Sistemas de Representação de Conhecimento e Raciocínio

-Trabalho prático Fase 1

***

Docente: José Carlos Ferreira Maia Neves

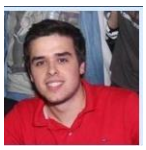
Desenvolvedores: Grupo 21

Daniel Cerveira Furtado Malhadas

A72293

Luís Manuel de Carvalho Ribeiro

A70804



Pedro Filipe Teixeira Costa

A71848

Pedro Miguel Braga do Vale

A72925

Resumo

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

Área de Aplicação: Desenho e arquitectura de Sistemas de Representação de Cohecimento e Raciocínio.

Palavras chave: Prolog; Programação em lógica e Invariantes; Bases de Conhecimento.

Tabela de Conteúdos

1.Introdução

***

1.1.Contextualização

O presente relatório serve de apoio à compreensão do projeto de sistemas de representação de conhecimento e raciocínio do presente ano letivo 2015/2016.

O projeto está dividido em várias fases a entregar por e-mail aos docentes, sendo que este relatório é referente somente à primeira fase.

1.2.Motivação e objetivos

Com este projeto pretende-se desenvolver um sistema de representação de conhecimento e raciocínio capaz de caracterizar instituições de saúde. Sendo que estas instituições normalmente estão associadas a uma grande afluência de dados, sejam eles utentes, serviços, profissionais, etc..., torna-se absolutamente necessário a existência de um sistema informático que permita a gestão de algo deste calibre não deixando que o potencial erro humano ponha em causa o, que se pretende que seja, o bom nome da(s) instituição(ões), sendo que tais erros, no caso específico de uma instituição de saúde, poderiam causar até perigos de vida para os utentes o que torna ainda mais urgente a existência então de um sistema informático capaz de gerir estes dados importantes e os apresentar com segurança e sem perdas aos funcionários qualificados que os solicitem.

Estabeleceu-se então que, para alcançar os objetivos pretendidos, determinadas condições mínimas tinham que se verificar no nosso projeto:

- ➔ Possibilidade de identificar os serviços existentes numa instituição;
- ➔ Possibilidade de identificar os utentes de uma instituição;
- ➔ Possibilidade de identificar os utentes de um determinado serviço;

- Possibilidade de identificar os utentes de um determinado serviço numa instituição;

- Possibilidade de identificar as instituições onde seja prestado um dado serviço ou conjunto de serviços;

- Possibilidade de identificar os serviços que não se podem encontrar numa instituição;

- Possibilidade de determinar as instituições onde um profissional presta serviço;

- Possibilidade de determinar todas as instituições (ou serviços, ou profissionais) a que um utente já recorreu;

- Possibilidade de registar utentes, profissionais, serviços ou instituições;

- Possibilidade de remover utentes (ou profissionais, ou serviços, ou instituições) dos registos.

Observando estas condições mínimas vemos que será necessário ter em conta várias entidades:

- ➔ Utente
- ➔ Instituição
- ➔ Serviço
- ➔ Profissional

Assim como as relações entre elas:

- ➔ Serviço / Instituição
- ➔ Profissional / Instituição / Serviço
- ➔ Utente / Serviço / Profissional / Instituição

1.3.Estrutura do relatório

No decorrer deste relatório tivemos em consideração duas questões que entendemos ser fundamentais e que explicamos de seguida:

- Estruturação:

Entendemos que não faz sentido começar imediatamente a construir "qualquer coisa" ou rápidamente começar a pensar nas mais complicadas maneiras de chegar ao fim se isso apenas significa ter algo mal pensado e mal estruturado que terá que vir a ser mudado no futuro. Não interessa quão bons sejam os programadores ou quão complexo seja o modelo do projeto que terão criado, se não tiver sido bem estruturado de raiz então apenas irá dar azo a uma incessante necessidade de modificações que poderão vir a pôr em risco a integridade do que já estava feito, criando necessidade de refazer partes significativas do projeto a longo prazo com a falsa esperança de "remediar" problemas que poderiam ter sido facilmente evitados numa fase de análise inicial. Isso é um exemplo de má estruturação e análise dos requisitos que sabemos ter de evitar a qualquer custo, visto que qualquer tempo "perdido" nesta fase será um indubitável ganho exponêncial em tempo na implementação. Por esta razão temos a noção da enorme importância que este primeiro obstáculo tem e tivemos todo o cuidado para o levar a sério e para o não apressar.

- Simplicidade:

É de grande importância para nós que a mensagem que desejamos transimitir com este relatório seja percebida na sua integridade sem que haja possibilidade de interpretações erradas ou alternativas. Sabendo nós que grande parte dos problemas com o software hoje em dia baseia-se na má comunicação entre os desenvolvedores e os seus empregadores quisemos que todos os esquemas e explicações neste relatório fossem sucintos, diretos ao assunto e simples de perceber. Desta forma emulamos a estrutura de um relatório que poderia ser relativo a um trabalho no mundo de tabalho fora da Universidade e onde o nosso empregador, mesmo com poucos conhecimentos na área seria, mesmo assim, capaz de ler, perceber e fazer uma crítica constructiva sobre a direcção do projeto. Isto permite evitar interpretações erradas do objetivo por parte dos programadores e também permite ao empregador compreender melhor se realmente os seus objectivos podem ser todos realizados da forma que idealizou ou se são demasiado optimistas.

2.Preliminares

***

1.1.Noções básicas de Bases de Dados

Note-se que é relevante conhecer certos conceitos, ou pelo menos ter uma pequena noção, para assim melhorar o entendimento do projeto e do relatório em questão. Como tal iremos aqui fazer a ponte de conhecimentos prévios para os presentes.

Sabemos já que este projeto exige que se guarde uma potencialmente grande quantidade de dados, sendo que teremos que validar esses dados para assim termos sempre a certeza da veracidade daquilo que vemos. Chamaremos então Bases de Conhecimento às bases usadas para guardar os dados e chamaremos invariantes às expressões que irão garantir a consistência dos dados que lá se encontram.

Estes invariantes são condições verificadas a cada inserção e irão garantir coisas como a não exist~encia de repetições ou a exist~encia de valores que não façam sentido no contexto do problema.

Com isto em mente, iremos agora descrever cada uma das bases de conhecimento.

3.Descrição do Trabalho e Análise dos Resultados

***

3.1.Bases de Conhecimento

Para desenvolver este projeto, pelas razões indicadas na **introdução**, compreendemos a necessidade de de alguma forma organizar os dados para posteriormente serem consultados e/ou verificados.

Usaremos então os conhecimentos derivados das noções explicitadas no capítulo **preliminares** para assim desenvolver bases de conhecimento que irão permitir guardar os dados das entidades existentes e das suas relações, tambem já indicadas na **introdução**.

Tendo em conta também os requesitos mínimos e as queries que entendemos ser necessário implementar, também já préviamente expostas na **introdução**, achamos essencial a existência das seguintes bases de conhecimento:

➔ Base de Conhecimento sobre **Utentes**

- Irá conter os utentes que existem no sistema
- Entende-se que a única restrição que se impõe à inserção nesta base de conhecimento é que não podem existir repetições.
- Na remoção não se impõem restrições sendo que apenas será necessário remover das restantes bases de conhecimento os registos que contêm o utente agora removido. No entanto existirá uma base de conhecimento, a referir em baixo que relaciona todas as entidades e servirá como um registo da atividade das instituições desde o início da sua existência, como tal, e para emular uma verdadeira instituição, esses registos não poderão ser eliminados, por isso ao remover um utente, os registos na base que referimos agora que o contenha não serão removidos.
- O sistema começa já com três utentes como exemplo: **joao, pedro, luis**.

- Pela restrição de inserção referida teremos que ter um invariante para esta base de conhecimento que garanta a sua consistência nesse sentido. Esse invariante irá garantir que não existem dois utentes iguais, para tal, depois de inserir, reúne numa lista todos os que têm o nome deste utente, caso essa lista tenha menos que dois elementos tudo correu bem e não se criaram inconsistências com esta inserção, caso sontrário teremos de voltar atrás e retirar o elemento agora inserido.

➔ Base de Conhecimento sobre **Serviços**

- Irá conter os serviços que existem no sistema

- Entende-se que a única restrição que se impõe à inserção nesta base de conhecimento é que não podem existir repetições.

- Na remoção não se impõem restrições sendo que apenas será necessário remover das restantes bases de conhecimento os registos que contêm o serviço agora removido. No entanto existirá uma base de conhecimento, a referir em baixo que relaciona todas as entidades e servirá como um registo da atividade das instituições desde o início da sua existência, como tal, e para emular uma verdadeira instituição, esses registos não poderão ser eliminados, por isso ao remover um serviço, os registos na base que referimos agora que o contenha não serão removidos. No entanto, quando este for eliminado iremos também eliminar todos os profissionais a trabalhar no serviço e iremos remover o serviço de todas as instituições ou apenas da especificada.

- O sistema começa já com três serviços como exemplo: **oncologia, cardiologia, consultas**.

- Pela restrição de inserção referida teremos que ter um invariante para esta base de conhecimento que garanta a sua consistência nesse sentido. Esse invariante irá garantir que não existem dois serviços iguais, para tal, depois de inserir, reúne numa lista todos os que têm o nome deste serviço, caso essa lista tenha menos que dois elementos tudo correu bem e não se criaram inconsistências com esta inserção, caso sontrário teremos de voltar atrás e retirar o elemento agora inserido.

➔ Base de Conhecimento sobre **Instituições**

- Irá conter as instituições que existem no sistema

- Entende-se que a única restrição que se impõe à inserção nesta base de conhecimento é que não podem existir repetições.

- Na remoção não se impõem restrições sendo que apenas será necessário remover das restantes bases de conhecimento os registos que contêm a instituição agora removida. No entanto existirá uma base de conhecimento, a referir em baixo que relaciona todas as entidades e servirá como um registo da atividade das instituições desde o início da sua existência, como tal, e para mimicar uma verdadeira instituição, esses registos não poderão ser eliminados, por isso ao remover uma instituição, os registos na base que referimos agora que a contenha não serão removidos. No entanto, quando este for eliminado iremos também eliminar todos os profissionais a trabalhar na instituição e iremos remover os serviços daquela instituição.

- O sistema começa já com três instituições como exemplo: l**isboa, porto, braga.**

- Pela restrição de inserção referida teremos que ter um invariante para esta base de conhecimento que garanta a sua consistência nesse sentido. Esse invariante irá garantir que não existem duas instituições repetidas, para tal, depois de inserir, reúne numa lista todas as que têm o nome desta instituição, caso essa lista tenha menos que dois elementos tudo correu bem e não se criaram inconsistências com esta inserção, caso sontrário teremos de voltar atrás e retirar o elemento agora inserido.

➔ Base de Conhecimento sobre **Serviços** de uma determinada **Instituição**

- Irá conter os serviços que cada instituição pode providenciar.

- Entende-se que as únicas restrições que se impõem à inserção nesta base de conhecimento é que não podem existir repetições assim como não podem ser inseridos registos que contenham serviços e/ou instituições que não constem das suas respetivas bases de conhecimento.

- Na remoção não se impõem restrições. No entanto, quando este for eliminado iremos também eliminar todos os profissionais a trabalhar no serviço e iremos remover o serviço de todas as instituições ou apenas da especificada.

- O sistema começa já com exemplos desta relação serviço/instituição:

oncologia/porto,

oncologia/lisboa,

cardiologia/lisboa,

cardiologia/porto,

consultas/braga,

consultas/lisboa.

- Pelas restrições de inserção referidas teremos que ter invariantes para esta base de conhecimento que garantam a sua consistência nesse sentido.

  Teremos que ter um invariante que garanta que não existam repetição, esse invariante irá garantir que não existem conjuntos serviço/instituição repetidos, para tal, depois de inserir, reúnem-se numa lista todos os registos com esta combinação, caso essa lista tenha menos que dois elementos tudo correu bem e não se criaram inconsistências com esta inserção, caso contrário teremos de voltar atrás e retirar o elemento agora inserido.

  Com os outros dois restantes invariantes iremos garantir que o serviço e a instituição a inserir constam das suas respetivas bases de conhecimento.

➔ Base de Conhecimento sobre **Profissionais** e seus respetivos **Serviços** nas respetivas **Instituições**.

- Irá conter os serviços que cada profissional exerce assim como a instituição onde o faz.

- Entende-se que as restrições que se impõem à inserção nesta base de conhecimento é que não podem existir repetições assim como não podem ser inseridos registos que contenham serviços e/ou instituições que não constem das suas respetivas bases de conhecimento. É também necessário verificar se a instituição pode providenciar o serviço a inserir, observando na base de conhecimento que relaciona essas duas entidades.

- Na remoção não se impõem restrições.

- O sistema começa já com exemplos desta relação profissional/serviço/instituição:

drjoao/oncologia/lisboa,
drjoao/oncologia/porto,

drpaulo/cardiologia/lisboa,
drpaulo/cardiologia/porto,

drajoana/consultas/braga,
drajoana/consultas/lisboa.

- Pelas restrições de inserção referidas teremos que ter invariantes para esta base de conhecimento que garantam a sua consistência nesse sentido.

  Teremos que ter um invariante que garanta que não existam repetições, esse invariante irá garantir que não existem conjuntos profissional/serviço/instituição repetidos, para tal, depois de inserir, reúnem-se numa lista todos os registos com esta combinação, caso essa lista tenha menos que dois elementos tudo correu bem e não se criaram inconsistências com esta inserção, caso contrário teremos de voltar atrás e retirar o elemento agora inserido.

  De seguida necessitamos somente de mais um invariante que garante que o serviço em questão pode ser providenciado pela instituição fornecida consultando a base de conhecimento que relaciona essas duas entidades. Não temos de ter um invariante que verifica se o serviço ou a instituição existem nas suas respetivas bases de conhecimento, pois se esta combinação serviço/instituição existir na base de conhecimento que relaciona essas entidades então de certeza que serviço e instituição existem nas suas respetivas bases de conhecimento também, pois isso já é verificado aquando da inserção de serviço/instituição com os invariantes dessa base de conhecimento.

➔ Base de Conhecimento sobre **Utentes**, as sua consultas com um **Profissional** num respetivo **Serviço** numa respetiva **Instituição**.

- Irá conter os utentes que recorreram a um determinado serviço de uma determinada instituição providenciado por um determinado profissional.

- Entende-se que a restrição que se impõem à inserção nesta base de conhecimento é a necessidade de verificar se o profissional pode exercer o serviço que se está a inserir na respetiva instituição, observando a base de conhecimento que relaciona essas três entidades. De notar que não há problema em que hajam repetições, pois um mesmo utente pode ir a um mesmo profissional num mesmo serviço numa mesma instituição repetidas vezes e tudo isso deve estar registado várias vezes para que assim os registos sejam verdadeiramente completos e não haja perda de informação.

- Estabelecemos que não se podem remover registos desta base de conhecimento. Isto acontece, pois, de modo a simular o funcionamente de uma instituição real, mesmo que utentes sejam removidos, profissionais mudados, serviços descontinuados ou instituições abandonadas, os registos daquilo que aconteceu (caso desta base de conhecimento) não devem, em situação alguma, ser eliminados para assim poderem ser consultados quando necessário.

- O sistema começa já com exemplos desta relação utente/profissional/serviço/instituição:

  joao/drjoao/oncologia/lisboa,
  joao/drajoana/consultas/lisboa,

  pedro/drjoao/oncologia/porto,
  pedro/drpaulo/cardiologia/porto,

  luis/drajoana/consultas/braga,
  luis/drajoana/consultas/braga.

- Pela restrição de inserção referida teremos que ter um invariante para esta base de conhecimento que garanta a sua consistência nesse sentido.

  Teremos que ter um invariante que o serviço em questão pode ser providenciado pela instituição fornecida e pelo profissional em questão consultando a base de conhecimento que relaciona essas três entidades. Não temos de ter um invariante que verifica se o serviço pode ser realizado nesta instituição, pois se esta combinação profissional/serviço/instituição existir na base de conhecimento que relaciona essas entidades então de certeza que o serviço se pode realizar na instituição e o serviço e a instituição existem nas suas respetivas bases de conhecimento também, pois isso já é verificado aquando da inserção de profissional/serviço/instituição com os invariantes dessa base de conhecimento.

Nota: Importa notar as seguintes considerações

- Note-se que os profissionais não estão vinculados a nenhuma instituição específica nem é possível fazê-lo, desta forma tenta-se emular uma instituição real onde não se impede o mesmo profissional de exercer em vários hospitais.

- Da mesma forma que os profissionais não estão vinculados a nenhuma instituição também os serviços e os utentes podem estar associados a diferentes instituições.

- Não vamos ter uma base de conhecimentos que guarde apenas o nome do profissional pois achamos que não fazia sentido, tratando-se da gestão de eventos num instituto hospitalar. Não vamos ter um profissional apenas, vamos ter um profissional que exerce um serviço numa ou mais instituições.

3.2.Queries

De modo a cumprir os requisitos a que nos proposemos e que enumeramos na **introdução** criamos alguns predicados que se apresentam de seguida:

➔ **Querie1**, criou-se uma extensão do predicado
servInst: instituicao,X -> {V,F}

- Permite identificar os serviços existentes numa instituição.
- Com recurso ao predicado **solucoes** reúne-se numa lista todos os serviços da instituição argumento.

➔ **Querie2**, criou-se uma extensão do predicado
utentInst: instituicao,X-> {V,F}

- Permite identificar os utentes que utilizaram determinada instituição.
- Com recurso ao predicado **solucoes** reúne-se numa lista todos os utentes que utilizaram a instituição argumento. No entanto como a base de dados que realaciona utentes e instituições poderá conter repetições é usado o predicado **rep** de seguida que remove repetições numa lista.

➔ **Querie3**, criou-se uma extensão do predicado

utentServ: servico,X -> {V,F}

- Permite identificar os utentes que recorreram a um determinado serviço.

- Com recurso ao predicado **solucoes** reúne-se numa lista todos os utentes que recorreram ao serviço argumento. No entanto como a base de dados que relaciona utentes e serviços poderá conter repetições é usado o predicado **rep** de seguida que remove repetições numa lista.

➔ **Querie4**, criou-se uma extensão do predicado

utentServInst: servico, instituicao, X -> {V,F}

- Permite identificar os utentes que recorreram a um determinado serviço numa determinada instituição.

- Com recurso ao predicado **solucoes** reúne-se numa lista todos os utentes que recorreram ao serviço argumento na instituição argumento. No entanto como a base de dados que relaciona utentes, serviços e instituições poderá conter repetições é usado o predicado **rep** de seguida que remove repetições numa lista.

➔ **Querie5**, criou-se uma extensão do predicado

inst: Instituicoes,Servicos -> {V,F}

- Permite identificar as instituições onde se fornece um serviço ou um conjunto de serviços.

- Com recurso ao predicado **solucoes** reúne-se numa lista todas as instituições onde se fornecem o(s) serviço(s) argumento(s). No entanto como um serviço pode ser fornecido por várias instituições diferentes é usado o predicado **rep** de seguida que remove repetições numa lista.

➔ **Querie6**, criou-se uma extensão do predicado

nao_encontra: Servicos,Instituicao -> {V,F}

- identifica os serviços que não se podem encontrar numa instituição.

- Com recurso ao predicado **solucoes** reúne-se numa lista todas as instituições onde se fornecem o serviço argumento. De seguida reune-se numa lista todos os serviços possíveis, também recorrendo ao predicado **solucoes**. Depois, com o predicado **delTodos**, retira-se desta lista a encontrada anteriormente, obtendo assim no final a lista pretendida.

➔ **Querie7**, criou-se uma extensão do predicado

prestaServ: profissional,X -> {V,F}

- Permite identificar as instituições onde um profissional presta serviços.

- Com recurso ao predicado **solucoes** reúne-se numa lista todos os profissionais que fornecem serviços na instituição argumento.

➔ **Querie8**, criou-se uma extensão do predicado

determ: utente, tag, Y -> {V,F}

- Determina todas as instituições (ou serviços, ou profissionais) a que um utente já recorreu

- Primeiramente interessa notar que é usada uma *tag* como segundo argumento que irá indicar se pretendemos obter serviços, instituições ou profissionais, sendo que essa tag será serv, inst ou prof, respetivamente. *Com* recurso ao predicado **solucoes** reúne-se numa lista todos os elementos pretendidos a que recorreu o utente argumento. No entanto como esses elementos poderão ter sido recorridos pelo utente várias vezes é usado o predicado **rep** de seguida que remove repetições numa lista.

➔ **Querie9**, criou-se uma extensão do predicado

inserir: P -> {V,F}

- Regista um utente, profissional, serviço ou instituição

- Com recurso ao predicado **soluções e testa**, após definidos os invariantes necessários e sendo esses testados de forma igual á apresentada na ficha pratica da semana 4, é inserido o predicado dado como paremetro, fazendo uso da função **assert**, inserindo assim este nas bases de conhecimentos apenas se não violar nenhuma regra imposta pelos invariantes definidos anteriormente.

➔ **Querie10**, criou-se uma extensão do predicado

remove: tag, X -> {V,F}

- Remove um utente, serviço, profissional ou instituição.

- Primeiramente interessa notar que é usada uma *tag* como primeiro argumento que irá indicar se pretendemos remover serviços, instituições, profissionais ou utentes, sendo que essa tag será servico, instituicao, profissional ou utente, respetivamente. De seguido com recurso a **retract** retira-se somente uma ocorrência do segundo argumento da base de conhecimento que corresponde à tag em primeiro argumento. A excepção são os profissionais, pois teremos que usar **retractAll** em vez de **retract** na sua base de conhecimento pois não existe uma base de conhecimento onde não possam haver repetições de profissionais. Para aquelas entidades em que seja necessário remover de mais que uma base de conhecimento (especificadas no sub-capítulo anterior) então usaremos **retractall** para eliminar todas as ocorrências do segundo argumento nas restantes bases de conhecimento relevantes.

➔ **Querie11**, criou-se uma extensão do predicado

profutente: profissional, X -> {V,F}

- identifica os utentes a que um dado profissional prestou serviço.

- Com recurso ao predicado **solucoes** reúne-se numa lista todos os utentes a que o profissional argumento prestou serviço. No entanto como um profissional pode ter fornecido serviços a um utente várias vezes é usado o predicado **rep** de seguida que remove repetições numa lista.

➔ **Querie12**, criou-se uma extensão do predicado

profutente: profissional, X -> {V,F}

- identifica os profissionais que prestam um dado serviço.

- Com recurso ao predicado **solucoes** reúne-se numa lista todos os profissionais que prestam o serviço argumento. No entanto como um serviço pode ser fornecido por um mesmo profissional em instituições diferentes é usado o predicado **rep** que remove repetições numa lista.

4.Conclusões e Sugestões

Terminada a primeira etapa/fase do projeto, depois de feita uma minunciosa análise de requisitos e avaliação do problema, chegamos a conclusões (apresentadas ao longo do corpo deste relatório) que nos deixaram satisfeitos. Pensamos ter sido capazes de apresentar explicações detalhadas no sentido em que toda a informação essencial para entender a implementação do nosso sistema de representação de conhecimento e raciocínio e o seu funcionamento se encontra presente sem nenhuma exceção, mas ao mesmo tempo bastante simples na maneira como será possível para alguém com pouco conhecimento de o que é prolog ou gestão de base de dados consiga ainda assim acompanhar e entender o que está a ser dito. Com isto pretendemos afirmar que estamos satisfeitos com as conclusões a que chegamos e com o plano de acção que elaboramos tendo conseguido realizar tudo o que foi proposto de forma simples e eficaz tendo ainda realizado vários testes para garantir o funcionamento correto dos invariantes que mantêm os dados sempre consistentes nunca obtendo resultados não aceitáveis.

Tendo isto em conta temos também a noção que nunca nenhum projeto se pode dar por realmente terminado ou por realmente perfeito, há sempre a possibilidade de melhorar algo e esse fato não deve nunca ser ignorado, por mais experiência que venhamos a ter, existe sempre um limite para quão boas as nossas primeiras impressões poderão ser.

Desta forma, estamos preparados para uma eventual necessidade de modificar algo que aqui poderá ter sido afirmado e/ou concluído. Isto é, o nosso projeto foi elaborado com a ideia de criar algo que permita uma fácil expansão tanto de funcionalidades como de novas ideias que venhamos a ter ou necessitemos de implementar na continuação do projeto. Isto é essencial em qualquer projeto, pois sabemos que no mundo real o empregador nem sempre sabe de raiz tudo aquilo que realmente quer implementar por isso torna-se natural que, num momento mais avançado da implementação, liste novas funcionalidades que quer ver presentes.

Um projeto mal pensado para expansões e modificações poderia ter de ser refeito de raiz para não dar problemas ou graus baixos de eficiência, no entanto tivemos a ideia presente de tentar criar algo que podesse minimizar todas essas possíveis perdas que poderiam acontecer se este fosse um projeto no mundo real e não um trabalho académico protegido pela redoma que acaba sempre por ser uma universidade.

Como exemplo de algo fácil de adicionar que poderia melhorar um pouco o projeto deixamos o conceito de data ou então meramente de sequência na base de conhecimento que relaciona utentes, serviços, profissionais e instituições. Desta forma poderiamos distinguir registos iguais (que podem existir pois nada evita que o mesmo utente recorra ao mesmo serviço na mesma instituição repetidas vezes e tal deve estar guardado para que os registos estejam completos). E poderiamos também obter todos os utentes que utilizaram o serviço numa determinada data, por exemplo.