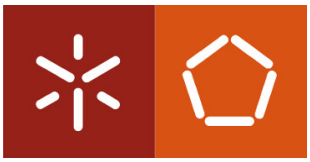
Universidade do Minho
Mestrado Integrado em Engenharia Informática



# Sistemas De Representação De conhecimento e Raciocínio


**GRUPO 44**
Ana Teresa Gião Gomes - A89536
Maria Quintas Barros - A89325
Maria Beatriz Araújo Lacerda - A89535
Francisco Franco - A89536
3º ano de MIEI


Abril de 2021

Figura 1: A89536
Ana Gomes

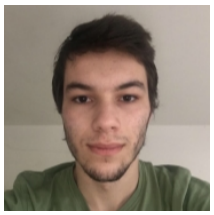


Figura 2: A89536
Francisco Franco

Figura 3: A89525
Maria Barros

Figura 4: A89535
Beatriz Lacerda

# Conteúdo

# 1 Introdução

Para este trabalho usamos a programação em lógica, ou seja, um estilo de programação que se baseia nos factos e que faz uso da lógica matemática para a criação de cada um dos vários programas.
Deste modo, este paradigma de programação tem como base da sua essência duas ideias principais: inferência, isto é, operação lógica que, partindo de premissas previamente definidas como verdadeiras, conseguimos derivar conclusões, e lógica, ou seja, uma área da filosofia que visa salientar o pensar correto e o uso do raciocínio.
Para além disso, utilizamos também o PROLOG, que é uma linguagem de programação muito comum no âmbito da Programação Lógica, isto é, é usada para resolver todos os problemas que tenham objetos e relações entre objetos.
Para este trabalho prático da primeira fase, foi nos proposto o desenvolvimento de um sistema que tenha a capacidade de caracterizar um universo de discurso na área da vacinação global da população do nosso país, no âmbito da situação atual pandémica devido ao COVID-19, no qual iremos utilizar os conceitos referidos em cima.

## 1.1 Motivação e Objetivos

Sendo este um dos intuitos do exercício, consideramos que iremos usufruir de uma melhor consolidação dos conteúdos lecionados nas aulas da UC, que, por consequência, se irá tornar importante para que todos estejamos cada vez mais à vontade neste novo paradigma de programação com o qual nunca tinhamos trabalhado e, por isso, não estavamos habituados.
Como objetivo principal deste projeto, temos a realização de um programa que consiga guardar todas as informações importantes relacionadas com a vacinação, para que esta seja feita da melhor forma possível e que resulte em bons resultados para o nosso país nesta luta contra a pandemia.

## 1.2 Estrutura do Relatório

Para estruturar da melhor forma o nosso relatório, consideramos prudente termos 6 capítulos.
Neste primeiro capítulo, intoduzimos o tema do projeto e a linguagem que usamos, onde não só explicamos qual o nosso objetivo enquanto grupo para este trabalho, como também referimos a nossa motivação.
Posteriormente, no segundo capítulo, apresentamos a linguagem que utilizamos, PROLOG, com o intuito de que quem leia perceba o projeto e o funcionamento da própria linguagem.
No terceiro capítulo, explicamos a base de conhecimento que utilizamos, assim como os invariantes, que nos ajudaram na implementação do trabalho.
No quarto capítulo, apresentamos a funcionalidades do programa, que são a base principal do projeto.
De seguida, no quinto capítulo, sentimos necessidade de explicar todas as funcionalidades extra que adicionamos de modo a que obtivéssemos um trabalho ainda mais completo.
Por último, no sexto capítulo, concluímos todo o trabalho desenvolvido e fizemos algumas observações sobre o que foi feito.

# 2 Programação em Lógica

A programação lógica é um paradigma de programação que utiliza Lógica Matemática. Deste modo, baseia-se em:

- declaração de **factos**, ou seja, afirmações que sabemos que são verdadeiras.
- definição de **regras**, isto é, imposições sobre os objetos e as suas relações.
- **predicados**, que estabelecem relações entre os objetos.

# 3 Descrição do Trabalho e Análise de Resultados

## 3.1 Base de conhecimento

A criação de uma base de conhecimentos é de extrema importância uma vez que precisamos de ter uma base de dados com todas as informações úteis para o nosso projeto. Deste modo, conseguimos que todas as questões expostas no enunciado sejam respondidas de forma eficaz e simples.

### 3.1.1 Utentes

Consideramos útil criar criar uma base de dados com os utentes a ter em conta para o projeto. Deste modo, acrescentamos a cada utente um conjunto de características que os vai definir, entre eles o id do próprio utente, o número da segurança social, o seu nome e data de nascimento, email, telefone, a morada, as doenças crónicas que apresenta, a profissão e o centro de saúde onde o utente vai. Perante isto, temos:

**utente (Idutente, Nº Segurança_Social, Nome, Data_Nasc, Email, Telefone, Morada, [Doenças_Crónicas], Profissão, CentroSaúde)**.

```
utente(1, 23456, 'Carlos', 24-04-1999,'carlos@gmail.com', '966345781','Rua do
    Falco',[Renite],'Professor',1).
utente(2, 23906, 'Rui', 13-03-1987,'rui@gmail.com', '989349235','Rua da
    Esquina',[],'Mdico',1).
utente(3, 12396, 'Carlota', 20-09-2005,'carlota@gmail.com', '963215637','Rua Alto
    Mar',[],'Estudante',1).
utente(4, 76590, 'Maria', 25-09-1935,'maria@gmail.com', '934674129','Rua do
    Bairro',[],'Carpinteira',1).
utente(5, 56851, 'Jorge', 29-05-1955,'jorge@gmail.com', '962458965','Rua de S.Bento da
    Vrzea',[],'Psicloga',1).
utente(6, 65975, 'Alexandra', 18-02-2009,'alexandra@gmail.com', '912358957','Rua Alto
    Mar',[],'Enfermeira',2).
utente(10, 75362,'Lisandro',05-06-2005,'lisandro@gmail.com', '963030147', 'Rua
    Lisboa',[],'Estudante',2).
```

### 3.1.2 Centros de sáude

Outro aspeto essencial é a criação de uma base de conhecimento para os centros de saúde, onde cada um deles vai ter também um conjunto de características a definí-los, entre eles o id de cada centro, o nome, a morada, o telefone e o email. Assim , ficamos com:

**centro_saude (Idcentro, Nomne, Morada, Telefone, email)**.

```
centro_saude(1,'Centro de Sade de Barcelos', 'Rua Alta', '253234674', 'centro1@gmail.com').
centro_saude(2,'Hospital De Braga', 'Rua das Paredes ', '253523489', 'centro2@gmail.com').
```

### 3.1.3 Staff

Para a base de conhecimento do Staff, consideramos importante acrescentar o id do staff, o id do centro onde ele trabalha, o nome e o email, ou seja, temos:

**staff(idstaff, Idcentro, Nome, email)**.

```
staff(1,1,'Rute Silva', 'rute@gmail.com').
staff(2,1,'Santiago Rua', 'santiago@gmail.com').
staff(3,2,'Vitor Castro', 'vitor@gmail.com').
staff(4,2,'Catarina Coutada', 'catarina@gmail.com').
```

### 3.1.4 Vacinação_covid

Por último, criamos também uma base de conhecimento para a vacinação ao covid, que inclui o id do staff, o id do utente vacinado, a data em que ocorreu a vacinação, a vacina escolhida e a toma. Deste modo, temos:

**vacinacao_covid (Staff, utente, Data, Vacina, Toma)**

```
vacinacao_covid(4,2,6-3-2021, Vacina2,1).
vacinacao_covid(1,5,6-1-2021, Vacina1, 1).
```

# 4 Funcionalidades do programa

## 4.1 Permitir a definição de fases de vacinação, definindo critérios de inclusão de utentes nas diferentes fases (e.g., doenças crónicas, idade, profissão)

O nosso grupo optou por definir 2 fases distintas de vacinação:

Numa primeira fase, que se deve realizar entre Janeiro de 2021 e Março de 2021, devem ser vacinados utentes que tenham uma idade superior a 50 anos, que tenham doenças ou que sejam profissionais da linha da frente (médicos e enfermeiros).

Para isto começamos por definir estes critérios:

```
idosos(R) :- resposta(U, (utente(U,_,_,D-M-A,_,_,_,_,_,_),X is 2021 - A, X>50), R).

doentes(R) :- resposta(U, (utente(U,_,_,_,_,_,_,_,L,_,_), comprimento(L,T), T>0),R).

profmedico(R) :- resposta(U, utente(U,_,_,_,_,_,_,_,_,'Medico',_),R).
profenfermeiro(R) :- resposta(U, utente(U,_,_,_,_,_,_,_,_,'Enfermeiro',_),R).
profenfermeira(R) :- resposta(U, utente(U,_,_,_,_,_,_,_,_,'Enfermeira',_),R).
profmedica(R) :- resposta(U, utente(U,_,_,_,_,_,_,_,_,'Medica',_),R).
```

O nosso resposta equivale a um findall. Sendo assim, o primeiro argumento equivale à informação que queremos obter, o segundo argumento equivale às condições que queremos cumpridas e no terceiro argumento temos a variável que vai armazenar o resultado final.

```
resposta(X,Y,Z):- findall(X,Y,Z).
```

Após a definição dos critérios conjugamos todas essas condições num só predicado. Assim o predicado "fase1"apresenta-nos o ID dos utentes que cumprem as condições para serem vacinadas nesta fase. Para atingir esse resultado, fomos concatenando as várias listas resultantes dos predicados acima. Definimos ainda o predicado "repetidos"que retira da lista os elementos repetidos devolvendo, assim, a lista final.

```
fase1(R) :- idosos(I), doentes(D),
    concat(I,D,L),profmedico(M),concat(M,L,X),profenfermeiro(J),concat(X,J,Y),profenfermeira(W),
    concat(W,Y,Z), profmedica(C), concat(Z,C,S),repetidos(S,R).
```

**Output 1:**
?- fase1(X).
X = [6, 2, 4, 5, 1, 7]

Assim o predicado "fase1"apresenta a lista de utentes que devem ser vacinados durante esta fase.
Na segunda fase devem ser vacinados os restantes utentes registados. Assim, utilizamos o predicado "resposta"para percorrer todos os utentes registados e excluímos aqueles que pertecem à primeira fase.

```
fase2(R) :- resposta((U),(utente(U,_,_,_,_,_,_,_,_,_), fase1(R),
     nao(pertence(U,R))),L), repetidos(L,R).
```

**Output 2:**
?- fase2(X).
X = [3, 10, 8, 9]

Assim o predicado "fase2"apresenta a lista de utentes que devem ser vacinados durante esta fase.

## 4.2 Identificar pessoas vacinadas

Consideramos pessoas vacinadas todas aquelas que já inciaram o processo de vacinação, isto é, têm pelo menos um registo no predicado "vacinacao_covid".

Para obtermos uma lista com os utentes vacinados utilizamos o predicado resposta para guardar em L a lista dos IDs de utentes da base de conhecimento. Em seguida criamos o predicado "utenteVac". Através deste predicados verificamos se os IDs dos utentes têm alguma vacina registada. Isto é feito utilizando novamente o predicado "resposta".

```
vacinados(R) :- resposta(U, utente(U,_,_,_,_,_,_,_,_,_),L),
                  utenteVac(L,R).

utenteVac([U],R) :- utenteVacinado(U,R).
utenteVac([U|US],R) :- utenteVacinado(U,F),
        utenteVac(US,P),
        concat(F,P,R).

utenteVacinado(U,R) :- resposta(U, vacinacao_covid(_,U,_,_,_),L),
           repetidos(L,R).
```

**Output:**
?- vacinados(R).
R = [1, 2, 3, 5, 10, 8]

Assim, obtivemos uma lista de seis utentes que já começaram o processo de vacinação.

## 4.3 Identificar pessoas não vacinadas

Para determinar os utentes não vacinados utilizamos o nosso predicado "resposta", para obter a lista dos utentes que não pertecem à lista dos vacinados, ou seja, ainda não iniciaram o processo de vacinação.

```
naovac(R) :- resposta((U,A,B,D,F),(utente(U,A,B,_,D,_,F,_,_,_),
    vacinados(R),nao(pertence(U,R))),L),
     repetidos(L,R).
```

5

**Output:**
?- naovac(R). R = [(4, 76590, 'Maria', 'maria@gmail.com', 'Rua do Bairro'), (6, 65975, 'Alexandra', 'alexandra@gmail.com', 'Rua Alto Mar'), (7, 45210, 'Cristiana', 'cristiana@gmail.com', 'Rua dos PeÃµes'), (9, 96203, 'Rafael', 'rafael@gmail.com', 'Rua da Boa Vontade')]

Neste caso, obtivemos um resultado de quatro utentes, em que para cada um temos as suas informações básicas, que ainda não iniciaram o processo de vacinação.

### 4.4 Identificar pessoas vacinadas indevidamente

Tal como referido anteriormente, consideramos que a primeira fase de vacinação tivesse lugar entre Janeiro (Mês 1) e Março (Mês 3) de 2021. Deste modo, consideramos que uma pessoa foi vacinada indevidamente quando o utente em questão foi vacinado entre estes meses mas não se encontra entre os utentes candidatos a vacinação na primeira fase por não cumprir nenhum dos requisitos impostos.

Para atingir este fim, utilizamos o predicado "vacinados"para verificar quais os utentes pertencentes à lista dos vacinados. Em seguida, criamos um predicado "vacinadosDentroData1fase", que verifica se o utente foi vacinado entre o priemiro e o terceiro mês de 2021. Por último, verificamos se o utente não pertence ao grupo da fase 1 de vacinação.

```
mal(R) :- resposta((U,NS,N), (utente(U,NS,N,_,_,_,_,_,_,_),vacinados(V), pertence(U,V),
    vacinadosDentroData1fase(Y), pertence(U,Y),fase1(X), nao(pertence(U,X))),L),
    repetidos(L,R).

vacinadosDentroData1fase(R) :-
    resposta(U,(utente(U,_,_,_,_,_,_,_,_,_),vacinacao_covid(_,U,D-M-A,_,_), M>=1, M<4),R).
```

**Output**:
?- mal(X).
X = [(3, 12396, 'Carlota'), (10, 75362, 'Lisandro')] .

Tal como podemos verificar na nossa base de conhecimento, existem apenas dois utentes mal vacinados. O utente de ID 3 começou o processo de vacinação em fevereiro durante a fase 1 quando não é profissional de saúde, tem mais de 50 anos ou tem doenças associadas.
O utende de iD 10 foi vacinado em Janeiro, também sem cumprir os critérios da primeira fase.

### 4.5 Identificar pessoas não vacinadas e que são candidatas a vacinação

Para cumprir este objetivo procuramos saber quais eram os utentes da nossa base de conhecimento que pertenciam à lista dos não vacinados, mas que ao mesmo tempo cumprissem os requisitos necessários que lhes permitia terem sido vacinados na primeira fase, sendo que evem agora ter acesso prioritário à vacina.

Assim, o predicado "candidatas", deve retornar uma lista com o ID dos utentes que durante a segunda fase terão prioridade em relação aos restantes utilizadores.
Tivemos ainda de criar o predicado "naovacID", visto que o nosso "naovac"(que fornece a lista de utentes não vacinados) não devolve apenas o ID e portanto a função pertence não conseguia comparar o ID.

```
candidatas(R) :- (resposta(U,(utente(U,_,_,_,_,_,_,_,_,_),
    naovacID(X),fase1(V),pertence(U,X),pertence(U,V)),L),repetidos(L,R)).


naovacID(R) :- resposta(U,(utente(U,_,_,_,_,_,_,_,_,_), vacinados(R),nao(pertence(U,R))),L),
    repetidos(L,R).
```

**Output:**
?- candidatas(X).
X = [4, 6, 7]

Neste exemplo que aqui apresentamos, dá-nos diretamente os ID's dos utentes em causa.

### 4.6 Identificar pessoas a quem falta a segunda toma da vacina

Para obtermos a lista de utentes que não tinham tomado a segunda dose da vacina, utilizamos o predicado "resposta". A partir deste predicado procuramos obter uma lista com o o número de utente (U) e o nome do utente (N), dos utentes que têm a primeira dose da vacina registada, mas não tem registo de uma segunda vacina.

```
primeiratoma(R) :- resposta((U,N),(utente(U,_,N,_,_,_,_,_,_,_),
    vacinacao_covid(_,U,_,_,1),nao(vacinacao_covid(_,U,_,_,2))),R).
```

**Output:**
?- primeiratoma(X).
X = [(1, 'Carlos'), (2, 'Rui'), (5, 'Jorge'), (10, 'Lisandro'), (8, 'Sílvio')].

Como podemos observar, há cinco utentes, Carlos, Rui, Jorge, Lisandro e Sílvio, que apenas tomaram a primeira dose da vacina.

### 4.7 Desenvolver um sistema de inferência capaz de implementar os mecanismos de raciocínio inerentes a estes sistemas.

Neste sistema, a questão é um predicado que queremos validar. Assim, podemos verificar se um certo membro do staff ou um centro existe na base de conhecimento, sendo que será devolvido verdadeiro ou false.

```
inferencia(Questao, falso) :- -Questao.
inferencia(Questao,verdadeiro) :- Questao.
```

**Output**
?- inferencia(staff(1,1,'Rute Silva','rute@gmail.com'),R).
R = verdadeiro.
?- inferencia(staff(6,1,'Vania Silva','vania@gmail.com'),R).
**false.**

# 5 Funcionalidades Extra

## 5.1 Evolução da base de conhecimento

### 5.1.1 Registo de Utentes

Para registar utentes utilizamos o predicado "evolução". No entanto, para introduzir um novo utente na base de conhcimento, tivemos que garantir que não era possível introduzir um utente com o mesmo ID ou Nº de segurança social que outro já existente. Para isto utilizamos dois invariantes.

```
registaUtente(ID,A,B,C,D,E,F,G,H,I) :- evolucao(utente(ID,A,B,C,D,E,F,G,H,I)).


evolucao(T) :- resposta(I, +T::I, L),
               insercao(T),
               teste(L).

insercao(T) :- assert(T).
insercao(T) :- retract(T), !, fail.

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


%Nao permte que seja adicionada informao repetida (utente com o mesmo ID)
+utente(ID,A,B,C,D,E,F,G,H,I)::(resposta(ID,(utente(ID,_,_,_,_,_,_,_,_,_)),L),
                                comprimento(L,R),
                                R==1).

%Nao deixa dois utentes ter o mesmo nmero de segurana social
+utente(ID,S,B,C,D,E,F,G,H,I)::(resposta(S,(utente(_,S,_,_,_,_,_,_,_,_)),L),
                                comprimento(L,R),
                                R==1).
```

**Output:**

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

utente(1, 23456, 'Carlos', 24-4-1999, 'carlos@gmail.com', '966345781', 'Rua do FalcÃ£o', [Renite], 'Professor', 1).
utente(2, 23906, 'Rui', 13-3-1987, 'rui@gmail.com', '989349235', 'Rua da Esquina', [], 'MÃ©dico', 1).
utente(3, 12396, 'Carlota', 20-9-2005, 'carlota@gmail.com', '963215637', 'Rua Alto Mar', [], 'Estudante', 1).
utente(4, 76590, 'Maria', 25-9-1935, 'maria@gmail.com', '934674129', 'Rua do Bairro', [], 'Carpinteira', 1).
utente(5, 56851, 'Jorge', 29-5-1955, 'jorge@gmail.com', '962458965', 'Rua de S.Bento da VÃ¡rzea', [], 'PsicÃ³loga', 1).
utente(6, 65975, 'Alexandra', 18-2-2009, 'alexandra@gmail.com', '912358957', 'Rua Alto Mar', [], 'Enfermeira', 2).
utente(10, 75362, 'Lisandro', 5-6-2005, 'lisandro@gmail.com', '963030147', 'Rua Lisboa', [], 'Estudante', 2).
utente(7, 45210, 'Cristiana', 18-8-2012, 'cristiana@gmail.com', '932013663', 'Rua dos PeÃµes', [Alergias, Bronquite], 'Tradutor', 2).
utente(8, 78563, 'SÃlvio', 14-11-2000, 'silvio@gmail.com', '932456901', 'Rua AntÃ³nio Cardodo', [], 'Eletricista', 2).
utente(9, 96203, 'Rafael', 3-10-1977, 'rafael@gmail.com', '963124032', 'Rua da Boa Vontade', [], 'Engenheira', 2).

**true.**

?- registaUtente(12, 5669, 'Renato', 5-5-1986, 'renato@gmail.com','965230125','Rua de Barros', [], 'Tradutor',2).

**true.**

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

utente(1, 23456, 'Carlos', 24-4-1999, 'carlos@gmail.com', '966345781', 'Rua do FalcÃ£o', [Renite], 'Professor', 1).
utente(2, 23906, 'Rui', 13-3-1987, 'rui@gmail.com', '989349235', 'Rua da Esquina', [], 'MÃ©dico', 1).
utente(3, 12396, 'Carlota', 20-9-2005, 'carlota@gmail.com', '963215637', 'Rua Alto Mar', [], 'Estudante', 1).
utente(4, 76590, 'Maria', 25-9-1935, 'maria@gmail.com', '934674129', 'Rua do Bairro', [], 'Carpinteira', 1).
utente(5, 56851, 'Jorge', 29-5-1955, 'jorge@gmail.com', '962458965', 'Rua de S.Bento da VÃ¡rzea', [], 'PsicÃ³loga', 1).
utente(6, 65975, 'Alexandra', 18-2-2009, 'alexandra@gmail.com', '912358957', 'Rua Alto Mar', [], 'Enfermeira', 2).
utente(10, 75362, 'Lisandro', 5-6-2005, 'lisandro@gmail.com', '963030147', 'Rua Lisboa', [], 'Estudante', 2).
utente(7, 45210, 'Cristiana', 18-8-2012, 'cristiana@gmail.com', '932013663', 'Rua dos PeÃµes', [Alergias, Bronquite], 'Tradutor', 2).
utente(8, 78563, 'SÃlvio', 14-11-2000, 'silvio@gmail.com', '932456901', 'Rua AntÃ³nio Cardodo', [], 'Eletricista', 2).
utente(9, 96203, 'Rafael', 3-10-1977, 'rafael@gmail.com', '963124032', 'Rua da Boa Vontade', [], 'Engenheira', 2).
utente(12, 5669, 'Renato', 5-5-1986, 'renato@gmail.com', '965230125', 'Rua de Barros', [], 'Tradutor', 2).

**true.**

Tal como foi possível verificar, o novo utente é registado. No entanto se tentarmos acrescentar um utente com o ID 8 (já existente) ou o mesmo número se seguranla social do utente 1, verificamos que o registo não é possível.

?- registaUtente(8, 5669, 'Renato', 5-5-1986, 'renato@gmail.com','965230125','Rua de Barros', [], 'Tradutor',2).
**false.**
?- registaUtente(12, 23456, 'Renato', 5-5-1986, 'renato@gmail.com','965230125','Rua de Barros', [], 'Tradutor',2).
**false.**

### 5.2 Regista Staff

Para registar novo staff seguimos a mesma lógica com o mesmo predicado "evolução"e fizemos um invariante que não permite que seja registado um novo menbro do staff com o mesmo ID.

```
registaStaff(A,B,C,D) :- evolucao(staff(A,B,C,D)).


% No  permte que seja adicionada informao repetida (staff com o mesmo ID)
+staff(A,B,C,D)::(resposta(A,(staff(A,_,_,_)), L),
                  comprimento(L,R),
                  R==1).
```

**Output:**
?- listing(staff).
:- dynamic staff/4.
staff(1, 1, 'Rute Silva', 'rute@gmail.com').
staff(2, 1, 'Santiago Rua', 'santiago@gmail.com').
staff(3, 2, 'Vitor Castro', 'vitor@gmail.com').
staff(4, 2, 'Catarina Coutada', 'catarina@gmail.com').

**true.**

?- registaStaff(5,2,'Rita Santos', 'rita@gmail.com').
**true.**

?- registaStaff(3,2,'Rita Santos', 'rita@gmail.com').
**false.**

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

staff(1, 1, 'Rute Silva', 'rute@gmail.com').
staff(2, 1, 'Santiago Rua', 'santiago@gmail.com').
staff(3, 2, 'Vitor Castro', 'vitor@gmail.com').
staff(4, 2, 'Catarina Coutada', 'catarina@gmail.com').
staff(5, 2, 'Rita Santos', 'rita@gmail.com').

**true.**

### 5.2.1 Registar novos centros

Para registar novos centros, para além do uso de "evolução", criamos também um invariante que impede o registo de um novo centro com um ID de centro de sáude já existente na nossa base de conhecimento.

```
registaCentro(A,B,C,D,E) :- evolucao(centro_saude(A,B,C,D,E)).

% No  permte que seja adicionada informao repetida (centro com o mesmo ID)
+centro_saude(A,B,C,D,E)::(resposta((A,B,D), (centro_saude(A,B,_,D,_)), L),
                        comprimento(L,R),
                        R==1).
```

**Output:**
?- listing(centro_saude).:- dynamic centro_saude/5.

centro_saude(1, 'Centro de SaÃºde de Barcelos', 'Rua Alta', '253234674', 'centro1@gmail.com').
centro_saude(2, 'Hospital De Braga', 'Rua das Paredes ', '253523489', 'centro2@gmail.com').

**true.**

?- registaCentro(3, 'Novo Centro', 'Rua Santo Antonio', '45236985', 'novo@gmail.com').
**true.**

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

centro_saude(1, 'Centro de SaÃºde de Barcelos', 'Rua Alta', '253234674', 'centro1@gmail.com').
centro_saude(2, 'Hospital De Braga', 'Rua das Paredes ', '253523489', 'centro2@gmail.com').
centro_saude(3, 'Novo Centro', 'Rua Santo Antonio', '45236985', 'novo@gmail.com').

**true.**
?- registaCentro(3, 'Novo Centro', 'Rua Santo Antonio', '45236985', 'novo@gmail.com').
**false.**

### 5.2.2 Registar Novos Vacinados

Para registar novos vacinados usamos o predicado "evolução"e criamos dois invariantes: um deles não permite que um utente tenha mais do que dois registos de vacina e outro que garante que a segunda toma é feita apenas depois da primeira.

```
registaVacina(A,B,C,D,E) :-evolucao(vacinacao_covid(A,B,C,D,E)).


%A primeira toma  antes da segunda
%Se o registo for da segunda toma obtemos a data da primeira dose e testamos atravs do
    "testaData" se a dose introduzida  posterior.
+vacinacao_covid(_,U,D-M-A,_,T)::(T == 1; T == 2, (vacinacao_covid(_,U,D1-M1-A1,_,1),
                                testaData(D,M,A,D1,M1,A1))).


%Apenas dois registos por utente
% Atravs  de um "resposta" verifica quantos registos o utente tem na base de conhecimento e
    no permite que seja maior que 2.
+vacinacao_covid(_,U,_,_,T)::((resposta((U, T),(vacinacao_covid(_,U,_,_,T)),L),
                            comprimento(L,N),
                            N =< 2)).


testaData(D,M,A,D1,M1,A1) :- (A > A1);(A == A1,M > M1);(A == A1,M == M1,D>D1).
```

**Output:**
?- listing(vacinacao_covid).
:- dynamic vacinacao_covid/5.

vacinacao_covid(1, 2, 6-3-2021, Vacina2, 1).
vacinacao_covid(2, 5, 6-1-2021, Vacina1, 1).
vacinacao_covid(2, 1, 20-4-2021, Vacina1, 1).
vacinacao_covid(2, 3, 5-2-2021, Vacina1, 1).
vacinacao_covid(2, 3, 5-4-2021, Vacina1, 2).
vacinacao_covid(3, 10, 12-1-2021, Vacina1, 1).
vacinacao_covid(4, 8, 5-8-2021, Vacina2, 1).

**true.**

?- registaVacina(2,3,8-6-2021,Vacina1,2).
**false.**

?- registaVacina(2,10,8-6-2021,Vacina1,2).
**true.**

?- registaVacina(2,8,20-2-2021,Vacina1,2).
**false.**

?- listing(vacinacao_covid).
:- dynamic vacinacao_covid/5.

vacinacao_covid(1, 2, 6-3-2021, Vacina2, 1).
vacinacao_covid(2, 5, 6-1-2021, Vacina1, 1).
vacinacao_covid(2, 1, 20-4-2021, Vacina1, 1).
vacinacao_covid(2, 3, 5-2-2021, Vacina1, 1).
vacinacao_covid(2, 3, 5-4-2021, Vacina1, 2).
vacinacao_covid(3, 10, 12-1-2021, Vacina1, 1).
vacinacao_covid(4, 8, 5-8-2021, Vacina2, 1).
vacinacao_covid(2, 10, 8-6-2021, Vacina1, 2).

**true.**

### 5.2.3 Remover Utentes

No caso da remoção de utentes, optamos por também remover os registos das vacinas desse mesmo utente.
Assim, vamos primeiramente obter em R a lista de vacinações daquele utente que pretendemos remover da base de conhecimento. De seguida criamos o "removeVacinacoes"que recebe a lista de vacinações do utente e remove-as uma a uma. Apenas depois da remoção das vacinações removemos o utente através do predicado "retroceder".

```
removeUtente(U) :- (resposta((A,U,B,C,D),(vacinacao_covid(A,U,B,C,D)),R),
                  removeVacinacoes(R),
                  retroceder(utente(U,_,_,_,_,_,_,_,_,_))).


removeVacinacoes([]).
removeVacinacoes([(A,U,B,C,D)|US]) :- removeVacinacao(U), removeVacinacoes(US).


retroceder(E) :- resposta(I,-E::I,L),
                teste(L),
                 remove(E).


remove(T) :- retract(T).
remove(T) :- assert(T),!,fail.
```

**Output:**
?- listing(utente).
:- dynamic utente/10.

utente(1, 23456, 'Carlos', 24-4-1999, 'carlos@gmail.com', '966345781', 'Rua do FalcÃ£o', [Renite], 'Professor', 1).
utente(2, 23906, 'Rui', 13-3-1987, 'rui@gmail.com', '989349235', 'Rua da Esquina', [], 'MÃ©dico', 1).
utente(3, 12396, 'Carlota', 20-9-2005, 'carlota@gmail.com', '963215637', 'Rua Alto Mar', [], 'Estudante', 1).
utente(4, 76590, 'Maria', 25-9-1935, 'maria@gmail.com', '934674129', 'Rua do Bairro', [], 'Carpinteira', 1).
utente(5, 56851, 'Jorge', 29-5-1955, 'jorge@gmail.com', '962458965', 'Rua de S.Bento da VÃ¡rzea', [], 'PsicÃ³loga', 1). utente(6, 65975, 'Alexandra', 18-2-2009, 'alexandra@gmail.com', '912358957', 'Rua Alto Mar', [], 'Enfermeira', 2).
utente(10, 75362, 'Lisandro', 5-6-2005, 'lisandro@gmail.com', '963030147', 'Rua Lisboa', [], 'Estudante', 2).

utente(7, 45210, 'Cristiana', 18-8-2012, 'cristiana@gmail.com', '932013663', 'Rua dos PeÃµes', [Alergias, Bronquite], 'Tradutor', 2).
utente(8, 78563, 'SÃlvio', 14-11-2000, 'silvio@gmail.com', '932456901', 'Rua AntÃ³nio Cardodo', [], 'Eletricista', 2).
utente(9, 96203, 'Rafael', 3-10-1977, 'rafael@gmail.com', '963124032', 'Rua da Boa Vontade', [], 'Engenheira', 2).

**true.**

?- listing(vacinacao$_c ovid$).
: $-dynamicvacinacao_c ovid/5$.

vacinacao$_c ovid(1, 2, 6-3-2021, Vacina2, 1)$.
$vacinacao_c ovid(2, 5, 6-1-2021, Vacina1, 1)$.
$vacinacao_c ovid(2, 1, 20-4-2021, Vacina1, 1)$.
$vacinacao_c ovid(2, 3, 5-2-2021, Vacina1, 1)$.
$vacinacao_c ovid(2, 3, 5-4-2021, Vacina1, 2)$.
$vacinacao_c ovid(3, 10, 12-1-2021, Vacina1, 1)$.
$vacinacao_c ovid(4, 8, 5-8-2021, Vacina2, 1)$.

**true.**

?- removeUtente(3).
**true.**

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

utente(1, 23456, 'Carlos', 24-4-1999, 'carlos@gmail.com', '966345781', 'Rua do FalcÃ£o', [Renite], 'Professor', 1).
utente(2, 23906, 'Rui', 13-3-1987, 'rui@gmail.com', '989349235', 'Rua da Esquina', [], 'MÃ©dico', 1).
utente(4, 76590, 'Maria', 25-9-1935, 'maria@gmail.com', '934674129', 'Rua do Bairro', [], 'Carpinteira', 1).
utente(5, 56851, 'Jorge', 29-5-1955, 'jorge@gmail.com', '962458965', 'Rua de S.Bento da VÃ¡rzea', [], 'PsicÃ³loga', 1).
utente(6, 65975, 'Alexandra', 18-2-2009, 'alexandra@gmail.com', '912358957', 'Rua Alto Mar', [], 'Enfermeira', 2).
utente(10, 75362, 'Lisandro', 5-6-2005, 'lisandro@gmail.com', '963030147', 'Rua Lisboa', [], 'Estudante', 2).
utente(7, 45210, 'Cristiana', 18-8-2012, 'cristiana@gmail.com', '932013663', 'Rua dos PeÃµes', [Alergias, Bronquite], 'Tradutor', 2).
utente(8, 78563, 'SÃlvio', 14-11-2000, 'silvio@gmail.com', '932456901', 'Rua AntÃ³nio Cardodo', [], 'Eletricista', 2).
utente(9, 96203, 'Rafael', 3-10-1977, 'rafael@gmail.com', '963124032', 'Rua da Boa Vontade', [], 'Engenheira', 2).

**true.**

?- listing(vacinacao$_c ovid$).
: $-dynamicvacinacao_c ovid/5$.

vacinacao$_c ovid(1, 2, 6-3-2021, Vacina2, 1)$.
$vacinacao_c ovid(2, 5, 6-1-2021, Vacina1, 1)$.
$vacinacao_c ovid(2, 1, 20-4-2021, Vacina1, 1)$.

$vacinacao_covid(3, 10, 12-1-2021, Vacina1, 1)$.
$vacinacao_covid(4, 8, 5-8-2021, Vacina2, 1)$.

**true.**

### 5.2.4 Remover Staff

Para remover um membro do staff existente na nossa base de conhecimento utilizamos novamente o predicado "retroceder", já utilizado anteriormente para remover utentes.

```
removeStaff(ID) :- retroceder(staff(ID,B,C,D)).
```

**Output:**
?- listing(staff).
:- dynamic staff/4.

staff(1, 1, 'Rute Silva', 'rute@gmail.com'). staff(2, 1, 'Santiago Rua', 'santiago@gmail.com'). staff(3, 2, 'Vitor Castro', 'vitor@gmail.com'). staff(4, 2, 'Catarina Coutada', 'catarina@gmail.com').
**true.**
?- removeStaff(2). **true.**
?- listing(staff). :- dynamic staff/4.
staff(1, 1, 'Rute Silva', 'rute@gmail.com'). staff(3, 2, 'Vitor Castro', 'vitor@gmail.com'). staff(4, 2, 'Catarina Coutada', 'catarina@gmail.com').
**true.**

### 5.2.5 Remover Centros

A mesma lógica utilizada para remover o staff foi usada para remover os centros de saúde.

```
removeCentros(ID) :- retroceder(centro_saude(ID,B,C,D,E)).
```

**Output:**
?- listing(centro_saude).
:- dynamic centro_saude/5.

centro_saude(1, 'Centro de SaÃºde de Barcelos', 'Rua Alta', '253234674', 'centro1@gmail.com'). centro_saude(2, 'Hospital De Braga', 'Rua das Paredes ', '253523489', 'centro2@gmail.com').

**true.**

?- removeCentros(1).
**true.**

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

centro_saude(2, 'Hospital De Braga', 'Rua das Paredes ', '253523489', 'centro2@gmail.com').

**true.**

### 5.2.6 Remover Vacinações

Para remover vacinações recorremos novamente ao predicado "evolução".

```
removeVacinacao(ID):- retroceder(vacinacao_covid(A,ID,B,C,D)).
```

**Output**

?- listing(vacinacao$_covid$).
: $-dynamicvacinacao_covid/5$.

vacinacao$_covid(1, 2, 6-3-2021, Vacina2, 1)$.
$vacinacao_covid(2, 5, 6-1-2021, Vacina1, 1)$.
$vacinacao_covid(2, 1, 20-4-2021, Vacina1, 1)$.
$vacinacao_covid(2, 3, 5-2-2021, Vacina1, 1)$.
$vacinacao_covid(2, 3, 5-4-2021, Vacina1, 2)$.
$vacinacao_covid(3, 10, 12-1-2021, Vacina1, 1)$.
$vacinacao_covid(4, 8, 5-8-2021, Vacina2, 1)$.

**true.**

?- removeVacinacao(10).
**true.**

?- listing(vacinacao$_covid$).
: $-dynamicvacinacao_covid/5$.

vacinacao$_covid(1, 2, 6-3-2021, Vacina2, 1)$.
$vacinacao_covid(2, 5, 6-1-2021, Vacina1, 1)$.
$vacinacao_covid(2, 1, 20-4-2021, Vacina1, 1)$.
$vacinacao_covid(2, 3, 5-2-2021, Vacina1, 1)$.
$vacinacao_covid(2, 3, 5-4-2021, Vacina1, 2)$.
$vacinacao_covid(4, 8, 5-8-2021, Vacina2, 1)$.

**true.**

## 5.3 Pesquisar utentes por critérios de seleção

Utilizamos estes predicados para obter um utente da nossa base de conhecimento baseado na sua informação.

```
%Obtem um utente a partir do seu ID
utenteID(ID,R) :- resposta((ID,A,B,C,D,E,F,G,H,I),utente(ID,A,B,C,D,E,F,G,H,I),R).

%Obtem um utente a partir do seu N de segurana Social
utenteSocial(SO,R) :- resposta((ID,SO,B,C,D,E,F,G,H,I),utente(ID,SO,B,C,D,E,F,G,H,I),R).

%Obtem um utente a partir do seu nome
utenteNome(NO,R) :- resposta((ID,A,NO,C,D,E,F,G,H,I),utente(ID,A,NO,C,D,E,F,G,H,I),R).

%Obtem um utente a partir da sua data de nascimento
utenteNascimento(DA,R) :- resposta((ID,A,B,DA,D,E,F,G,H,I),utente(ID,A,B,DA,D,E,F,G,H,I),R).
```

```
%Obtem um utente a partir do seu email
utenteEmail(EM,R) :- resposta((ID,A,B,C,EM,E,F,G,H,I),utente(ID,A,B,C,EM,E,F,G,H,I),R).

%Obtem um utente a partir do seu telemovel
utenteTelemovel(TE, R) :- resposta((ID,A,B,C,D,TE,F,G,H,I),utente(ID,A,B,C,D,TE,F,G,H,I),R).

%Obtem um utente a partir da sua morada
utenteMorada(MO,R) :- resposta((ID,A,B,C,D,E,MO,G,H,I),utente(ID,A,B,C,D,E,MO,G,H,I),R).

%Obtem utentes de um centro
utenteCentro(CE,R) :- resposta((ID,A,B,C,D,E,F,G,H,CE),utente(ID,A,B,C,D,E,F,G,H,CE),R).
```

**Output**

?- utenteID(3,R). R = [(3, 12396, 'Carlota', 20-9-2005, 'carlota@gmail.com', '963215637', 'Rua Alto Mar', [], ..., ...)].

?- utenteCentro(1,R). R = [(1, 23456, 'Carlos', 24-4-1999, 'carlos@gmail.com', '966345781', 'Rua do FalcÃ£o', [...], ..., ...), (2, 23906, 'Rui', 13-3-1987, 'rui@gmail.com', '989349235', 'Rua da Esquina', ..., ...), (3, 12396, 'Carlota', 20-9-2005, 'carlota@gmail.com', '963215637', ..., ...), (4, 76590, 'Maria', ... - ... - 1935, 'maria@gmail.com', ..., ...), (5, 56851, 'Jorge', ... - ..., ..., ...)].

?- utenteMorada('Rua do Bairro',R). R = [(4, 76590, 'Maria', 25-9-1935, 'maria@gmail.com', '934674129', 'Rua do Bairro', [], ..., ...)].

## 5.4 Identificar quantos utentes um centro de sáude tem

Para identificar o número de utentes num dado centro criamos um predicado que recebe o número de ID desse mesmo centro. Em seguida e através do "resposta"coloca em L a lista de todos os utentes desse centro. Por último, através do predicado "comprimento", obtemos o comprimento da lista L na variável R.

```
utentePorCentro(C, R) :- (resposta((U,C), (utente(U,_,_,_,_,_,_,_,_,_)),L),
                         comprimento(L,R)).
```

**Output:**

?- utentePorCentro(2,R).
R = 5.

Por este exemplo, conseguimos perceber que, se procurarmos pelo número de utentes no centro com id igual a 2, obtemos uma resposta de 5.

### 5.5 Identificar os utentes vacinados por um membro do staff

Para obter os utentes vacinados por um certo membro do staff utilizamos novamente uma "resposta", que pesquisa na nossa base de conhecimento os vários registos de vacinas existentes. Assim, obtemos o ID e o nome dos utentes que têm um registo de vacina com o staff identificado por S.

```
utentePorStaff(S, R) :- (resposta((ID,N),(vacinacao_covid(S,ID,_,_,_),
    utente(ID,_,N,_,_,_,_,_,_,_,_)),L),repetidos(L,R)).
```

**Output:**
?- utentePorStaff(2,R).
R = [(5, 'Jorge'), (1, 'Carlos'), (3, 'Carlota')]

Aqui, ao indicarmos o id de um membro do staff igual a 2, temos que este vacinou três utentes diferentes.

### 5.6 Identificar utentes que já terminaram a vacinação

De forma a termos acesso aos utentes que já terminaram totalmente o processo de vacinação (já tomaram ambas as doses) utilizamos novamente o predicado "resposta".

```
terminou(R) :- resposta((U,N), (utente(U,_,N,_,_,_,_,_,_,_,_),vacinados(V), pertence(U,V),
    vacinacao_covid(_,U,_,_,2)),L),repetidos(L,R).
```

**Output**
?- terminou(X).
X = [(3, 'Carlota')]

Deste modo, sabemos que quem não terminou todo o processo de vacinação foi apenas a Carlota, cujo ID é 3.

## 6 Conclusão

Primeiramente, consideramos que foi um enorme desafio para nós enquanto grupo desenvolver um projeto em programação lógica, utlizando o PROLOG, uma vez que, para além dos exercícios resolvidos durante as aulas da UC, nunca tinhamos trabalhado em algo sequer parecido.
Assim, ao utilizarmos esta linguagem, tinhamos como objetivo a representação de um sistema que conseguisse caracterizar o ambiente de conhecimento e raciocínio necessário para a resolução do nosso problema, que está relacionado com a criação de um plano de vacinação.
Por fim, apesar das dificuldades, estamos satisfeitos com o resultado final e não temos dúvidas de que este prjeto foi muito útil para nos deixar mais à vontade com a linguagem e mais confiantes para os trabalhos que irão vir no futuro.