Universidade do Minho
Escola de Engenharia

Trabalho Prático - Grupo 32

Vacinação da Populacão Portuguesa no contexto COVID-19

Sistemas de Representação de Conhecimento e Raciocínio

Bruno Filipe de Sousa Dias A89583
Luís Enes Sousa A89597
Pedro Miguel de Soveral Pacheco Barbosa A89529

9 de abril de 2021

Conteúdo

1 Resumo

Este relatório foi elaborado no âmbito da unidade curricular de Sistemas de Representação de Conhecimento e Raciocínio em conjunto com a resolução do trabalho de grupo prático proposto pelos docentes da unidade curricular. A realização deste exercício pretende motivar os alunos para a utilização da linguagem de programação PROLOG, no âmbito da representação de conhecimento e construção de mecanismos de raciocínio para a resolução de problemas.

Primeiramente foi feita uma introdução do projeto, seguida da explicação do raciocínio utilizado pelo grupo tanto para a resolução das funcionalidades especificamente pedidas no enunciado como das extras adicionadas. Por último, foi feita uma apreciação critíca da globalidade do trabalho.

2 Introdução

Este relatório surge na elaboração do exercício em grupo proposto na unidade curricular de Sistemas de Representação de Conhecimento e Raciocínio. Este exercício tem como base a programação em lógica cujo objetivo é o desenvolvimento de um programa assente em factos e em predicados que, por sua vez, estão associados a factos e regras. Assim, foi utilizada a linguagem de programação PROLOG para ser possível atingir esse mesmo objetivo.

O trabalho prático consiste em desenvolver um sistema de representação de conhecimento e raciocínio com capacidade para caracterizar um universo de discurso na área da vacinação da população portuguesa no contexto COVID-19 que estamos a viver.

A base de conhecimento do sistema é constituída por utentes, staff, centros de saúde e vacinações ao covid e este deverá ser capaz de definir diferentes fases de vacinação, definindo critérios de inclusão de utentes nas diferentes fases consoante diferentes critérios de seleção, identificar pessoas não vacinadas e pessoas vacinadas, identificar pessoas vacinadas indevidamente e pessoas não vacinadas que são candidatas a vacinação, identificar pessoas a quem falta a segunda toma da vacina e desenvolver um sistema de inferência capaz de implementar mecanismos de raciocínio inerentes a estes sistemas.

3 Motivação e Objetivos

Depois de adquiridos os conhecimentos base de programação em lógica e de PROLOG a resolução deste trabalho prático motivou a consolidação de conhecimentos lecionados nas aulas da unidade curricular. Todos estes conhecimentos foram utilizados com o objetivo de desenvolver um sistema de vacinações de covid feitos por staffs a utentes e através deste conhecimento ser possível solucionar problemas com este relacionados.

4 Estrutura do Relatório

5 Descrição do trabalho e Análise de Resultados

5.1 Base de Conhecimento

A base de conhecimento define a base de dados ou de conhecimento adquirido sobre um determinado assunto. Desta forma, foi necessário começarmos com uma base de conhecimento inicial de modo a podermos logo testar a nossa informação sem termos de inserir conhecimento. Deste modo, começamos por preencher a mesma de acordo com as restrições dadas no enunciado do projeto.

- Utentes

  Um utente é identificado pelo seu ID único, número de Segurança Social, nome, data de nasicmento, email, telefone, morada, profissão, lista com as suas doenças crónicas e o ID do centro de saúde ao qual se encontra associado. Assim, a assinatura de um Utente será:

  utente(Id,Nss,N,D,E,T,M,P,DC,CS).

  Exemplo da base de conhecimento criada para os utentes.

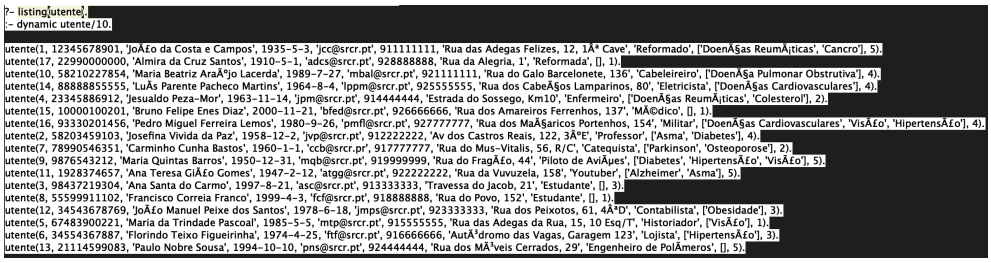


Figura 1: Base de Conhecimento Utentes

- Staff

  Um staff é identificado pelo seu ID único, ID do centro de saúde ao qual se encontra associado, nome e email. Assim, a assinatura de um Staff será:

  staff(Id,Idcs,N,E).

  Exemplo da base de conhecimento criada para os staff.

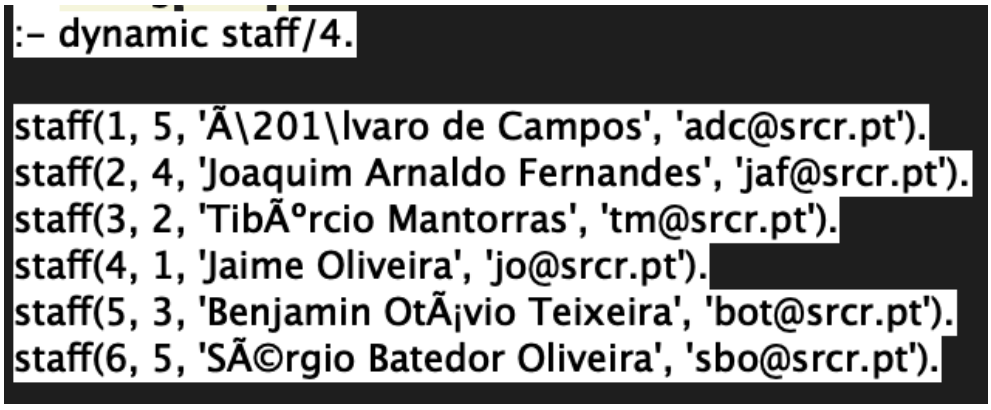


Figura 2: Base de Conhecimento Staffs

- Centros de Saúde

  Um centro de saúde é identificado pelo seu ID único, nome, morada, telefone e email. Assim, a assinatura de um Centro de Saúde será:

  centrosaude(Id,N,M,T,E).

  Exemplo da base de conhecimento criada para os Centro de Saúde.

```
:- dynamic centro_saude/5.

centro_saude(1, 'Centro de SaÃºde Mem Martins', 'Rua Pedro Bispo, 725', 951111111, 'csmm@srcr.pt').
centro_saude(2, 'Centro de SaÃºde Chelas', 'Rua Samuel Mira, 666', 952222222, 'csc@srcr.pt').
centro_saude(3, 'Centro de SaÃºde SÃ£o VÃtor', 'Rua dos CabeÃ§udos, 80', 953333333, 'cssv@srcr.pt').
centro_saude(4, 'Centro de SaÃºde Amares', 'Rua da Lagars, 137', 954444444, 'csa@srcr.pt').
centro_saude(5, 'Centro de SaÃºde Viena do Forte', 'Rua Anthony FeijÃ³, 154', 955555555, 'csvf@srcr.pt').
```

Figura 3: Base de Conhecimento Centros de Saúde

- Vacinações contra COVID-19

  Uma vacinação ao COVID-19 é identificada pelo ID do staff que a vai dar, ID do utente que a irá receber, data, marca da vacina e o número da toma(1 ou 2).

  Assim, a assinatura de uma Vacinação contra o COVID-19 será:

  vacinacaoCovid(Ids,Idu,D,V,T).

  Exemplo da base de conhecimento criada para as vacinações covid.

```
:- dynamic vacinacao_Covid/5.

vacinacao_Covid(1, 1, 2021-1-5, 'Pfizer', 1).
vacinacao_Covid(1, 17, 2021-1-5, 'Pfizer', 1).
vacinacao_Covid(1, 10, 2021-1-5, 'Pfizer', 1).
vacinacao_Covid(1, 14, 2021-1-5, 'Pfizer', 1).
vacinacao_Covid(6, 1, 2021-1-19, 'Pfizer', 2).
vacinacao_Covid(2, 2, 2021-1-20, 'Pfizer', 1).
vacinacao_Covid(3, 7, 2021-5-24, 'Astrazeneca', 1).
vacinacao_Covid(3, 7, 2021-5-15, 'Astrazeneca', 2).
vacinacao_Covid(6, 8, 2021-8-15, 'Astrazeneca', 1).
vacinacao_Covid(2, 6, 2021-6-1, 'Pfizer', 1).
vacinacao_Covid(2, 6, 2021-6-16, 'Pfizer', 2).
vacinacao_Covid(1, 13, 2021-7-31, 'Astrazeneca', 1).
```

Figura 4: Base de Conhecimento Vacinações contra COVID-19

5.2 Invariantes

Os **Invariantes** têm como objetivo garantir que o conhecimento é consistente e não repetido, sendo por estas razões essenciais para este trabalho. Desta forma, implementamos dois tipos de invariantes: invariantes para a inserção e invariantes para a remoção de conhecimento.

Para definir os invariantes necessários foi necessário definir o predicado procura que executa o predicado pré definido do PROLOG *findall* e que encontra todas as soluções que satisfazem um ou mais predicados. Este predicado foi extremamente necessário para todos os predicados elaborados ao longo do projeto.

De seguida apresentamos o predicado **procura**.

```
% ------------------------------------------------------------------------------------------------
% Predicado que permite a procura de Conhecimento.
% Extensão do predicado procura: Termo,Predicado,Lista -> {V,F}.
% ------------------------------------------------------------------------------------------------

procura(T,P,L) :-
    findall(T,P,L).
```

Figura 5: Predicado Procura

Assim, recorrendo a este predicado foi possível criar todos os invariantes necessários quer para a inserção quer para a remoção de conhecimento.

Para o conjunto de **invariantes associados à inserção**, encontramos invariantes que impedem a repetição de conhecimento (utentes, staffs, centros de saúde ou vacinações). Deste modo, após a inserção de um determinado conhecimento, o predicado verifica se o seu ID é único, caso contrário, retira-o da base de conhecimento.

Apresentamos assim os diferentes invariantes explicados em cima.

```
% ------------------------------------------------------------------------------------------------
% ------------------------------------Invariantes para a inserção---------------------------------
% ------------------------------------------------------------------------------------------------

+utente(Id,_,_,_,_,_,_,_,_):: (procura(Id,utente(Id,_,_,_,_,_,_,_,_),L),
                               comprimento(L,R),
                               R == 1).

+centro_saude(Id,_,_,_,_):: (procura(Id,centro_saude(Id,_,_,_,_),L),
                             comprimento(L,R),
                             R == 1).

+staff(Id,_,_,_):: (procura(Id,staff(Id,_,_,_),L),
                    comprimento(L,R),
                    R == 1).

+vacinacao_Covid(Ids,Idu,D,_,_):: (procura((Ids,Idu,D),vacinacao_Covid(Ids,Idu,D,_,_),L),
                                   comprimento(L,R),
                                   R == 1).
```

Figura 6: Invariantes para a inserção

Ainda nos invariantes de inserção adicionamos um último que só permite que sejam inseridas vacinações contra COVID-19 caso o utente e o staff a esta relacionados existam.

```
% Ids do utente e do staff têm de existir para inserir uma vacinação.
+vacinacao_Covid(Ids,Idu,D,V,T):: (procura((Ids,Idu),(utente(Idu,_,_,_,_,_,_,_,_,_),staff(Ids,_,_,_)),R),
                                   comprimento(R,N), N==1).
```

Figura 7: Invariante para a inserção de uma vacinação

Para além destes invariantes, foi ainda preciso adicionar mais alguns de modo a garantir que qualquer inserção feita para uma vacinação ao COVID-19 teria de respeitar certas regras. Desta forma, adicionamos três invariantes que garantem que um utente está a tomar sempre vacinas da mesma marca, Pfizer ou Astrazeneca, que toma a segunda toma da vacina após a primeira toma da mesma e que a data de toma da segunda dose tem de ser posterior à toma da primeira fase.

```
% tomar a mesma vacina
+vacinacao_Covid(_,Idu,_,V,T):: (
                                (T==1;(procura(Idu,(vacinacao_Covid(_,Idu,_,V,_)),R),
                                comprimento(R,N),N==2 )
                                )).

% toma a segunda depois da primeira
+vacinacao_Covid(_,Idu,_,_,T):: (
                                (utente(Idu,_,_,_,_,_,_,_,_,_)),
                                qtdVacUtente(Idu,QT), T=<2,
                                T =:= QT
                                ).

% segunda dose depois da primeira
+vacinacao_Covid(_,Idu,A-M-D,_,T)::(
                                    (T==1;
                                    (vacinacao_Covid(_,Idu,A2-M2-D2,_,1)),
                                    T==2,
                                    (D>D2,M==M2,A==A2;M>M2,A==A2;A>A2)
                                    )).
```

Figura 8: Invariante para garantir corretas inserções de vacinação

Para o conjunto de **invariantes associados à remoção**, foi necessário verificar que há conhecimento para remover.

Foi ainda necessário adicionar dois invariantes associados à remoção de conhecimento de modo a não ser possível remover um utente ou um staff se estes tiverem associados a alguma vacinação existente na base de conhecimento.

```
% ------------------------------------------------------------------------------------------
% ------------------------------------Invariantes para a remoção------------------------------------
% ------------------------------------------------------------------------------------------

-utente(Id,Nss,N,D,E,T,M,P,DC,CS):: (procura(Id,utente(Id,Nss,N,D,E,T,M,P,DC,CS),L),
                                     comprimento(L,R),
                                     R == 1).

-centro_saude(Id,N,M,T,E):: (procura(Id,centro_saude(Id,N,M,T,E),L),
                             comprimento(L,R),
                             R == 1).

-staff(Id,Idcs,N,E):: (procura(Id,staff(Id,Idcs,N,E),L),
                       comprimento(L,R),
                       R == 1).

-vacinacao_Covid(Ids,Idu,D,V,T):: (procura(Idu,vacinacao_Covid(Ids,Idu,D,V,T),L),
                                   comprimento(L,R),
                                   R == 1).
```

Figura 9: Invariantes para a remoção

```
% Não permitir a remoção de utentes e de staff se existirem vacinações de covid a eles associadas.
-utente(Id,Nss,N,D,E,T,M,P,DC,CS):: (procura(Id,vacinacao_Covid(Ids,Id,D,V,T),L),
                                     comprimento(L,R),
                                     R == 0).

-staff(Id,Idcs,N,E):: (procura(Id,vacinacao_Covid(Id,Idu,D,V,T),L),
                       comprimento(L,R),
                       R == 0).
```

Figura 10: Invariantes para a remoção de um utente ou um staff

5.3 Funções auxiliares

Nesta secção do relatório procederemos à explicação de algumas funções retiradas das aulas práticas ou feitas por nós que foram essenciais para a realização do trabalho prático.

O predicado **pertence** foi uma das funções mais úteis ao longo do trabalho.

```
% ------------------------------------------------------------------------------------------------
% Predicado que verifica se um elemento pertece a uma determinada lista.
% Extensão do predicado pertence: X, Lista -> {V,F}.
% ------------------------------------------------------------------------------------------------

pertence(H,[H|_T]).
pertence(X,[H|_T]) :-
    X \= H,
pertence(X,_T).
```

Figura 11: Predicado Pertence

Este predicado verifica se um certo elemente é igual ao elemento da cabeça de uma lista e caso não o seja verifica se este é igual a algum dos elementos da cauda da mesma. Se algum destes casos se verificar o elemento pertence à lista, caso contrário, não pertence.

De seguida focaremos no predicado **concat**.

```
% ------------------------------------------------------------------------------------------------
% Predicado que concatena duas listas.
% Extensão do predicado concat: Lista, Lista -> {V,F}.
% ------------------------------------------------------------------------------------------------

concat([], L, L).
concat([H|T], L, R) :- add(H, N, R), concat(T, L, N).
```

Figura 12: Predicado Concat

Este predicado como o próprio nome indica concatena duas listas. Basicamente, vai pegando no elemento que está à cabeça de uma das listas e adiciona-o à cabeça da outra lista até a primeira lista ficar vazia.

Seguem-se as funções **nao** e **comprimento**, sendo que a primeira verifica o contrário de um predicado e a segunda calcula o comprimento de uma determinada lista.

```
% ------------------------------------------------------------------------------------------------------------
% Predicado que verifica o contrário de outro predicado.
% Extensão do predicado não: Q -> {V,F}.
% ------------------------------------------------------------------------------------------------------------

nao(Q) :-
    Q, !, fail.
nao(Q).

% ------------------------------------------------------------------------------------------------------------
% Predicado que verifica qual o comprimento de uma lista.
% Extensão do predicado comprimento: Lista,Tamanho -> {V,F}.
% ------------------------------------------------------------------------------------------------------------

comprimento([],0).
comprimento([_|T],S) :-
    comprimento(T,G),
    S is G+1.
```

Figura 13: Predicados Nao e Comprimento

O predicado **comparaDatas** verifica qual é a maior data entre duas determinadas datas comparando os anos, os meses e os dias das mesmas para o conseguir.

```
% ------------------------------------------------------------------------------------------------------------
% Predicado que verifica qual a maior data entre duas.
% Extensão do predicado data_Maior: Lista,Tamanho -> {V,F}.
% ------------------------------------------------------------------------------------------------------------

comparaDatas(Y1-M1-D1,Y2-M2-D2,Y1-M1-D1) :- Y2 < Y1;
                                            (Y2 == Y1, M2 < M1);
                                            (Y2 == Y1, M2 == M1, D2 < D1).
comparaDatas(Y1-M1-D1,Y2-M2-D2,Y2-M2-D2) :- Y2 > Y1;
                                            (Y2 == Y1, M2 > M1);
                                            (Y2 == Y1, M2 == M1, D2 >= D1).
```

Figura 14: Predicado comparaDatas

Por último, duas das funções auxiliares mais fulcrais do nosso projeto sendo elas a **removeRepetidos** e a **eliminaTuplos**. A primeira retira os elementos repetidos de uma lista criando uma lista nova sem os mesmos. A segunda, tendo em conta que a maior parte das listas apresentadas como resultado no nosso trabalho são listas de tuplos **(ID,Nome)**, elimina os tuplos que fazem parte de uma lista e que estão presentes noutra lista.

```
% ------------------------------------------------------------------------------------------------------------
% Predicado que cria uma nova lista sem os repetidos da lista original.
% Extensão do predicado removeRepetidos: Lista,Lista -> {V,F}.
% ------------------------------------------------------------------------------------------------------------

removeRepetidos([],[]).
removeRepetidos([H|T],R) :-
    pertence(H,T),
    removeRepetidos(T,R).
removeRepetidos([H|T],[H|R]) :-
    nao(pertence(H,T)),
    removeRepetidos(T,R).
```

Figura 15: Predicado removeRepetidos

```
% ------------------------------------------------------------------------------------------------------------
% Predicado que permite a remoção de um tuplo de uma lista de outra lista (Remove os tuplos da segunda lista da primeira lista).
% Extensão do predicado remocao: Lista, Lista, Lista -> {V,F}.
% ------------------------------------------------------------------------------------------------------------

eliminaTuplos([], _, []) :- !.
eliminaTuplos([E|T], D, R) :-
    memberchk(E, D), !,
    eliminaTuplos(T, D, R).
eliminaTuplos([H|T], D, [H|R]) :-
    eliminaTuplos(T, D, R).
```

Figura 16: Predicado eliminaTuplos

6 Funcionalidades

- Definição da 1ª Fase de Vacinação

A primeira fase de vacinação estende-se desde dia 1 de janeiro de 2021 até dia 31 de março de 2021.

Para a primeira fase de vacinação começamos por definir que em termos de **idade** seriam vacinados utentes com idade superior a 80 anos. Assim, elaboramos o seguinte predicado auxiliar que retorna uma lista com os utentes com idade superior a 80 anos.

```
% ------------------------------------------------------------------------------------
% Predicado que identifica as pessoas elegíveis para vacinação na primeira fase devido à sua idade.
% Extensão do predicado idade1_fase_vacinacao: Lista -> {V,F}.
% ------------------------------------------------------------------------------------

idade1_fase_vacinacao(V) :-
    procura((Id,Nome),(utente(Id,_,Nome,D,_,_,_,_,_,_),comparaDatas(1942-01-01,D,1942-01-01)),V).
```

Figura 17: Predicado idade1 fase vacinacao

Em relação à **profissão** decidimos que seriam vacinados médicos, enfermeiros e militares, ou seja, profissionais ligados à saúde e à segurança pública. Desta forma, criamos um predicado auxiliar que verifica na base de conhecimento dos utentes quais são os que exercem estas profissões e retorna uma lista com os mesmos.

```
% ------------------------------------------------------------------------------------
% Predicado que identifica as pessoas elegíveis para vacinação na primeira fase devido à sua profissão.
% Extensão do predicado profissao1_fase_vacinacao: Lista -> {V,F}.
% ------------------------------------------------------------------------------------

profissao1_fase_vacinacao(V) :-
    procura((Id,Nome),(utente(Id,_,Nome,_,_,_,_,P,_,_),(P == 'Médico';P == 'Enfermeiro';P == 'Militar')),V).
```

Figura 18: Predicado profissao1 fase vacinacao

De seguida definimos os indivíduos que seriam vacinados consoante as **doenças crónicas** que têm e, deste modo, decidimos que iriam ser vacinados utentes com doenças pulmonares obstrutivas e com doenças cardiovasculares. Assim, pesquisamos na base de conhecimento dos utentes quais tinham estas mesmas doenças crónicas e devolvemos uma lista com os mesmos.

```
% ------------------------------------------------------------------------------------
% Predicado que identifica as pessoas elegíveis para vacinação na primeira fase devido às suas doenças crónicas.
% Extensão do predicado doencas1_fase_vacinacao: Lista -> {V,F}.
% ------------------------------------------------------------------------------------

doencas1_fase_vacinacao(V) :-
    procura((Id,Nome),(utente(Id,_,Nome,_,_,_,_,_,L,_),(pertence('Doença Pulmonar Obstrutiva',L);pertence('Doenças Cardiovasculares',L))),V).
```

Figura 19: Predicado doencas1 fase vacinacao

Por fim, criamos um predicado que concatenava todas estas listas eliminando os utentes que se repetiam por estarem inseridos em mais do que um dos critérios de inclusão da primeira fase de vacinação. Começamos por juntar a lista dos utentes incluidos por idade e pela profissão, de seguida concatenamos esta última lista com a dos utentes incluídos por doença e acabamos o predicado a retirar da lista final os utentes repetidos.

```
% ------------------------------------------------------------------------------------------------
% Predicado que identifica as pessoas elegíveis para vacinação na primeira fase.
% Extensão do predicado fase1_vacinacao: Lista -> {V,F}.
% ------------------------------------------------------------------------------------------------

fase1_vacinacao(V) :-
    idade1_fase_vacinacao(I),
    profissao1_fase_vacinacao(P),
    concat(I,P,L),
    doencas1_fase_vacinacao(D),
    concat(L,D,V1),
    removeRepetidos(V1,V).
```

Figura 20: Predicado fase1 vacinacao

- Definição da 2ª Fase de Vacinação

A segunda fase de vacinação estende-se desde dia 1 de abril de 2021 até dia 31 de julho de 2021.

Para a segunda fase de vacinação começamos por definir que em termos de **idade** seriam vacinados utentes com idade superior a 50 anos. Assim, elaboramos o seguinte predicado auxiliar que retorna uma lista com os utentes com idade superior a 50 anos.

```
% ------------------------------------------------------------------------------------------------
% Predicado que identifica as pessoas elegíveis para vacinação na segunda fase devido à sua idade.
% Extensão do predicado idade2_fase_vacinacao: Lista -> {V,F}.
% ------------------------------------------------------------------------------------------------

idade2_fase_vacinacao(V) :-
    procura((Id,Nome),(utente(Id,_,Nome,D,_,_,_,_,_,_),comparaDatas(1972-01-01,D,1972-01-01),comparaDatas(1942-01-01,D,D)),V).
```

Figura 21: Predicado idade2 fase vacinacao

Em relação à **profissão** decidimos que seriam vacinados professores e estudantes, ou seja, todas as pessoas ligadas à educação. Desta forma, criamos um predicado auxiliar que verifica na base de conhecimento dos utentes quais são os que exercem estas profissões e retorna uma lista com os mesmos.

```
% ------------------------------------------------------------------------------------------------
% Predicado que identifica as pessoas elegíveis para vacinação na segunda fase devido à sua profissão.
% Extensão do predicado profissao2_fase_vacinacao: Lista -> {V,F}.
% ------------------------------------------------------------------------------------------------

profissao2_fase_vacinacao(V) :-
    procura((Id,Nome),(utente(Id,_,Nome,_,_,_,_,P,_,_),(P == 'Professor';P == 'Estudante')),V).
```

Figura 22: Predicado profissao2 fase vacinacao

De seguida definimos os indivíduos que seriam vacinados consoante as **doenças crónicas** que têm e, deste modo, decidimos que iriam ser vacinados utentes com diabetes, obesidade, asma e cancro. Assim, pesquisamos na base de conhecimento dos utentes quais tinham estas mesmas doenças crónicas e devolvemos uma lista com os mesmos.

```
% ------------------------------------------------------------------------------
% Predicado que identifica as pessoas elegíveis para vacinação na segunda fase devido às suas doenças crónicas.
% Extensão do predicado doencas2_fase_vacinacao: Lista -> {V,F}.
% ------------------------------------------------------------------------------

doencas2_fase_vacinacao(V) :-
    procura((Id,Nome),(utente(Id,_,Nome,_,_,_,_,_,L,_),(pertence('Diabetes',L);pertence('Obesidade',L);pertence('Asma',L);pertence('Cancro',L))),V).
```

Figura 23: Predicado doencas2 fase vacinacao

Por fim, criamos um predicado que concatenava todas estas listas eliminando os utentes que se repetiam por estarem inseridos em mais do que um dos critérios de inclusão da segunda fase de vacinação. Começamos por juntar a lista dos utentes incluidos por idade e pela profissão, de seguida concatenamos esta última lista com a dos utentes incluídos por doença e acabamos o predicado a retirar da lista final os utentes repetidos. A única diferença deste predicado para o predicado definido para a primeira fase de vacinação é que também utilizamos o predicado auxiliar eliminaTuplos de forma a eliminarmos os utentes que já se encontravam incluídos na primeira fase de vacinação da segunda fase da mesma, pois seguimos o raciocínio utilizado na realidade que é o de se uma pessoa não for vacinada na primeira fase é porque o recusou ser e, assim sendo, passa para o fim da lista das pessoas a ser vacinadas e não é elegível na segunda fase de vacinação.

```
% ------------------------------------------------------------------------------
% Predicado que identifica as pessoas elegíveis para vacinação na segunda fase.
% Extensão do predicado fase2_vacinacao: Lista -> {V,F}.
% ------------------------------------------------------------------------------

fase2_vacinacao(V) :-
    fase1_vacinacao(F),
    idade2_fase_vacinacao(I),
    profissao2_fase_vacinacao(P),
    concat(I,P,L),
    doencas2_fase_vacinacao(D),
    concat(L,D,V1),
    removeRepetidos(V1,V2),
    eliminaTuplos(V2,F,V).
```

Figura 24: Predicado fase2 vacinacao

- Identificar Pessoas Não Vacinadas

Para ser possível identificar pessoas não vacinadas simplesmente procuramos pelos IDs e nomes de utentes que existiam na base de conhecimento dos utentes e não existiam na base de conhecimento da vacinação ao COVID-19, guardando os mesmos numa lista. Para esta funcionalidade assumimos que uma pessoa não está vacinada se não tiver qualquer tipo de toma de uma vacina.

```
% ------------------------------------------------------------------------------------------------
% Predicado que identifica as pessoas não vacinadas.
% Extensão do predicado naoVacinados: Lista -> {V,F}.
% ------------------------------------------------------------------------------------------------

naoVacinados(V) :-
    procura((Id,Nome),(utente(Id,_,Nome,_,_,_,_,_,_),nao(vacinacao_Covid(_,Id,_,_,_))),V).
```

Figura 25: Predicado naoVacinados

- Identificar Pessoas Vacinadas

Para ser possível identificar pessoas vacinadas simplesmente procuramos pelos IDs e nomes de utentes que existiam na base de conhecimento dos utentes e existiam na base de conhecimento da vacinação ao COVID-19 com a segunda toma já tomada, guardando os mesmos numa lista. Para esta funcionalidade assumimos que uma pessoa só está vacinada se tiver a segunda toma da vacina.

```
% ------------------------------------------------------------------------------------------------
% Predicado que identifica as pessoas vacinadas.
% Extensão do predicado vacinados: Lista -> {V,F}.
% ------------------------------------------------------------------------------------------------

vacinados(V) :-
    procura((Id,Nome),(utente(Id,_,Nome,_,_,_,_,_,_),vacinacao_Covid(_,Id,_,_,2)),V1),
    removeRepetidos(V1,V).
```

Figura 26: Predicado vacinados

- Identificar Pessoas Mal Vacinadas

O nosso grupo decidiu dividir esta funcionalidade em duas funções, uma que verifica os utentes mal vacinados na primeira fase e outra que faz o mesmo em relação à segunda fase.

Para o primeiro predicado começamos por fazer um *findall* que encontra todos os utentes que existem na base de conhecimento dos mesmos e na base de conhecimento da vacinação ao COVID-19 entre as datas em que se realiza a primeira fase de vacinação. De seguida, removemos os utentes repetidos desta lista, ou seja, os utentes que entre estas datas iriam aparecer duas vez por já terem tomado as duas doses da vacina. Depois, fomos buscar uma lista com os utentes candidatos à vacinação da primeira fase e concatenamos ambas as listas. Finalmente, removemos os utentes candidatos à vacinação na primeira fase da lista final de modo a apenas ficarmos com as pessoas vacinadas entre as datas em que se realizou a primeira fase de vacinação mas que não eram elegíveis para o ser.

```
% ------------------------------------------------------------------------------------------
% Predicado que identifica as pessoas mal vacinadas na 1ª fase.
% Extensão do predicado malVacinadosfase1: Lista -> {V,F}.
% ------------------------------------------------------------------------------------------

malVacinadosfase1(V) :-
    procura((Id,Nome),(utente(Id,_,Nome,_,_,_,_,_,_),vacinacao_Covid(_,Id,D,_,_),comparaDatas(2021-01-01,D,D),comparaDatas(2021-03-31,D,2021-03-31)),V1),
    removeRepetidos(V1,V2),
    fase1_vacinacao(L),
    concat(V2,L,V3),
    eliminaTuplos(V3,L,V).
```

Figura 27: Predicado malVacinadosfase1

Para o segundo predicado o raciocínio utilizado foi o mesmo sendo que apenas alteramos as datas para aquelas entre as quais se realizou a segunda fase de vacinação.

```
% ------------------------------------------------------------------------------------------
% Predicado que identifica as pessoas mal vacinadas na 2ª fase.
% Extensão do predicado malVacinadosfase2: Lista -> {V,F}.
% ------------------------------------------------------------------------------------------

malVacinadosfase2(V) :-
    procura((Id,Nome),(utente(Id,_,Nome,_,_,_,_,_,_),vacinacao_Covid(_,Id,D,_,_),comparaDatas(2021-04-01,D,D),comparaDatas(2021-07-31,D,2021-07-31)),V1),
    removeRepetidos(V1,V2),
    fase2_vacinacao(L),
    concat(V2,L,V3),
    eliminaTuplos(V3,L,V).
```

Figura 28: Predicado malVacinadosfase2

- Identificar Pessoas Não Vacinadas e que são Candidatas a Vacinação

Para esta funcionalidade também fizemos duas funções diferentes, uma relacionada com a primeira fase de vacinação e outra com a segunda.

Para o primeiro predicado, começamos por procurar os utentes existentes na base de conhecimento dos utentes e da vacinação ao COVID-19 entre as datas entre as quais se realizou a primeira fase de vacinação e adiciona-los a uma lista, removendo os repetidos. De seguida, juntamos os utentes que se encontram candidatos a ser vacinados na primeira fase da vacinação numa lista. Por último, retiramos desta última lista os utentes existentes na primeira lista ficando apenas com aqueles candidatos a serem vacinados na primeira fase mas que ainda não o foram.

```
% ------------------------------------------------------------------------------------------------
% Predicado que identifica as pessoas não vacinadas e que são candidatas a vacinação na 1ª fase.
% Extensão do predicado candidatosVacinacaofase1: Lista -> {V,F}.
% ------------------------------------------------------------------------------------------------

candidatosVacinacaofase1(V) :-
    procura((Id,Nome),(utente(Id,_,Nome,_,_,_,_,_,_),vacinacao_Covid(_,Id,D,_,_),comparaDatas(2021-01-01,D,D),comparaDatas(2021-03-31,D,2021-03-31)),V1),
    removeRepetidos(V1,V2),
    fase1_vacinacao(F),
    eliminaTuplos(F,V2,V).
```

Figura 29: Predicado candidatosVacinacaofase1

Para o segundo predicado o raciocínio utilizado foi o mesmo sendo que apenas alteramos as datas para aquelas entre as quais se realizou a segunda fase de vacinação.

```
% ------------------------------------------------------------------------------------------------
% Predicado que identifica as pessoas não vacinadas e que são candidatas a vacinação na 2ª fase.
% Extensão do predicado candidatosVacinacaofase2: Lista -> {V,F}.
% ------------------------------------------------------------------------------------------------

candidatosVacinacaofase2(V) :-
    procura((Id,Nome),(utente(Id,_,Nome,_,_,_,_,_,_),vacinacao_Covid(_,Id,D,_,_),comparaDatas(2021-04-01,D,D),comparaDatas(2021-07-31,D,2021-07-31)),V1),
    removeRepetidos(V1,V2),
    fase2_vacinacao(F),
    eliminaTuplos(F,V2,V).
```

Figura 30: Predicado candidatosVacinacaofase2

- identificar Pessoas a Quem Falta a Segunda Toma da Vacina

Para esta funcionalidade, simplesmente procuramos na base de conhecimento os utentes que pertenciam à base de conhecimento dos utentes e à da vacinação ao COVID-19 e que tinham recebido a primeira toma mas que não tinham recebido a segunda.

```
% ------------------------------------------------------------------------------------------------
% Predicado que identifica as pessoas a quem falta a segunda toma da vacina.
% Extensão do predicado vacinadosPrimeiraToma: Lista -> {V,F}.
% ------------------------------------------------------------------------------------------------

vacinadosPrimeiraToma(V) :-
    procura((Id,Nome),(utente(Id,_,Nome,_,_,_,_,_,_),vacinacao_Covid(_,Id,_,_,1),nao(vacinacao_Covid(_,Id,_,_,2))),V).
```

Figura 31: Predicado vacinadosPrimeiraToma

- Sistema de Inferência

De forma a desenvolver um sistema de inferência capaz de implementar mecanismos de raciocínio inerentes a estes sistemas simplesmente elaboramos um predicado que já tinha sido abordado nas aulas. O predicao **si** simplesmente verifica se determinado conhecimento existe na base de conhecimento.

```
% ------------------------------------------------------------------------------------------
% Predicado que verifica se o conhecimento existe na base de conhecimento.
% Extensão do predicado si: Questao, Resposta -> {V,F}.
% ------------------------------------------------------------------------------------------

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

Figura 32: Predicado si

7 Funcionalidades Extra

- Adicionar e Remover Termos

Uma das funcionalidades extra do nosso trabalho prático é conseguirmos **adicionar** e **remover** termos da nossa base de conhecimento. Para estas funcionalidades funcionarem da melhor forma, estas encontram-se intrinsecamente ligadas aos invariantes de inserção e de remoção explicados anteriormente no relatório.

Desta forma, para **adicionarmos** termos à nossa base de conhecimento utilizamos o predicado auxiliar evolução. Este predicado permite adicionar um termo, respeitando todos os invariantes definidos, à base de conhecimento. Desta forma, pega numa lista onde já tem o termo verificado com os respetivos invariantes, insere-o na base de conhecimento e testa se o mesmo ficou inserido na base de conhecimento. Para isto, também são utilizados dois predicados auxiliares um que permite a inserção de um termos e outro que permite o teste de invariante de uma lista.

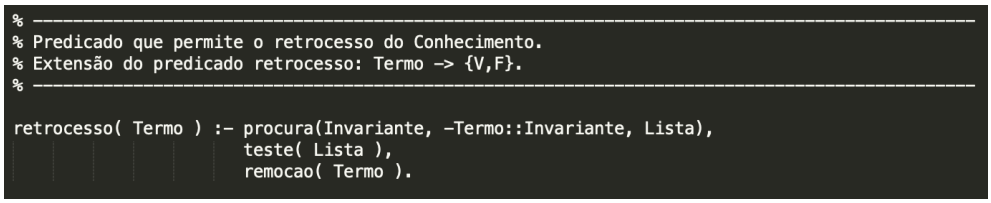
```
% -------------------------------------------------------------------------------------------
% Predicado que permite a evolução do Conhecimento.
% Extensão do predicado evolução: Termo -> {V,F}.
% -------------------------------------------------------------------------------------------

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

Figura 33: Predicado evolucao

```
% -------------------------------------------------------------------------------------------
% Predicado que permite o teste de invariante de uma lista.
% Extensão do predicado teste: Lista -> {V,F}.
% -------------------------------------------------------------------------------------------

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

% -------------------------------------------------------------------------------------------
% Predicado que permite a inserção de um termo.
% Extensão do predicado insercao: Termo -> {V,F}.
% -------------------------------------------------------------------------------------------

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

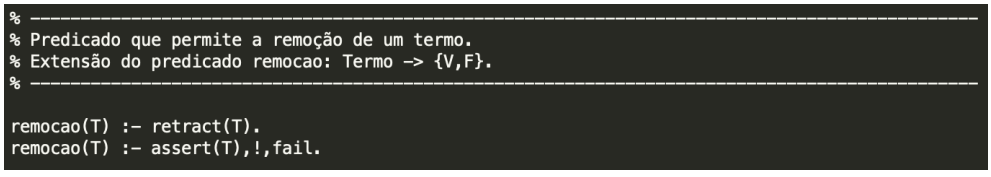
Figura 34: Predicados auxiliares do predicado evolucao

Para **removermos** termos da nossa base de conhecimento utilizamos a involução, num predicado ao qual chamamos de retrocesso. Este predicado permite remover um termo, respeitando todos os invariantes de remoção definidos, da nossa base de conhecimento. Deste modo, a partir de uma lista com o termos onde já foram verificados os invariantes respetivos, testa se o termo existe na base de conhecimento e, caso isto se verifique, retira-o da base de conhecimento.



```
% ------------------------------------------------------------------------------------------------
% Predicado que permite o retrocesso do Conhecimento.
% Extensão do predicado retrocesso: Termo -> {V,F}.
% ------------------------------------------------------------------------------------------------

retrocesso( Termo ) :- procura(Invariante, -Termo::Invariante, Lista),
                       teste( Lista ),
                       remocao( Termo ).
```

Figura 35: Predicado retrocesso



```
% ------------------------------------------------------------------------------------------------
% Predicado que permite a remoção de um termo.
% Extensão do predicado remocao: Termo -> {V,F}.
% ------------------------------------------------------------------------------------------------

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

Figura 36: Predicado auxiliar do predicado retrocesso

Após estes predicados, temos todas as condições reunidas para conseguir registar e remover conhecimento da nossa base de conhecimento. Passando os parâmetros necessários a cada um dos respetivos conhecimentos, utentes, staffs, centros de saúde e vacinações, utilizamos os predicados evolução e retrocesso para o conseguir.

```
% ------------------------------------------------------------------------------------------------
% Predicado que regista Utentes.
% Extensão do predicado registarUtente: Id_Utente, Número de Segurança_Social, Nome, Data Nascimento, Email, Telefone, Morada, Profissão, [Doenças_Crónicas], CentroSaúde ->
{V,F}.
% ------------------------------------------------------------------------------------------------

registarUtente(Id,Nss,N,D,E,T,M,P,DC,CS) :-
    evolucao(utente(Id,Nss,N,D,E,T,M,P,DC,CS)).

% ------------------------------------------------------------------------------------------------
% Predicado que regista Centros de Saúde.
% Extensão do predicado registarCentro: Id_Centro, Nome, Morada, Telefone, Email -> {V,F}.
% ------------------------------------------------------------------------------------------------

registarCentro(Id,N,M,T,E) :-
    evolucao(centro_saude(Id,N,M,T,E)).

% ------------------------------------------------------------------------------------------------
% Predicado que regista Staffs.
% Extensão do predicado registarStaff: Id_Staff, Id_Centro, Nome, Email -> {V,F}.
% ------------------------------------------------------------------------------------------------

registarStaff(Id,Idcs,N,E) :-
    evolucao(staff(Id,Idcs,N,E)).

% ------------------------------------------------------------------------------------------------
% Predicado que regista Vacinações de Covid.
% Extensão do predicado registarVacinacao: Id_Staff, Id_Utente, Data, Vacina, Toma -> {V,F}.
% ------------------------------------------------------------------------------------------------

registarVacinacao(Ids,Idu,D,V,T) :-
    evolucao(vacinacao_Covid(Ids,Idu,D,V,T)).
```

Figura 37: Predicados de registo de conhecimento

```
% ------------------------------------------------------------------------------------------------
% Predicado que remove Utentes.
% Extensão do predicado removerUtente: Id_Utente, Número de Segurança_Social, Nome, Data Nascimento, Email, Telefone, Morada, Profissão, [Doenças_Crónicas], CentroSaúde ->
{V,F}.
% ------------------------------------------------------------------------------------------------

removerUtentes(Id,Nss,N,D,E,T,M,P,DC,CS) :-
    retrocesso(utente(Id,Nss,N,D,E,T,M,P,DC,CS)).

% ------------------------------------------------------------------------------------------------
% Predicado que remove Centros de Saúde.
% Extensão do predicado removerCentros: Id_Centro, Nome, Morada, Telefone, Email -> {V,F}.
% ------------------------------------------------------------------------------------------------

removerCentros(Id) :-
    retrocesso(centro_saude(Id,N,M,T,E)).

% ------------------------------------------------------------------------------------------------
% Predicado que remove Staff.
% Extensão do predicado removerStaff: Id_Staff, Id_Centro, Nome, Email -> {V,F}.
% ------------------------------------------------------------------------------------------------

removerStaff(Id) :-
    retrocesso(staff(Id,Idcs,N,E)).

% ------------------------------------------------------------------------------------------------
% Predicado que remove Vacinações de Covid.
% Extensão do predicado removerVacinacao: Id_Staff, Id_Utente, Data, Vacina, Toma -> {V,F}.
% ------------------------------------------------------------------------------------------------

removerVacinacao(Ids,Idu,D) :-
    retrocesso(vacinacao_Covid(Ids,Idu,D,V,T)).
```

Figura 38: Predicado de remoção de conhecimento

- Staffs que trabalham num determinado centro de saúde

Esta funcionalidade permite obter uma lista de todos os staffs que trabalham num determinado centro de saúde. Assim, dado um id de um determinado centro de saúde procuramos na base de conhecimento os staffs que estaão associados a esse mesmo id.

```
% ------------------------------------------------------------------------------------------------
% Predicado que retorna os staffs que trabalham num determinado centro de saúde.
% Extensão do predicado getStaffCS: Id_Centro, Lista -> {V,F}.
% ------------------------------------------------------------------------------------------------

getStaffCS(CS,V) :-
    procura(Nome,staff(_,CS,Nome,_),V).
```

Figura 39: Predicado getStaffCS

- Utentes associados a um determinado centro de saúde

Esta funcionalidade permite obter uma lista de todos os utentes que são atendidos num determinado centro de saúde. Assim, dado um id de um determinado centro de saúde procuramos na base de conhecimento os utentes que estão associados a esse mesmo id.

```
% ------------------------------------------------------------------------------------------------
% Predicado que retorna os utentes que estão associados a um determinado centro de saúde.
% Extensão do predicado getUtenteCS: Id_Centro, Lista -> {V,F}.
% ------------------------------------------------------------------------------------------------

getUtenteCS(CS,V) :-
    procura(Nome,utente(_,_,Nome,_,_,_,_,_,_,CS),V).
```

Figura 40: Predicado getUtenteCS

- Número de vacinações feitas nas fases de vacinação

Esta funcionalidade tem como principal objetivo saber quantas vacinações foram feitas nas diferentes fases de vacinação ao COVID-19. Assim, elaboramos dois predicados idênticos em que procuramos na base de conhecimento todas as vacinações efetuadas entre as datas entre as quais as duas fases de vacinação são efetudas. Finalmente, utilizamos um predicado que retorna o tamanho da lista.

```
% ------------------------------------------------------------------------------------------------
% Predicado que retorna o número de vacinações feitas na 1ª fase de Vacinação.
% Extensão do predicado numeroVacinacoesfase1: Lista -> {V,F}.
% ------------------------------------------------------------------------------------------------

numeroVacinacoesfase1(V) :-
    procura(Id,(vacinacao_Covid(_,Id,D,_,_),comparaDatas(2021-01-01,D,D),comparaDatas(2021-03-31,D,2021-03-31)),L),
    comprimento(L,T),
    V is T.

% ------------------------------------------------------------------------------------------------
% Predicado que retorna o número de vacinações feitas na 2ª fase de Vacinação.
% Extensão do predicado numeroVacinacoesfase2: Lista -> {V,F}.
% ------------------------------------------------------------------------------------------------

numeroVacinacoesfase2(V) :-
    procura(Id,(vacinacao_Covid(_,Id,D,_,_),comparaDatas(2021-04-01,D,D),comparaDatas(2021-07-31,D,2021-07-31)),L),
    comprimento(L,T),
    V is T.
```

Figura 41: Predicados numeroVacinacoesfase1 e numeroVacinacoesfase2

- Número de pessoas vacinadas nas fases de vacinação

Esta funcionalidade tem como principal objetivo saber quantos utentes diferentes foram vacinados nas diferentes fases de vacinação ao COVID-19. Assim, elaboramos dois predicados idênticos em que procuramos na base de conhecimento todas as vacinações efetuadas entre as datas entre as quais as duas fases de vacinação são efetudas, removendo os elementos repetidos. Por último, utilizamos um predicado que retorna o tamanho da lista.

```
% ------------------------------------------------------------------------------------------------
% Predicado que retorna o número de pessoas vacinadas na 1ª fase de Vacinação.
% Extensão do predicado numeroPessoasVacinadasfase1: Lista -> {V,F}.
% ------------------------------------------------------------------------------------------------

numeroPessoasVacinadasfase1(V) :-
    procura(Id,(vacinacao_Covid(_,Id,D,_,_),comparaDatas(2021-01-01,D,D),comparaDatas(2021-03-31,D,2021-03-31)),V1),
    removeRepetidos(V1,L),
    comprimento(L,T),
    V is T.

% ------------------------------------------------------------------------------------------------
% Predicado que retorna o número de pessoas vacinadas na 2ª fase de Vacinação.
% Extensão do predicado numeroPessoasVacinadasfase2: Lista -> {V,F}.
% ------------------------------------------------------------------------------------------------

numeroPessoasVacinadasfase2(V) :-
    procura(Id,(vacinacao_Covid(_,Id,D,_,_),comparaDatas(2021-04-01,D,D),comparaDatas(2021-07-31,D,2021-07-31)),V1),
    removeRepetidos(V1,L),
    comprimento(L,T),
    V is T.
```

Figura 42: Predicados numeroPessoasVacinadasfase1 e numeroPessoasVacinadasfase2

- Vacinações efetuadas entre duas datas

Est funcionalidade permite obter uma lista com as vacinações que foram efetuadas entre duas determinadas datas. Desta forma, elaboramos um predicado que cria um limite entre duas datas e utilizamos este predicado para procurar as vacinações da nossa base de conhecimento para obter aquelas que se realizaram entre duas determinadas alturas do ano.

```
% ------------------------------------------------------------------------------------------
% Predicado que retorna as vacinações feitas entre duas datas.
% Extensão do predicado vacinacaoEntreDatas: Data, Data, Lista -> {V,F}.
% ------------------------------------------------------------------------------------------

datasLimiteVacinacao(D1,D2,V) :-
    comparaDatas(D1,V,V),comparaDatas(D2,V,D2).

vacinacaoEntreDatas(D1,D2,V) :-
    procura((Ids,Idu,D,M,T),(vacinacao_Covid(Ids,Idu,D,M,T),datasLimiteVacinacao(D1,D2,D)),L),
    removeRepetidos(L,V).
```

Figura 43: Predicado vacinacaoEntreDatas

8 Conclusão

Em suma, a realização deste trabalho prático permitiu a consolidação de diversos conhecimentos obtidos nas aulas práticas e teóricas da unidade curricular de Sistemas de Representação de Conhecimento e Raciocínio e também relativamente à linguagem de programação PROLOG.

O nosso projeto permite representar todo o conhecimento pedido no enunciado e todas as funcionalidades pedidas estão funcionais e eficientes. Acrescentamos algumas funcionalidades extra que achamos convenientes dado o propósito do projeto e a ideia base em que foi feito, ou seja, a vacinação da população portuguesa ao COVID-19.

Achamos que poderíamos ter aumentado um pouco mais a nossa base de conhecimento assim como adicionar outros tipos de conhecimento de forma a termos um trabalho ainda mais aproximado da realidade.

Concluindo, todos os objetivos propostos pelos docentes da unidade curricular foram conseguidos e pensamos que, no geral, todo o trabalho prático foi bem conseguido pelo grupo.