

Exercício Prático N°1 - Grupo 24 - Sistemas de Representação de Conhecimento e Raciocínio

Ângelo Dias Teixeira
a73312



Bruno Manuel Arieira
a70565



João Miguel Palmeira
a73864



Pedro Manuel Almeida
a74301



Março 2018

1 Resumo

Neste documento encontra-se apresentado o método de resolução do primeiro exercício do trabalho prático de Sistemas de Representação de Conhecimento e Raciocínio. Apresentam-se as decisões que nos conduziram à resolução do problema, mostrando código e explicando o mesmo.

Contents

1	Resumo	2
2	Introdução	6
3	Preliminares	7
4	Descrição do Trabalho e Análise de Resultado	8
4.1	Base de Conhecimento	8
4.2	Requisitos do Enunciado	10
4.3	Funcionalidades	11
4.3.1	Predicado Evolução (alínea 1)	11
4.3.2	Predicado inewvolução (alínea 2)	13
4.3.3	Predicado Identificar utentes por critérios de seleção (Alínea 3)	15
4.3.4	Predicado Identificar as instituições prestadoras de cuidados de saúde (Alínea 4)	19
4.3.5	Predicado Identificar cuidados de saúde prestados por instituição/cidade/datas (Alínea 5)	20
4.3.6	Predicado Identificar os utentes de um prestador/especialidade/instituição (Alínea 6)	23
4.3.7	Predicado Identificar cuidados de saúde realizados por utente/instituição/prestador (Alínea 7)	28
4.3.8	Predicado Determinar todas as instituições/prestadores a que um utente já recorreu (Alínea 8)	29
4.3.9	Predicado Calcular o custo total dos cuidados de saúde por utente/especialidade/prestador/datas (Alínea 9)	31
4.4	Outros Predicados	36
4.4.1	Predicado Teste	36
4.4.2	Predicado Comprimento	36
4.4.3	Predicado Concatenar	37
4.4.4	Predicado Somatório	37
4.4.5	Predicado Remover Elemento	37
4.4.6	Predicado Remover Repetidos	38
4.4.7	Predicado Não Negativo	38
4.4.8	Predicado Média	38
5	Novas Funcionalidades	39
5.1	Média de idades dos utentes	39
5.2	Média de idades por morada	39
5.3	Número de Prestadores de uma Instituição	40
5.4	Número de Utentes de uma Instituição	40
5.5	Número de Cuidados de uma Instituição	41
6	Conclusão e Trabalho Futuro	43

List of Figures

1	Declarações iniciais	8
2	Declaraões iniciais	8
3	InvarianteS Estruturais	9
4	Factos	10
5	Predicado evolucao e predicado insercao	11
6	Predicado evolução de um Utente	12
7	Predicado evolução de um Prestador	12
8	Predicado evolução de um Cuidado	13
9	Predicado inewolucao	13
10	Predicado inewolução um Utente	14
11	Predicado inewolução um Prestador	14
12	Predicados listarUtenteI, listarUtenteM, listarUtenteNI, listarUtenteNM, listarUtenteIM, listarUtenteNIM, listarUtenteID	15
13	Predicado listarUtenteI	16
14	Predicado listarUtenteM	16
15	Predicado listarUtenteNI	17
16	Predicado listarUtenteNM	17
17	Predicado listarUtenteIM	18
18	Predicado listarUtenteNIM	18
19	Predicado listarUtenteID	19
20	Predicado listarInstituicoes	19
21	Predicado Identificar uma Instituição Prestadora	20
22	Predicado prestadorInstituicao, cuidadosInstituicao, icuidados, cuidadosData	21
23	Predicado cuidadosInstituicao	22
24	Predicado utenteMorada	22
25	Predicado cuidadosData	23
26	Predicado utentesDeUmPrestador, uPaux, prestadorEspecialidade, utentesEspecialidade, uEaux, utentesPrestador	24
27	Predicado utentesInstituicao, uIaux, prestadoresInstituicao	25
28	Predicado Utentes de um determinado Prestador	26
29	Predicado Utentes de uma determinada Instituição	26
30	Predicado Utentes de uma determinada Especialidade	27
31	Predicado cuidadoUtente, cuidadoPrestador	28
32	Predicado cuidado de saude realizador por utente	29
33	Predicado cuidado de saude realizador por prestador	29
34	Predicado prestadoresUtente, instituicoesUtente, iu, idPrestadoresUtente	30
35	Predicado Prestadores de um Utente	31
36	Predicado Instituições de um Utente	31
37	Predicado custoUtente, prestadorEspecialidade, custoEspecialidade, cEaux, custoPrestador, custoData	33
38	Predicado custo total dos cuidados de um Utente	34
39	Predicado custo total dos cuidados de um Prestador	34

40	Predicado custo total dos cuidados de uma determinada Data	35
41	Predicado custo total dos cuidados de uma determinada Especialidade	35
42	Predicado teste	36
43	Predicado comprimento	36
44	Predicado concatenar	37
45	Predicado somatorio	37
46	Predicado removerElemento	37
47	Predicado removerRepetidos	38
48	Predicado naoNegativo	38
49	Predicado media	38
50	Predicado mediaIdades_utentes	39
51	Predicado que fornece a média de idades de todos os utentes	39
52	Predicado mediaIdades_morada	39
53	Predicado que fornece a média de idades dos utentes de uma morada dada	40
54	Predicado prestadoresN_Instituicao	40
55	Predicado que fornece o número de prestadores de uma instituição	40
56	Predicado nUtentesInstituicao	41
57	Predicado que fornece o número de utentes de uma instituição	41
58	Predicado nCuidadosInstituicao	41
59	Predicado que fornece o número de Cuidados de uma instituição	42

2 Introdução

No presente ano letivo foi proposto aos alunos de Sistemas de Representação de Conhecimento e Raciocínio que desenvolvessem um projeto de programação em lógica que emulasse o funcionamento de uma instituição de saúde. O tema escolhido possibilita a construção de bases de conhecimento de complexidade média e assim permite a demonstração de conhecimentos de programação em lógica e de invariantes que garantam a consistência das bases de conhecimento.

3 Preliminares

Para a realização deste trabalho prático, é necessário ter os conhecimentos adquiridos nas aulas práticas e teóricas da UC. Adicionalmente, existiu também uma breve pesquisa sobre Programação em lógica e exemplos do seu uso. Posto isto, realizaremos agora uma breve introdução à linguagem Prolog:

- Facto - constata algo que se reconhece e se sabe verdadeiro;
- Predicado - implementa uma relação;
- Regra - utilizada para definir um novo predicado;
- ”. - Utilizado para terminar uma declaração;
- ”:- - Representa o ”se”;
- ”, - Representa o ”e”;
- ”; - Representa o ”ou”;
- // Representa a unificação;
- Invariante - Regra definida que tem obrigatoriamente de ser válida ao longo da execução do programa;

4 Descrição do Trabalho e Análise de Resultado

4.1 Base de Conhecimento

Para iniciar a elaboração deste trabalho, decidimos começar por definir a quantidade de argumentos relativos a cada conhecimento, usando para tal o dynamic. Pela leitura do enunciado, optamos que cada utente tem associado quatro parâmetros, o prestador também e o cuidado contém cinco.

```
:– op( 900,xfy,'::' ).  
:– dynamic utente/4.  
:– dynamic prestador/4.  
:– dynamic cuidado/5.
```

Figure 1: Declarações iniciais

Após essa definição, alteramos três flags do PROLOG para evitar ter warnings ou erros aquando da compilação do ficheiro usado para o desenvolvimento deste sistema, podendo assim ser executado corretamente e de acordo com o tema desta fase da disciplina.

```
:– set_prolog_flag( discontiguous_warnings,off ).  
:– set_prolog_flag( single_var_warnings,off ).  
:– set_prolog_flag( unknown,fail ).
```

Figure 2: Declarações iniciais

De seguida, definimos invariantes estruturais que se encontram dispostos na figura abaixo. Os invariantes estruturais dizem respeito a operações de inserção e remoção de conhecimento no sistema

```

%----- %
% Invariante Estrutural: nao permitir a insercao de numero de utente repetido
+utente( X,Y,Z,W ) :: (findall( X,(utente( X,B,C,D )),S ),
                        comprimento( S,N ),
                        N == 1
                      ).

%----- %
% Invariante Estrutural: nao permitir a insercao de numero de prestador repetido
+prestashop( X,Y,Z,W ) :: (findall(X, (prestashop( X,B,C,D )),S ),
                            comprimento( S,N ),
                            N == 1
                          ).

%----- %
% Invariante Estrutural: nao permitir a insercao de cuidado repetido
+cuidado( X,Y,Z,W,W1 ) :: (findall( (X,Y,Z,W,W1),(cuidado( X,Y,Z,W,W1 )),S ),
                            comprimento( S,N ),
                            N == 1
                          ).

%----- %
% Invariante Estrutural: nao permitir a insercao de custo negativo
+cuidado(X,Y,Z,W,W1) :: (findall(W1,cuidado(A,B,C,D,W1)),S),
                        naoNegativo(S)
                      ).

%----- %
% Invariante Estrutural: nao permitir a insercao de cuidado com utente inexistente
+cuidado( X,Y,Z,W,W1 ) :: (findall( A,(utente( A,B,C,D )),S ),
                            contem( Y,S )
                          ).

%----- %
% Invariante Estrutural: nao permitir a insercao de cuidado com prestador inexistente
+cuidado( X,Y,Z,W,W1 ) :: (findall( A,(prestashop( A,B,C,D )),S ),
                            contem( Z,S )
                          ).

%----- %
% Invariante Estrutural: nao permitir a remocao de utente caso exista cuidado associado
-utente(X,Y,W,Z) :: (findall(X, cuidado(A,X,C,D,E)),S),
                     comprimento(S,N),
                     N == 0).

%----- %
% Invariante Estrutural: nao permitir a remocao de prestador caso exista cuidado associado
-prestador(X,Y,W,Z) :: (findall(X, cuidado(A,B,X,D,E)), S),
                        comprimento(S,N),
                        N == 0).

```

Figure 3: Invariantes Estruturais

Além disso, definimos os factos que são sempre verdadeiros, sendo eles:

- Utente : As informações pessoais são o nome a idade e a morada, tendo ainda associado um id.
- Prestador : Tem associado as informações pessoais tais como o nome, a especialidade e a instituição onde trabalha bem como um id associado.
- Cuidado : Tem associado uma data, um id do utente, um id do prestador, uma descrição e um custo.

```

%----- Base de Conhecimento -----
%----- Utente -----
% Utente: #idUt, Nome, Idade, Morada

utente(3, joao, 21, braga).
utente(5, angelo, 22, braga).
utente(7, bruno, 22, viana).
utente(9, pedro, 21, barcelos).
utente(28, luis, 34, lisboa).
utente(21, josefina, 25, faro).

%----- Prestador -----
% Prestador: #Prest, Nome, Especialidade, Instituicao

prestashop(5, tiago, ortopedia, hospitalViana).
prestashop(2, fernando, oftalmologia, hospitalPorto).
prestashop(32, jose, podologia, hospitalBraga).
prestashop(14, francisco, cardiologia, hospitalBraga).
prestashop(23, manuel, psiquiatria, hospitalPorto).
prestashop(28, jorge, otorrinolaringologia, hospitalLisboa).

%----- Cuidados -----
% Cuidado: Data, #idUt, #idPrest, Descricao, Custo

cuidado(01-03-2015, 5, 14, enfartedomiocardio, 60).
cuidado(15-12-2014, 7, 28, timpanofurado, 45).
cuidado(09-08-2016, 5, 14, colocacaopacemaker, 52).
cuidado(17-05-2013, 3, 23, criseexistencial, 56).
cuidado(23-01-2018, 21, 32, remocaoadoutero, 65).
cuidado(16-02-2014, 9, 2, testedevisao, 15).
cuidado(03-06-2011, 20, 5, entorce, 41).

```

Figure 4: Factos

Por fim, foram definidos os predicados propriamente ditos, estando estes divididos em duas secções: predicados mais "gerais", que podem e são utilizados por outros predicados, e os predicados que permitem resolver as necessidades de demonstração das funcionalidades apresentadas no enunciado do primeiro exercício prático.

Os predicados globais e definidos para o uso de outros predicados são:

- evolucao
- teste
- inevolucao
- comprimento
- concatenar
- somatorio

4.2 Requisitos do Enunciado

O enunciado deste exercício previa a existência dos seguintes requisitos, sendo que estes serão abordados na seção "Funcionalidades":

- Registar utentes, prestadores e cuidados de saúde;
- Remover utentes, prestadores e cuidados de saúde;
- Identificar utentes por critérios de seleção;
- Identificar as instituições prestadoras de cuidados de saúde;
- Identificar cuidados de saúde prestados por instituição/cidade/datas;
- Identificar os utentes de um prestador/especialidade/instituição;
- Identificar cuidados de saúde realizados por utente/instituição/prestador;
- Determinar todas as instituições/prestadores a que um utente já recorreu;
- Calcular o custo total dos cuidados de saúde por utente/especialidade/prestador/datas;

4.3 Funcionalidades

4.3.1 Predicado Evolução (alínea 1)

O primeiro predicado é usado para registar utentes, prestador e cuidados. Deste modo, sempre que se pretende registar algum deste tipo de conhecimento, os invariantes estruturais são testados. Este teste aos invariantes estruturais permite não inserir nem utentes, nem prestadores nem cuidados que já estejam registados na base de conhecimento. O teste dos invariantes garante que não é inserido um utente com um IdUt já existente, nem que é inserido um prestador com um IdPrest já utilizado e que também não é adicionado um cuidado com uma combinação de IdUt e IdPrest já existentes. Além disso, dois dos invariantes estruturais permitem garantir que não é adicionado nenhum cuidado com um utente (IdUt) inexistente na base de conhecimento e que não é inserido um cuidado com um prestador (IdPrest) não existente.

```
%----- %
% Extensão do predicado que permite a evolução do conhecimento: P -> {V,F} 1
%
evolucao( Termo ) :-
    findall( Invariante,+Termo::Invariante,Lista ),
    insercao( Termo ),
    teste( Lista ).

insercao( Termo ) :-
    assert( Termo ).

insercao( Termo ) :-
    retract( Termo ),!,fail.
```

Figure 5: Predicado evolucao e predicado insercao

- Output

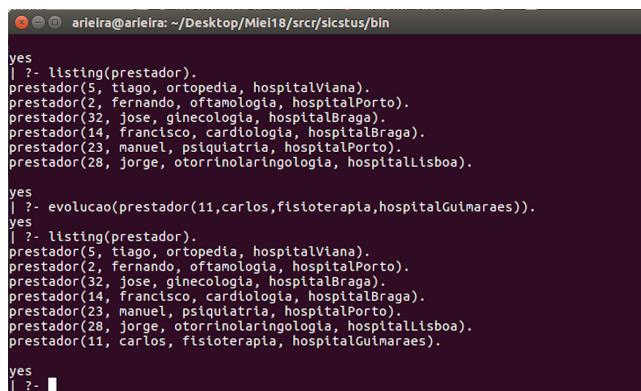


```
msec -112 bytes
arieira@arieira: ~/Desktop/MieI18/srcr/sicstus/bin
yes
| ?- listing(utente).
utente(3, joao, 21, braga).
utente(5, angelo, 22, braga).
utente(7, bruno, 22, viana).
utente(9, pedro, 21, barcelos).
utente(20, luis, 34, lisboa).
utente(21, josefina, 25, faro).

yes
| ?- evolucao(utente(4,joel,43,braganca)).
yes
| ?- listing(utente).
utente(3, joao, 21, braga).
utente(5, angelo, 22, braga).
utente(7, bruno, 22, viana).
utente(9, pedro, 21, barcelos).
utente(20, luis, 34, lisboa).
utente(21, josefina, 25, faro).
utente(4, joel, 43, braganca).

yes
| ?-
```

Figure 6: Predicado evolução de um Utente



```
arieira@arieira: ~/Desktop/MieI18/srcr/sicstus/bin
yes
| ?- listing(prestador).
prestador(5, tiago, ortopedia, hospitalViana).
prestador(2, fernando, oftamologia, hospitalPorto).
prestador(32, jose, ginecologia, hospitalBraga).
prestador(14, francisco, cardiologia, hospitalBraga).
prestador(23, manuel, psiquiatria, hospitalPorto).
prestador(28, jorge, otorrinolaringologia, hospitalLisboa).

yes
| ?- evolucao(prestador(11,carlos,fisioterapia,hospitalGuimaraes)).
yes
| ?- listing(prestador).
prestador(5, tiago, ortopedia, hospitalViana).
prestador(2, fernando, oftamologia, hospitalPorto).
prestador(32, jose, ginecologia, hospitalBraga).
prestador(14, francisco, cardiologia, hospitalBraga).
prestador(23, manuel, psiquiatria, hospitalPorto).
prestador(28, jorge, otorrinolaringologia, hospitalLisboa).
prestador(11, carlos, fisioterapia, hospitalGuimaraes).

yes
| ?-
```

Figure 7: Predicado evolução de um Prestador

```

arieira@arieira ~/Desktop/MIEI18/srcr/sicstus/bin
| ?- listing(cuidado).
cuidado(1-3-2015, 5, 14, enfartedomicardio, 60).
cuidado(15-12-2014, 7, 28, timpanofurado, 45).
cuidado(9-8-2016, 5, 14, colocacaopacemaker, 52).
cuidado(17-5-2013, 3, 23, criseexistencial, 56).
cuidado(23-1-2018, 21, 32, remocaouterco, 65).
cuidado(16-2-2014, 9, 2, testedevisao, 15).
cuidado(3-6-2011, 20, 5, entorce, 41).

yes
| ?- evolucao(cuidado(25-10-2015, 21, 28, limpeza, 30)).
yes
| ?- listing(cuidado).
cuidado(15-12-2014, 7, 28, timpanofurado, 45).
cuidado(9-8-2016, 5, 14, colocacaopacemaker, 52).
cuidado(17-5-2013, 3, 23, criseexistencial, 56).
cuidado(23-1-2018, 21, 32, remocaouterco, 65).
cuidado(16-2-2014, 9, 2, testedevisao, 15).
cuidado(3-6-2011, 20, 5, entorce, 41).
cuidado(25-10-2015, 21, 28, limpeza, 30).

yes
| ?-

```

Figure 8: Predicado evolução de um Cuidado

4.3.2 Predicado inevolução (alínea 2)

Este predicado é o oposto do predicado evolucao, já apresentado e explicado acima. Ou seja, como o seu nome sugere, permite retirar conhecimento já disponível no sistema. Para tal, recebe um F que corresponderá ao conhecimento que se pretende retirar, e ele é retirado caso tal seja possível. Caso contrário, quando não é permitido retirar tal conhecimento, pois desrespeita algum invariante, a remoção falha e o conhecimento não é retirado do sistema.

```

% Extensão do predicado que permite a remoção do conhecimento: Termo -> {V,F}

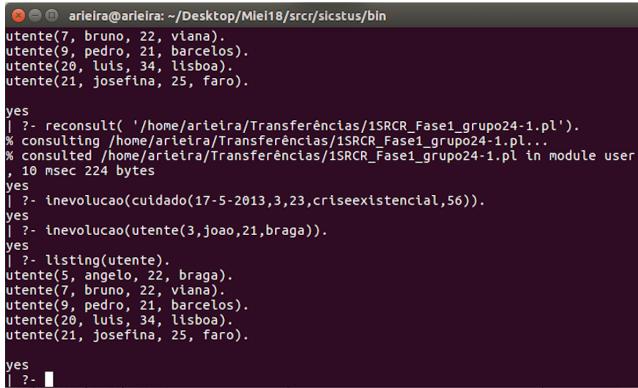
inevolucao(Termo) :-
    findall( Invariante, -Termo::Invariante, Lista),
    remocao(Termo),
    teste(Lista).

remocao(Termo) :-
    retract(Termo).
remocao(Termo) :-
    assert(Termo), !, fail.

```

Figure 9: Predicado inevolucao

- Output



```

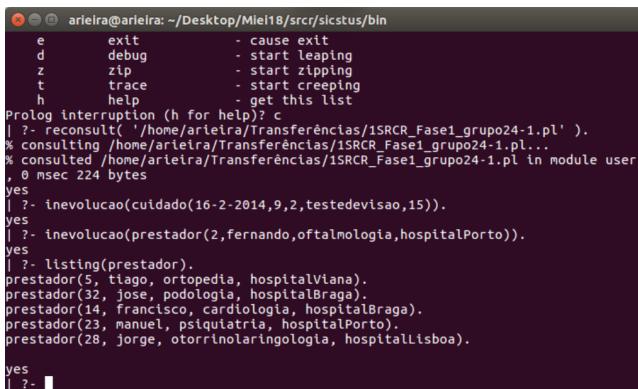
arieira@arieira: ~/Desktop/Mlei18/src/sicstus/bin
utente(7, bruno, 22, viana).
utente(9, pedro, 21, barcelos).
utente(20, luis, 34, lisboa).
utente(21, josefina, 25, faro).

yes
| ?- reconsult( 'home/arieira/Transferências/1SRCR_Fase1_grupo24-1.pl').
% consulting /home/arieira/Transferências/1SRCR_Fase1_grupo24-1.pl...
% consulted /home/arieira/Transferências/1SRCR_Fase1_grupo24-1.pl in module user
, 10 msec 224 bytes
yes
| ?- inevolucao(cuidado(17-5-2013,3,23,criseexistencial,56)).
yes
| ?- inevolucao(utente(3,joao,21,braga)).
yes
| ?- listing(utente).
utente(5, angelo, 22, braga).
utente(7, bruno, 22, viana).
utente(9, pedro, 21, barcelos).
utente(20, luis, 34, lisboa).
utente(21, josefina, 25, faro).

yes
| ?-

```

Figure 10: Predicado inevolução um Utente



```

arieira@arieira: ~/Desktop/Mlei18/src/sicstus/bin
e      exit          - cause exit
d      debug         - start leaping
z      zip           - start zipping
t      trace         - start creeping
h      help          - get this list
Prolog interruption (h for help)? c
| ?- reconsult( 'home/arieira/Transferências/1SRCR_Fase1_grupo24-1.pl').
% consulting /home/arieira/Transferências/1SRCR_Fase1_grupo24-1.pl...
% consulted /home/arieira/Transferências/1SRCR_Fase1_grupo24-1.pl in module user
, 0 msec 224 bytes
yes
| ?- inevolucao(cuidado(16-2-2014,9,2,testedevisaو,15)).
yes
| ?- inevolucao(prestador(2,fernando,oftalmologia,hospitalPorto)).
yes
| ?- listing(prestador).
prestador(5, tiago, ortopedia, hospitalViana).
prestador(32, jose, podologia, hospitalBraga).
prestador(14, francisco, cardiology, hospitalBraga).
prestador(23, manuel, psiquiatria, hospitalPorto).
prestador(28, jorge, otorrinolaringologia, hospitalLisboa).

yes
| ?-

```

Figure 11: Predicado inevolução um Prestador

4.3.3 Predicado Identificar utentes por critérios de seleção (Alínea 3)

Para identificar os utentes por critérios de seleção, criamos os predicados que recebem cada um a característica correspondente, isto é, o nome, o id, entre outros e procura na lista dos utentes o utente com essa característica com o auxílio do predicado findall.

```
%-----  
% Extensao do predicado listarUtenteN: N,S -> {V,F}      3  
  
listarUtenteN(N,S):-  
    findall(X,utente(X,N,Z,W),S).  
%-----  
% Extensao do predicado listarUtenteI: I,S -> {V,F}      3  
  
listarUtenteI(I,S):-  
    findall(X,utente(X,Y,I,W),S).  
%-----  
% Extensao do predicado listarUtenteM: M,S -> {V,F}      3  
  
listarUtenteM(M,S):-  
    findall(X,utente(X,Y,Z,M),S).  
%-----  
% Extensao do predicado listarUtenteNI: N,I,S -> {V,F}      3  
  
listarUtenteNI(N,I,S):-  
    findall(X,utente(X,N,I,W),S).  
%-----  
% Extensao do predicado listarUtenteNM: N,M,S -> {V,F}      3  
  
listarUtenteNM(N,M,S):-  
    findall(X,utente(X,N,Z,M),S).  
%-----  
% Extensao do predicado listarUtenteIM: I,M,S -> {V,F}      3  
  
listarUtenteIM(I,M,S):-  
    findall(X,utente(X,Y,I,M),S).  
%-----  
% Extensao do predicado listarUtenteNIM: N,I,M,S -> {V,F}      3  
  
listarUtenteNIM(N,I,M,S):-  
    findall(X,utente(X,N,I,M),S).  
%-----  
% Extensao do predicado listarUtenteID: ID,S -> {V,F}      3  
  
listarUtenteID(ID,S):-  
    findall((ID,N,Idade,M),utente(ID,N,Idade,M),S).
```

Figure 12: Predicados listarUtenteI, listarUtenteM, listarUtenteNI, listarUtenteNM, listarUtenteIM, listarUtenteNIM, listarUtenteID

- Output

```

arieira@arieira:~/Desktop/Miei18/srcr/sicstus/bin
arieira@arieira:~/Desktop/Miei18/srcr/sicstus$ cd bin
arieira@arieira:~/Desktop/Miei18/srcr/sicstus/bin$ ./sicstus
bash: ./sicstus: Ficheiro ou directorio inexistente
arieira@arieira:~/Desktop/Miei18/srcr/sicstus/bin$ ./sicstus
SICStus 4.3.0 (x86_64-linux-glibc2.12): Thu May  8 05:34:25 PDT 2014
Licensed to studentSP4.3di.uninho.pt
| ?- consult(~/home/arieira/Transferências/SRCR_Fase1_grupo24.pl).
% consulting /home/arieira/Transferências/SRCR_Fase1_grupo24.pl...
% consulted /home/arieira/Transferências/SRCR_Fase1_grupo24.pl in module user, 1
0 msec 148416 bytes
yes
| ?- listing(utente).
utente(3, joao, 21, braga).
utente(5, angelo, 22, braga).
utente(7, bruno, 22, viana).
utente(9, pedro, 21, barcelos).
utente(20, luis, 34, lisboa).
utente(21, josefina, 25, faro).

yes
| ?- listarUtenteI(22,S).
S = [(5,angelo,22,braga),(7,bruno,22,viana)] ?
yes
| ?-

```

Figure 13: Predicado listarUtenteI

```

arieira@arieira:~/Desktop/Miei18/srcr/sicstus/bin
utente(3, joao, 21, braga).
utente(5, angelo, 22, braga).
utente(7, bruno, 22, viana).
utente(9, pedro, 21, barcelos).
utente(20, luis, 34, lisboa).
utente(21, josefina, 25, faro).

yes
| ?- listarUtenteI(22,S).
S = [(5,angelo,22,braga),(7,bruno,22,viana)] ?
yes
| ?- listing(utente).
utente(3, joao, 21, braga).
utente(5, angelo, 22, braga).
utente(7, bruno, 22, viana).
utente(9, pedro, 21, barcelos).
utente(20, luis, 34, lisboa).
utente(21, josefina, 25, faro).

yes
| ?- listarUtenteM(braga,S).
S = [(3,joao,21,braga),(5,angelo,22,braga)] ?
yes
| ?-

```

Figure 14: Predicado listarUtenteM

```

arieira@arieira: ~/Desktop/Mlei18/srcr/sicstus/bin
| ?- listing(utente).
utente(3, joao, 21, braga).
utente(5, angelo, 22, braga).
utente(7, bruno, 22, viana).
utente(9, pedro, 21, barcelos).
utente(20, luis, 34, lisboa).
utente(21, josefina, 25, faro).

yes
| ?- listarUtente(angelo,22,S).
no
| ?- listing(utente).
utente(3, joao, 21, braga).
utente(5, angelo, 22, braga).
utente(7, bruno, 22, viana).
utente(9, pedro, 21, barcelos).
utente(20, luis, 34, lisboa).
utente(21, josefina, 25, faro).

yes
| ?- listarUtenteNI(angelo,22,S).
S = [(5,angelo,22,braga)] ?
yes
| ?-

```

Figure 15: Predicado listarUtenteNI

```

arieira@arieira: ~/Desktop/Mlei18/srcr/sicstus/bin
utente(3, joao, 21, braga).
utente(5, angelo, 22, braga).
utente(7, bruno, 22, viana).
utente(9, pedro, 21, barcelos).
utente(20, luis, 34, lisboa).
utente(21, josefina, 25, faro).

yes
| ?- listarUtenteNI(angelo,22,S).
S = [(5,angelo,22,braga)] ?
yes
| ?- listing(utente).
utente(3, joao, 21, braga).
utente(5, angelo, 22, braga).
utente(7, bruno, 22, viana).
utente(9, pedro, 21, barcelos).
utente(20, luis, 34, lisboa).
utente(21, josefina, 25, faro).

yes
| ?- listarUtenteNM(bruno,viana,S).
S = [(7,bruno,22,viana)] ?
yes
| ?-

```

Figure 16: Predicado listarUtenteNM

```

arieira@arieira:~/Desktop/Mlei18/srcr/sicstus/bin
| ?- listing(utente).
utente(3, joao, 21, braga).
utente(5, angelo, 22, braga).
utente(7, bruno, 22, viana).
utente(9, pedro, 21, barcelos).
utente(20, luis, 34, lisboa).
utente(21, josefina, 25, faro).

yes
| ?- listarUtenteIM(22,braga).
no
| ?- listing(utente).
utente(3, joao, 21, braga).
utente(5, angelo, 22, braga).
utente(7, bruno, 22, viana).
utente(9, pedro, 21, barcelos).
utente(20, luis, 34, lisboa).
utente(21, josefina, 25, faro).

yes
| ?- listarUtenteIM(22,braga,S).
S = [(5,angelo,22,braga)] ?
yes
| ?-

```

Figure 17: Predicado listarUtenteIM

```

arieira@arieira:~/Desktop/Mlei18/srcr/sicstus/bin
utente(3, joao, 21, braga).
utente(5, angelo, 22, braga).
utente(7, bruno, 22, viana).
utente(9, pedro, 21, barcelos).
utente(20, luis, 34, lisboa).
utente(21, josefina, 25, faro).

yes
| ?- listarUtenteIM(22,braga,S).
S = [(5,angelo,22,braga)] ?
yes
| ?- listing(utente).
utente(3, joao, 21, braga).
utente(5, angelo, 22, braga).
utente(7, bruno, 22, viana).
utente(9, pedro, 21, barcelos).
utente(20, luis, 34, lisboa).
utente(21, josefina, 25, faro).

yes
| ?- listarUtenteNIM(pedro,21,barcelos,S).
S = [(9,pedro,21,barcelos)] ?
yes
| ?-

```

Figure 18: Predicado listarUtenteNIM

```

arieira@arieira: ~/Desktop/Mlei18/srcr/sicstus/bin
| ?- listing(utente).
utente(3, joao, 21, braga).
utente(5, angelo, 22, braga).
utente(7, bruno, 22, viana).
utente(9, pedro, 21, barcelos).
utente(20, luis, 34, lisboa).
utente(21, josefina, 25, faro).

yes
| ?- listarUtente(20,S).
no
| ?- listing(utente).
utente(3, joao, 21, braga).
utente(5, angelo, 22, braga).
utente(7, bruno, 22, viana).
utente(9, pedro, 21, barcelos).
utente(20, luis, 34, lisboa).
utente(21, josefina, 25, faro).

yes
| ?- listarUtenteID(21,S).
S = [(21,josefina,25,faro)] ?
yes
| ?-

```

Figure 19: Predicado listarUtenteID

4.3.4 Predicado Identificar as instituições prestadoras de cuidados de saúde (Alínea 4)

Neste caso o nosso objetivo é identificarmos as instituições prestadoras de cuidados de saúde, para isso, foi preciso criarmos um predicado listarInstituicoes, que irá devolver uma lista das instituições (S). Para conseguirmos encontrar essas devidas instituições foi necessário recorrermos a outro predicado chamado findall onde irá ser utilizado também outro predicado, o predicado prestador, que irá retornar uma lista com as Instituições dentro do Prestador, tal como pedido.

```

%----- %
% Extensão do predicado listarInstituicoes: S -> {V,F} 4
%
listarInstituicoes( S ) :-
    findall( W, prestador(X,Y,Z,W), S ).
```

Figure 20: Predicado listarInstituicoes

- Output

```

arieira@arieira- ~/Desktop/Mel18/src/sicstus/bin
utente(5, angelo, 22, braga).
utente(7, bruno, 22, viana).
utente(9, pedro, 21, barcelos).
utente(20, luis, 34, lisboa).
utente(21, josefina, 25, faro).

yes
| ?- listarUtenteID(21,S).
S = [(21,josefina,25,faro)] ?
yes
| ?- listing(prestador).
prestashop(5, tiago, ortopedia, hospitalViana).
prestashop(2, fernando, oftamologia, hospitalPorto).
prestashop(32, jose, ginecologia, hospitalBraga).
prestashop(14, francisco, cardiologia, hospitalBraga).
prestashop(23, manuel, psiquiatria, hospitalPorto).
prestashop(28, jorge, otorrinolaringologia, hospitalLisboa).

yes
| ?- listarInstituicoes(S).
S = [hospitalViana,hospitalPorto,hospitalBraga,hospitalBraga,hospitalPorto,hospitalLisboa] ?
yes
| ?-

```

Figure 21: Predicado Identificar uma Instituição Prestadora

4.3.5 Predicado Identificar cuidados de saúde prestados por instituição/cidade/datas (Alínea 5)

Primeiramente no predicado prestadoresInstituicao irá ser recebido uma instituição e uma lista (I/S) e será invocado o predicado findall que posteriormente invocará o predicado prestador de modo a verificar se o prestador existe na instituição. No predicado icuidados será concatenada a lista dos cuidados procurados (findall) sobre o prestador.

Já no predicado cuidadosInstituicao (predicado que determina os cuidados de saúde prestados por uma instituição) irão ser utilizados os outros dois predicados anteriores, fornecendo o resultado pretendido.

Já o predicado cuidadosData irá usar o predicado findall e este irá receber um cuidado e um predicado cuidado.

```

%----- %
% Lista os cuidados oferecidos por instituição
% Extensão do predicado cuidadosPorInstituicao: Instituicao,S -> {V,F}

cuidadosPorInstituicao(I,S) :-
    findall(X, prestador(A,B,X,I),S).

%----- %
% Fornece IDs dos utentes de uma dada cidade(morada)
% Extensão do predicado utenteMorada: M,S -> {V,F}

utenteMorada(M,S) :-
    findall(X, utente(X,B,C,M),S).

%----- %
% Fornece os cuidados prestados por cidade
% Extensão do predicado cuidadosCidade: I,S -> {V,F}

cuidadosCidade(I,S) :-
    utenteMorada(I,U),
    cCaux(U,S).

cCaux([],[]).
cCaux([H|T],S) :-
    findall((A,H,C,D,E), cuidado(A,H,C,D,E),NL),
    concatenar(N,NL,S),
    cCaux(T,N).

%----- %
% Lista os cuidados prestados por Data
% Extensão do predicado cuidadosData: Data,S -> {V,F}

cuidadosData(Data,S) :-
    findall((Data,B,C,D,E), cuidado(Data,B,C,D,E),S).

```

Figure 22: Predicado prestadorInstituicao, cuidadosInstituicao, icuidados, cuidadosData

- Output

```

cuidado(1-3-2015, 5, 14, enfartedomicardio, 60).
cuidado(15-12-2014, 7, 28, tmapanofurado, 45).
cuidado(9-8-2016, 5, 14, colocacaopacemaker, 52).
cuidado(23-1-2018, 21, 32, criseexistencial, 50).
cuidado(23-1-2018, 21, 32, remocadoutero, 65).
cuidado(10-2-2014, 9, 2, testedevisao, 15).
cuidado(3-6-2011, 20, 5, entorce, 41).

yes
| ?- cuidadoInstituicao(hospitalBraga,S).
no
| ?- listing(prestador).
prestador(1, joao, cardiologo, hospitalViana).
prestador(2, fernando, oftalmologia, hospitalPorto).
prestador(32, jose, ginecologia, hospitalBraga).
prestador(18, francisco, cardilogia, hospitalBraga).
prestador(33, joao, ortopedico, hospitalPorto).
prestador(28, jorge, storinolaringologo, hospitalLisboa).

yes
| ?- listing(utente).
utente(3, joao, 21, braga).
utente(5, angelo, 22, braga).
utente(7, bruno, 22, viana).
utente(9, pedro, 21, barcelos).
utente(20, luis, 34, lisboa).
utente(21, josefina, 25, faro).

yes
| ?- listing(cuidado).
cuidado(1-3-2015, 5, 14, enfartedomicardio, 60).
cuidado(15-12-2014, 7, 28, tmapanofurado, 45).
cuidado(9-8-2016, 5, 14, colocacaopacemaker, 52).
cuidado(17-5-2013, 3, 23, criseexistencial, 50).
cuidado(23-1-2018, 21, 32, remocadoutero, 65).
cuidado(23-1-2018, 21, 32, remocadoutero, 65).
cuidado(3-6-2011, 20, 5, entorce, 41).

yes
| ?- cuidadosInstituicao(hospitalBraga,S).
S = [[(1-3-2015,5,14,enfartedomicardio,60),(9-8-2016,5,14,colocacaopacemaker,52),(23-1-2018,21,32,remocadoutero,65)]]
yes
| ?- ■

```

Figure 23: Predicado cuidadosInstituicao

```

| ?- reconsult( '/home/arleira/Transferências/ISRCR_Fase1_grupo24-1.pl' ).
% consulting /home/arleira/Transferências/ISRCR_Fase1_grupo24-1.pl...
% consulted /home/arleira/Transferências/ISRCR_Fase1_grupo24-1.pl in module user
, 10 msec 2432 bytes
yes
| ?- utenteMorada(braga,S).
S = [3,5] ?
yes
| ?- listing(utente).
utente(3, joao, 21, braga).
utente(5, angelo, 22, braga).
utente(7, bruno, 22, viana).
utente(9, pedro, 21, barcelos).
utente(20, luis, 34, lisboa).
utente(21, josefina, 25, faro).

yes
| ?- utenteMorada(braga,S).
S = [3,5] ?
yes
| ?- ■

```

Figure 24: Predicado utenteMorada

```

cuidado(15-12-2014, 7, 29, timplanofurado, 45), 
cuidado(9-8-2010, 5, 14, colocacaopacemaker, 52). 
cuidado(17-5-2013, 3, 23, criseexistencial, 50). 
cuidado(16-2-2014, 9, 2, testeovario, 15). 
cuidado(3-6-2011, 20, 5, entorce, 41).
yes
| ?- cuidadosInstituicao(hospitalBraga,S).
S = [(1-3-2015, 5, 14, enfartedomicardio, 60), (9-8-2016, 5, 14, colocacaopacemaker, 52), (23-1-2018, 21, 32, remocaoadoutero, 65)] ?
yes
| ?- listing(prestador).
prestador(5, tiago, ortopedia, hospitalViana).
prestador(1, fernando, cardilogia, hospitalPorto).
prestador(32, lise, ginecologia, hospitalBraga).
prestador(14, francisco, cardilogia, hospitalBraga).
prestador(23, manuel, psiquiatria, hospitalPorto).
prestador(28, jorge, otorrinolaringologia, hospitalLisboa).
yes
| ?- listing(utente).
utente(3, joao, 21, braga).
utente(5, angelo, 22, braga).
utente(1, fernando, 23, porto).
utente(9, pedro, 21, barcelos).
utente(20, luis, 34, lisboa).
utente(21, josefina, 25, faro).
utente(21, josefina, 25, faro).
yes
| ?- listing(cuidado).
cuidado(1-3-2015, 5, 14, enfartedomicardio, 60).
cuidado(15-12-2014, 7, 28, timplanofurado, 45).
cuidado(9-8-2010, 5, 14, colocacaopacemaker, 52).
cuidado(17-5-2013, 3, 23, criseexistencial, 50).
cuidado(16-2-2014, 9, 2, testeovario, 15).
cuidado(3-6-2011, 20, 5, entorce, 41).
yes
| ?- cuidadosData(17-5-2013,S).
S = [(17-5-2013, 3, 23, criseexistencial, 50)] ?
yes
| ?-

```

Figure 25: Predicado cuidadosData

4.3.6 Predicado Identificar os utentes de um prestador/especialidade/instituição (Alínea 6)

No primeiro predicado (utentesDeUmPrestador) a estratégia passa por encontrar todos os cuidados realizados pelo prestador ao utente, extrair os ids dos utentes atendidos pelo prestador e posteriormente remover os repetidos.

Já no predicado utentesEspecialidade são invocados 2 predicados: prestadorEspecialidade (onde invocados o predicado findall que nos vai fornecer uma lista com os Ids dos prestadores de uma especialidade) e utentesPrestador (que concatena as listas dos ids dos utentes de cada prestador). Através destes dois predicados conseguimos gerar a lista dos utentes de uma especialidade fornecida.

Por último, o predicado utentesInstituicao convoca outros dois predicados (à semelhança do anterior): prestadorInstituicao (que usa o predicado findall para gerar gerar uma lista dos ids dos prestadores de uma determinada instituição) e o utentesPrestador. Através destes é possível gerar a lista pretendida.

```

%----- -
% Fornece os utentes de um prestador
% Extensao do predicado utentesDeUmPrestador: I,S -> {V,F}

utentesDeUmPrestador(I,S) :-
    findall(X,cuidado(A,X,I,D,E),NL),
    removerRepetidos(NL,U),
    uPaux(U,S).

uPaux([],[]).
uPaux([H|T],S) :-
    findall((H,Y,W,Z),utente(H,Y,W,Z),NL),
    concatenar(N,NL,S),
    uPaux(T,N).

%----- -
% Fornece os Ids dos prestadores de uma dada especialidade
% Extensao do predicado prestadorEspecialidade: Especialidade,S -> {V,F}

prestashoporEspecialidade(I,S) :-
    findall(X,prestashopor(X,B,I,D),S).

%----- -
% Fornece a lista de utentes de uma especialidade
% Extensao do predicado utentesEspecialidade: I,S -> {V,F}

utentesEspecialidade(I,S) :-
    prestadorEspecialidade(I,X),
    utentesPrestador(X,U),
    removerRepetidos(U,Y),
    uEaux(Y,S).

uEaux([],[]).
uEaux([H|T],S) :-
    findall((H,Y,Z,W),utente(H,Y,Z,W),NL),
    concatenar(N,NL,S),
    uEaux(T,N).

%----- -
% Fornece a lista dos Ids dos utentes que receberam cuidados dos prestadores cujos Ids estão contidos no Input
% Extensao do predicado utentesPrestador: ListaPrestadores,S -> {V,F}

utentesPrestador([],[]).
utentesPrestador([H|T],S) :-
    findall(X,cuidado(A,X,H,D,E),NL),
    concatenar(N,NL,S),
    utentesPrestador(T,N).

```

Figure 26: Predicado utentesDeUmPrestador, uPaux, prestadorEspecialidade, utentesEspecialidade, uEaux, utentesPrestador

```

%----- -
% Fornece a lista de utentes de uma instituição
% Extensão do predicado utentesInstituicao: I,S -> {V,F} 6

utentesInstituicao(I,S) :-
    prestadoresInstituicao(I,X),
    utentesPrestador(X,U),
    removerRepetidos(U,L),
    uIaux(L,S).

uIaux([],[]).
uIaux([H|T],S) :-
    findall((H,Y,W,Z),utente(H,Y,Z,W),NL),
    concatenar(N,NL,S),
    uIaux(T,N).

%----- -
% Fornece lista dos IDs dos prestadores da instituição em questão
% Extensão do predicado prestadoresInstituicao: Instituicao,S -> {V,F}

prestadoresInstituicao(I,S) :-
    findall(X,prestador(X,Y,Z,I),S).

```

Figure 27: Predicado utentesInstituicao, uIaux, prestadoresInstituicao

- Output

```

arieira@arieira: ~/Desktop/Mlei18/src/sicstus/bin
prestador(2, fernando, oftamologia, hospitalPorto).
prestador(32, jose, ginecologia, hospitalBraga).
prestador(14, francisco, cardiolgia, hospitalBraga).
prestador(23, manuel, psiquiatria, hospitalPorto).
prestador(28, jorge, otorrinolaringologia, hospitalLisboa).

yes
| ?- listing(cuidados).
+ listing(user:cuidados) - no matching predicate
yes
| ?- listing(utente),
utente(3, joao, 21, braga),
utente(5, angelo, 22, braga),
utente(7, bruno, 22, viana),
utente(9, pedro, 21, barcelos),
utente(20, luis, 34, lisboa),
utente(21, josefina, 25, faro).

yes
| ?- listing(prestador).
prestador(5, tiago, ortopedia, hospitalViana).
prestador(2, fernando, oftamologia, hospitalPorto).
prestador(32, jose, ginecologia, hospitalBraga).
prestador(14, francisco, cardiolgia, hospitalBraga).
prestador(23, manuel, psiquiatria, hospitalPorto).
prestador(28, jorge, otorrinolaringologia, hospitalLisboa).

yes
| ?- listing(cuidado),
cuidado(1-3-2015, 7, 14, enfartedomicardio, 60),
cuidado(15-12-2014, 7, 28, timpanofurado, 45),
cuidado(9-8-2016, 5, 14, colocacaopacemaker, 52),
cuidado(17-5-2013, 3, 23, criseexistencial, 56),
cuidado(23-1-2018, 21, 32, remocadoutero, 65),
cuidado(16-2-2014, 9, 2, testedevisao, 15),
cuidado(3-6-2011, 20, 5, entorce, 41).

yes
| ?- utentesDeUmPrestador(14,S).
S = [[idsDosUtentes, S]] ?
yes
| ?-

```

Figure 28: Predicado Utentes de um determinado Prestador

```

arieira@arieira: ~/Desktop/Mlei18/src/sicstus/bin
yes
| ?- listing(prestador),
prestador(5, tiago, ortopedia, hospitalViana),
prestador(2, fernando, oftamologia, hospitalPorto),
prestador(32, jose, ginecologia, hospitalBraga),
prestador(14, francisco, cardiolgia, hospitalBraga),
prestador(23, manuel, psiquiatria, hospitalPorto),
prestador(28, jorge, otorrinolaringologia, hospitalLisboa).

yes
| ?- listing(cuidado),
cuidado(1-3-2015, 5, 14, enfartedomicardio, 60),
cuidado(15-12-2014, 7, 28, timpanofurado, 45),
cuidado(9-8-2016, 5, 14, colocacaopacemaker, 52),
cuidado(17-5-2013, 3, 23, criseexistencial, 56),
cuidado(23-1-2018, 21, 32, remocadoutero, 65),
cuidado(16-2-2014, 9, 2, testedevisao, 15),
cuidado(3-6-2011, 20, 5, entorce, 41).

yes
| ?- utentesInstituicao(hospitalBraga,S).
S = [5,5,21] ?
yes
| ?-

```

Figure 29: Predicado Utentes de uma determinada Instituição

```

arieira@arieira: ~/Desktop/MieI18/src/sicstus/bin
% consulting /home/arieira/Transferências/SRCR_Fase1_grupo24.pl...
% consulted /home/arieira/Transferências/SRCR_Fase1_grupo24.pl in module user, 0
  msec -304 bytes
yes
| ?- listing(prestador).
prestador(5, tiago, ortopedia, hospitalViana).
prestador(2, fernando, oftalmologia, hospitalPorto).
prestador(32, jose, ginecologia, hospitalBraga).
prestador(14, francisco, cardiologia, hospitalBraga).
prestador(23, manuel, psiquiatria, hospitalPorto).
prestador(28, jorge, otorrinolaringologia, hospitalLisboa).

yes
| ?- listing(cuidado).
cuidado(1-3-2015, 5, 14, enfartedomlocardio, 60).
cuidado(15-12-2014, 7, 28, timpanofurado, 45).
cuidado(9-8-2016, 5, 14, colocacaopacemaker, 52).
cuidado(17-5-2013, 3, 23, criseexistencial, 56).
cuidado(23-1-2018, 21, 32, remocaodoutero, 65).
cuidado(16-2-2014, 9, 2, testedevisao, 15).
cuidado(3-6-2011, 20, 5, entorce, 41).

yes
| ?- listing(utente).
utente(3, joao, 21, braga).
utente(5, angelo, 22, braga).
utente(7, bruno, 22, viana).
utente(9, pedro, 21, barcelos).
utente(20, luis, 34, lisboa).
utente(21, josefina, 25, faro).

yes
| ?- utentesEspecialidade(cardiologia,S).
S = [(5,angelo,22,braga)] ?
yes
yes
| ?-

```

Figure 30: Predicado Utentes de uma determinada Especialidade

4.3.7 Predicado Identificar cuidados de saúde realizados por utente/instituição/prestador (Alínea 7)

Neste parâmetro aplicou-se a mesma estratégia tanto no predicado cuidadosUtente (cuidados de saúde realizados por um utente) tanto no cuidadosPrestador (cuidados de saúde fornecidos por um prestador), fornece-se um utente ou prestador ao predicado findall e o cuidado com o mesmo id do utente ou prestador. Obtem-se como resultado a lista dos cuidados de saúde pretendida.

```
%----- %
% Fornece a lista dos cuidados de que o utente usufruiu
% Extensão do predicado cuidadosUtente: IdUtente,S -> {V,F}

cuidadosUtente(I,S) :-
    findall((A,I,C,D,E),cuidado(A,I,C,D,E),S).

%----- %
% Fornece a lista de cuidados realizados por uma instituição
% Extensão do predicado cuidadosInstituicao: Instituição,S -> {V,F}

cuidadosInstituicao(I,S) :-
    prestatoresInstituicao(I,X),
    icuidados(X,S).

icuidados([],[]).
icuidados([H|T],S) :-
    findall((X,Y,H,W,Z),cuidado(X,Y,H,W,Z),NL),
    concatenar(N,NL,S),
    icuidados(T,N).

%----- %
% Fornece a lista de cuidados realizados por um prestador
% Extensão do predicado cuidadosPrestador: IdPrestador,S -> {V,F}

cuidadosPrestador(I,S) :-
    findall((A,B,I,D,E),cuidado(A,B,I,D,E),S).
```

Figure 31: Predicado cuidadoUtente, cuidadoPrestador

- Output

```

arieira@arieira: ~/Desktop/MieI18/srcr/sicstus/bin
| ?- utentesEspecialidade(psicopatologia,S).
S = [] ?
yes
| ?- utentesInstituicao(hospitalBraga,S).
S = [5,5,21] ?
yes
| ?- cuidadoUtente(S,S).
S = [(1-3-2015,5,14,infarto,60),(9-8-2016,5,14,colocacaopacemaker,52)]
yes
| ?- listing(cuidado).
cuidado(1-3-2015, 5, 14, infarto, 60).
cuidado(15-12-2014, 7, 28, timpanofurado, 45).
cuidado(9-8-2016, 5, 14, colocacaopacemaker, 52).
cuidado(17-5-2013, 3, 23, criseexistencial, 56).
cuidado(23-1-2018, 21, 32, remoçãodoutero, 65).
cuidado(16-2-2014, 9, 2, testedevisao, 15).
cuidado(3-6-2011, 20, 5, entorce, 41).

yes
| ?- cuidadoUtente(9,S).
S = [(16-2-2014,9,2,testedevisao,15)] ?
yes
| ?-

```

Figure 32: Predicado cuidado de saude realizador por utente

```

arieira@arieira: ~/Desktop/MieI18/srcr/sicstus/bin
Prolog interruption (h for help)? h
Prolog interrupt options:
  a      abort      - cause abort
  b      break      - cause break
  c      continue   - do nothing
  e      exit       - cause exit
  d      debug      - start leaping
  z      zip        - start zipping
  t      trace      - start creeping
  h      help       - get this list
Prolog interruption (h for help)? c
| ?- listing(prestador).
prestador(5,tiago,ortopedia,hospitalViana).
prestador(2,fernando,oftamologia,hospitalPorto).
prestador(32,jose,ginecologia,hospitalBraga).
prestador(14,francisco,cardiologia,hospitalBraga).
prestador(23,manuel,psiquiatria,hospitalPorto).
prestador(28,jorge,otorrinolaringologia,hospitalLisboa).

yes
| ?- cuidadoPrestador(32,S).
S = [(23-1-2018,21,32,remoçãodoutero,65)] ?
yes
| ?-

```

Figure 33: Predicado cuidado de saude realizador por prestador

4.3.8 Predicado Determinar todas as instituições/prestadores a que um utente já recorreu (Alínea 8)

No primeiro predicado (prestadoresUtente), no caso de um utente recorrer a um prestador invocamos o predicado findall que extrai os Ids dos prestadores que atenderam o cliente e posteriormente é invocado o predicado pUaux de maneira a listar toda a informação sobre os prestadores em questão.

O predicado instituicoesUtente pretende determinar todas as instituições a que um utente já recorreu. Iremos receber um id de utente e irá ser devolvido uma lista com as instituições. A nossa estratégia foi a seguinte: primeiro invocamos o predicado idPrestadoresUtente de onde obtivemos os Ids dos prestadores que atenderam o utente e de seguida invocamos o predicado iu de modo a listarmos as instituições às quais os prestadores em questão pertencem.

```

%----- %
% Fornece lista de instituições a que um utente recorreu
% Extensão do predicado instituicoesUtente: IdUtente,S -> {V,F}

instituicoesUtente(I,S) :-
    idPrestadoresUtente(I,X),
    iu(X,S).

iu([],[]).
iu([H|T],S) :-
    findall(X,prestador(H,B,C,X),NL),
    concatenar(N,NL,S),
    iu(T,N).

%----- %
% Fornece lista de IDs dos prestadores a que um utente recorreu
% Extensão do predicado idPrestadoresUtente: IdUtente,S -> {V,F}

idPrestadoresUtente(I,S) :-
    findall(X,cuidado(A,I,X,D,E),U),
    removerRepetidos(U,S).

%----- %
% Fornece lista de prestadores a que um utente recorreu
% Extensão do predicado prestadoresUtente: IdUtente,S -> {V,F}

prestadoresUtente(I,S) :-
    findall(X,cuidado(A,I,X,D,E),NL),
    removerRepetidos(NL,U),
    pUaux(U,S).

pUaux([],[]).
pUaux([H|T],S) :-
    findall((H,Y,W,Z),prestador(H,Y,W,Z),NL),
    concatenar(N,NL,S),
    pUaux(T,N).

```

Figure 34: Predicado prestadoresUtente, instituicoesUtente, iu, idPrestadoresUtente

- Output

```

arieira@arieira:~/Desktop/Mlei18/srcr/sicstus/bin
arieira@arieira:~/Desktop/Mlei18/srcr/sicstus$ cd sicstus
arieira@arieira:~/Desktop/Mlei18/srcr/sicstus$ cd bin
arieira@arieira:~/Desktop/Mlei18/srcr/sicstus/bin$ ./sicstus
SICStus 4.3.0 (x86_64-linux-glibc2.12): Thu May  8 05:34:25 PDT 2014
Licensed to studentSP4.3di.uninho.pt
| ?- consult( '/home/arieira/Desktop/SRCR1-GRUPO24/SRCR_Fase1_grupo24.pl' ). 
% consulting /home/arieira/Desktop/SRCR1-GRUPO24/SRCR_Fase1_grupo24.pl...
% consulted /home/arieira/Desktop/SRCR1-GRUPO24/SRCR_Fase1_grupo24.pl in module
user, 20 msec 165000 bytes
yes
| ?- listing(cuidado).
cuidado(1-3-2015, 5, 14, enfartedomicardio, 60).
cuidado(15-12-2014, 7, 28, timpanofurado, 45).
cuidado(9-8-2016, 5, 14, colocacaopacemaker, 52).
cuidado(17-5-2013, 3, 23, criseexistencial, 56).
cuidado(23-1-2018, 21, 32, remocaoadoutero, 65).
cuidado(16-2-2014, 9, 2, testedevisao, 15).
cuidado(3-6-2011, 20, 5, entorce, 41).

yes
| ?- prestadoresUtente(21,S).
S = [(32,jose,podologta,hospitalBraga)] ?
yes
| ?-

```

Figure 35: Predicado Prestadores de um Utente

```

arieira@arieira:~/Desktop/Mlei18/srcr/sicstus/bin
yes
| ?- listing(prestashop).
prestashop(5, tiago, ortopedia, hospitalViana).
prestashop(2, fernando, oftamologia, hospitalPorto).
prestashop(32, jose, ginecologia, hospitalBraga).
prestashop(14, francisco, cardiologia, hospitalBraga).
prestashop(23, manuel, psiquiatria, hospitalPorto).
prestashop(28, jorge, otorrinolaringologia, hospitalLisboa).

yes
| ?- listing(cuidado).
cuidado(1-3-2015, 5, 14, enfartedomicardio, 60).
cuidado(15-12-2014, 7, 28, timpanofurado, 45).
cuidado(9-8-2016, 5, 14, colocacaopacemaker, 52).
cuidado(17-5-2013, 3, 23, criseexistencial, 56).
cuidado(23-1-2018, 21, 32, remocaoadoutero, 65).
cuidado(16-2-2014, 9, 2, testedevisao, 15).
cuidado(3-6-2011, 20, 5, entorce, 41).

yes
| ?- instituicoesUtente(3,S).
S = [hospitalPorto] ?
yes
| ?-

```

Figure 36: Predicado Instituições de um Utente

4.3.9 Predicado Calcular o custo total dos cuidados de saúde por utente/especialidade/prestador/datas (Alínea 9)

- Custo total numa determinada Data (custoData) : Este predicado tem como objetivo calcular o custo total dos cuidados por data. Para isto foi necessário recebermos uma data, e a partir dessa mesma, através do predicado findall extraí-mos os custos de todos os cuidados relativos a essa data, sendo a sua soma calculada de seguida pelo predicado somatorio.
- Custo total de um Utente (custoUtente) : Neste predicado pretendemos calcular o custo total dos cuidados por id de utente. Foi utilizado exatamente o mesmo raciocínio que no anterior, sendo a única diferença na utilização da variável que iremos receber, sendo neste caso o id de utente (I).

- Custo total de um Prestador (custoPrestador) : Neste predicado também usamos exatamente o mesmo raciocínio do anterior, mas em vez de passarmos o id de utente, fornecemos o id do prestador.
- Custo total de uma Especialidade (custoEspecialidade) : Para este predicado será necessário calcular o custo total dos cuidados por especialidade. Vamos receber uma especialidade (I), e devolver o custo total (S). A nossa primeira tarefa foi recorrermos a outro predicado chamado prestadoresEspecialidade, sendo devolvidos todos os ids de prestador da especialidade em questão, de seguida recorremos a outro predicado chamada cEaux, predicado este que agrupa numa lista todos os custos dos cuidados referentes aos prestadores previamente obtidos sendo, estes custos finalmente processados pelo predicado somatorio de modo a obtermos o resultado pretendido.

```

%----- %
% Custo total associado a um utente
% Extensao do predicado custoUtente: IdUtente,S -> {V,F}

custoUtente(I,S) :-
    findall(E,cuidado(A,I,C,D,E),NL),
    somatorio(NL,S).

%----- %
% Fornece lista dos ids dos prestadores de uma dada especialidade
% Extensao do predicado prestadoresEspecialidade: Especialidade,S -> {V,F}

prestadoresEspecialidade(I,S) :-
    findall(X,prestashop(X,B,I,D),S).

%----- %
% Custo total associado a uma especialidade
% Extensao do predicado custoEspecialidade: Especialidade,S -> {V,F}

custoEspecialidade(I,S) :-
    prestadoresEspecialidade(I,X),
    cEaux(X,Y),
    somatorio(Y,S).

cEaux([],[]).
cEaux([H|T],S) :-
    findall(X,cuidado(A,B,H,D,X),NL),
    concatenar(N,NL,S),
    cEaux(T,N).

%----- %
% Custo total associado a um prestador
% Extensao do predicado custoPrestador: IdPrestador,S -> {V,F}

custoPrestador(I,S) :-
    findall(X,cuidado(A,B,I,D,X),NL),
    somatorio(NL,S).

%----- %
% Custo total associado a uma data
% Extensao do predicado custoData: Data,S -> {V,F}

custoData(I,S) :-
    findall(X,cuidado(I,B,C,D,X),NL),
    somatorio(NL,S).

```

Figure 37: Predicado custoUtente, prestadorEspecialidade, custoEspecialidade, cEaux, custoPrestador, custoData

- Output

```

arielra@arielra: ~/Desktop/MieI18/src/sicstus/bin
cuidado(9-8-2016, 5, 14, colocacaopacemaker, 52).
cuidado(17-5-2013, 3, 23, criseexistencial, 56).
cuidado(23-1-2018, 21, 32, remocaoadoutero, 65).
cuidado(16-2-2014, 9, 2, testedevisao, 15).
cuidado(3-6-2011, 20, 5, entorce, 41).

yes
| ?- instituicoesUtente(3,S).
S = [hospitalPorto] ?
yes
| ?- listing(cuidado).
cuidado(1-3-2015, 5, 14, enfartedomiocardio, 60).
cuidado(15-12-2014, 7, 28, timpanofurado, 45).
cuidado(9-8-2016, 5, 14, colocacaopacemaker, 52).
cuidado(17-5-2013, 3, 23, criseexistencial, 56).
cuidado(23-1-2018, 21, 32, remocaoadoutero, 65).
cuidado(16-2-2014, 9, 2, testedevisao, 15).
cuidado(3-6-2011, 20, 5, entorce, 41).

yes
| ?- custoUtente(5,S).
S = 112 ?
yes
| ?- 
```

Figure 38: Predicado custo total dos cuidados de um Utente

```

arielra@arielra: ~/Desktop/MieI18/src/sicstus/bin
yes
| ?- listing(prestador).
prestador(5, tiago, ortopedia, hospitalViana).
prestador(2, fernando, oftamologia, hospitalPorto).
prestador(32, jose, ginecologia, hospitalBraga).
prestador(14, francisco, cardiologia, hospitalBraga).
prestador(23, manuel, psiquiatria, hospitalPorto).
prestador(28, jorge, otorrinolaringologia, hospitalLisboa).

yes
| ?- listing(cuidado).
cuidado(1-3-2015, 5, 14, enfartedomiocardio, 60).
cuidado(15-12-2014, 7, 28, timpanofurado, 45).
cuidado(9-8-2016, 5, 14, colocacaopacemaker, 52).
cuidado(17-5-2013, 3, 23, criseexistencial, 56).
cuidado(23-1-2018, 21, 32, remocaoadoutero, 65).
cuidado(16-2-2014, 9, 2, testedevisao, 15).
cuidado(3-6-2011, 20, 5, entorce, 41).

yes
| ?- custoPrestador(14,S).
S = 112 ?
yes
| ?- 
```

Figure 39: Predicado custo total dos cuidados de um Prestador

```

arielra@arieira:~/Desktop/Mlei18/srcr/sicstus/bin
cuidado(9-8-2016, 5, 14, colocacaopacemaker, 52).
cuidado(17-5-2013, 3, 23, criseexistencial, 56).
cuidado(23-1-2018, 21, 32, remocaoadoutero, 65).
cuidado(16-2-2014, 9, 2, testedevisao, 15).
cuidado(3-6-2011, 20, 5, entorce, 41).

yes
| ?- custoPrestador(14,S).
S = 112 ?
yes
| ?- listing(cuidado).
cuidado(1-3-2015, 5, 14, enfartedomiocardio, 60).
cuidado(15-12-2014, 7, 28, timpanofurado, 45).
cuidado(9-8-2016, 5, 14, colocacaopacemaker, 52).
cuidado(17-5-2013, 3, 23, criseexistencial, 56).
cuidado(23-1-2018, 21, 32, remocaoadoutero, 65).
cuidado(16-2-2014, 9, 2, testedevisao, 15).
cuidado(3-6-2011, 20, 5, entorce, 41).

yes
| ?- custoData(1-3-2015,S).
S = 60 ?
yes
| ?-

```

Figure 40: Predicado custo total dos cuidados de uma determinada Data

```

arielra@arieira:~/Desktop/Mlei18/srcr/sicstus/bin
% consulting /home/arieira/Transferências/SRCR_Fase1_grupo24.pl...
% consulted /home/arieira/Transferências/SRCR_Fase1_grupo24.pl in module user, 0
msec -176 bytes
yes
| ?- listing(cuidados).
* listing(user:cuidados) - no matching predicate
yes
| ?- reconsult( '/home/arieira/Transferências/SRCR_Fase1_grupo24.pl' ).
% consulting /home/arieira/Transferências/SRCR_Fase1_grupo24.pl...
% consulted /home/arieira/Transferências/SRCR_Fase1_grupo24.pl in module user, 0
msec -176 bytes
yes
| ?- listing(cuidado).
cuidado(1-3-2015, 5, 14, enfartedomiocardio, 60).
cuidado(15-12-2014, 7, 28, timpanofurado, 45).
cuidado(9-8-2016, 5, 14, colocacaopacemaker, 52).
cuidado(17-5-2013, 3, 23, criseexistencial, 56).
cuidado(23-1-2018, 21, 32, remocaoadoutero, 65).
cuidado(16-2-2014, 9, 2, testedevisao, 15).
cuidado(3-6-2011, 20, 5, entorce, 41).

yes
| ?- custoEspecialidade(cardiologia,S).
S = 112 ?
yes
| ?-

```

Figure 41: Predicado custo total dos cuidados de uma determinada Especialidade

4.4 Outros Predicados

Os predicados auxiliares utilizados na resolução deste trabalho prático.

4.4.1 Predicado Teste

O predicado apresentado na figura abaixo testa se os elementos de uma lista são todos verdadeiros e é usado nos predicados evolucao e involucao de modo a garantir que não há inserção nem remoção de informação da base de conhecimento que coloque o sistema num estado de inconsistência, por outras palavras, apenas permite inserções e remoções se os devidos invariantes forem respeitados.

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

Figure 42: Predicado teste

4.4.2 Predicado Comprimento

O seguinte predicado, como o próprio nome indica, permite calcular o comprimento de uma lista. Este cálculo é feito de forma recursiva, isto é, incrementando uma variável em uma unidade sempre que houver um elemento à cabeça da lista, sendo o valor desta variável o resultado final deste predicado.

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

Figure 43: Predicado comprimento

4.4.3 Predicado Concatenar

O predicado concatena, por sua vez, permite concatenar duas listas, L1 e L2, sendo o resultado da concatenação das mesmas devolvido na lista L3.

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

Figure 44: Predicado concatenar

4.4.4 Predicado Somatório

Este predicado permite somar todos os valores de uma lista. Para tal, recebe uma lista com os elementos que se pretende somar e retorna um R que terá o valor da soma dos elementos da lista recebida. Se a lista for uma lista vazia, a soma dos seus valores é igual a zero. Caso contrário, quando a lista recebida não é vazia, o predicado somatorio é chamado recursivamente para a cauda da lista e é somado ao valor resultante de somar os elementos da cauda ao valor que estava ‘a cabeça’ da lista, sendo retornado o valor final.

```
%-----  
% Extensao do predicado somatorio: L,R -> {V,F}  
  
somatorio( [],0 ).  
somatorio( [H|T],R ) :- somatorio(T,N),  
    R is H+N.
```

Figure 45: Predicado somatorio

4.4.5 Predicado Remover Elemento

Este predicado recebe uma lista L1 e um elemento Y e remove da lista todas as ocorrências do elemento, sendo a lista resultante de tal remoção devolvida em L2. Caso não haja ocorrências de Y em L1, a própria é retornada.

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

Figure 46: Predicado removerElemento

4.4.6 Predicado Remover Repetidos

O predicado removerRepetidos recebe uma lista e, com recurso ao predicado acima apresentado(removerElemento), devolve a mesma se esta não tiver duplicações, caso contrário devolve uma nova lista sem qualquer repetições.

```
%-----  
% remove os elementos repetidos de uma lista  
% Extencao do predicado removerRepetidos: L1, L2 -> {V,F}  
  
removerRepetidos( [],[] ).  
removerRepetidos( [X|L],[X|NL] ) :-  
    removerElemento( L,X,TL ), removerRepetidos( TL,NL ).
```

Figure 47: Predicado removerRepetidos

4.4.7 Predicado Não Negativo

Através deste predicado podemos verificar se os elementos de uma lista são todos não negativos, ou seja, todos maiores que zero.

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

Figure 48: Predicado naoNegativo

4.4.8 Predicado Média

Neste predicado auxiliar realiza-se a média de uma lista, isto é, utilizamos o predicado somatorio para realizar a soma de todos os elementos da lista e utilizamos o predicado comprimento para determinar o comprimento da lista.

Por último, dividimos o somatório pelo comprimento da lista e obtemos á média.

```
%-----  
% Faz a media de uma lista  
% Extencao do predicado media: L,R -> {V,F}  
  
media([H|T],S) :-  
    somatorio([H|T],S1),comprimento([H|T],S2), S is S1/S2.
```

Figure 49: Predicado media

5 Novas Funcionalidades

5.1 Média de idades dos utentes

Neste predicado verificamos a invocação de outros dois predicados: primeiro o findall para obtermos a lista das idades de todos os utentes existentes na nossa base de conhecimento e em segundo a media, que recebe a lista gerada anteriormente e devolve o resultado da média das idades dos mesmos.

```
%-----  
% Média de todos os utentes  
% Extensão do predicado mediaIdades_utentes: S -> {V,F}  
  
mediaIdades_utentes(S) :-  
    findall(I,utente(Id,_,I,_),S1),  
    media(S1,S).
```

Figure 50: Predicado mediaIdades_utentes

- Output

```
| ?- reconsult( '/home/arieira/Transferências/SRCR_Fase1_grupo24.pl' ).  
% consulting /home/arieira/Transferências/SRCR_Fase1_grupo24.pl...  
% consulted /home/arieira/Transferências/SRCR_Fase1_grupo24.pl in module user, 1  
0 msec 1344 bytes  
yes  
| ?- mediaIdades_utentes(S).  
S = 24.166666666666668 ?  
yes  
| ?- █
```

Figure 51: Predicado que fornece a média de idades de todos os utentes

5.2 Média de idades por morada

Neste predicado verificamos a invocação de outros dois predicados: primeiro o findall para obtermos as idades dos utentes da morada em questão e em segundo a media que recebe a lista gerada anteriormente e devolve o resultado da média das idades dos mesmos.

```
%-----  
% Dá a média de idades correspondentes aos utentes com a morada dada  
% Extensão do predicado mediaIdades_utentes: Morada,S -> {V,F}  
  
mediaIdade_morada(morada,M,S) :-  
    findall(Id, utente(_,_,Id,M), S1),  
    media(S1,S).
```

Figure 52: Predicado mediaIdades_morada

- Output

```
% consulted /home/arieira/Transferências/SRCR_Fase1_grupo24.pl in module user, 0
  msec 624 bytes
yes
| ?- listing(utente).
utente(3, joao, 21, braga).
utente(5, angelo, 22, braga).
utente(7, bruno, 22, viana).
utente(9, pedro, 21, barcelos).
utente(20, luis, 34, lisboa).
utente(21, josefina, 25, faro).

yes
| ?- mediaIdade_morada(localizacao,braga,S).
S = 21.5 ?
yes
| ?-
```

Figure 53: Predicado que fornece a média de idades dos utentes de uma morada dada

5.3 Número de Prestadores de uma Instituição

Este predicado tem como finalidade calcular o número de utentes numa determinada instituição. Para tal, primeiramente obtivemos os Ids dos prestadores da instituição em questão, de seguida foram agrupados numa lista os Ids dos utentes dos prestadores previamente mencionados e, por fim, após a remoção de repetições nesta lista, é calculado o seu comprimento, ou seja, o número de utentes na instituição em questão.

```
%-----%
% Dá o numero de prestadores correspondente a uma instituição
% Extenso do predicado prestadoresN_Instituicao: Instituicao,S -> {V,F}

prestadoresN_Instituicao(I,S) :-
    findall(P,prestashop(_,P,_,I),R),
    comprimento(R,S).
```

Figure 54: Predicado prestadoresN_Instituicao

- Output

```
| ?- listing(prestador).
prestashop(5, tiago, ortopedia, hospitalViana).
prestashop(2, fernando, oftalmologia, hospitalPorto).
prestashop(32, jose, podologia, hospitalBraga).
prestashop(14, francisco, cardiologia, hospitalBraga).
prestashop(23, manuel, psiquiatria, hospitalPorto).
prestashop(28, jorge, otorrinolaringologia, hospitalLisboa).

yes
| ?- prestadoresN_Instituicao(hospitalBraga,S).
S = 2 ?
yes
| ?-
```

Figure 55: Predicado que fornece o número de prestadores de uma instituição

5.4 Número de Utentes de uma Instituição

Neste predicado, tem como funcionalidade, dar o numero de utentes que tem registo numa determinada instituição. Para tal, tivemos de recorrer aos

prestadores para verificar a instituição e depois aos utentes para fazer o devido somatorio, retirando os repetidos.

```
%-----  
% Fornece o número de utentes por instituição  
% Extensoa do predicado nUtentesInstituicao Instituicao,S -> {V,F}  
  
nUtentesInstituicao(I,S) :-  
    prestadoresInstituicao(I,X),  
    utentesPrestador(X,U),  
    removerRepetidos(U,L),  
    comprimento(L,S).
```

Figure 56: Predicado nUtentesInstituicao

- Output

```
| ?- listing(cuidado).  
cuidado(1-3-2015, 5, 14, enfartedomiocardio, 60).  
cuidado(15-12-2014, 7, 28, timpanofurado, 45).  
cuidado(9-8-2016, 5, 14, colocacaopacemaker, 52).  
cuidado(17-5-2013, 3, 23, criseexistencial, 56).  
cuidado(23-1-2018, 21, 32, remoacaodoutero, 65).  
cuidado(16-2-2014, 9, 2, testedevisao, 15).  
cuidado(3-6-2011, 20, 5, entorce, 41).  
  
yes  
| ?- nUtentesInstituicao(hospitalViana,S).  
S = 1 ?  
yes  
| ?- ■
```

Figure 57: Predicado que fornece o número de utentes de uma instituição

5.5 Número de Cuidados de uma Instituição

Este predicado, dando de entrada uma instituição, fornece o número de cuidados relativos a essa mesma instituição. Para tal foi necessário obter os Ids dos prestadores referentes à instituição em causa, o que nos permitiu, obter o número de cuidados da instituição em questão.

```
%-----  
% Fornece o número de cuidados por instituição  
% Extensoa do predicado nCuidadosInstituicao Instituicao,S -> {V,F}  
  
nCuidadosInstituicao(I,S) :-  
    prestadoresInstituicao(I,X),  
    nCuidadosPrestador(X,C),  
    comprimento(C,S).  
  
nCuidadosPrestador([],[]).  
nCuidadosPrestador([H|T],S) :-  
    findall(X,cuidado(X,B,H,D,E),NL),  
    concatenar(N,NL,S),  
    nCuidadosPrestador(T,N).
```

Figure 58: Predicado nCuidadosInstituicao

- Output

```
| ?- ncuidadosInstituicao(hospitalPorto,S).  
S = 2 ?  
yes  
| ?-
```

Figure 59: Predicado que fornece o número de Cuidados de uma instituição

6 Conclusão e Trabalho Futuro

Este primeiro trabalho prático consistiu em aplicar os nossos conhecimentos da linguagem de programação em lógica PROLOG, para a resolução de um problema proposto e representação de conhecimento fiável, usando para isso mecanismos e metodologias de forma a permitir retirar informações semanticamente corretas e realizar com elas raciocínios de forma segura..

Para iniciar o trabalho, foi necessário fazer um estudo prévio e superficial sobre o tema em causa, sendo que, partindo disso, conseguimos construir uma base de conhecimento sólida que nos permitiu implementar todas as funcionalidades propostas.

Com base nas aulas práticas da disciplina, podemos ”reutilizar” código que nos foi útil na implementação das funções auxiliares. Além disso, também nos permitiu implementar funcionalidades extras que achamos que se enquadrariam no contexto do exercício propostos.

Para cada funcionalidade definida fizemos o respetivo teste no SICSTUS e verificamos que tudo o que implementamos apresentava o resultado correto e esperado, confirmando-se assim a solidez do trabalho prático.

Assim, podemos dizer que estamos satisfeitos com o trabalho realizado, uma vez que foram cumpridos todos os requisitos e tivemos a oportunidade de aumentar o nosso conhecimento neste tipo de programação o que facilita a realização de futuros projetos.