**openEHR**

O EHR surgiu essencialmente devido a estudos feitos pelos centros médicos académicos e organizações clinicas governamentais em meados dos anos 60 e 70. A Universidade de Utah together juntamente com a Corporação 3M desenvolveram um dos primeiros EHRs (Health Evaluation through Logical Processing - HELP). Ao longo dos anos foram desenvolvidas várias outras ferramentas nesta área como: Allscripts, COSTAR, DHCP, etc. O EHR surge da necessidade de resolver vários problemas como: a constante necessidade de adaptação das bases de dados, a dificuldade em manter compatibilidade com dados anteriores, a dificuldade em documentar alterações, a heterogeneidade na forma de introdução de dados, as dúvidas na interpretação na leitura, o fato dos conceitos evoluírem e a falta de avaliação na qualidade dos dados. Além destes problemas as estruturas de dados (e a qualidade) dependem muito do objetivo na sua escolha (prestação de cuidados, investigação clinica, gestão/financeira, ensino, legal).

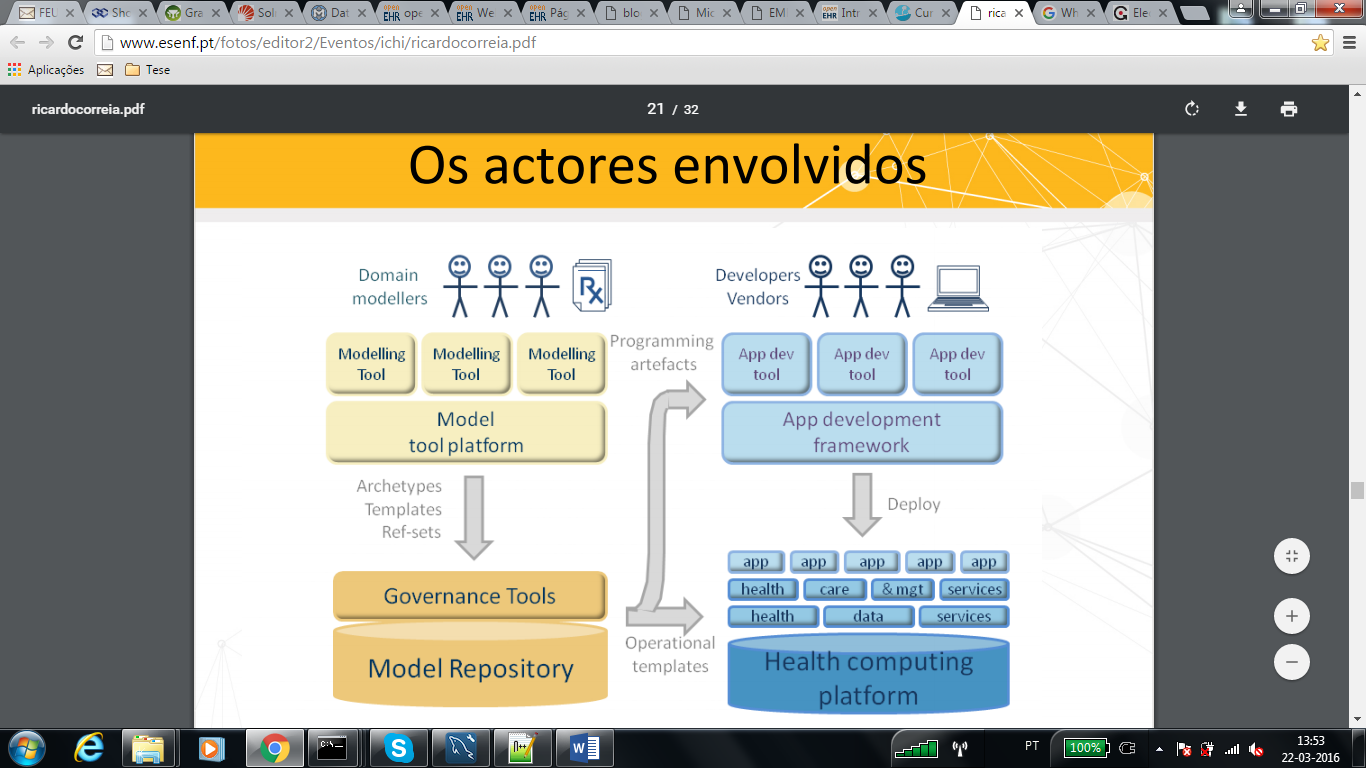
Perante os problemas referidos e, uma vez que o EHR pretende corrigi-los, é possível descreve-lo como uma especificação (norma) que permite a representação de conceitos clínicos complexos.

A norma EHR está construída da seguinte forma:

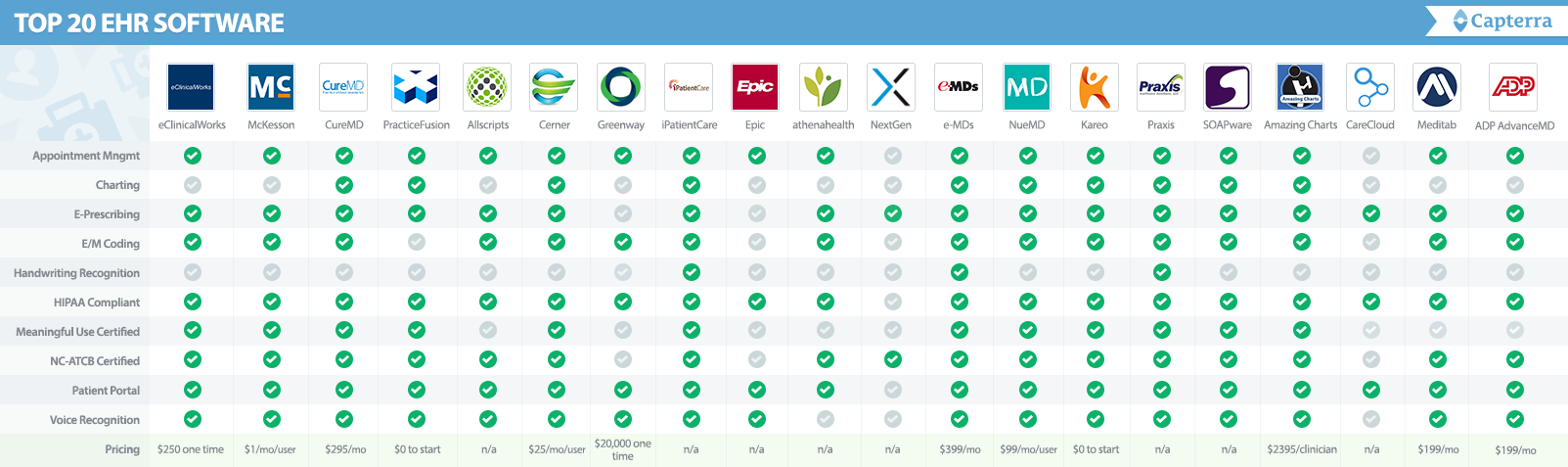
* Arquétipo: conceito mais pequeno nesta estrutura (equivalente a uma peça de LEGO), consiste em dados médicos, como por exemplo, altura, peso, sumário de gravidez e ecocardiograma;
* Template: consiste num conjunto de peças LEGO, isto é, num conjunto de arquétipos.

Perante estes conceitos é possível construir um esquema que define a estrutura de um documento podendo, assim, adaptar a informação presente nos documentos existentes de forma a homogeneizar os seus dados.

Com o EHR, e como já referido, surgem diversas ferramentas, entre as quais se realça o openEHR. Este consiste num ambiente onde é possível colocar várias formas, tendo estas formas a possibilidade de comunicar entre si. Uma das vantagens principais na utilização desta ferramenta é o fato de os utilizadores poderem alterar os arquétipos e templates sem alterar o software, ou seja, estamos perante uma ferramenta bastante geral, que aceita qualquer tipo de documento (desde que seja elaborado um bom template para o receber). Os atores envolvidos no openEHR são apresentados na imagem seguinte:



Em suma, o EHR garante que os dados sejam compreensíveis no futuro, facilita a utilização das bases de dados comuns e permite até uma melhor cooperação entre países e organizações.



**Importação de dados do EHR para SOLR**

Para importação dos dados começou por ser testada uma ferramenta existente no SOLR denominada por data import handler (DIH). Esta permite que a informação existente nas tabelas de uma base de dados realcional seja extraída e estruturada a partir dos dados definidos nos ficheiros schema.xml, solrconfig.xml e data-config.xml. Além destas possibilidades a ferramenta em causa permite também atualizar os índices quando um documento é modificado ou um novo é introduzido. Apesar da sua facilidade de utilização o DIH em termos de desempenho é bastante lento, especialmente se a quantidade de dados for grande, pois apenas importa um documento de cada vez. Além disto, é de realçar também, que a quantidade de pedidos feitos à base de dados influencia a velocidade do DIH.

Assim sendo e, numa primeira experiência, foram introduzidos cerca de 5 documentos na base de dados e foram registados cerca de 467 pedidos os quais, juntamente com o processo de indexação, tiveram uma duração de cerca de 6-8 segundos. Numa tentativa de otimizar os pedidos, foi feita uma nova experiência com queries diferentes e com 8 documentos na qual foram registados cerca de 475 pedidos, com duração de 4-6 segundos. Como é possível observar, uma ligeira redução na quantidade de pedidos traduziu-se numa grande alteração na duração da operação. Todavia o problema de importar um documento de cada vez continuava. Para isso foram realizados testes com threads para concluir se agilizava ou não o processo. Primeiramente foi necessário desistir do DIH e dos ficheiros xml que este incluía como método de estruturação da informação recebida. Após isto, e com utilização do SolrNet, em c#, passei aos pedidos à base de dados os quais, juntamente com threads apresentaram resultados bem mais satisfatórios, cerca de 1-1.5 segundos para 8 documentos. Com isto finalizei a análise de importação de dados para o SOLR através de base de dados relacionais, concluindo que com threads e com a quantidade de pedidos atuais é possível realizar importação e indexação em tempo aceitável.

Relativamente à importação de ficheiros, foram realizados testes com ficheiros xml existentes num exemplo do SolrNet. Rapidamente chegou-se à conclusão que, mesmo sem threads, a importação destes era bem mais rápida de que a da base de dados. No entanto, e apesar dos bons resultados, foram na mesma efetuados testes com threads, e novamente estes mostraram-se mais eficientes do que o processo sem threads. A tabela dos resultados de teste encontra-se a seguir:

|  |  |  |  |
| --- | --- | --- | --- |
| **Número de ficheiros** | **1705** | **10230** | **20460** |
| **Sem Threads** | 9.3s | 22.4s | 48.3s |
| **Com Threads** | 8.4s | 16.4s | 33.8s |

Este teste foi realizado com ficheiros xml, mas prevê-se que a utilização de outra extensão de ficheiros terá resultados semelhantes.

Partindo agora para um plano em que os ficheiros estão a ser constantemente atualizados ou novos estão a ser inseridos, é importante perceber que fazer novamente uma importação completa dos dados não é correto. Este problema é fácil de perceber, pois se fosse realizada novamente a importação de todos os ficheiros a solução a apresentar seria tão lenta que não existiriam condições para usa-la. Assim para resolver este problema tem apenas interesse realizar importações parciais dos dados, ou seja, importando apenas os dados novos ou aqueles que foram modificados. Com o fim de realizar estas importações parciais surgem alguns parâmetros disponíveis pelo SOLR designados por deltaQuery e parentDeltaQuery. O deltaQuery corresponde à query que seleciona a linha que foi modificada, enquanto o parentDeltaQuery seleciona as linhas modificadas para fornecer linhas às tabelas pai na hierarquia de pedidos.

Mas como se verifica que uma nova linha foi modificada? Para esta pergunta a resposta é, criando uma coluna em cada uma das tabelas utilizadas designada por: last\_modified. Esta coluna deve guardar a data na qual a linha teve a sua última modificação. Assim, comparando com a data da última indexação, é possível saber se uma linha foi modificada ou não.

Uma vez obtida a linha modificada, através das queries existentes no parâmetro query do mesmo xml, é feita a importação do documento em causa, eliminando assim a necessidade de uma importação completa dos documentos.

**Pesquisa (Stemming e stopwords)**

Como já foi referido no capítulo do estado da arte as stemming, stopwords e sinónimos são conceitos bastante importantes no que diz respeito à indexação e pesquisa de informação. Face a esta importância, foram realizados testes com dicionários deste género de palavras para verificar a sua utilidade nas operações inerentes aos mesmos. Como esperado, as stopwords foram de grande ajuda na remoção de palavras sem interesse para a indexação, assim como as stemming words, reduzindo as palavras para a sua raiz. Ajuda maior notou-se no processo de pesquisa, o qual se tornou muito mais natural essencialmente devido às stemming words e aos sinónimos. Os testes realizados foram feitos com dicionários em espanhol, uma vez que a base de dados se encontra nesse idioma. No entanto, outros dicionários já se encontram preparados caso seja necessário alterar o idioma.

**Associação de meta-informação aos documentos dos pacientes**

A solução que se pretende implementar nesta dissertação pretende apresentar, após uma pesquisa de um utilizador, além dos dados de um documento, a meta-informação do doente ao qual o documento se refere. Para isso, atualmente uma pesquisa realizada no SOLR retorna os dados de um documento o qual é constituído por: id do documento, data do documento (por causa das versões), texto do documento e o número do EHR que a ele está inerente. Uma vez que temos informação sobre o EHR é possível saber a qual paciente o documento com o EHR referido diz respeito. Uma vez sabendo qual é o paciente é apenas necessário extrair a informação deste das tabelas correspondentes. Desta forma consegue-se uma informação do doente, tal como proposto como tema desta dissertação (ajudar os médicos na sua pesquisa por informação dos pacientes).

**Arquitetura (modelo conceptual)**

Após os testes realizados com as diversas ferramentas foi possível definir o que utilizar e qual a estrutura necessária para a solução proposta. Assim sendo, está descrito a seguir, passo a passo, a estrutura da mesma:

1. **Base de dados:** local onde se encontra toda a informação clínica dos doentes. Armazena todos os documentos.
2. **EHR:** local onde se encontra toda a informação dos documentos de forma estruturada e organizada de acordo com os templates criados. Possivelmente alguns documentos não respeitarão a estrutura dos templates e por isso serão guardados num módulo à parte ou num template mais geral que guarde todo o texto.
3. **Solr:** ferramenta na qual será realizada indexação e na qual serão guardados os índices resultantes da mesma. Para importação dos dados para o SOLR serão usados os métodos já referidos anteriormente.
4. **Pesquisa:** A pesquisa também será realizada com a ajuda do SOLR e, tendo em conta as stemming, stopwords e sinónimos pretende-se que seja o mais natural possível e que retorne uma maior quantidade de informação útil para os médicos.

O modelo conceptual encontra-se na imagem seguinte: