                            comprimento( S,N ), N == 1 ).

-automovel( M,MARCA,MODELO ) :: (solucoes((C),cor( M,C),S ),
                            comprimento( S,N), N == 0 ).

-automovel( M,MARCA,MODELO ) :: (solucoes((P),preco( M,P),S ),
                            comprimento( S,N), N == 0 ).

-automovel( M,MARCA,MODELO ) :: (solucoes((DD,MM,AA),dataFabrico( M,DD,MM,AA),S ),
                            comprimento( S,N), N == 0 ).

-automovel( M,MARCA,MODELO ) :: (solucoes((E),estado( M,E),S ),
                            comprimento( S,N), N == 0 ).

-automovel( M,MARCA,MODELO ) :: (solucoes((P),proprietario( M,P),S ),
                            comprimento( S,N), N == 0 ).
```

Figura 11 - Invariantes relativos à inserção e remoção do predicado automóvel

## Predicado estado

```
% --------- Invariante Estado ----------------
%%%% Inserção
% Só pode ser atribuído um estado a um automóvel se este mesmo existir na base
+estado( M,E ) :: (solucoes( (MARCAS,MODS),(automovel( M,MARCAS,MODS )), S),
                   comprimento( S,N ), N==1
                   ).

% Um automóvel só pode ter um estado
+estado( M,E ) :: (solucoes( (E2),(estado( M,E2 )),S ),
                   comprimento( S,N ), N == 1
                   ).


%%%% Remoção
%Só se pode remover o estado dum automóvel se esse mesmo existir
-estado( M,E ) :: (solucoes((M1,M2),automovel( M,M1,M2 ),S ),
                              comprimento( S,N ), N == 1
                            ).
```

Figura 12 - Invariantes relatvos à inserção e remoção do predicado estado

## Predicado proprietario

```
% ----------- Invariante Proprietario ----------
%%%% Inserção
% Só pode ser atribuído um proprietário a um automóvel se este mesmo existir na base
+proprietario( M,P ) :: (solucoes( (MARCAS,MODS),(automovel( M,MARCAS,MODS )), S),
                        comprimento( S,N ), N==1
                        ).

% Um carro só pode ter um proprietário
+proprietario( M,P ) :: (solucoes( (P2),(proprietario( M,P2 )),S ),
                        comprimento( S,N ), N == 1
                        ).


%%%% Remoção
%Só se pode remover um proprietário dum automóvel se esse mesmo existir
-proprietario( M,P ) :: (solucoes((M1,M2),automovel( M,M1,M2 ),S ),
                                   comprimento( S,N ), N == 1
                                 ).
```

Figura 13 - Invariantes relativos á inserção e remoção do predicado proprietário

## Predicado dataFabrico

```
% --------- Invariante DataFabrico ----------
%Só pode ser atribuída data de fabrico se o automóvel já existir na base.
%%%% Inserção
+dataFabrico( M, D, MM, A ) :: (solucoes( (MARCA,MOD),(automovel( M,MARCA,MOD )),S ),
                                comprimento( S,N ), N == 1
                                ).

% Um carro só pode ter uma data de fabrico
+dataFabrico( M,D,MM,A ) :: (solucoes( (D1,MM1,A1),(dataFabrico( M,D1,MM1,A1 )),S ),
                             comprimento( S,N ), N == 1
                             ).

%%%% Remoção
%Só se pode remover a data de fabrico dum automóvel se esse mesmo existir
-dataFabrico( M,D,MM,A ) :: (solucoes((M1,M2),automovel( M,M1,M2 ),S ),
                                comprimento( S,N ), N == 1
                               ).
```

Figura 14 - Invariante relativo à inserção e remoção do predicado dataFabrico

# 4. Aplicação Stand Vitória

Foi desenvolvida em JAVA uma interface para interação com o SICStus PROLOG. Nesta pequena interface apenas incorporamos uma consola (*JTextArea*) que funciona como uma linha de comandos, fazendo o parse adequado das queries introduzidas e o também das respetivas respostas para o output da consola.

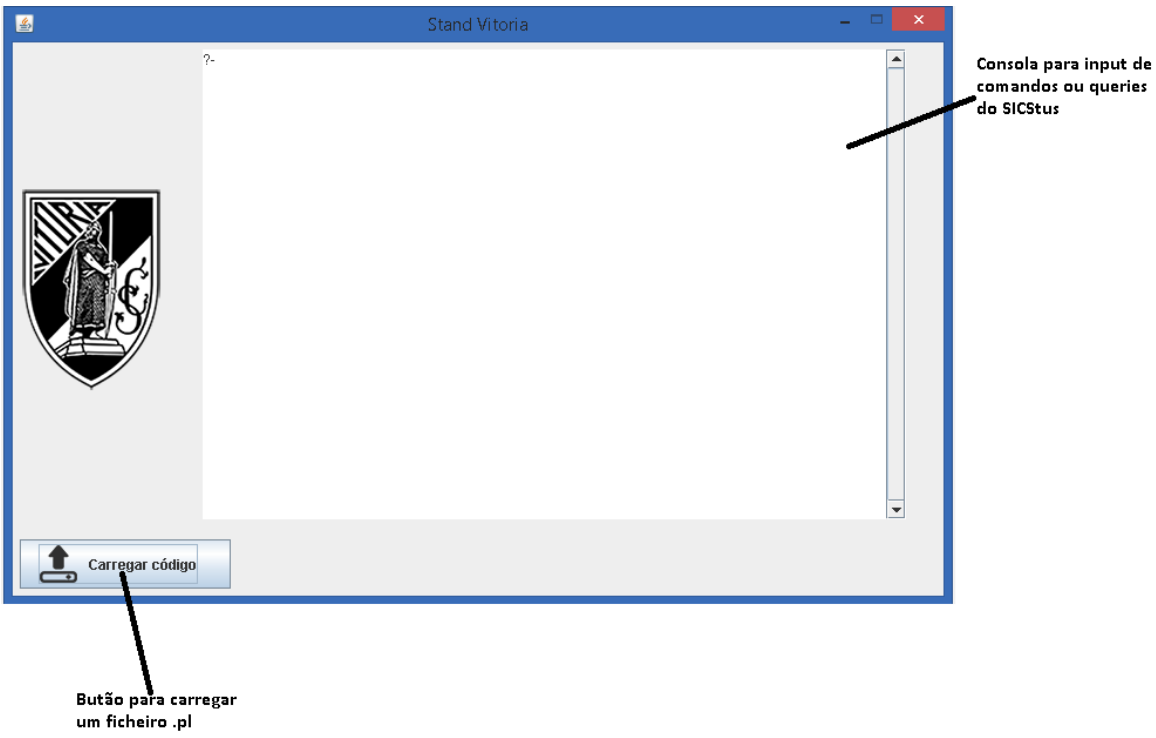



Figura 15 - Interface da aplicação.

Funcionalidades:

- Executar queries e obter respostas num formato legível e apresentável;
- Histórico de comandos (primir tecla para cima para aceder ao comando anterior);

**Carregar um ficheiro**

Para carregar um ficheiro escolha-se através de um *filechooser* o ficheiro que se deseja carregar e insira-se o comando load.

## 4.1 Arquitetura da aplicação

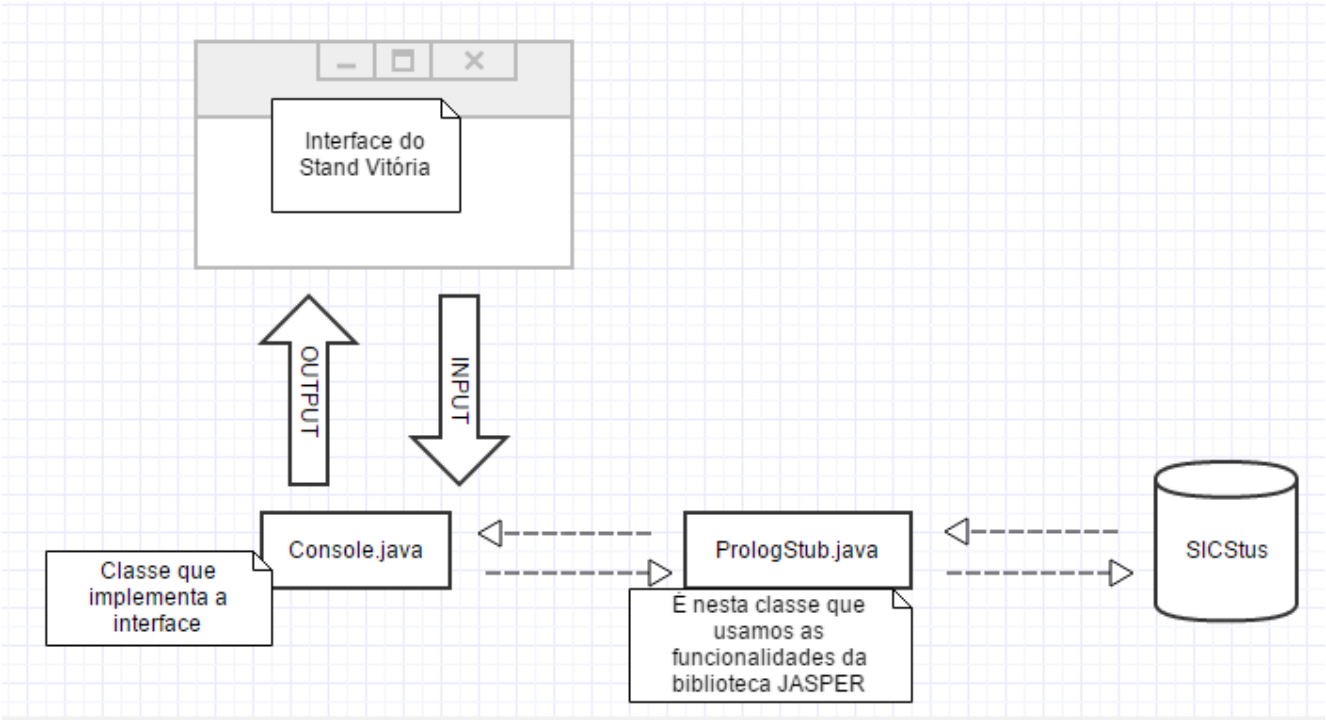


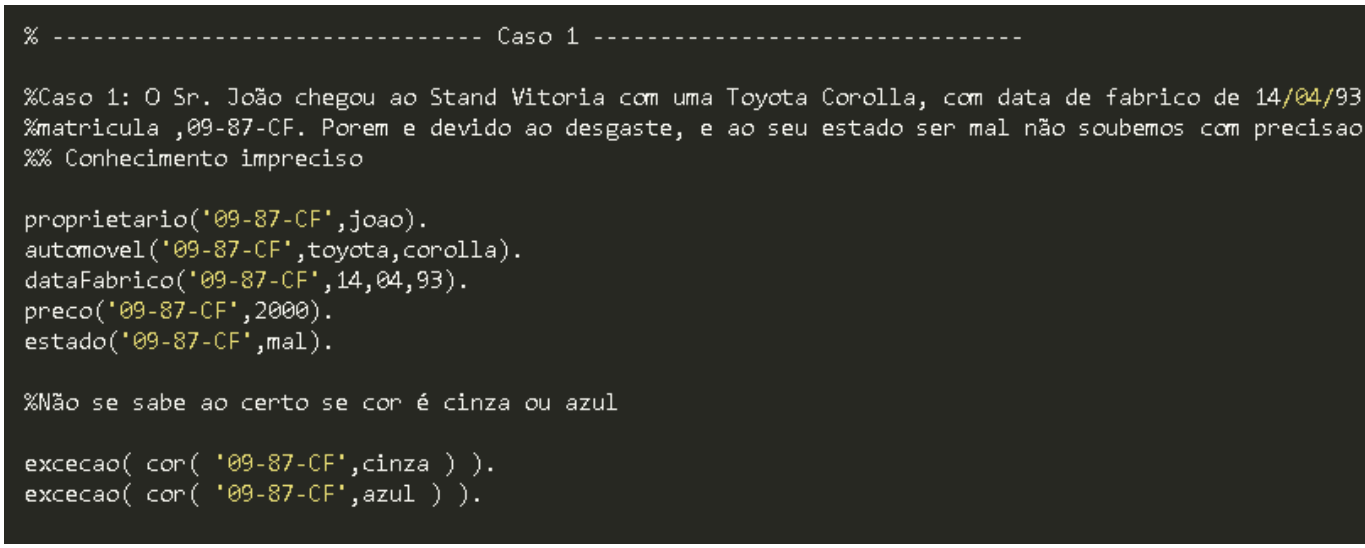

Figura 16 - Arquitetura da nossa aplicação.

No esquema da figura 10 podemos observar claramente as diversas componentes da nossa aplicação e o fluxo desde a interface até à base de conhecimento do SICStus. Basicamente, implementamos uma classe para manipular o input proveniente da interface e uma outra que faz chamadas remotas ao SICStus para obter resultados de queries.

# 5. Solução dos Casos de estudo

Nesta secção apresentaremos uma série de imagens que são as soluções implementadas para resolver estes problemas refletidos nos casos de estudo.

## 5.1 Caso 1



```
% ------------------------------ Caso 1 ------------------------------

%Caso 1: O Sr. João chegou ao Stand Vitoria com uma Toyota Corolla, com data de fabrico de 14/04/93
%matricula ,09-87-CF. Porem e devido ao desgaste, e ao seu estado ser mal não soubemos com precisao
%% Conhecimento impreciso

proprietario('09-87-CF',joao).
automovel('09-87-CF',toyota,corolla).
dataFabrico('09-87-CF',14,04,93).
preco('09-87-CF',2000).
estado('09-87-CF',mal).

%Não se sabe ao certo se cor é cinza ou azul

excecao( cor( '09-87-CF',cinza ) ).
excecao( cor( '09-87-CF',azul ) ).
```

Figura 17 - Solução do caso 1.

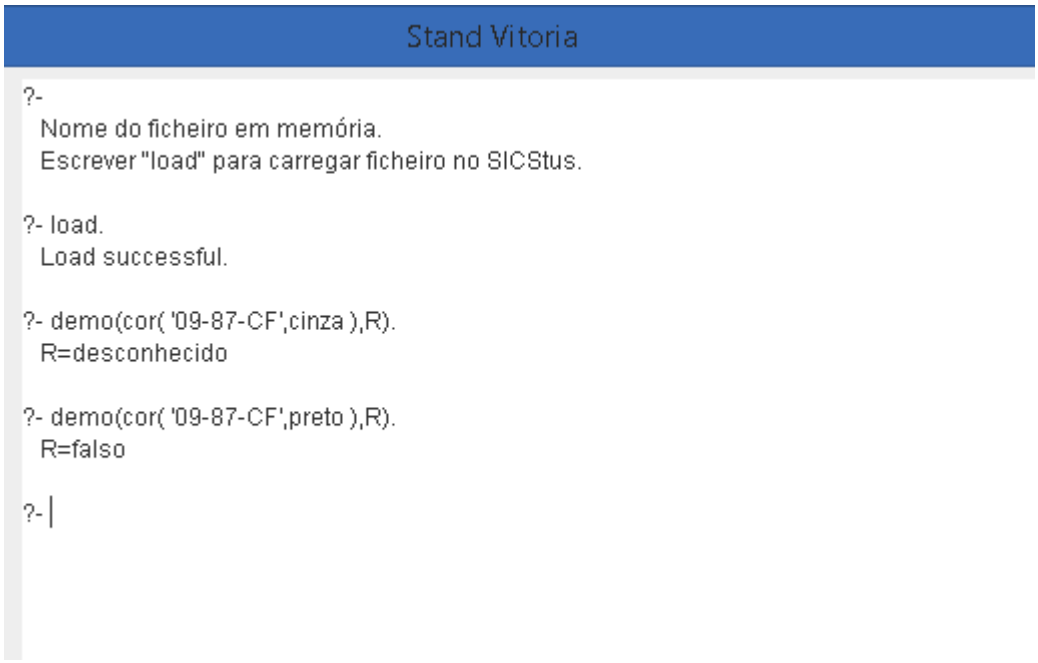



Figura 18 - Resultados do caso 1.

## 5.2 Caso 2

```
%Caso 2: A Dona Rosa chegou ao Stand Vitoria com o carro de seu pai, ja falecido. Um Datsun,de cor verde, mas devido ao estado ja na
%E por nao saber do livrete nao se consegui saber o data de fabrico.
%% Conhecimento nulo

automovel('67-34-PT',datsun,xptop).
excecao( automovel( M,M,M ) ) :-
    automovel( M,M,xptop ).
+automovel('67-34-PT',A,Q) :: (solucoes(X, (automovel('67-34-PT',datsun,X), nao(nulo(X))),S),
                    comprimento(S,N), N==0).

proprietario('67-34-PT',rosa).
cor('67-34-PT',verde).
preco('67-34-PT',2000).

dataFabrico('67-34-PT',ddd,mmm,aaa).
excecao( dataFabrico( M,D,M,A ) ) :-
    dataFabrico( M,ddd,mmm,aaa ).
```

Figura 19 - Solução do caso 2.

```
?-
?- evolucao( dataFabrico('67-34-PT',1,2,2000)).
  no.

?- demo(dataFabrico('67-34-PT',1,2,2000),R).
  R=falso

?-
```

Figura 20 - Resultados do caso 2.

## 5.3 Caso 3

```
%Caso 3: O Sr. Cesar viu, de relance,  um Lamborghini no Stand Vitória, porem ele nao sabe se é um huracan ou um gallardo mas tem a certeza que nao é um
%aventador. O carro é branco, matricula, 08-07-XP, ano de fabrico 30/6/2014
% Conhecimento negativo / Conhecimento impreciso

proprietario('08-07-XP',cesar).
dataFabrico('08-07-XP',30,6,2014).
cor('08-07-XP',branco).
preco('08-07-XP',345000).


excecao(automovel('08-07-XP',lamborghini,huracan)).
excecao(automovel('08-07-XP',lamborghini,gallardo)).
-automovel('08-07-XP',lamborghini,aventador).
```

Figura 21 - Solução do caso 3.

```
?-
?- demo(automovel('08-07-XP',lamborghini,aventador),R).
  R=falso
  R=falso

?- demo(automovel('08-07-XP',lamborghini,huracan),R).
  R=desconhecido

?-
```

Figura 22 - Resultados do caso 3.

## 5.4 Caso 4

```
% Caso 4:  O Júlio Mendes avistou um Audi R8 em bom estado no Stand Vitoria. Estava tão atento a todos os por
%reparou na matrícula ('22-33-FX') , da data de fabrico do carro
%  (5/6/2009)  no preço exorbitante de 230 000 € mas, quando chegou a casa, já não se lembrava da cor.

automovel('22-23-FX',audi,r8).
dataFabrico('22-23-FX', 05,06,2009).
cor('22-23-FX', xpto23).
preco('22-23-FX', 230000).
estado('22-23-FX', bom).

excecao( cor( M,C) ) :-
    cor( M,xpto23 ).
```

Figura 23 - Solução do caso 4.

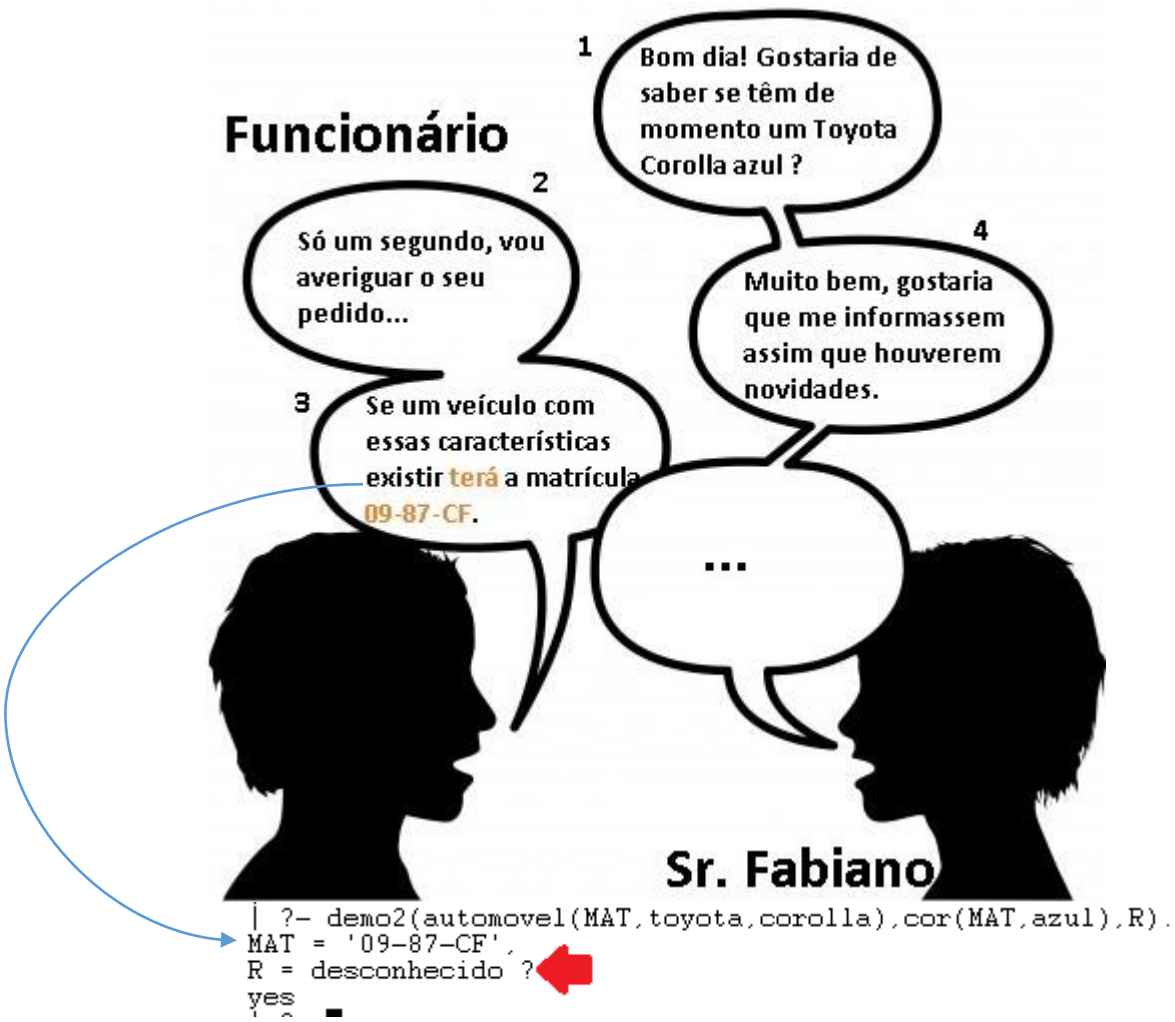
```
Load successful.

?- demo(cor('22-23-FX', azul),R).
  R=desconhecido

?- demo(cor('22-23-FX', preto),R).
  R=desconhecido
```

Figura 24 - Resultados do caso 4.

# 6. Caso de estudo 5

Nesta pequena demonstração pretendemos criar uma ligação mais concreta daquilo que desenvolvemos com o mundo real, ao mesmo tempo que documentamos a capacidade do nosso sistema representar conhecimento imperfeito.

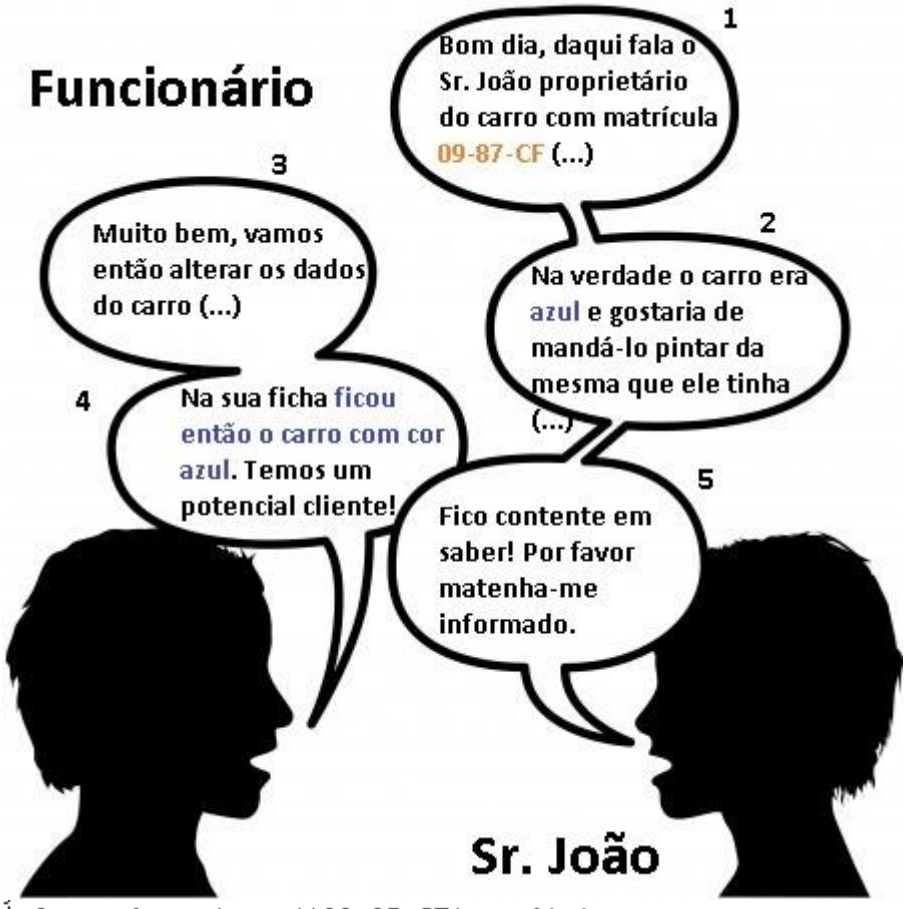
Iremos neste passo situarmo-nos o mais próximo possível daquilo que poderia ser uma situação real no contexto do problema dos stands de automóveis, apresentando uma banda desenhada e traduzindo as ações e diálogos em programação em lógica estendida com a linguagem PROLOG.



Nesta primeira banda desenhada e *queries* em PROLOG podemos ver uma situação típica de conhecimento imperfeito, o funcionário do stand tem informação acerca do modelo e marca do carro e batem certo com os requisitos do Sr. Fabiano, no entanto é desconhecida a cor desse carro com essa marca e esse modelo.
O grupo usa o predicado demo2 já explicado anteriormente neste relatório.

## Nota de leitura prévia

É crucial reler o 1º caso de estudo apresentado em que o Sr. João (que também participará neste pequeno teatro) sabe o modelo e marca do seu carro (Toyota Corolla) mas devido ao desgaste da tinta não se sabe se é azul ou não.




Nesta segunda banda desenhado uma nova personagem entra em cena, desta vez o Sr. João dirige-se ao Stand Vitória para alterar o registo do seu Toyota Corolla. Ele sabe agora que o seu veículo era de cor azul e pede que seja feita nova pintura com a cor antiga. O funcionário regista no sistema a alteração, portanto irá escrever na aplicação do stand a frase na imagem em cima onde é usado o predicado evolução para fazer asserção de conhecimento positivo.

De seguida o funcionário liga ao Sr. Fabiano a transmitir a nova informação…


```
| ?- demo2(automovel(MAT,toyota,corolla),cor(MAT,azul),R).
MAT = '09-87-CF',
R = verdadeiro ?
yes
| ?- remocao(proprietario('09-87-CF',joao)).
yes
| ?- remocao(dataFabrico('09-87-CF',14,04,93)).
yes
| ?- remocao(preco('09-87-CF',2000)).
yes
| ?- remocao(estado('09-87-CF',mal)).
yes
| ?- remocao(cor('09-87-CF',azul)).
yes
| ?- remocao(automovel('09-87-CF',toyota,corolla)).
yes
| ?
```

Nesta banda desenhada final o funcionário verifica através da primeira query à aplicação que de facto o automóvel com o requisitos do Sr. Fabiano encontra-se disponível e tem matrícula ’09-87-CF’, o conhecimento imperfeito foi concretizado em conhecimento verdadeiro pela asserção da cor do carro (banda desenhada anterior).

Como o Sr. Fabiano decide comprar o carro temos de remover os registos do mesmo da base de conhecimento da aplicação daí o bloco laranja de remoções seguido da query de verificação simultânea de condições de procura.

# 5 Conclusões

Com este trabalho prático conseguimos interiorizar conceitos relativos à programação em lógica estendida usando como ferramenta o PROLOG. É necessário ter em mente que a assunção do mundo aberto e domínio aberto tem de fazer sentido no contexto do problema, mas em termos generalistas podemos concluir que quando a nossa base desconhece uma dada entidade ele nunca pode afirmar ou negar essa entidade a menos que seja explícito na base de conhecimento essa negação ou essa afirmação ou seja, a representação de conhecimento negativo ou positivo daquela entidade. Em vez disso é introduzido um novo conceito, o conceito “desconhecido”, ou seja para aquelas entidades externas ao nosso mundo nós vamos dizer que o valor de conhecimento relativo a elas é “desconhecido”.

Relativamente ao desempenho do grupo fica a nota de cumprimento dos requisitos básicos do trabalho, a forma original de como foram apresentados os nossos resultados e a implementação da consola que apesar de uma simples linha de comandos não é uma solução trivial.

# Bibliografia

[Analide, 2001] ANALIDE, Cesar, NOVAIS, Paulo, NEVES, José, “Sugestões para a Elaboração de Relatórios”, Relatório Técnico, Departamento de Informática, Universidade do Minho, Portugal, 2001.

# Anexos

# I. Anexo 1- JAVADOC da aplicação Stand Vitória

## Class PrologStub

java.lang.Object
    PrologStub

---

```
public class PrologStub
extends java.lang.Object
```

Classe de comunicação com o SICStus Prolog. Contém métodos que efetuam pré-validação de querys, verificação e parse de resultados, contém mecanismos de detação de erros e excepção por forma a conseguir reproduzir de forma credível a resposta dada pelo SICStus.

**Version:**

2015.06.08

**Author:**

Daniel Caldas, José Cortez, Susana Mendes, Xavier Rodrigues

### Field Summary

**Fields**

| Modifier and Type | Field and Description |
|---|---|
| static java.lang.String | OK |

### Constructor Summary

**Constructors**

| Constructor and Description |
|---|
| PrologStub(java.lang.String file)<br>Construtor. |

### Method Summary

**Methods**

| Modifier and Type | Method and Description |
|---|---|
| java.lang.String | checkError(java.lang.String s)<br>Pré-verificação de erros de sintaxe. |
| java.lang.String | interpret(java.lang.String queryS)<br>Função que chama o sicstus para intrepertar uma dada query. |
| boolean | loadOK()<br>Verificar se código foi carregado sem problemas. |

### Field Detail

#### OK

```
public static java.lang.String OK
```

### Constructor Detail

#### PrologStub

```
public PrologStub(java.lang.String file)
```

Construtor.

**Parameters:**

`file` - do ficheiro com código em PROLOG que queremos carregar.

## Method Detail

### loadOK

```
public boolean loadOK()
```

Verificar se código foi carregado sem problemas.

**Returns:**

booleano

### checkError

```
public java.lang.String checkError(java.lang.String s)
```

Pré-verificação de erros de sintaxe.

**Parameters:**

`s` - uma query em PROLOG

**Returns:**

String com mensagem de confirmação ou erro

### interpret

```
public java.lang.String interpret(java.lang.String queryS)
```

Função que chama o sicstus para intrepertar uma dada query.

**Parameters:**

`queryS` - query em PROLOG.

**Returns:**