



Universidade do Minho
Escola de Engenharia

Sistemas de Representação de Conhecimento e Raciocínio

Avaliação em Grupo - 1^a Fase

Grupo 20

Guilherme Martins a89532 - Jaime Oliveira a89598 - João
Pereira a89607 - José Costa a89519 - Tiago Freitas a89570



a89532

a89598

a89607

a89519

a89570

Mestrado Integrado em Engenharia Informática

3^o Ano - 2^o Semestre

1 Resumo

Este relatório foi desenvolvido no âmbito da realização do primeiro trabalho da disciplina de Sistema de Representação de Conhecimento e Raciocínio.

Este trabalho apresenta como principal desafio a utilização do *PROLOG* como a principal linguagem para a realização de programação em lógica, como também para o recurso a Invariantes.

Índice

1	Resumo	1
2	Introdução	6
3	Preliminares	7
4	Descrição do Trabalho e Análise de Resultados	8
4.1	Predicados de Evolução de Conhecimento	8
4.2	Invariantes	8
4.2.1	Estruturais	9
4.2.2	Referenciais	15
4.3	Predicados Gerais	22
4.4	Definições	25
4.5	Base de Conhecimento Inicial	25
5	Funcionalidades	27
5.1	Definição de fases de vacinação	27
5.1.1	Registo	28
5.1.2	Verificar se um utente é candidato a uma fase de vacinação	28
5.1.3	Lista de candidatos a uma determinada fase de vacinação	29
5.2	Identificar pessoas não vacinadas	29
5.2.1	Verificar se um utente não foi vacinado	29
5.2.2	Lista pessoas não vacinadas	29
5.3	Identificar pessoas vacinadas	32
5.3.1	Verificar se um utente já foi vacinado	32
5.3.2	Lista de pessoas vacinadas	32
5.4	Identificar pessoas vacinadas indevidamente	33
5.4.1	Verificar se um utente está vacinado indevidamente . .	33
5.4.2	Lista de utentes vacinados indevidamente	35
5.5	Identificar pessoas não vacinadas e que são candidatas a vacinação	36
5.5.1	Verificar se um utente não foi vacinado e é candidato a uma determinada fase	36
5.5.2	Lista de utentes não vacinados e candidatos a uma determinada fase	36
5.6	Identificar pessoas a quem falta a segunda toma da vacina . .	38
5.6.1	Lista de pessoas a quem falta a segunda toma	38
6	Funcionalidades Extra	39
6.1	Lista das pessoas vacinadas num determinado centro de saúde	39
6.2	Lista das diferentes vacinas dadas num determinado centro de saúde	39
6.3	Lista das pessoas que tomaram uma determinada vacina . . .	39

6.4	Lista das pessoas que receberam a vacina por uma determinada pessoa do staff	40
6.5	Lista das pessoas que têm a vacinação completa	40
7	Conclusão	41

Figuras

1	Predicado inserir	8
2	Predicado evolucao	8
3	Invariante - Utente (sem ID repetido)	9
4	Output: Invariante - Utente (sem ID repetido)	9
5	Invariante - Utente (sem NSS repetido)	9
6	Output: Invariante - Utente (sem NSS repetido)	10
7	Invariante - Centro de Saúde (sem ID repetido)	10
8	Output: Invariante - Centro de Saúde (sem ID repetido)	10
9	Invariante - Centro de Saúde (sem telefone repetido)	11
10	Output: Invariante - Centro de Saúde (sem telefone repetido)	11
11	Invariante - Centro de Saúde (sem email repetido)	11
12	Output: Invariante - Centro de Saúde (sem email repetido)	12
13	Invariante - Staff (sem ID repetido)	12
14	Output: Invariante - Staff (sem ID repetido)	13
15	Invariante - Staff (sem email repetido)	13
16	Output: Invariante - Staff (sem email repetido)	14
17	Invariante - Vacinação (toma válida)	14
18	Output: Invariante - Vacinação (toma válida)	15
19	Invariante - Utente (Centro de Saúde existente)	15
20	Output: Invariante - Utente (Centro de Saúde existente)	15
21	Invariante - Staff (Centro de Saúde existente)	16
22	output: Invariante - Staff (Centro de Saúde existente)	16
23	Invariante - Vacinação (Staff existente)	17
24	Output: Invariante - Vacinação (Staff existente)	17
25	Invariante - Vacinação (Utente existente)	17
26	Output: Invariante - Vacinação (Utente existente)	18
27	Invariante - Vacinação (Primeira dose existente)	18
28	Output: Invariante - Vacinação (Primeira dose existente)	19
29	Invariante - Vacinação (Duas doses iguais)	19
30	Output: Invariante - Vacinação (Duas doses iguais)	20
31	Invariante - Vacinação (Sem vacinação antes da primeira dose)	20
32	Output: Invariante - Vacinação (Sem vacinação antes da primeira dose)	21
33	Invariante - Vacinação (Segunda dose depois da primeira)	21
34	Output: Invariante - Vacinação (Segunda dose depois da primeira)	22
35	teste	22
36	mostrarRegistos	22
37	date	23
38	anterior	23
39	idade	23
40	solucoes	23
41	solucoesSRep	24

42	comprimento	24
43	pertence	24
44	head	24
45	append	24
46	nao	25
47	si	25
48	Definições	25
49	Utentes	26
50	Centros de Saúde	26
51	Staff	26
52	Vacinações	26
53	Registar	28
54	Candidato à fase 1	29
55	Candidato à fase 2	29
56	Candidato à fase 3	30
57	Output: Candidatos a cada fase	30
58	Listas dos candidatos a cada fase	30
59	Output: Listas dos candidatos a cada fase	30
60	Utentes não vacinados	30
61	Output: Utentes não vacinados	31
62	Utentes vacinados	32
63	Output: Utentes vacinados	32
64	Utente vacinado indevidamente	33
65	Output: Utente vacinado indevidamente	34
66	Lista de utentes vacinados indevidamente	35
67	Lista de utentes vacinados indevidamente	35
68	Utentes não vacinados e candidatos a cada uma das fases	36
69	Output: Utentes não vacinados e candidatos a cada uma das fases	37
70	Utentes a quem falta a segunda toma	38
71	Output: Utentes a quem falta a segunda toma	38
72	Predicado da funcionalidade extra 1	39
73	Predicado da funcionalidade extra 2	39
74	Predicado da funcionalidade extra 3	39
75	Predicado da funcionalidade extra 4	40
76	Predicado da funcionalidade extra 5	40

2 Introdução

Neste documento apresentamos a nossa solução para o primeiro trabalho prático, que consiste em desenvolver um sistema de representação de conhecimento e raciocínio (através do uso da linguagem **Prolog**) com capacidade para caracterizar uma base de conhecimento relativa à vacinação global no contexto COVID-19.

Desta forma poderemos *perguntar* ao programa várias informações relevantes dessa área.

Inicialmente indicamos os requisitos iniciais para o desenvolvimento do nosso sistema, na secção **Preliminares**. Posteriormente descrevemos o trabalho, desde os predicados de evolução de conhecimento implementados, até à base de conhecimento inicial, em **Descrição do Trabalho e Análise de Resultados**. Depois apresentamos as **Funcionalidades** mínimas implementadas, e como as utilizar no contexto do nosso sistema, no seguimento disto também apresentamos as **Funcionalidades Extra** criadas pela nossa equipa. Por fim, fazemos algumas conclusões e sugestões sobre o trabalho em equipa, e sobre a implementação final e resultados obtidos em **Conclusões e Sugestões**.

3 Preliminares

Numa primeira fase do desenvolvimento do sistema, a equipa reuniu os requisitos fundamentais para iniciar o processo de implementação do programa.

Em primeiro lugar, verificamos as características dos predicados principais da nossa base de conhecimento. Estes são: utente, centro de saúde, staff e registo de vacinação.

- **utente:** constituído por Idutente, N^o Segurança Social, Nome, Data Nascimento, Email, Telefone, Morada, Profissão, Lista de doenças crónicas e Id do centro de saúde;
- **centro_saude:** constituído por Idcentro, Nome, Morada, Telefone e Email;
- **staff:** constituído por Idstaff, Idcentro, Nome e email;
- **vacinacao_Covid:** constituído por IdStaf, IdUtente, Data, Vacina e Toma.

Posteriormente identificamos os diferentes componentes necessários para implementar um sistema consistente e íntegro. Chegamos à conclusão de que teria de existir uma base de conhecimento predefinida (para possuir dados e testar), com algum povoamento das entidades acima enunciadas; um predicado que nos permitisse evoluir conhecimento; invariantes para garantir a integridade referencial e estrutural; implementar predicados que nos permitissem ter várias funcionalidades e predicados mais gerais e auxiliares para as funções principais.

4 Descrição do Trabalho e Análise de Resultados

4.1 Predicados de Evolução de Conhecimento

```
%-----  
% Inserir predicados  
  
inserir(Termo) :- assert(Termo).  
inserir(Termo) :- retract(Termo), !, fail.
```

Figure 1: Predicado inserir

```
%-----  
% Extensao do predicado que permite a evolucao do conhecimento  
  
evolucao( Termo ) :- solucoes(Invariante,+Termo::Invariante,Lista),  
                    inserir(Termo),  
                    teste(Lista).
```

Figure 2: Predicado evolucao

Estes predicados têm como função apoiar,desenvolver e expandir a nossa base de conhecimento, isto é, são responsáveis por adicionar informação e, além disso, não permitem qualquer outra ação enquanto decorrem, o que torna a base de conhecimento consistente.

4.2 Invariantes

Os invariantes têm uma extrema importância pois permitem termos controlo sobre a informação que entra na nossa base de conhecimento. Neste trabalho foram implementados dois tipos: **estruturais** e **referenciais**. Os invariantes **estruturais** têm como função não permitir a inserção de conhecimento repetido e/ou inválido, enquanto que os **referenciais** bloqueiam a adição de uma entidade que tenciona relacionar-se com outra que nao existe, no momento da inserção, tal como vamos mostrar nos *outputs*.

4.2.1 Estruturais

```
% Utente - Id
+utente(Id,_,_,_,_,_,_,_,_) ::
    (solucoes(Id,
        (utente(Id,_,_,_,_,_,_,_,_)),S),
        comprimento(S,N),
        N == 1).
```

Figure 3: Invariante - Utente (sem ID repetido)

```
?- mostrarRegistos(utente).
:- dynamic utente/10.

utente(1, 123456789, 'Pedro Oliveira', (19, 2, 1934), 'po@gmail.com', 253123451, 'Barcelos', 'Bombeiro', [], 1).
utente(2, 123123123, 'Manuel Faria', (13, 3, 1945), 'mf@gmail.com', 253429351, 'Barcelos', 'Medico', [], 1).
utente(3, 523183123, 'Carla Castro', (2, 12, 1977), 'cc@gmail.com', 253459320, 'Viana do Castelo', 'Jornalista', [], 2).
utente(4, 256331909, 'Roberto Carlos', (21, 1, 1955), 'rc@gmail.com', 253919559, 'Guimarães', 'Engenheiro Informático', ['Hipertensão'], 2).
utente(5, 436329091, 'Rita Neves', (21, 1, 2001), 'rn@gmail.com', 253010123, 'Viana do Castelo', 'Engenheira de Materiais', [], 1).

true.

?- registaUtente(1, 987654321, 'Rui Oliveira', (10, 2, 1956), 'po@gmail.com', 253123451, 'Braga', 'Carpinteiro', [], 1).
false.

?- mostrarRegistos(utente).
:- dynamic utente/10.

utente(1, 123456789, 'Pedro Oliveira', (19, 2, 1934), 'po@gmail.com', 253123451, 'Barcelos', 'Bombeiro', [], 1).
utente(2, 123123123, 'Manuel Faria', (13, 3, 1945), 'mf@gmail.com', 253429351, 'Barcelos', 'Medico', [], 1).
utente(3, 523183123, 'Carla Castro', (2, 12, 1977), 'cc@gmail.com', 253459320, 'Viana do Castelo', 'Jornalista', [], 2).
utente(4, 256331909, 'Roberto Carlos', (21, 1, 1955), 'rc@gmail.com', 253919559, 'Guimarães', 'Engenheiro Informático', ['Hipertensão'], 2).
utente(5, 436329091, 'Rita Neves', (21, 1, 2001), 'rn@gmail.com', 253010123, 'Viana do Castelo', 'Engenheira de Materiais', [], 1).
```

Figure 4: Output: Invariante - Utente (sem ID repetido)

```
% Utente - Nº Segurança Social
+utente(_,Nss,_,_,_,_,_,_,_) ::
    (solucoes(Nss,
        (utente(_,Nss,_,_,_,_,_,_,_)),S),
        comprimento(S,N),
        N == 1).
```

Figure 5: Invariante - Utente (sem NSS repetido)

```
?- mostrarRegistos(utente).
:- dynamic utente/10.

utente(1, 123456789, 'Pedro Oliveira', (19, 2, 1934), 'po@gmail.com', 253123451, 'Barcelos', 'Bombeiro', [], 1).
utente(2, 123123123, 'Manuel Faria', (13, 3, 1945), 'mf@gmail.com', 253429351, 'Barcelos', 'Medico', [], 1).
utente(3, 523183123, 'Carla Castro', (2, 12, 1977), 'cc@gmail.com', 253459320, 'Viana do Castelo', 'Jornalista', [], 2).
utente(4, 256331909, 'Roberto Carlos', (21, 1, 1955), 'rc@gmail.com', 253919559, 'Guimarães', 'Engenheiro Informático', ['Hipertensão'], 2).
utente(5, 436329091, 'Rita Neves', (21, 1, 2001), 'rn@gmail.com', 253010123, 'Viana do Castelo', 'Engenheira de Materiais', [], 1).

true.

?- registaUtente(6, 123456789, 'Rui Oliveira', (10, 2, 1956), 'ro@gmail.com', 253123451, 'Braga', 'Carpinteiro', [], 1).
false.

?- mostrarRegistos(utente).
:- dynamic utente/10.

utente(1, 123456789, 'Pedro Oliveira', (19, 2, 1934), 'po@gmail.com', 253123451, 'Barcelos', 'Bombeiro', [], 1).
utente(2, 123123123, 'Manuel Faria', (13, 3, 1945), 'mf@gmail.com', 253429351, 'Barcelos', 'Medico', [], 1).
utente(3, 523183123, 'Carla Castro', (2, 12, 1977), 'cc@gmail.com', 253459320, 'Viana do Castelo', 'Jornalista', [], 2).
utente(4, 256331909, 'Roberto Carlos', (21, 1, 1955), 'rc@gmail.com', 253919559, 'Guimarães', 'Engenheiro Informático', ['Hipertensão'], 2).
utente(5, 436329091, 'Rita Neves', (21, 1, 2001), 'rn@gmail.com', 253010123, 'Viana do Castelo', 'Engenheira de Materiais', [], 1).
```

Figure 6: Output: Invariante - Utente (sem NSS repetido)

```
% Centro de Saúde - Id
+centro_saude(IdCS,_,_,_) ::
    (solucoes(IdCS,
    (centro_saude(IdCS,_,_,_)),S),
    comprimento(S,N),
    N == 1).
```

Figure 7: Invariante - Centro de Saúde (sem ID repetido)

```
?- mostrarRegistos(centro_saude).
:- dynamic centro_saude/5.

centro_saude(1, 'Centro de saúde de Viana', 'Viana do Castelo', 253456712, 'csv@gmail.com').
centro_saude(2, 'Centro de saúde de Guimarães', 'Guimarães', 253921733, 'csg@gmail.com').
centro_saude(3, 'Centro de saúde de Barcelos', 'Barcelos', 253004239, 'csb@gmail.com').

true.

?- registaCentro(1, 'Centro de Saúde', 'Lisboa', 238761548, 'email').
false.

?- mostrarRegistos(centro_saude).
:- dynamic centro_saude/5.

centro_saude(1, 'Centro de saúde de Viana', 'Viana do Castelo', 253456712, 'csv@gmail.com').
centro_saude(2, 'Centro de saúde de Guimarães', 'Guimarães', 253921733, 'csg@gmail.com').
centro_saude(3, 'Centro de saúde de Barcelos', 'Barcelos', 253004239, 'csb@gmail.com').
```

Figure 8: Output: Invariante - Centro de Saúde (sem ID repetido)

```
% Centro de Saúde - Telefone
+centro_saude(_,_,_,Tel,_) ::
    (solucoes(Tel,
    (centro_saude(_,_,_,Tel,_)),S),
    comprimento(S,N),
    N == 1).
```

Figure 9: Invariante - Centro de Saúde (sem telefone repetido)

```
?- mostrarRegistos(centro_saude).
:- dynamic centro_saude/5.

centro_saude(1, 'Centro de saúde de Viana', 'Viana do Castelo', 253456712, 'csv@gmail.com').
centro_saude(2, 'Centro de saúde de Guimarães', 'Guimarães', 253921733, 'csg@gmail.com').
centro_saude(3, 'Centro de saúde de Barcelos', 'Barcelos', 253004239, 'csb@gmail.com').

true.

?- registaCentro(4, 'Centro de Saúde', 'Lisboa', 253456712, 'email').
false.

?- mostrarRegistos(centro_saude).
:- dynamic centro_saude/5.

centro_saude(1, 'Centro de saúde de Viana', 'Viana do Castelo', 253456712, 'csv@gmail.com').
centro_saude(2, 'Centro de saúde de Guimarães', 'Guimarães', 253921733, 'csg@gmail.com').
centro_saude(3, 'Centro de saúde de Barcelos', 'Barcelos', 253004239, 'csb@gmail.com').
```

Figure 10: Output: Invariante - Centro de Saúde (sem telefone repetido)

```
% Centro de Saúde - Email
+centro_saude(_,_,_,Email) ::
    (solucoes(Email,
    (centro_saude(_,_,_,Email)),S),
    comprimento(S,N),
    N == 1).
```

Figure 11: Invariante - Centro de Saúde (sem email repetido)

```

?- mostrarRegistos(centro_saude).
:- dynamic centro_saude/5.

centro_saude(1, 'Centro de sa de de Viana', 'Viana do Castelo', 253456712, 'csv@gmail.com').
centro_saude(2, 'Centro de sa de de Guimar es', 'Guimar es', 253921733, 'csg@gmail.com').
centro_saude(3, 'Centro de sa de de Barcelos', 'Barcelos', 253004239, 'csb@gmail.com').

true.

?- registaCentro(4, 'Centro de Sa de', 'Lisboa', 253456712, 'csv@gmail.com').
false.

?- mostrarRegistos(centro_saude).
:- dynamic centro_saude/5.

centro_saude(1, 'Centro de sa de de Viana', 'Viana do Castelo', 253456712, 'csv@gmail.com').
centro_saude(2, 'Centro de sa de de Guimar es', 'Guimar es', 253921733, 'csg@gmail.com').
centro_saude(3, 'Centro de sa de de Barcelos', 'Barcelos', 253004239, 'csb@gmail.com').

```

Figure 12: Output: Invariante - Centro de Sa de (sem email repetido)

```

% Staff - Id
+staff(Id,_,_,_) ::
    (solucoes(Id,
        staff(Id,_,_,_),S),
        comprimento(S,N),
        N == 1).

```

Figure 13: Invariante - Staff (sem ID repetido)

```

?- mostrarRegistos(staff).
:- dynamic staff/4.

staff(1, 2, 'Jose Sa', 'js@gmail.com').
staff(2, 1, 'Joao Marques', 'jm@gmail.com').
staff(3, 1, 'Maria Matos', 'm&m@gmail.com').
staff(4, 3, 'Renata Peixoto', 'rp@gmail.com').
staff(5, 2, 'Marta Domingues', 'md@gmail.com').

true.

?- registaStaff(1,3,'Ana','email').
false.

?- mostrarRegistos(staff).
:- dynamic staff/4.

staff(1, 2, 'Jose Sa', 'js@gmail.com').
staff(2, 1, 'Joao Marques', 'jm@gmail.com').
staff(3, 1, 'Maria Matos', 'm&m@gmail.com').
staff(4, 3, 'Renata Peixoto', 'rp@gmail.com').
staff(5, 2, 'Marta Domingues', 'md@gmail.com').

```

Figure 14: Output: Invariante - Staff (sem ID repetido)

```

% Staff - Email
+staff(?,?,?,Email) ::
    (solucoes(Email,staff(?,?,?,Email),S),
     cumprimento(S,N),
     N == 1).

```

Figure 15: Invariante - Staff (sem email repetido)

```

?- mostrarRegistos(staff).
:- dynamic staff/4.

staff(1, 2, 'Jose Sa', 'js@gmail.com').
staff(2, 1, 'Joao Marques', 'jm@gmail.com').
staff(3, 1, 'Maria Matos', 'm&m@gmail.com').
staff(4, 3, 'Renata Peixoto', 'rp@gmail.com').
staff(5, 2, 'Marta Domingues', 'md@gmail.com').

true.

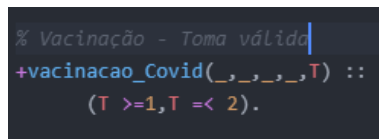
?- registaStaff(6,3,'Ana','js@gmail.com').
false.

?- mostrarRegistos(staff).
:- dynamic staff/4.

staff(1, 2, 'Jose Sa', 'js@gmail.com').
staff(2, 1, 'Joao Marques', 'jm@gmail.com').
staff(3, 1, 'Maria Matos', 'm&m@gmail.com').
staff(4, 3, 'Renata Peixoto', 'rp@gmail.com').
staff(5, 2, 'Marta Domingues', 'md@gmail.com').

```

Figure 16: Output: Invariante - Staff (sem email repetido)



```

% Vacinação - Toma válida
+vacinacao_Covid(?,?,?,T) ::
    (T >=1,T <= 2).

```

Figure 17: Invariante - Vacinação (toma válida)

```

?- mostrarRegistos(vacinacao_Covid).
:- dynamic vacinacao_Covid/5.

vacinacao_Covid(4, 2, (23, 3, 2021), 'Astrazeneca', 1).
vacinacao_Covid(4, 2, (6, 4, 2021), 'Astrazeneca', 2).
vacinacao_Covid(2, 5, (1, 4, 2021), 'Pfizer', 1).

true.

?- registaVacinacao(3,1,(1,4,2021),'Pfizer',5).
false.

?- mostrarRegistos(vacinacao_Covid).
:- dynamic vacinacao_Covid/5.

vacinacao_Covid(4, 2, (23, 3, 2021), 'Astrazeneca', 1).
vacinacao_Covid(4, 2, (6, 4, 2021), 'Astrazeneca', 2).
vacinacao_Covid(2, 5, (1, 4, 2021), 'Pfizer', 1).

```

Figure 18: Output: Invariante - Vacinação (toma válida)

4.2.2 Referenciais

```

% Utente exige um Centro de Saúde existente
+utente(_____,_____,_____,_____,_____,IdCS) ::
    (solucoes(IdsCS,
    (centro_saude(IdCS,_____,_____,_____,_____,S),
    pertence(IdCS,S))).

```

Figure 19: Invariante - Utente (Centro de Saúde existente)

```

?- mostrarRegistos(utente).
:- dynamic utente/10.

utente(1, 123456789, 'Pedro Oliveira', (19, 2, 1934), 'po@gmail.com', 253123451, 'Barcelos', 'Bombeiro', [], 1).
utente(2, 123123123, 'Manuel Faria', (13, 3, 1945), 'mf@gmail.com', 253429351, 'Barcelos', 'Medico', [], 1).
utente(3, 523183123, 'Carla Castro', (2, 12, 1977), 'cc@gmail.com', 253459320, 'Viana do Castelo', 'Jornalista', [], 2).
utente(4, 256331909, 'Roberto Carlos', (21, 1, 1955), 'rc@gmail.com', 253919559, 'Guimarães', 'Engenheiro Informático', ['Hipertensão'], 2).
utente(5, 436329091, 'Rita Neves', (21, 1, 2001), 'rn@gmail.com', 253010123, 'Viana do Castelo', 'Engenheira de Materiais', [], 1).

true.

?- registaUtente(6, 333333333, 'Rui Oliveira', (10, 2, 1956), 'ro@gmail.com', 253123451, 'Braga', 'Carpinteiro', [], 5).
false.

?- mostrarRegistos(utente).
:- dynamic utente/10.

utente(1, 123456789, 'Pedro Oliveira', (19, 2, 1934), 'po@gmail.com', 253123451, 'Barcelos', 'Bombeiro', [], 1).
utente(2, 123123123, 'Manuel Faria', (13, 3, 1945), 'mf@gmail.com', 253429351, 'Barcelos', 'Medico', [], 1).
utente(3, 523183123, 'Carla Castro', (2, 12, 1977), 'cc@gmail.com', 253459320, 'Viana do Castelo', 'Jornalista', [], 2).
utente(4, 256331909, 'Roberto Carlos', (21, 1, 1955), 'rc@gmail.com', 253919559, 'Guimarães', 'Engenheiro Informático', ['Hipertensão'], 2).
utente(5, 436329091, 'Rita Neves', (21, 1, 2001), 'rn@gmail.com', 253010123, 'Viana do Castelo', 'Engenheira de Materiais', [], 1).

```

Figure 20: Output: Invariante - Utente (Centro de Saúde existente)


```
% Staff exige um Centro de Saúde existente
+staff(_, IdC, _, _) ::
    (solucoes(IdsCS,
    (centro_saude(IdsCS, _, _, _)), S),
    pertence(IdC, S)).
```

Figure 21: Invariante - Staff (Centro de Saúde existente)

```
?- mostrarRegistos(staff).
:- dynamic staff/4.

staff(1, 2, 'Jose Sa', 'js@gmail.com').
staff(2, 1, 'Joao Marques', 'jm@gmail.com').
staff(3, 1, 'Maria Matos', 'm&m@gmail.com').
staff(4, 3, 'Renata Peixoto', 'rp@gmail.com').
staff(5, 2, 'Marta Domingues', 'md@gmail.com').

true.

?- registaStaff(6, 5, 'Ana', 'ana@gmail.com').
false.

?- mostrarRegistos(staff).
:- dynamic staff/4.

staff(1, 2, 'Jose Sa', 'js@gmail.com').
staff(2, 1, 'Joao Marques', 'jm@gmail.com').
staff(3, 1, 'Maria Matos', 'm&m@gmail.com').
staff(4, 3, 'Renata Peixoto', 'rp@gmail.com').
staff(5, 2, 'Marta Domingues', 'md@gmail.com').
```

Figure 22: output: Invariante - Staff (Centro de Saúde existente)

```
% Vacinação exige Staff existente
+vacinacao_Covid(Staff,_,_,_,_) ::
    (solucoes(IdS,
    (staff(IdS,_,_,_)),S),
    pertence(Staff,S)).
```

Figure 23: Invariante - Vacinação (Staff existente)

```
?- mostrarRegistos(vacinacao_Covid).
:- dynamic vacinacao_Covid/5.

vacinacao_Covid(4, 2, (23, 3, 2021), 'Astrazeneca', 1).
vacinacao_Covid(4, 2, (6, 4, 2021), 'Astrazeneca', 2).
vacinacao_Covid(2, 5, (1, 4, 2021), 'Pfizer', 1).

true.

?- registaVacinacao(8,1,(1,4,2021),'Pfizer',1).
false.

?- mostrarRegistos(vacinacao_Covid).
:- dynamic vacinacao_Covid/5.

vacinacao_Covid(4, 2, (23, 3, 2021), 'Astrazeneca', 1).
vacinacao_Covid(4, 2, (6, 4, 2021), 'Astrazeneca', 2).
vacinacao_Covid(2, 5, (1, 4, 2021), 'Pfizer', 1).
```

Figure 24: Output: Invariante - Vacinação (Staff existente)

```
% Vacinação exige Utente existente
+vacinacao_Covid(_,U,_,_,_) ::
    (solucoes(IdU,
    (utente(IdU,_,_,_,_,_,_,_,_,_)),S),
    pertence(U,S)).
```

Figure 25: Invariante - Vacinação (Utente existente)

```

?- mostrarRegistos(vacinacao_Covid).
:- dynamic vacinacao_Covid/5.

vacinacao_Covid(4, 2, (23, 3, 2021), 'Astrazeneca', 1).
vacinacao_Covid(4, 2, (6, 4, 2021), 'Astrazeneca', 2).
vacinacao_Covid(2, 5, (1, 4, 2021), 'Pfizer', 1).

true.

?- registaVacinacao(3,9,(1,4,2021),'Pfizer',1).
false.

?- mostrarRegistos(vacinacao_Covid).
:- dynamic vacinacao_Covid/5.

vacinacao_Covid(4, 2, (23, 3, 2021), 'Astrazeneca', 1).
vacinacao_Covid(4, 2, (6, 4, 2021), 'Astrazeneca', 2).
vacinacao_Covid(2, 5, (1, 4, 2021), 'Pfizer', 1).

```

Figure 26: Output: Invariante - Vacinação (Utente existente)

```

% Segunda dose de uma vacina exige uma primeira já existente
+vacinacao_Covid(Staff,Utente,_,_,2) ::
    (solucoes((Staff,Utente,1),
    (vacinacao_Covid(Staff,Utente,_,_,1)),R),
    comprimento(R,N),
    N == 1).

```

Figure 27: Invariante - Vacinação (Primeira dose existente)

```

?- mostrarRegistos(vacinacao_Covid).
:- dynamic vacinacao_Covid/5.

vacinacao_Covid(4, 2, (23, 3, 2021), 'Astrazeneca', 1).
vacinacao_Covid(4, 2, (6, 4, 2021), 'Astrazeneca', 2).
vacinacao_Covid(2, 5, (1, 4, 2021), 'Pfizer', 1).

true.

?- registaVacinacao(3,1,(1,4,2021),'Pfizer',2).
false.

?- mostrarRegistos(vacinacao_Covid).
:- dynamic vacinacao_Covid/5.

vacinacao_Covid(4, 2, (23, 3, 2021), 'Astrazeneca', 1).
vacinacao_Covid(4, 2, (6, 4, 2021), 'Astrazeneca', 2).
vacinacao_Covid(2, 5, (1, 4, 2021), 'Pfizer', 1).

```

Figure 28: Output: Invariante - Vacinação (Primeira dose existente)

```

% Segunda dose de uma vacina exige uma vacina igual à primeira
+vacinacao_Covid(Staff,Utente,_,Nome,2) ::
    (solucoes((Staff,Utente,Nome,1),
    (vacinacao_Covid(Staff,Utente,_,Nome,1)),R),
    comprimento(R,N),
    N == 1).

```

Figure 29: Invariante - Vacinação (Duas doses iguais)

```

?- mostrarRegistos(vacinacao_Covid).
:- dynamic vacinacao_Covid/5.

vacinacao_Covid(4, 2, (23, 3, 2021), 'Astrazeneca', 1).
vacinacao_Covid(4, 2, (6, 4, 2021), 'Astrazeneca', 2).
vacinacao_Covid(2, 5, (1, 4, 2021), 'Pfizer', 1).

true.

?- registaVacinacao(2,5,(10,4,2021),'Astrazeneca',2).
false.

?- mostrarRegistos(vacinacao_Covid).
:- dynamic vacinacao_Covid/5.

vacinacao_Covid(4, 2, (23, 3, 2021), 'Astrazeneca', 1).
vacinacao_Covid(4, 2, (6, 4, 2021), 'Astrazeneca', 2).
vacinacao_Covid(2, 5, (1, 4, 2021), 'Pfizer', 1).

```

Figure 30: Output: Invariante - Vacinação (Duas doses iguais)

```

% Primeira dose de uma vacina exige que esse Utente nao tenha sido vacinado ainda
+vacinacao_Covid(_,U,_,1) ::
    (solucoes(U,
        (vacinacao_Covid(_,U,_,_)),R),
    comprimento(R,N),
    N == 0).

```

Figure 31: Invariante - Vacinação (Sem vacinação antes da primeira dose)

```

?- mostrarRegistros(vacinacao_Covid).
:- dynamic vacinacao_Covid/5.

vacinacao_Covid(4, 2, (23, 3, 2021), 'Astrazeneca', 1).
vacinacao_Covid(4, 2, (6, 4, 2021), 'Astrazeneca', 2).
vacinacao_Covid(2, 5, (1, 4, 2021), 'Pfizer', 1).

true.

?- registaVacinacao(2,2,(10,4,2021),'Astrazeneca',1).
false.

?- mostrarRegistros(vacinacao_Covid).
:- dynamic vacinacao_Covid/5.

vacinacao_Covid(4, 2, (23, 3, 2021), 'Astrazeneca', 1).
vacinacao_Covid(4, 2, (6, 4, 2021), 'Astrazeneca', 2).
vacinacao_Covid(2, 5, (1, 4, 2021), 'Pfizer', 1).

```

Figure 32: Output: Invariante - Vacinação (Sem vacinação antes da primeira dose)

```

% Segunda dose de uma vacina exige que seja depois da primeira (data)
+vacinacao_Covid(_,U,Data2,V,2) ::
    (solucoes((D1,M1,A1),
    (vacinacao_Covid(_,U,(D1,M1,A1),V,1)),R),
    head(R,X),
    anterior(X,Data2)).

```

Figure 33: Invariante - Vacinação (Segunda dose depois da primeira)

```

?- mostrarRegistos(vacinacao_Covid).
:- dynamic vacinacao_Covid/5.

vacinacao_Covid(4, 2, (23, 3, 2021), 'Astrazeneca', 1).
vacinacao_Covid(4, 2, (6, 4, 2021), 'Astrazeneca', 2).
vacinacao_Covid(2, 5, (1, 4, 2021), 'Pfizer', 1).

true.

?- registaVacinacao(2,5,(10,3,2021),'Pfizer',2).
false.

?- mostrarRegistos(vacinacao_Covid).
:- dynamic vacinacao_Covid/5.

vacinacao_Covid(4, 2, (23, 3, 2021), 'Astrazeneca', 1).
vacinacao_Covid(4, 2, (6, 4, 2021), 'Astrazeneca', 2).
vacinacao_Covid(2, 5, (1, 4, 2021), 'Pfizer', 1).

```

Figure 34: Output: Invariante - Vacinação (Segunda dose depois da primeira)

4.3 Predicados Gerais

No decorrer do trabalho, deparamo-nos com situações em que a utilização de predicados auxiliares seria muito benéfica, os quais são apresentados nesta secção.

```

%-----
% Predicado teste
teste([]).
teste([R|LR]) :- R, teste(LR).

```

Figure 35: teste

```

%-----
% Mostrar registos
mostrarRegistos(P) :- listing(P).
%

```

Figure 36: mostrarRegistos

```

%-----
% Data atual
date(Day,Month,Year) :-
    get_time(Stamp),
    stamp_date_time(Stamp, DateTime, local),
    date_time_value(year, DateTime, Year),
    date_time_value(month, DateTime, Month),
    date_time_value(day, DateTime, Day).

```

Figure 37: date

```

%-----
% Verificar se uma data é anterior a outra

anterior((_,_,A1),(_,_,A2)) :- A1 < A2.
anterior((_,M1,A1),(_,M2,A2)) :- A1 == A2, M1 < M2.
anterior((D1,M1,A1),(D2,M2,A2)) :- A1 == A2, M1 == M2, D1 < D2.

```

Figure 38: anterior

```

%-----
% Calcular a idade de um utente
idade((_,M,A),I):- date(_,Y,Z), I is Z-A, M<Y.
idade((D,M,A),I):- date(X,Y,Z), I is Z-A, M==Y, D<X.
idade((D,M,A),I):- date(X,Y,Z), I is Z-A-1, M==Y, D>X.
idade((_,M,A),I):- date(_,Y,Z), I is Z-A-1, M>=Y.

```

Figure 39: idade

```

%-----
% Predicado solucoes
solucoes(X,P,S) :- findall(X,P,S).

```

Figure 40: solucoes


```
%-----
% Predicado solucoes sem repeticoes
solucoesSRep(X,Y,Z1) :-
    findall(X,Y,Z),
    list_to_set(Z,Z1).
```

Figure 41: solucoesSRep

```
%-----
% Comprimento da Lista
comprimento(S,N) :- length(S,N).
```

Figure 42: comprimento

```
%-----
% Pertencer a uma Lista
pertence(H,[H|_]) :-!,true.
pertence(X,[H|T]) :-
    X \= H,
    pertence(X,T).
```

Figure 43: pertence

```
%-----
% Cabeça de uma Lista
head([H],H).
head([H|_],H).
```

Figure 44: head

```
%-----
% Concatenar uma Lista
append([ ], L, L).
append([H|L1], L2, [H|L3]) :- append(L1, L2, L3).
```

Figure 45: append

```
%-----
% Extensao do meta-predicado nao: Questao -> {V,F}
nao( Questao ) :-
    Questao, !, fail.
nao( _ ).
```

Figure 46: nao

```
%-----
% Extensao do sistema de inferencia si: Questao, (Valor -> {V,F})
si(Questao, verdadeiro) :-
    Questao.
si(Questao, falso) :-
    nao(Questao).
```

Figure 47: si

4.4 Definições

A nossa base de conhecimento está pronta para receber informação de utentes, centros de saúde, staff e sobre vacinações.

```
%-----
% Definicoes iniciais

:- op( 900,xfy,'::' ).
:- dynamic utente/10.
:- dynamic centro_saude/5.
:- dynamic staff/4.
:- dynamic vaccinacao_Covid/5.
```

Figure 48: Definições

4.5 Base de Conhecimento Inicial

Como só é permitida a adição de informação na nossa base de conhecimento, optamos por incluir uma pequena quantidade de conhecimento inicial de forma a que o utilizador tenha uma maior margem de manobra na utilização do nosso programa.

```
%-----
% Extensao do predicado utente: Idutente, Nª Segurança_Social,
% Nome, Data_Nasc, Email, Telefone, Morada,
% Profissão, [Doenças_Crónicas], CentroSaúde -> {V,F}

utente(1,123456789,'Pedro Oliveira',(19,02,1934),'po@gmail.com',253123451,'Barcelos','Bombeiro',[],1).
utente(2,123123123,'Manuel Faria',(13,03,1945),'mf@gmail.com',253429351,'Barcelos','Medico',[],1).
utente(3,523183123,'Carla Castro',(02,12,1977),'cc@gmail.com',253459320,'Viana do Castelo','Jornalista',[],2).
utente(4,256331909,'Roberto Carlos',(21,01,1955),'rc@gmail.com',253919559,'Guimarães','Engenheiro Informático',['Hipertensão'],2).
utente(5,436329091,'Rita Neves',(21,01,2001),'rn@gmail.com',253010123,'Viana do Castelo','Engenheira de Materiais',[],1).
```

Figure 49: Utentes

```
%-----
% Extensao do predicado centro_saúde: Idcentro, Nome, Morada, Telefone, Email -> {V,F}

centro_saude(1,'Centro de saúde de Viana','Viana do Castelo',253456712,'csv@gmail.com').
centro_saude(2,'Centro de saúde de Guimarães','Guimarães',253921733,'csg@gmail.com').
centro_saude(3,'Centro de saúde de Barcelos','Barcelos',253004239,'csb@gmail.com').
```

Figure 50: Centros de Saúde

```
%-----
% Extensao do predicado staff: Idstaff, Idcentro, Nome, email -> {V,F}

staff(1,2,'Jose Sa','js@gmail.com').
staff(2,1,'Joao Marques','jm@gmail.com').
staff(3,1,'Maria Matos','m&m@gmail.com').
staff(4,3,'Renata Peixoto','rp@gmail.com').
staff(5,2,'Marta Domingues','md@gmail.com').
```

Figure 51: Staff

```
%-----
% Extensao do predicado vacinacao_Covid: Staff, Utente, Data, Vacina, Toma -> {V,F}

vacinacao_Covid(4,2,(23,03,2021),'Astrazeneca',1).
vacinacao_Covid(4,2,(06,04,2021),'Astrazeneca',2).
vacinacao_Covid(2,5,(01,04,2021),'Pfizer',1).
```

Figure 52: Vacinações

5 Funcionalidades

5.1 Definição de fases de vacinação

Esta funcionalidade contém duas principais subfuncionalidades para permitir ao utilizador retirar informações acerca das fases de vacinação, como por exemplo, conhecer todos os utentes candidatos a cada uma das fases de vacinação (1, 2 ou 3), ou pedir ao sistema de inferência para verificar se um utente é candidato a uma determinada fase.

Inicialmente, a nossa equipa decidiu definir datas de início de cada fase de vacinação (definidas através de predicados **dataFaseX**, em que X representa a fase), assim como os critérios para ser candidato a cada uma das fases.

- **Fase 1:** Começa a partir de 1/12/2020 e para ser candidato a esta fase, consideramos que o utente deve ter a profissão de médico ou enfermeiro (para tal criamos um predicado **profissoesFase1** que nos *guarda* estas profissões) ou então possuir pelo menos uma doença crónica e ter mais de 80 anos (utilizamos o predicado **idade** para calcularmos a idade de cada utente em função da data de nascimento).
- **Fase 2:** Começa a partir de 1/4/2021 e é candidato a esta fase qualquer utente que não seja candidato à primeira e que tenha pelo menos uma doença crónica e mais de 50 anos, ou então que tenha mais de 65 anos sem doenças crónicas.
- **Fase 3:** Fase relativa a toda a população que não seja candidata às duas fases anteriores, começa a partir de 1/7/2021.

5.1.1 Registo

A inserção de informação na base de conhecimento é feita através do teorema **evolucao** como ja foi referido anteriormente e é dividido em 4 registos.

```
%-----  
% Registar Utentes  
  
registarUtente(Id,Nss,Nome,Data,Email,Tel,Mor,Prof,Dc,Cs) :-  
    evolucao(utente(Id,Nss,Nome,Data,Email,Tel,Mor,Prof,Dc,Cs)).  
  
%-----  
% Registar Centro de Saúde  
  
registarCentro(Id,Nome,Mor,Tel,Email) :-  
    evolucao(centro_saude(Id,Nome,Mor,Tel,Email)).  
  
%-----  
% Registar Staff  
  
registarStaff(Id,Idcentro,Nome,Email) :-  
    evolucao(staff(Id,Idcentro,Nome,Email)).  
  
%-----  
% Registar Vacinação  
  
registarVacinacao(Idstaff,Idutente,Data,Vac,T) :-  
    evolucao(vacinacao_Covid(Idstaff,Idutente,Data,Vac,T)).
```

Figure 53: Registar

5.1.2 Verificar se um utente é candidato a uma fase de vacinação

Para esta funcionalidade temos 3 predicados, **candidata1**, **candidata2** e **candidata3**. Passando ao predicado o Id da pessoa verificamos se é candidata à fase 1, 2 ou 3 respetivamente.

Relativamente à fase 1, na primeira entrada do predicado verificamos se o utente existe e se a sua profissão pertence à lista de profissões. Na segunda verificamos se existe, se a sua idade é superior ou igual a 80 anos, verificamos se tem doenças crónicas.

Na fase 2, fazemos o processo quase igual à fase 1, no entanto no primeiro caso a idade tem de estar entre 50 e 65 anos, deve ter pelo menos uma doença crónica e o utente não deve ser candidato à fase 1, e no segundo caso a idade deve ser maior ou igual a 65 anos.

Na fase 3 apenas verificamos se o utente existe e não é candidato a nenhuma das outras fases.

```

candidata1(Id):-
    utente(Id,_,_,_,_,_,P,_,_),
    profissoesFase1(Ps),
    pertence(P,Ps).
candidata1(Id) :-
    utente(Id,_,_,(D,M,A),_,_,_,P,Ds,_,_),
    idade((D,M,A),R),
    R >= 80,
    comprimento(Ds,N),
    N >= 1,
    profissoesFase1(Ps),
    nao(pertence(P,Ps)).

```

Figure 54: Candidato à fase 1

```

candidata2(Id) :-
    utente(Id,_,_,(D,M,A),_,_,_,Ds,_,_),
    idade((D,M,A),R),
    R >= 50,
    R < 65,
    comprimento(Ds,N),
    N >= 1,
    nao(candidata1(Id)).
candidata2(Id) :-
    utente(Id,_,_,(D,M,A),_,_,_,_,_,_),
    idade((D,M,A),R),
    R >= 65,
    nao(candidata1(Id)).

```

Figure 55: Candidato à fase 2

5.1.3 Lista de candidatos a uma determinada fase de vacinação

Esta funcionalidade devolve-nos a lista com os candidatos a cada fase, de acordo com o predicado em causa, que pode ser **fase1**, **fase2** ou **fase3**. Para implementá-la utilizamos o predicado **solucoes** que se baseia num *findall*, e também o predicado **utente** e **candidataX**, em que X define a fase.

5.2 Identificar pessoas não vacinadas

5.2.1 Verificar se um utente não foi vacinado

No nosso sistema temos uma funcionalidade que nos indica se um utente já foi vacinado, por isso, a estratégia de implementação desta foi utilizar o predicado **nao** e o **vacinada**, que nos diz se uma pessoa já foi vacinada.

5.2.2 Lista pessoas não vacinadas

Esta funcionalidade utiliza a anterior para nos dar a lista de todos os não vacinados existentes no sistema. Utilizamos o predicado **solucoes** e damos a lista com todos os utentes não vacinados.

```

candidata3(Id) :-
    utente(Id,_,_,_,_,_,_,_,_,_),
    nao(candidata1(Id)),
    nao(candidata2(Id)).

```

Figure 56: Candidato à fase 3

```

?- si(candidata1(1),V).
V = falso.

?- si(candidata2(1),V).
V = verdadeiro .

?- si(candidata2(2),V).
V = falso.

?- si(candidata3(2),V).
V = falso.

?- si(candidata1(2),V).
V = verdadeiro .

```

Figure 57: Output: Candidatos a cada fase

```

fase1(Lista) :- solucoes((X,Nomes),(utente(X,_,Nomes,_,_,_,_,_,_,_),candidata1(X)),Lista).
fase2(Lista) :- solucoes((X,Nomes),(utente(X,_,Nomes,_,_,_,_,_,_,_),candidata2(X)),Lista).
fase3(Lista) :- solucoes((X,Nomes),(utente(X,_,Nomes,_,_,_,_,_,_,_),candidata3(X)),Lista).

```

Figure 58: Listas dos candidatos a cada fase

```

?- si(fase1(L),V).
L = [(2, 'Manuel Faria')],
V = verdadeiro .

?- si(fase2(L),V).
L = [(1, 'Pedro Oliveira'), (4, 'Roberto Carlos')],
V = verdadeiro .

?- si(fase3(L),V).
L = [(3, 'Carla Castro'), (5, 'Rita Neves')],
V = verdadeiro .

```

Figure 59: Output: Listas dos candidatos a cada fase

```

nao_vacinada(X) :- nao(vacinada(X)).

nao_vacinadas(Lista) :-
    solucoes((Ids,Nomes),
        (utente(Id,_,Nomes,_,_,_,_,_,_,_),nao_vacinada(Id)),
        Lista).

```

Figure 60: Utentes não vacinados

```
?- si(nao_vacinada(1),V).  
V = verdadeiro .  
  
?- si(nao_vacinada(2),V).  
V = falso.  
  
?- si(nao_vacinadas(L),V).  
L = [(1, 'Pedro Oliveira'), (3, 'Carla Castro'), (4, 'Roberto Carlos')],  
V = verdadeiro .
```

Figure 61: Output: Utentes não vacinados

5.3 Identificar pessoas vacinadas

5.3.1 Verificar se um utente já foi vacinado

Esta funcionalidade terá como principal objetivo verificar em todas as vacinações ocorridas e registadas na nossa base de conhecimento, se existe uma vacinação ao utente com o id que foi passado como argumento.

5.3.2 Lista de pessoas vacinadas

Com ajuda da funcionalidade anterior, conseguimos implementar uma que nos listasse todos os utentes que já tinham sido vacinados, devolvendo o respetivo nome e id da pessoa.

```
vacinada(X):- vacinacao_Covid(_,X,_,_,_).  
  
vacinadas(Lista) :-  
    (solucoesSRep((Ids,Nomes),  
        (utente(Id,_,Nome,_,_,_,_,_),vacinada(Id)),  
        Lista)).
```

Figure 62: Utentes vacinados

```
?- si(vacinada(1),V).  
V = falso.  
  
?- si(vacinada(2),V).  
V = verdadeiro .  
  
?- si(vacinadas(L),V).  
L = [(2, 'Manuel Faria'), (5, 'Rita Neves')],  
V = verdadeiro .
```

Figure 63: Output: Utentes vacinados

5.4 Identificar pessoas vacinadas indevidamente

Esta funcionalidade permite-nos saber quais as pessoas que foram vacinadas indevidamente, de acordo com as datas de vacinação. É considerada uma vacina indevida, se o utente tiver sido vacinado antes da data inicial da fase em que é candidato.

5.4.1 Verificar se um utente está vacinado indevidamente

Este predicado verifica a vacinação indevida para apenas um utente, com base no que foi dito em cima. Deste modo, temos de verificar para cada fase, se o utente está vacinado, se é candidato a uma fase e foi vacinado numa data inválida para essa fase. O predicado auxiliar **anterior** verifica se uma data é anterior a outra.

```
vacina_indevida(X) :-
    vacinada(X),
    candidat1(X),
    vaccinacao_Covid(_,X,(D,M,A),_,_),
    dataFase1((D1,M1,A1)),
    anterior((D,M,A),(D1,M1,A1)).
vacina_indevida(X) :-
    vacinada(X),
    candidata2(X),
    vaccinacao_Covid(_,X,(D,M,A),_,_),
    dataFase2((D1,M1,A1)),
    anterior((D,M,A),(D1,M1,A1)).
vacina_indevida(X) :-
    vacinada(X),
    candidata3(X),
    vaccinacao_Covid(_,X,(D,M,A),_,_),
    dataFase3((D1,M1,A1)),
    anterior((D,M,A),(D1,M1,A1)).
```

Figure 64: Utente vacinado indevidamente

```
?- vacina_indevida(5).  
true .  
  
?- si(vacina_indevida(2),V).  
V = falso.  
  
?- si(vacina_indevida(5),V).  
V = verdadeiro .  
  
?- si(vacinas_indevidas(L),V).  
L = [(5, 'Rita Neves')],  
V = verdadeiro .
```

Figure 65: Output: Utente vacinado indevidamente

5.4.2 Lista de utentes vacinados indevidamente

Com o auxílio do predicado anterior, obtemos a funcionalidade principal. O predicado **solucoesSRep** é parecido com o **solucoes**, a diferença é que nos devolve a **Lista** sem elementos repetidos.

```
vacinas_indevidas(Lista) :-  
    (solucoesSRep((Ids,Nomes),  
    (utente(Id,_),Nome,_,_,_,_,_,_),vacina_indevida(Id)),  
    Lista)).
```

Figure 66: Lista de utentes vacinados indevidamente

```
?- vacina_indevida(5).  
true .  
  
?- si(vacina_indevida(2),V).  
V = falso.  
  
?- si(vacina_indevida(5),V).  
V = verdadeiro .  
  
?- si(vacinas_indevidas(L),V).  
L = [(5, 'Rita Neves')],  
V = verdadeiro .
```

Figure 67: Lista de utentes vacinados indevidamente

5.5 Identificar pessoas não vacinadas e que são candidatas a vacinação

Esta funcionalidade divide-se em 3 predicados que se centram em cada uma das fases e desenvolvendo um processo idêntico para solucionar o problema.

5.5.1 Verificar se um utente não foi vacinado e é candidato a uma determinada fase

Em cada uma das fases, no seu respetivo predicado, verificamos se o id passado como argumento não foi vacinado e, de seguida, verificamos se cumpre os requisitos estipulados para essa mesma fase.

5.5.2 Lista de utentes não vacinados e candidatos a uma determinada fase

Desenvolvida a funcionalidade anterior, o processo de construção de uma funcionalidade que listasse utentes não vacinados e candidatos a cada fase não foi uma tarefa complicada, pois seguimos a lógica de todas as funcionalidades até agora, devolvendo sempre o id e o utente.

```
% Pessoas nao vacinadas e candidatas a fase 1
nao_vacinada1(X) :-
    nao_vacinada(X),
    candidata1(X).

nao_vacinadas1(Lista) :-
    (solucoes((Ids,Nomes),
    (utente(Id,_,Nomes,_,_,_,_,_,_),nao_vacinada1(Id)),
    Lista)).

% Pessoas nao vacinadas e candidatas a fase 2
nao_vacinada2(X) :-
    nao_vacinada(X),
    candidata2(X).

nao_vacinadas2(Lista) :-
    (solucoes((Ids,Nomes),
    (utente(Id,_,Nomes,_,_,_,_,_,_),nao_vacinada2(Id)),
    Lista)).

% Pessoas nao vacinadas e candidatas a fase 3
nao_vacinada3(X) :-
    nao_vacinada(X),
    candidata3(X).

nao_vacinadas3(Lista) :-
    (solucoes((Ids,Nomes),
    (utente(Id,_,Nomes,_,_,_,_,_,_),nao_vacinada3(Id)),
    Lista)).
```

Figure 68: Utentes não vacinados e candidatos a cada uma das fases

```

?- si(candidata1(1),V).
V = falso.

?- si(candidata2(1),V).
V = verdadeiro .

?- si(candidata2(2),V).
V = falso.

?- si(candidata3(2),V).
V = falso.

?- si(candidata1(2),V).
V = verdadeiro .

?- si(fase1(L),V).
L = [(2, 'Manuel Faria')],
V = verdadeiro .

?- si(fase2(L),V).
L = [(1, 'Pedro Oliveira'), (4, 'Roberto Carlos')],
V = verdadeiro .

?- si(fase3(L),V).
L = [(3, 'Carla Castro'), (5, 'Rita Neves')],
V = verdadeiro .

```

Figure 69: Output: Utentes não vacinados e candidatos a cada uma das fases

5.6 Identificar pessoas a quem falta a segunda toma da vacina

5.6.1 Lista de pessoas a quem falta a segunda toma

Com estas duas funcionalidades conseguimos obter uma lista com todos os utentes a quem falta a segunda toma da vacina.

```
falta_2toma(X):-
    (solucoes(Ys,
    (vacinacao_Covid(_,X,_,Ys)),
    Res),
    comprimento(Res,N),
    N == 1,
    [H|_] = Res,
    H == 1).

falta_2tomaLista(Lista) :-
    (solucoes((Ids,Nomes),
    (utente(Id,_,Nome,_,_,_,_,_),falta_2toma(Id)),
    Lista)).
```

Figure 70: Utentes a quem falta a segunda toma

```
?- si(falta_2toma(2),V).
V = falso.

?- si(falta_2toma(5),V).
V = verdadeiro .

?- si(falta_2tomaLista(L),V).
L = [(5, 'Rita Neves')],
V = verdadeiro .
```

Figure 71: Output: Utentes a quem falta a segunda toma

6 Funcionalidades Extra

Para facilitar a consulta da informação, e complementar o sistema e avaliar o estado do mesmo, foram criadas desenvolvidas algumas funcionalidades extra.

6.1 Lista das pessoas vacinadas num determinado centro de saúde

```
% Lista das pessoas vacinadas num determinado centro de saúde  
  
pessoas_vacinadas_centro(Idcentro,L):- solucoesSRep((Idu,Nome),  
    (utente(Idu,_,Nome,_,_,_,_,_,Idcentro),  
    vacinacao_Covid(_,Idu,_,_,_)),  
    L).
```

Figure 72: Predicado da funcionalidade extra 1

6.2 Lista das diferentes vacinas dadas num determinado centro de saúde

```
% Lista das diferentes vacinas dadas num determinado centro de saúde  
  
vacinas_centro(Idcentro,L):- solucoesSRep(Vacina,  
    (vacinacao_Covid(_,_,_,Vacina,_),  
    staff(_,Idcentro,_,_)),  
    L).
```

Figure 73: Predicado da funcionalidade extra 2

6.3 Lista das pessoas que tomaram uma determinada vacina

```
% Lista das pessoas que tomaram uma determinada vacina  
pessoas_vacina(Vacina,L) :- solucoesSRep(Nome,  
    (vacinacao_Covid(_,Idu,_,Vacina,_),  
    utente(Idu,_,Nome,_,_,_,_,_)),  
    L).
```

Figure 74: Predicado da funcionalidade extra 3

6.4 Lista das pessoas que receberam a vacina por uma determinada pessoa do staff

```
% Lista das pessoas que receberam a vacina por uma determinada pessoa do staff
pessoas_staff(Staff,(NomeStaff,Nomes)) :- solucoesSRep(N,
    (vacinacao_Covid(Staff,U,_,_,_),
    utente(U,_,N,_,_,_,_,_,_)),
    Nomes),
    staff(Staff,NomeStaff,_,_)).
```

Figure 75: Predicado da funcionalidade extra 4

6.5 Lista das pessoas que têm a vacinação completa

```
% Lista das pessoas que têm a vacinação completa
vacinacao_completa(R) :- solucoesSRep((Idu,Nome),
    (vacinacao_Covid(_,Idu,_,_,2),
    utente(Idu,_,Nome,_,_,_,_,_,_)),
    R).
```

Figure 76: Predicado da funcionalidade extra 5

7 Conclusão

Este primeiro trabalho prático da unidade curricular de Sistemas de Representação de Conhecimento e Raciocínio, fez-nos aprofundar e familiarizar-nos com a linguagem *Prolog* que até agora se tem revelado ser bastante interessante e fácil de manipular.

O grupo sente-se satisfeito com o trabalho desenvolvido, já que atingimos todos os objetivos estipulados e conseguimos ainda ter tempo de sobra para nos divertir-mos com o tema e poder-mos melhorar o projeto com funcionalidades extra.

Por fim, reconhecemos que este primeiro trabalho foi importante para consolidar algumas bases que serão necessárias nos próximos projetos que teremos até ao fim do semestre.