Universidade de Aveiro
2012
Departamento de Eletrónica, Telecomunicações e Informática

André Filipe Fidalgo Vechina

Representação de redes semânticas de termos biomédicos

Representation of semantic networks of biomedical terms

Universidade de Aveiro
2012
Departamento de Eletrónica, Telecomunicações e Informática

André Filipe Fidalgo Vechina

Representação de redes semânticas de termos biomédicos

Representation of semantic networks of biomedical terms

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários para a obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor José Luís Oliveira e do Doutor Joel Arrais.

o júri

presidente

Prof. Doutor Joaquim Manuel Henriques de Sousa Pinto
Professor Auxiliar da Universidade de Aveiro

vogais

Prof. Doutor Carlos Manuel Jorge da Silva Pereira
Professor Adjunto do Dep. de Engenharia Informática e de Sistemas do Instituto Superior de Engenharia de Coimbra

Prof. Doutor José Luís Guimarães Oliveira
Professor Associado da Universidade de Aveiro (Orientador)

Prof. Doutor Joel Perdiz Arrais
Professor Auxiliar Convidado do Dep. de Engenharia Informática da Faculdade de Ciências e Tecnologias da Universidade de Coimbra (Co-Orientador)

agradecimentos

Queria agradecer a todas as pessoas que, direta ou indiretamente, tornaram possível a realização desta dissertação.

Ao Prof. José Luís Oliveira e ao Prof. Joel Arrais pelas orientações, críticas e sugestões prestadas durante todo o desenvolvimento deste trabalho.

Aos meus pais, irmã, familiares e amigos que, com muito apoio e incentivo, não mediram esforços para que superasse mais este desafio académico.

palavras-chave Rede, Grafo, Termos Biomédicos, Aplicação Web

resumo

Os recentes avanços tecnológicos têm sido de importância preponderante na evolução da biologia e da biomedicina. A aplicação de métodos computacionais na integração de dados e na análise de resultados experimentais de natureza biológica tem permitido expandir o conjunto de associações conhecidas entre diferentes termos biomédicos. No entanto, o conhecimento encontra-se disperso sobre inúmeras bases de dados independentes.

Através da integração de diferentes tipos de termos biomédicos e das suas associações numa única vista, obtém-se um quadro mais abrangente e globalizado que permita extrair evidências biomoleculares relevantes. Representando uma rede de termos biomédicos através de grafos, é possível transportar os conhecimentos matemáticos dessa teoria e aplicá-los na área da bioinformática.

Esta dissertação teve como objetivo fundamental desenvolver uma plataforma Web que, através da integração de dados biológicos, permita a visualização, o estudo e a compreensão das relações entre doenças, genes e outros elementos/termos biomédicos. Este sistema disponibiliza uma série de métodos e ferramentas computacionais que visam dar resposta a este problema.

keywords Network, Graph, Biomedical Terms, Web Application

abstract

Computer science and informatics have proved to be of overriding importance on the recent evolution of biology sciences. The integration of data, the analysis of experimental results and the application of several other computational methods have made possible to expand the set of well-known associations between different biomedical terms. However, this knowledge is spread over several independent databases.

By merging the different types of biomedical terms and the associations between them, into a single network, it's attainable to get a very inclusive big picture which allows the extraction of relevant biomolecular evidences. Representing networks of biomedical terms through graphs, it's possible to carry the mathematical findings in graph theory and apply them in bioinformatics.

The fundamental objective of this dissertation was to develop a Web platform that, through the integration of biological data, enables to visualize, to study and to understand the relationships between diseases, genes and other biomedical terms. This system provides a variety of different computational tools and methods that were designed to address this problem.

Índice

Lista de Figuras

Lista de Tabelas

Acrónimos

Termo	Descrição
ACID	Atomicidade, Consistência, Isolamento e Durabilidade
API	*Application Programming Interface*
DB	*Database*
GO	*Gene Ontology*
GOA	*Gene Ontology Annotation*
JDBC	*Java Database Connectivity*
JSF	*JavaServer Faces*
JSP	*JavaServer Pages*
JUNG	*Java Universal Network/Graph*
KEGG	*Kyoto Encyclopedia of Genes and Genomes*
MVC	*Model–View–Controller*
NoSQL	*Not only SQL*
OMIM	*Online Mendelian Inheritance in Man*
PPI	*Protein-Protein Interaction*
REST	*REpresentational State Transfer*
SaaS	*Software as a Service*
SQL	*Structured Query Language*
WS	*Web Service*
XML	*eXtensible Markup Language*

Capítulo 1
Introdução

1.1 Enquadramento

Com a digitalização sucessiva de processos e serviços, os sistemas de informação estão hoje em dia um pouco por toda a parte. Estes sistemas destacam-se pelas capacidades de armazenamento, de manipulação, de edição e de representação de dados. Na maioria dos sistemas de informação, os dados são componentes individuais que possuem valor por si só e são capazes de representar a informação pretendida. No entanto, existem situações em que, para além dos componentes individuais, a informação reside na forma como esses componentes interagem entre si e na topologia que apresentam. Um cenário concreto de uma destas situações consiste na representação de redes.

A representação de redes trata-se de um tema atual. Na última década, o estudo de redes ganhou novo ânimo, desencadeado pelo interesse espontâneo pelas "redes sociais". Sítios como Facebook [1], Linkedin [2], MySpace [3] ou Twitter [4] passaram a fazer parte dos mais visitados na Web. Esses sítios são exemplos de aplicações que fazem uso intensivo (e quase exclusivo) da rede social real, em que pessoas e/ou entidades interagem entre si através das suas relações sociais. Para além dos portais Web que criaram, estas "redes sociais" disponibilizam vários serviços que estão ainda longe de totalmente explorados.

No entanto, as redes sociais são apenas um entre muitos exemplos. Existem múltiplos tipos de dados que podem ser considerados elementos de uma rede: rede de computadores, rede de estradas, rede de citações, entre muitas outras.

No contexto específico deste trabalho, pretende-se explorar uma rede de termos biomédicos. Podem identificar-se genes, proteínas, vias metabólicas e doenças como

exemplos de elementos da rede. No entanto, são as associações entre esses elementos que tornam possível representar sistemas biológicos através desta abstração. As interações podem revelar influências físicas ou funcionais entre elementos, mas também existem interações de outros tipos e baseadas em conhecimento experimental ou induzido por técnicas computacionais.

Nas últimas décadas, novos conhecimentos na área da bioinformática têm surgido a uma cadência impressionante. Estas constantes reformulações não podem, de forma alguma, ser desassociadas da evolução tecnológica [5]. Este aumento do conhecimento implica uma dilatação da rede de termos biomédicos em dois aspetos essenciais. Primeiro, no crescimento do número total de elementos, onde se podem identificar descobertas de novas doenças, processos biológicos, etc. Segundo, no aumento das interações conhecidas. Essas interações são muitas vezes fruto da capacidade crescente de processamento informático e de novas técnicas e algoritmos de previsão ou dedução de associações.

1.2 Objetivos

O principal objetivo deste projeto consistiu em desenvolver uma *framework* de software capaz de armazenar, processar e representar informação relativa a uma rede de elementos biomédicos. A solução a desenvolver deveria ser acompanhada de um estudo de tecnologias que permita fundamentar as decisões tomadas na fase de desenvolvimento.

Como produto final, foi criada uma aplicação Web, BioMedNet, que permite estudar e compreender as relações entre doenças, genes e outros elementos/termos biomédicos. O propósito essencial da aplicação é permitir a visualização e análise das interações entre elementos biomédicos, tirando partido de uma representação sob a forma de grafo.

1.3 Estrutura da Dissertação

Esta dissertação encontra-se dividida, para além deste, em quatro principais capítulos nos quais são abordados os seguintes assuntos:

Capítulo 2: Descrição detalhada sobre o estado de arte relativo ao âmbito da solução a desenvolver. Apresentação de conhecimentos teóricos consolidados, tanto na área de fundamentos matemáticos da teoria de grafos, como na área informática.

Análise de soluções existentes e tecnologias disponíveis, convenientemente seguida de avaliações comparativas.

Capítulo 3: Descrição do problema e apresentação dos requisitos do sistema a desenvolver.

Capítulo 4: Apresentação do modelo de dados implementado. Explicação pormenorizada da solução desenvolvida, nomeadamente de armazenamento e consulta de dados. Descrição de detalhes de desenvolvimento da aplicação web e dos serviços de acesso a dados. Exposição do resultado final da aplicação desenvolvida.

Capítulo 5: Apresentação e análise do resultado final. Avaliação do trabalho desenvolvido e dos conhecimentos adquiridos. Proposta de futuro trabalho a desenvolver.

Capítulo 2
Redes e Grafos

2.1 Redes e Grafos em Informática

As redes serão um dos temas transversais ao longo de todo o trabalho. Como já foi referido, o estudo de redes tem sido alvo de grande interesse nos últimos anos. A proliferação dos computadores, o aparecimento da Internet e o aumento da capacidade de processamento tornaram possível criar e analisar redes de larga escala [6].

No âmbito da rede social Facebook, foi introduzido recentemente aquilo a que se deu o nome de "grafo social" ("*social graph*"). O Facebook apresentou a sua ideia de mapear, no mesmo grafo, todas as ligações entre pessoas e os seus interesses. O termo utilizado gerou bastante debate na Web, uma vez que, "grafo social" e "rede social" eram utilizados como termos sinónimos.

No entanto, verifica-se que existem diferenças entre o termo "rede" e "grafo". Conforme descrito, uma rede é um sistema de entidades que, de alguma forma, se interligam. Por sua vez, um grafo é essencialmente um conceito matemático. Um grafo é um objeto composto por um conjunto de vértices que se ligam entre si por arestas. Este elemento matemático permite modelar, representar e descrever redes, independentemente do tipo de dados em estudo.

Em ciências da computação, um grafo pode ser representado através de um tipo de dados abstrato. Essa estrutura computacional deve ser capaz de implementar o conceito matemático definido pela teoria de grafos.

Esta capacidade de modelar uma rede sob a forma de um grafo torna possível transitar todos os conhecimentos de teoria de grafos para as ciências que se dedicam à análise de redes. Na próxima secção será feita uma introdução a esse tema.

2.2 Teoria de Grafos

A teoria de grafos tem sido objeto de estudo desde o século XVIII, altura em que foi publicado o primeiro documento sobre o tema [7]. Esta secção introduz alguns conceitos básicos de teoria de grafos essenciais à compreensão da solução desenvolvida.

De uma forma muito simplista um grafo pode ser visto como um conjunto de elementos representados por vértices que estão relacionados entre si através de arestas. Podem, porém, existir diferentes notações para representar a mesma estrutura.

Um grafo pode ser descrito como sendo composto por três elementos devidamente ordenados [8]:

$$G = (V, E, I_G).$$

O elemento V representa o conjunto de vértices ou nós do grafo G. E é o conjunto de arestas (do inglês *edges*) que traduzem as relações entre os vértices presentes em V. E o componente I_G é a função responsável por mapear a cada aresta um par de vértices incidentes.

No entanto, uma forma mais comum e semanticamente idêntica, utilizada por Steen [9], será a terminologia adotada em descrições matemáticas neste documento.

Pode-se então dizer que, o grafo G é constituído por um conjunto de vértices V e um conjunto de arestas E:

$$G = (V, E).$$

O conjunto dos vértices de G é identificado por $V(G)$, enquanto que conjunto de arestas por $E(G)$.

Qualquer elemento $e \in E$ é uma aresta incidente em dois vértices. Se uma aresta e une os vértices $u, v \in V$ então diz-se que $e = (u, v)$. Nesta situação os vértices u e v são denominados por adjacentes. O grupo de vizinhos de um determinado vértice corresponde ao conjunto de vértices com os quais partilha uma das extremidades de

uma aresta, ou por outras palavras, todos os vértices que são adjacentes ao vértice definido.

É importante referir que, nesta situação, a ordem pela qual se representam os vértices é indiferente. Então, a aresta $e = (u, v)$ pode também ser representada como $e = (v, u)$, exprimindo ambas as representações o facto dos vértices u e v serem adjacentes. O mesmo não se poderá dizer caso se trate de um grafo orientado, como será apresentado em seguida.

Os grafos orientados, grafos direcionados ou digrafos, representam uma possível variante aos grafos anteriormente apresentados. Num grafo orientado todas as arestas são orientadas segundo uma direção própria. Uma vez direcionadas, as arestas passam a representar pares ordenados de vértices e a designarem-se de arcos, arestas orientadas ou setas.

Com o intuito de prever este cenário, a definição de grafos orientados sofre uma ligeira alteração:

$$G = (V, A),$$

onde A representa o conjunto de arcos ou,

$$G = (V, E, f),$$

onde o elemento f é uma função responsável por mapear para cada uma das arestas um par ordenado de vértices.

O arco $a = (u, v)$ ou $a = (\overrightarrow{u, v})$ é descrito como orientado de u para v. Assim, a ordem pela qual se apresentam os vértices é determinante para identificar a orientação do arco.

Considerando ainda o arco $a = (u, v)$, o vértice u representa o vértice de origem ou cauda do arco enquanto o vértice v é o vértice de destino ou cabeça do arco. Nesta situação pode ainda dizer-se que v é um sucessor de u e que u é um predecessor de v.

Uma terceira e última variante de grafos, considerada no âmbito deste projeto, são os denominados grafos pesados ou grafos com arestas pesadas. Como o próprio nome indica, os grafos pesados têm a característica de terem associadas a cada uma das suas arestas um peso. O peso de cada aresta pode representar, entre outras medidas, a distância, o custo ou a relevância de cada relação. Associada ao conjunto de arestas existe uma função $w: E \to \mathbb{R}$ que relaciona cada aresta com um número real característico. O peso de uma aresta $e \in E$ é definido por $w(e)$.

A Figura 2.1 apresenta três exemplos de grafos: um grafo, um grafo orientado ou digrafo e um grafo orientado pesado ou digrafo pesado. Todos os grafos apresentam quatro vértices e quatro arestas/arcos, tendo portanto: $|V| = 4$ e $|E| = 4$. Os grafos A, B e C podem ainda ser descritos de formas mais pormenorizadas, para além da cardinalidade dos conjuntos V e E, da seguinte forma:

- Grafo A:

$$V = \{V_1, V_2, V_3, V_4\}$$
$$E = \{(V_2, V_1), (V_1, V_4), (V_2, V_4), (V_3, V_2)\}$$

- Digrafo B:

$$V = \{V_1, V_2, V_3, V_4\}$$
$$E = \{(V_1, V_2), (V_1, V_4), (V_4, V_2), (V_2, V_3)\}$$

- Digrafo pesado C:

$$V = \{V_1, V_2, V_3, V_4\}$$
$$E = \{(V_1, V_2, 1), (V_1, V_4, 9), (V_4, V_2, 3), (V_2, V_3, 4)\}$$

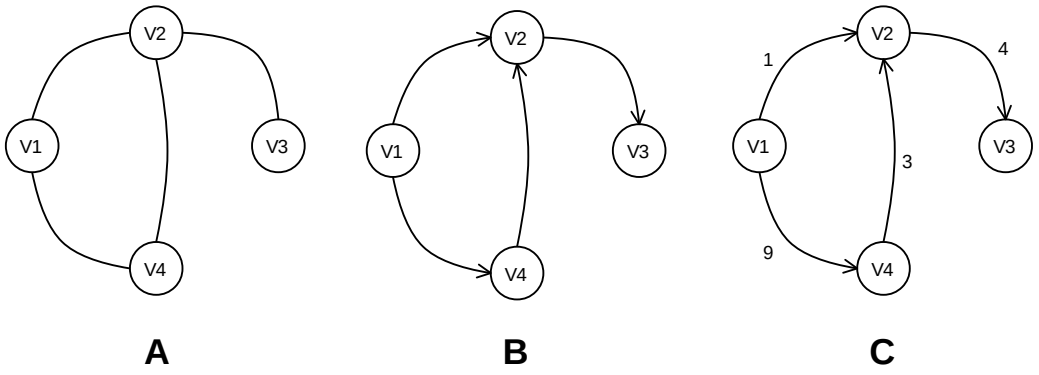


Figura 2.1 – Grafo, digrafo e digrafo pesado (adaptado de [10]*)*

Além da natureza específica de cada tipo de grafos existem muitas outras características partilhadas entre todos os tipos. Será feita, em seguida, uma descrição detalhada sobre algumas dessas características.

Dois tipos muito particulares de arestas são os laços e as arestas múltiplas. A definição de grafo não inibe a possibilidade de existirem estes dois tipos de aresta. Exemplo de um laço é a aresta $e = (u, u), u \in V$, uma vez que ambas as extremidades são incidentes no mesmo vértice. Relativamente às arestas múltiplas, pode dizer-se que a sua presença se verifica quando duas arestas distintas unem o mesmo par de vértices, seguindo como exemplo as arestas e_1 e e_2 em que $e_1 = (u, v), e_2 = (u, v)$, $u, v \in V$. Se se confirmar a presença de laços ou arestas múltiplas num grafo, pode

afirmar-se que esse grafo apresenta características de multigrafo, caso contrário trata-se de um grafo simples.

O grau de um vértice $v \in V$, comumente representado por $deg(v)$, refere o número de arestas conectadas a esse vértice. No caso do vértice *v* fazer parte de um digrafo devem ser considerados dois graus distintos: grau de saída e grau de entrada. O grau de saída ou $deg_{out}(v)$ é o número de arcos no qual *v* é vértice de origem, enquanto que o grau de entrada ou $deg_{in}(v)$ é número de arcos no qual *v* é vértice de destino.

Se $G = (V, E)$ for um grafo, diz-se que $G_1 = (V_1, E_1)$ é um subgrafo de G se $V_1 \subseteq V$, $E_1 \subseteq E$ e todos as arestas pertencentes a E_1 são incidentes em vértices de V_1.

Outro apontamento importante sobre grafos são os caminhos. Um caminho é uma sequência de vértices em que existe sempre uma aresta de um vértice para o vértice seguinte. Um caminho é então uma sequência de vértices $(v_1, v_2, \dots, v_i)$ em que $\{(v_1, v_2), (v_2, v_3), \dots, (v_{i-1}, v_i)\} \subseteq E$. Dois tipos de caminhos muito específicos são os caminhos simples e os ciclos. Os caminhos simples têm a característica especial de não terem vértices repetidos. Os ciclos são caminhos em que o primeiro e último vértice do caminho são o mesmo $v_1 = v_i$. A distância de um caminho corresponde ao número de arestas que constituem o próprio caminho.

A distância entre dois vértices u e v é representada por $\delta(u, v)$ e corresponde à distância do caminho mais curto de u para v. Caso não exista qualquer caminho de u para v diz-se que $\delta(u, v) = \infty$. O caminho mais curto entre dois vértices pode também basear-se na soma do peso das ligações e, nesse cenário específico, deixa de estar associado ao número de arestas e passa a representar o peso dessas arestas.

Tendo como referência os tópicos apresentados, torna-se possível compreender alguns métodos de classificação de vértices. A classificação de vértices pode basear-se em conhecimentos e indicadores muito distintos, mas uma capacidade que é partilhada por todos os métodos é o facto de permitirem compreender e interpretar um grafo de forma mais aprofundada. Apesar do levantamento apresentado não ser exaustivo, os mecanismos de classificação que serão considerados no âmbito deste projeto são apenas três tipos de centralidades:

- Centralidade baseada em grau (*Degree Centrality*). Este tipo de centralidade permite compreender a quantidade de ligações incidentes num determinado vértice. A centralidade de um vértice é dada pelo valor do grau do próprio vértice:

$$C_d(v) = \deg(v)$$

- Centralidade baseada em distância (*Closeness Centrality*). Segundo este modelo de centralidade os vértices mais importantes comunicam de forma mais rápida com os restantes vértices do grafo ou, por outras palavras, a distância desse vértice aos restantes é relativamente baixa. Matematicamente define-se a centralidade do vértice u como:

$$C_{clo}(v) = \frac{1}{\sum_{u \in V}^{|V|} \delta(v,u)}$$

- Centralidade baseada em intermediação (*Betweenness Centrality*). Este tipo de centralidade torna possível analisar a influência de um vértice nas interações entre os restantes vértices. Se um determinado vértice apresenta um valor de centralidade elevado, pode concluir-se que, este faz parte do caminho mais curto de dois vértices aleatórios de forma muito frequente. Considerando σ_{uv} como o número total de caminhos mais curtos entre u e v e $\sigma_{uv}(w)$ o número de caminhos mais curtos entre u e v que passam por w, pode calcular-se a centralidade de um determinado vértice w como:

$$C_b(w) = \sum_{(u,v) \in V(w)} \frac{\sigma_{uv}(w)}{\sigma_{uv}}$$

A teoria de grafos é uma área matemática que se estende muito para além dos conteúdos apresentados ao longo desta secção. No entanto, esta breve introdução teórica é suficiente para a compreensão do projeto.

2.3 Dados Biomédicos

As redes possibilitam a modelação de sistemas biológicos bastante completos [11]. Existem várias fontes de dados biomédicos que podem facilmente ser consultadas. Muitos desses dados podem ser acedidos de forma livre. No entanto, dependendo das políticas de cada fonte, pode ser exigida uma licença de utilização. Em alguns casos essa licença é gratuita quando a utilização se destina a fins académicos.

2.3.1 Modos de Acesso à Informação

O acesso a estes dados biomédicos pode ser assegurado através de diferentes estratégias. Uma das opções frequentemente adotada é a de disponibilizar os dados em *flat files* (ficheiros de texto simples) capazes de comportar *datasets* completos de dados. Este modelo de acesso a dados não representa a solução para todos os problemas, uma vez que exige que seja feito o *download* e *parse* completo do ficheiro. Um dos problemas que este modelo de acesso a dados apresenta é o facto de não

permitir a consulta de um subconjunto pequeno de dados num tempo suficientemente curto, uma vez que requer o processamento do conjunto completo da informação. Outro problema desta solução reside no facto destes ficheiros serem recriados periodicamente pelas respetivas fontes, o que pode não garantir o acesso aos dados atualizados, num determinado momento.

No entanto, no contexto deste projeto, esta será a solução preferida sempre que for necessário aceder a um volume de dados significativo. O motivo desta escolha prende-se com o facto de este método garantir um acesso rápido e fácil a um conjunto de dados avultado. Esses dados serão devidamente interpretados e armazenados na solução desenvolvida.

Contudo, nem todos os dados essenciais para o projeto podem ser obtidos através deste modelo de *flat files*. Em algumas situações isso deve-se ao facto de as próprias fontes da informação não disponibilizarem os dados nesse formato. Por outro lado, será também necessário obter os dados de forma praticamente imediata ou num espaço de tempo tão curto que não interfira na usabilidade da aplicação. Nessas situações o recurso a Web Services (WS) é atualmente a solução preferível. Embora haja diferentes modelos de Web Services os únicos utilizados serão do tipo REST, devido ao seu formato simplista, à sua rapidez e por representarem o estado de arte da tecnologia [12].

2.3.2 Fontes de Dados

As seguintes fontes de dados foram utilizadas para obter os dados necessários a este projeto.

STRING [13] é uma base de dados de interações conhecidas e/ou deduzidas entre proteínas. STRING disponibiliza uma extensa rede de interações entre proteínas (*Protein-Protein Interaction* PPI) mas apenas serão utilizadas as proteínas às quais se reconheça a ligação ao ser humano (*Homo sapiens*). Os identificadores correspondentes às proteínas presentes no *flat file* STRING encontram-se no formato Ensembl Protein [14].

A base de dados OMIM [15] mantém um conjunto de associações entre genes e distúrbios humanos. Através desta fonte é possível obter as associações entre os fenótipos associados a doenças ou distúrbios e os genes humanos. Esses dados serão então integrados com os dados recolhidos do STRING fazendo o mapeamento dos genes humanos em proteínas. Todos os ficheiros recolhidos da OMIM utilizam a representação interna da instituição. Para além desses extensos ficheiros de informação, a OMIM também disponibiliza um WS REST que permite aceder a vários

dos seus recursos. A aplicação resultante deste projeto deverá fazer uso do serviço de pesquisa que esta entidade disponibiliza.

Através da informação recolhida (*flat files*) da base de dados Gene Ontology Annotation [16] podem obter-se ontologias conhecidas para vários genes. Essas ontologias representam termos de três tipos: processos biológicos, componentes celulares e funções moleculares. Cada um desses termos está reconhecido por um identificador único e diretamente relacionado com um ou vários genes em formato UniProt. Tirando partido das relações conhecidas entre genes e proteínas pode juntar-se estes dados aos descritos anteriormente. Recorrendo aos ficheiros disponibilizados pela Gene Ontology [17] é ainda possível obter o nome completo e o tipo de cada um dos termos.

A base de dados KEGG [18] é uma conhecida base de dados de termos biomédicos. O acesso programático a esses dados é livremente garantido através do seu recente WS REST. Desta fonte serão recolhidas as associações conhecidas entre vias metabólicas e genes humanos. O serviço permite converter os identificadores de genes KEGG em notação UniProt. É ainda possível obter os nomes completos das vias metabólicas.

Outro WS que será utilizado pela aplicação é o disponibilizado pela UniProt [19]. Os dados recolhidos desses serviços permitirão quer fazer pesquisas sob a sua basta coleção de proteínas, quer mapear em identificados UniProt os dados recolhidos pelas fontes anteriores.

2.4 Armazenamento de dados

O modelo de armazenamento de dados é um dos primeiros problemas que se coloca quando se pretende desenvolver qualquer sistema de informação. Num sistema onde os grafos são a base da informação, como é o caso deste projeto, a mesma questão mantém-se. Foram testadas duas abordagens a este problema, uma baseada nas tradicionais bases de dados relacionais, outra nas emergentes bases de dados *NoSQL*.

2.4.1 Bases de Dados SQL

As bases de dados relacionais são, na esmagadora maioria dos casos, a solução de eleição quando se pretende ter um registo permanente de informação estruturada. Neste problema específico, podemos encontrar nessa tecnologia uma solução possível.

As bases de dados relacionais contam com um vasto leque de características que as tornam muito atrativas, pelo que têm sido exaustivamente utilizadas ao longo das últimas décadas, tanto em projetos académicos como em soluções profissionais. A enorme penetração destes produtos no mercado obrigou também a sistemáticas otimizações e correções, tornando-a mais robusta e estável [20].

Uma das facilidades desta tecnologia é o facto das implementações da linguagem SQL não diferirem muito entre os diferentes produtos, o que garante a familiarização dos programadores com os mecanismos de utilização, diminuindo tempos de aprendizagem ou adaptação e maximizando a produtividade. Para além disso, esta característica, uniforme a todas as soluções de dados, permite uma mudança entre implementações sem grandes esforço e/ou custo.

Dentro desta tecnologia podemos encontrar uma diversidade grande de produtos, muitos de qualidade reconhecida. Bases de dados como Microsoft SQL Server [21], Oracle DB [22] ou MySQL [23] são exemplo disso.

Microsoft SQL Server e Oracle DB são produtos de duas das principais empresas na área tecnológica, Microsoft e Oracle respetivamente. Estas soluções comerciais são fruto de projetos dispendiosos que garantem a qualidade e desempenho dos seus produtos. Além disso, contam ainda com uma completa documentação e um suporte eficaz.

Por outro lado, a solução gratuita MySQL (também propriedade da Oracle) faz-se acompanhar de uma ampla comunidade. A qualidade do produto, juntamente com a licença gratuita do mesmo, fizeram desta base de dados uma das escolhas preferidas para projetos com reduzida capacidade de investimento.

As bases de dados SQL conseguiram, aos olhos do mundo, a reputação de fidedignas e seguras. Com algum esforço de desenvolvimento podem ser utilizadas como uma solução de armazenamento no cenário do problema em estudo.

2.4.2 Bases de Dados NoSQL

Embora as bases de dados relacionais se façam acompanhar de todas as qualidades referidas na secção anterior, tem-se assistido a um aumento significativo de popularidade de outros tipos de bases de dados: as bases de dados NoSQL ("*Not only SQL*"[1]).

[1] "Não apenas SQL" (tradução do autor)

As bases de dados NoSQL são uma alternativa ao modelo relacional. Estas bases de dados garantem o armazenamento de dados e *frameworks* exclusivas de acesso. Foram pensadas e criadas com o intuito de garantir capacidades onde as bases de dados relacionais são bastante ineficientes ou simplesmente desadequadas.

Um dos momentos mais importantes na vida das bases de dados NoSQL aconteceu em 2007, quando a Amazon [24] apresentou a sua nova solução de armazenamento de dados, o Dynamo [25]. Esta solução baseou-se num modelo "chave-valor", um dos três grandes paradigmas NoSQL.

Os três principais paradigmas de bases de dados NoSQL são [25]:

- Chave-valor (*Key-value Stores*): em que a base de dados pode ser vista como uma tabela de dispersão (tabela *hash*). Dynamo é um exemplo deste modelo.
- Orientadas a colunas (*Column-oriented Database*): modelo adotado por Bigtable (da Google [26]) e Cassandra (do Facebook) entre outros, guarda e processa a informação em colunas ao contrário do conhecido modelo orientado a registos (linhas).
- Orientadas a documentos (*Document Database*): bases de dados muito semelhantes às chave-valor desenvolvidas, no entanto, com o pretexto específico de armazenar coleções de documentos. MongoDB [27] e Apache CouchDB [28] implementam este paradigma.

No entanto, para além dos três grandes tipos, existem ainda bases de dados do tipo grafo (*Graph Database*). Pelo próprio nome se percebe que estas representarão uma das principais alternativas no contexto deste projeto.

As bases de dados baseadas em grafos operam sobre dados na forma de grafos e os seus principais elementos: vértices e arestas. A base de dados Neo4j [29] é uma das mais populares no género.

Os criadores do Neo4j descrevem a sua base de dados como um modelo de representação de dados em esquema de grafo bastante intuitiva. Além disso, os autores desta ferramenta salientam o seu desempenho e capacidade de escalabilidade.

O Neo4j pode ser utilizado quer embutido em projetos (como biblioteca) quer como aplicação autónoma (como servidor). A ferramenta disponibiliza uma API Java orientada a objetos bastante simples e atraente.

A Figura 2.2 representa, de um modo esquemático, o modelo de armazenamento da base de dados Neo4j. Os vértices relacionam-se através de arestas que podem ser direcionadas. A informação é armazenada nos vértices e arestas através de propriedades, que funcionam segundo um modelo "chave-valor".

Existem no entanto fatores desfavoráveis à utilização desta ferramenta, uma vez que se trata de uma tecnologia nova e ainda insuficientemente explorada.

O Neo4j implementa uma interface de acesso aos dados armazenados no grafo mas não apresenta qualquer modelo *standard* de acesso, como o *SQL* nas bases de dados relacionais. Não existindo uma API definida, cada base de dados grafo pode implementar o seu próprio modelo de acesso a dados. Porém, uma possível mudança para outra solução de armazenamento de dados implicará um esforço de desenvolvimento considerável.

A falta de suporte e algumas dúvidas relativamente a segurança podem, também, contribuir para o desinteresse na ferramenta [20]. Em ambiente de produção será um problema grave ficar dependente de uma ferramenta pouco madura.

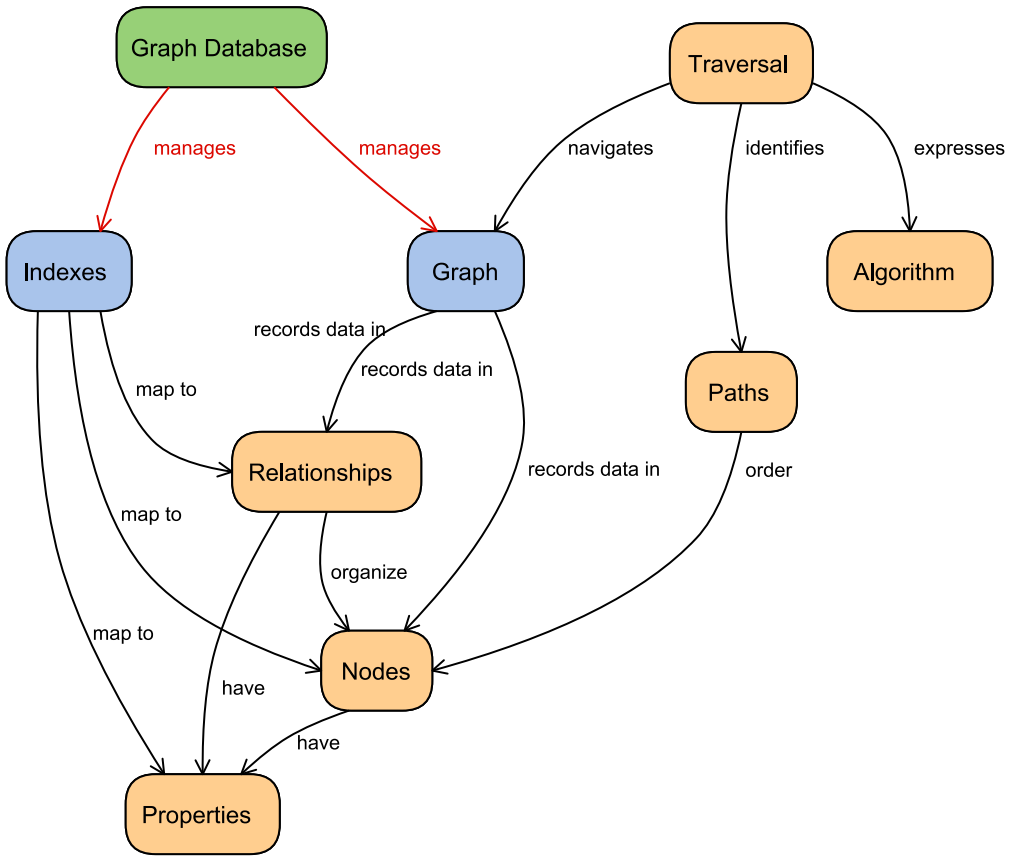


Figura 2.2 – Conceitos da base de dados Neo4j (adaptada de [30]*)*

2.5 Representação de Grafos

Depois de criado um grafo é importante conseguir representá-lo, pois essa será uma das possíveis formas de acesso aos grafos. Este acesso poderá ser tanto por parte de utilizadores como de outras aplicações.

Por um lado existe a representação gráfica. Esta representação permite a visualização dos vértices e arestas de um grafo. Esta será, certamente, a forma mais fácil de interação com um utilizador humano. A possibilidade de observar e analisar visualmente um grafo pode ter um papel preponderante sobre a interpretação do valor semântico da rede.

Por outro lado existe a representação lógica de grafos. Este tipo de representação pode ser conseguido com o auxílio de formatos de descrição capazes de simbolizar a topologia de uma rede. Esta representação deverá permitir a troca de informação sob a anatomia do grafos entre diferentes aplicações e/ou componentes. Este modelo de representação poderá também ser utilizado para importar ou exportar dados.

2.5.1 Plugins de Visualização Web

Uma vez que o sistema a desenvolver deverá ter uma aplicação/portal Web, a visualização gráfica deverá ser facilitada através de uma página Web. Existem várias ferramentas que se propõem a conseguir este objetivo, nomeadamente: Cytoscape Web [31], JavaScript InfoVis Toolkit [32], Cobweb [33] e Arbor.js [34].

O Cytoscape Web é um componente visual (Figura 2.3) que atua sobre o lado do cliente (navegador Web). Esta ferramenta é capaz de incorporar diferentes redes numa página HTML, segundo diferentes estilos visuais. Esta ferramenta dispõe de uma API JavaScript bastante extensa. Essa interface para além de grande é também muito rica, permitindo um controlo pleno do componente de visualização. Os dados que são exibidos pelo Cytoscape Web tanto podem ser carregados a partir de ficheiros com um dos formatos suportados, como introduzidos manualmente através da interface de utilização [35]. O Cytoscape Web é capaz de interpretar uma gama de formatos de representação muito variada.

O sistema de navegação pelo grafo é muito intuitivo e fácil de utilizar. Servindo-se exclusivamente do rato e/ou alguns controlos de navegação, o utilizador é capaz de movimentar-se pela área de representação, aumentar e diminuir a dimensão dos elementos, selecionar vértices ou arestas, etc. As redes podem ser exibidas segundo diferentes esquemas de representação: radial, circular, *force-directed*, entre outros. Os estilos visuais são amplamente personalizáveis, desde as formas e legendas dos

vértices até às espessura e cor das arestas. É também possível adicionar aos elementos gráficos eventos de interação com o rato que facilitem a utilização ao utilizador.

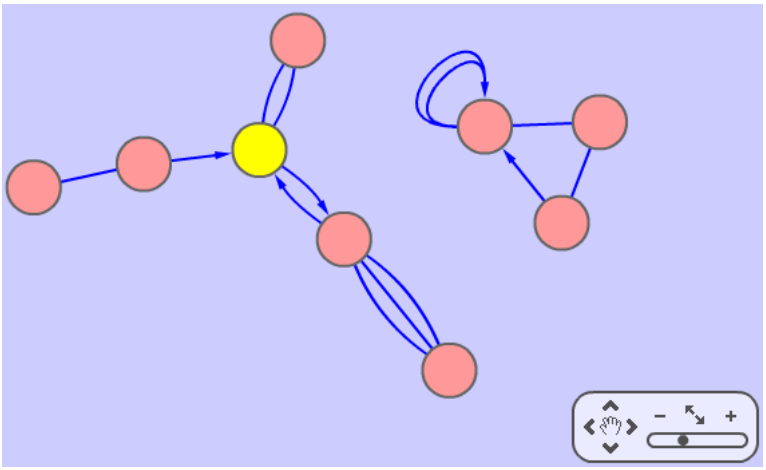


Figura 2.3 – Cytoscape Web [31]

Lopes e outros [35] realizaram testes num computador portátil que mostraram que o Cytoscape Web consegue ter um funcionamento bastante competente até aos 2000 elementos (800 vértices e 1200 arestas).

O Cytoscape Web é um projeto *open source* aberto à colaboração de qualquer programador. No seu sítio Web pode ser encontrada documentação de qualidade, assim como vários exemplos demonstrativos.

O Cytoscape Web apresenta, no entanto, uma desvantagem que pode facilmente desmotivar a sua utilização. Essa reside no facto de a implementação interna do seu componente de visualização ter sido concebida em Flex/ActionSctipt. Esse fator obrigará o suporte à tecnologia Flash por parte do navegador Web. Embora o suporte Flash seja garantido nos principais navegadores em ambiente *desktop*, tem-se verificado, nos últimos três anos, uma tendência de mudança de tecnologias. O início desta transição pode ter sido precipitado quando a Apple inibiu o suporte à tecnologia Flash em todos os seus dispositivos móveis [36]. A verdade é que o HTML5 tem-se mostrado uma séria alternativa ao Flash e tem crescido em popularidade e aceitação. A própria Abobe acabou por descontinuar o desenvolvimento do Flash Player para ambientes móveis.

Motivados por esta possível mudança de tecnologia foram, entretanto, produzidos alguns desenvolvimentos numa nova versão do Cytoscape Web, exclusivamente baseada em HTML5 [37]. A manter-se inalterada a API JavaScript fornecida pela atual

versão, o processo de migração de conteúdos, para esta nova versão, deverá ser simples.

Face a todas as virtudes supra-mencionadas e após uma avaliação comparativa aos recursos disponibilizados pelas restantes ferramentas, a escolha acabou por recair sobre o Cytoscape Web.

2.5.2 Representação Lógica

Perante à necessidade de acesso e/ou partilha de dados é importante que se garanta também uma representação lógica dos grafos. A representação gráfica é a interface mais fácil e intuitiva para o utilizador humano, porém não é um modelo que permita o acesso aos dados por diferentes componentes informáticos. Para essas situações deverão ser utilizadas formas de representação lógica. Existem formatos de descrição de grafos capazes de atingir esses propósitos. De entre os vários formatos já existentes uns são mais simples e minimalistas enquanto outros são mais complexos mas com um poder de representação maior.

O Cytoscape Web, descrito na secção anterior, é capaz de consumir dados em múltiplos formatos distintos: GraphML [38], XGMML [39], SIF, entre outros. Uma vez que a escolha para o *plugin* de representação gráfica recaiu sobre essa ferramenta, faz todo o sentido eleger por uma das opções que ele próprio também utiliza. Dessa forma, para além de ser possível disponibilizar grafos a terceiros, torna-se também possível alimentar a própria representação Web com esses dados.

O SIF é um formato de texto demasiado simples e que, por isso, não tem o poder de representação necessário para a conceção deste projeto. Já o GraphML e o XGMML são dois formatos de representação de grafos bastante complexos. Ambas as normas orientam documentos de texto baseados em XML e com um poder de descrição que vai para além das necessidades deste projeto.

O GraphML permite a representação de um ou mais grafos por documento. Os grafos podem ser direcionados ou não, é possível associar-se pesos às arestas e podem representar-se arestas múltiplas e laços. Para além de descrever a topologia do grafo, é possível definir atributos em todos os elementos: grafos, vértices ou arestas. Os atributos são definidos por uma chave que os identifica e podem ser de um dos seguintes tipos: *boolean*, *int*, *long*, *float*, *double* ou *string*. A norma GraphML prevê ainda conceitos mais complexos e que não fazem parte do âmbito deste projeto.

No Anexo A pode ser encontrada, a pretexto de exemplo, a descrição de um grafo em GraphML e a respetiva representação visual.

2.6 Processamento de Grafos

Atualmente existem diversas ferramentas que permitem a modelação de dados que podem ser representados sobre a forma de grafo. De um modo geral, essas ferramentas implementam APIs que, para além da modelação, permitem efetuar análise e processamento dos dados. Frequentemente estas ferramentas incluem visualizadores com a capacidade de representar os dados que o sistema modela. É possível identificar, entre outras, algumas dessas ferramentas: JGraphT [40] , JDSL [41], JUNG [42], etc.

Embora, de modo geral, qualquer uma das alternativas aparente ser de boa qualidade, todas apresentam debilidades no que diz respeito a modelos de armazenamento de dados.

A biblioteca de dados JUNG acabou por fazer parte da solução desenvolvida neste projeto. A arquitetura desta ferramenta permite a representação de uma grande variedade de grafos. Além disso, o extenso conjunto de funcionalidades e algoritmos que acompanham a biblioteca JUNG permite analisar e processar os dados do grafo através de diferentes formas. Uma vez que se trata de um projeto *open-source* e que se faz acompanhar por uma documentação de qualidade, é possível criar, de forma simples e compreensível, extensões ou *plugins* que completem a ferramenta e se adequem às necessidades de cada projeto.

Capítulo 3
Requisitos do Sistema BioMedNet

O objetivo principal deste projeto é desenvolver uma ferramenta que permita estudar as interações conhecidas entre doenças, proteínas e outros termos biomédicos. Esta ferramenta deverá permitir a qualquer utilizador o acesso a uma representação visual e intuitiva da rede semântica de termos biomédicos disponibilizada pela aplicação.

Neste capítulo são apresentados os requisitos que foram identificados para a aplicação BioMedNet e que permitem atingir os objetivos aos quais o projeto se propõe.

3.1 Visão do Projeto

A evolução na área da informática provocou uma completa revolução em várias áreas da biologia, como na genética ou biologia molecular, mas existem muitos outros exemplos [5], [43]. A chegada da informática ao mundo da biologia colaborou, de forma crucial, para um crescimento drástico de todo o conhecimento científico. A descoberta de elementos biológicos e de novas associações biológicas levou à expansão, de forma muito significativa, da rede semântica de termos biomédicos. Existem atualmente várias bases de dados públicas que conservam essas redes de larga escala.

As principais redes de termos biomédicos, aquelas que são mais populares e vastamente utilizadas, foram geradas através de quatro principais métodos [11]: seleção manual com base em subconjuntos reduzidos de resultados experimentais, técnicas computacionais de alto rendimento (*High-Throughput Screening*), mineração de dados automática de literatura científica publicada e previsão computacional

através da ampla integração de dados. Naturalmente, a tradicional seleção manual baseada em relatórios de dados de pequenas experiências é o método que produz as interações mais fidedignas. Apesar da inferior precisão dos métodos computacionais, estes são uma mais-valia em todo o processo de descoberta de possíveis interações, uma vez que permitem identificar rapidamente uma listagem de prováveis elementos candidatos. Têm existido vários desenvolvimentos em novas técnicas computacionais por forma a conseguir resultados mais eficazes diminuindo a quantidade de falsos positivos [44], [45].

Apesar da evolução de todos os processos computacionais que permitem gerar novo conhecimento, a ação humana continua a ser preponderante para os resultados do sistema. Toda a experiência e capacidade de interpretação e análise de um cientista, face a um conjunto de elementos e associações de uma rede, não devem ser menosprezadas. Além disso, também é essencial que qualquer profissional, das áreas de biologia ou biomedicina, seja capaz de compreender as possíveis razões que levaram um método computacional a prever uma possível associação entre dois termos distintos.

Uma forma de simplificar a intervenção humana neste processo seria disponibilizar uma representação visual da rede em estudo. Seria assim possível retirar partido de todo o poder de representação de um grafo e aplicá-lo a este problema em concreto. Reunindo numa visão unificada os principais elementos e associações de uma rede, tornar-se-á possível, aos olhos humanos, ter uma compreensão mais específica e pormenorizada ou mais geral e globalizada, quanto necessária.

Certamente que uma ferramenta que se mostrasse capaz de cumprir esse propósito teria um público-alvo bastante específico. Essa ferramenta destinar-se-ia essencialmente a profissionais nas áreas de biologia, biomedicina e bioinformática. Cientistas, docentes e estudantes das respetivas áreas de intervenção, deverão ser os principais beneficiários dessa ferramenta. Qualquer outro conhecedor da natureza dos dados abrangidos pela rede poderá encontrar nessa funcionalidade outros motivos de interesse. Não se espera, no entanto, que um eventual utilizador, sem conhecimentos em matéria de biologia, consiga retirar informações relevantes dessa ferramenta.

Strogatz [46] responde à pergunta: "*Why is network anatomy so important to characterize? Because structure always affects function."*[2]. É exatamente motivado por essa necessidade de compreensão do funcionamento que com o desenvolvimento deste projeto, se propõem a implementação de um serviço Web que permita estudar e analisar as interações conhecidas entre doenças, genes e outros termos biomédicos.

3.2 BioMedNet como *Software as a Service*

Os serviços disponibilizados através da Internet, ou *Cloud Computing Services*, têm crescido de popularidade e aceitação na área das tecnologias de informação e comunicação. Através desses serviços da *Cloud* torna-se possível aceder, através da Internet, a múltiplos recursos [47]. Recursos de infraestrutura, plataforma e software conseguem ser garantidos através dos três principais tipos de serviços da *Cloud* (Figura 3.1): *Infrastructure as a Service (IaaS)*, *Platform as a Service (PaaS)* e *Software as a Service (SaaS)*.

As aplicações desenvolvidas para Web encontram-se na camada de SaaS e esta camada representa o topo da pilha de serviços, uma vez que a sua existência é suportada pelas camadas de serviços inferiores. As aplicações desenvolvidas segundo este modelo podem facilmente ser acedidas pelos utilizadores finais através de portais Web com recurso a Web Services [48] e tecnologias Web 2.0 [49].

São muitas as vantagens que este modelo de desenvolvimento oferece, tanto do ponto de vista do programador como do utilizador final [50]. Aos olhos do programador o desenvolvimento de uma aplicação seguindo o modelo SaaS, conduz a várias facilidades. Nestas condições torna-se possível descartar, desde logo, problemas de heterogeneidade de *hardware* e sistemas, uma vez que a infraestrutura e sistema representam um ambiente controlado. Por outro lado, a gestão de uma aplicação Web é também mais simples, uma vez que a implementação e testes necessitam apenas de ser feitos numa única e determinada plataforma. De outro ponto de vista, os utilizadores também podem tirar partido de uma experiência de utilização melhorada, uma vez que a aplicação encapsula a plataforma e a infraestrutura. Ao contrário dos softwares tradicionais, o único requisito para aceder a uma aplicação desenvolvida segundo o modelo de SaaS é um navegador Web. Esta facilidade de acesso tem contribuído para que os utilizadores tendam, de forma gradual, a substituir as suas tradicionais aplicações *desktop* por serviços Web com as mesmas funcionalidades. Os tradicionais clientes de *email* e editores de texto são

[2] "Porque é que a anatomia de uma rede é tão importante de caracterizar? Porque a estrutura afeta sempre o funcionamento." (tradução do autor)

agora frequentemente substituídos por aplicações Web. A esta ubiquidade de acesso pode também acrescentar-se a facilidade de gestão, tendo em conta que para além das próprias aplicações também todos os dados e *backups* passam a ser guardados e manipulados na *Cloud*.

Para seguir as tendências de desenvolvimento de software, a plataforma BioMedNet foi desenvolvida e implementada segundo um modelo de *Software as a Service*. Qualquer utilizador terá acesso simples e permanente à aplicação e aos dados disponibilizados, desde que se faça acompanhar de uma conexão à Internet (Figura 3.2). Assim, os utilizadores serão libertados das obrigações de gerir e atualizar, mas sobretudo armazenar, um volume de dados que ainda se espera considerável.

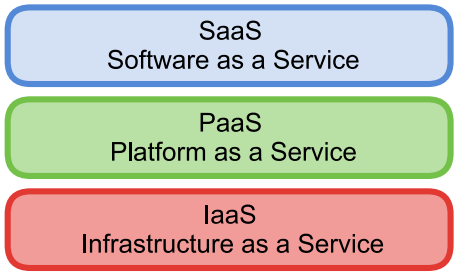


Figura 3.1 – Serviços da Cloud

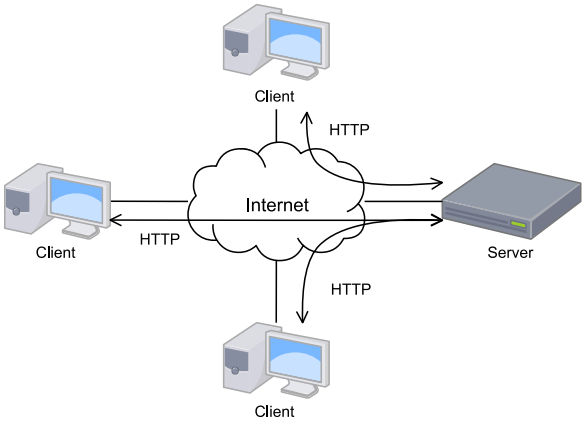


Figura 3.2 – Modelo Cliente-Servidor da aplicação BioMedNet

3.3 Requisitos Funcionais

Com vista a atingir os objetivos aos quais a aplicação se propôs, foram identificados os principais requisitos funcionais. Esses requisitos representaram as principais funcionalidades que viriam a ser implementadas. A Tabela 3.1 apresenta uma lista dos principais requisitos, acompanhados de uma breve descrição.

Tabela 3.1 – Lista de requisitos funcionais

Requisito	Descrição
Visualizar uma subsecção da rede	Deve ser utilizada a estrutura de um grafo por forma a representar uma subsecção da rede. Uma vez que a extensão da rede é tão basta, apenas uma pequena subsecção pode ser visualizada.
Adicionar um elemento como elemento base	A lista de elementos base deve ser utilizada no processo de filtragem da rede. A subsecção filtrada deve ter como base os elementos que constituem esta lista.
Procurar elementos da rede	A procura de elementos da rede é obrigatória para o reconhecimento do primeiro constituinte da lista de elementos base. Para a deteção dos restantes e facultativos constituintes é opcional.
Aplicar filtros à rede	Deve ser possível definir vários filtros que serão tomados em conta na filtragem da rede. A filtragem deve limitar tanto o número de elementos como as características das associações e da subsecção do grafo gerado. Serão considerados vários tipos de filtragem: por número máximo de vértices ou arestas, por profundidade na máxima na rede, por confiança das arestas e por máximo de elementos por cada nível de profundidade.
Procurar associações entre dois elementos	A procura de associações entre dois elementos deve facilitar o estudo de relações entre tais elementos.
Visualizar métricas de elementos	Para efeitos de estudo, deve ser possível aplicar métodos de classificação de vértices (da teoria de grafos) aos elementos representados no grafo.
Ver detalhes de um elemento	Para se conhecer um pouco mais sobre cada elemento deve ser disponibilizada alguma informação extraordinária: o nome completo, o tipo e a fonte. Sempre que possível também deve ser exibida uma ligação para se obter mais informação na respetiva fonte.

Ver associações de um elemento	Com base num elemento determinado, deve ser possível obter uma lista de todas as interações em que esse elemento participa.
Ver detalhes de uma associação	Tal como acontece com os elementos, também alguma informação complementar deve ser apresentada sobre cada associação, como: o tipo, a confiança da associação. Sempre que existam mais tipos de confiança estes também devem ser apresentados.

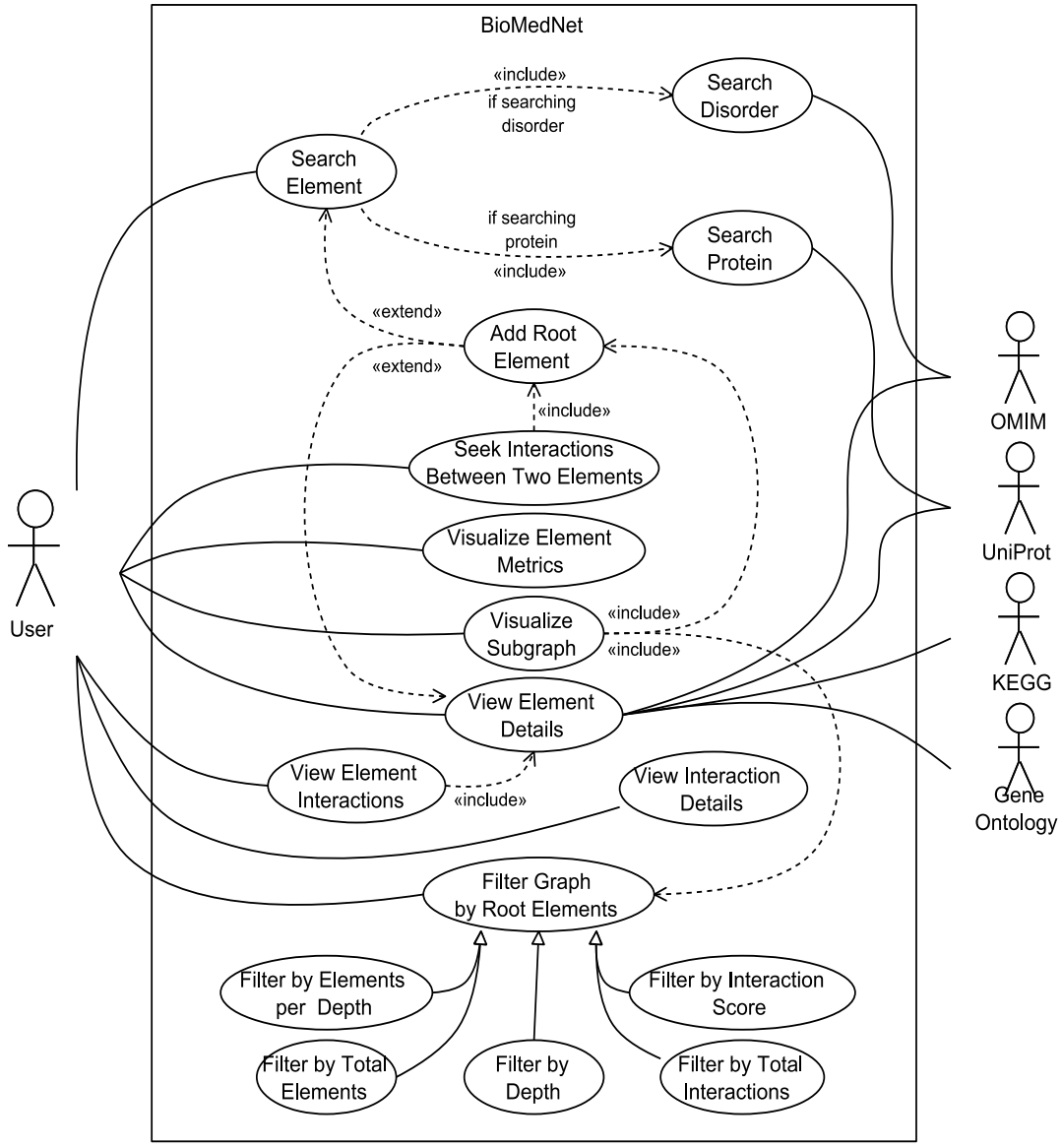


Figura 3.3 – Diagrama de Casos de Utilização

A aplicação foi desenvolvida com base nos requisitos que foram identificados anteriormente. Considerando todas as funcionalidades que foram implementadas e todas as fontes de dados que foram utilizadas construiu-se o diagrama de casos de utilização (Figura 3.3).

Só existe um tipo de utilizador do sistema. Os utilizadores são representados pelo ator do lado esquerdo do sistema. Os atores que se encontram sobre o lado direito representam as entidades externas que interagem com o sistema.

Os casos de utilização de *Procura de Doença* (*Search Disorder*) e *Procura de Proteína* (*Search Protein*) têm relações com os atores OMIM e UniProt uma vez que consomem dados que são disponibilizados pelos Web Services das respetivas entidades. Já o caso de utilização de *Ver Detalhes de Elemento* (*View Element Details*) encontra-se diretamente relacionado com todas as entidades externas, onde estas relações representam hiperligações para páginas dos sistemas das respetivas entidades.

3.4 Requisitos Não Funcionais

Para além dos requisitos funcionais existiam necessidades não funcionais que o sistema BioMedNet deveria ser capaz de cumprir. Essas características deveriam garantir a qualidade geral do sistema.

3.4.1 Qualidades do Sistema

A facilidade de acesso do SaaS, apresentada anteriormente, constituía uma das principais exigências deste sistema. Sempre que um biólogo, biomédico ou bioinformático pretenda utilizar o sistema, este deverá mostrar-se disponível. Esse é um fator decisivo para a criação de um sistema eficiente para todos os interessados.

Um dos fatores dos quais depende a qualidade do sistema é a confiabilidade dos dados explorados. Para maximizar as potencialidades desta ferramenta teriam de ser utilizados dados biomédicos provenientes de diversas fontes fidedignas, uma vez que qualquer análise baseada em informação pouco rigorosa será inevitavelmente mais imprecisa.

Algumas preocupações de desempenho deviam também ser rigorosamente cumpridas. O sistema deveria ser responsivo e capaz de processar pedidos em tempos suficientemente curtos (definitivamente inferior a quinze segundos) para não forçar o utilizador a períodos prolongados de espera. Era esperado que a utilização a dar à *framework* solicitasse o armazenamento e consulta de um volume de dados

relativamente grande. Também era previsto que esse fator influenciasse, de forma decisiva, o tempo de espera a acesso a dados. A capacidade de resposta do sistema era de importância fulcral e preocupação permanente durante toda a fase de desenvolvimento. Uma vez que como produto final teria de ser criado um visualizador gráfico de elementos da rede, baseado na Web, era importante que o tempo de acesso a uma subsecção de dados fosse suficientemente curto no contexto de utilização do sistema. No entanto, tornar-se-ia inevitável que para operar com grandes volumes de dados, o tempo de acesso à informação fosse gradualmente superior. Não se esperava pois, que operações que solicitassem uma desmedida quantidade de dados armazenados pudessem ser utilizadas, por exemplo, num contexto de visualização na Web. A aptidão para aceder a um subconjunto de dados requeridos por um qualquer utilizador seria elemento primordial e fator decisivo na escolha da solução a desenvolver.

3.4.2 Interface do Sistema

O sistema BioMedNet deveria resultar numa aplicação Web e, por isso, comunicar com os utilizadores através de um conjunto de páginas Web.

Uma vez que não existe diferenciação entre utilizadores, o acesso ao sistema deve ser sujeito a um modelo padronizado e singular de visualização, pois todos os utilizadores terão ao seu dispor a mesma interface de utilização. Motivado por esse fator, tornou-se importante desenvolver uma interface que facilmente se adaptasse ao perfil dos utilizadores esperados. O sistema deve ser intuitivo e autoexplicativo, fazendo-se acompanhar sempre que necessário de instruções de utilização.

Numa fase ainda embrionária do desenvolvimento do projeto, foi concebido um protótipo, daquela que seria a página principal da interface de utilização da aplicação (Figura 3.4). Uma larga zona da página é exclusivamente dedicada à representação do grafo, dando destaque aos elementos e à anatomia da rede. Esta área de representação deveria apresentar funcionalidades de navegação e interação, permitindo ao utilizador mover-se e ampliar o grafo, assim como selecionar elementos do mesmo. As restantes funcionalidades do sistema deveriam estar diretamente acessíveis através de um menu lateral, na página principal. As ações dos utilizadores sobre os menus de funcionalidades deveriam ter efeito imediato na perspetiva de visualização sobre o grafo, assemelhando-se ao comportamento de uma aplicação *desktop*. A separação nestas duas principais áreas permite uma visualização permanente do conteúdo do grafo e um controlo rápido e eficaz de todas as funcionalidades disponíveis.

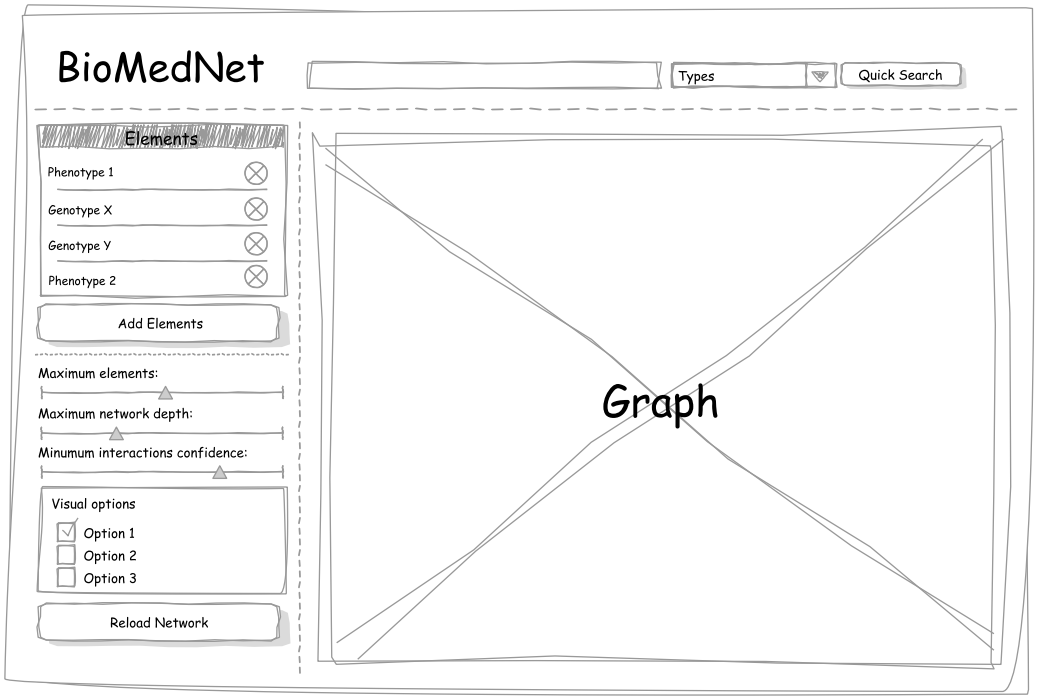


Figura 3.4 – Protótipo da interface gráfica da aplicação Web

3.4.3 Interface com Sistemas Externos

Para além de serem exibidas hiperligações para páginas Web de outros sistemas, o sistema de pesquisa da aplicação BioMedNet depende diretamente da comunicação com outros sistemas externos. Conforme apresentado nos requisitos funcionais da aplicação, a pesquisa de elementos dos tipos doença e proteína está dependente dos dados disponibilizados pela OMIM e pela UniProt, respetivamente. Esses dados provenientes de sistemas externos serão consumidos através dos Web Services REST de ambos os sistemas.

3.5 Modelo e Arquitetura

Uma vez identificados os requisitos que o sistema devia atender, foi modelada toda a solução a desenvolver.

3.5.1 Componentes

O sistema BioMedNet foi desenvolvido tendo por base três componentes essenciais (Figura 3.5): BMNetwork, BioMedNet e BioMedNetPortal. Todos os componentes foram inteiramente desenvolvidos no âmbito deste projeto e totalmente implementados com recurso exclusivo a tecnologias Java. Mais informação sobre o desenvolvimento dos três componentes será apresentada no Capítulo 4.

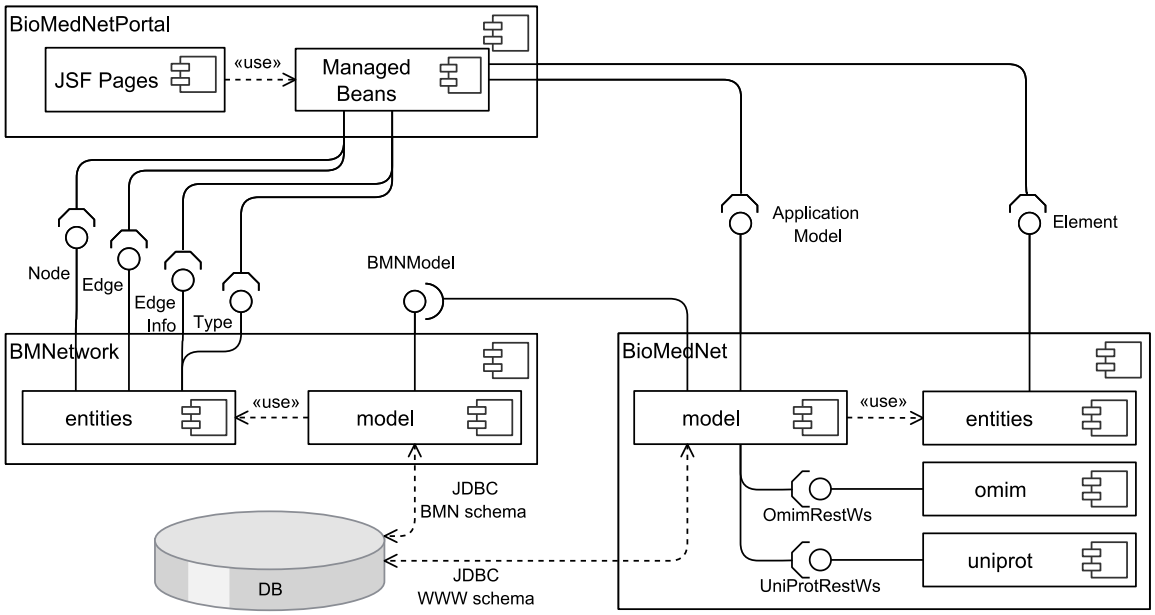


Figura 3.5 – Diagrama de componentes

3.5.1.1 Componente BMNetwork

O componente BMNetwork corresponde a uma *framework* com a capacidade de armazenar, editar e aceder a dados de um grafo. A *framework* é totalmente genérica, pois permite liberdade total sobre o tipo de vértices e arestas do grafo que serão utilizadas.

Esta ferramenta depende de um módulo de armazenamento de dados que cumpra uma interface bem definida. Com a realização deste projeto, e uma vez que a solução de armazenamento eleita foi o SQL Server da Microsoft, foi implementado um módulo responsável pelo armazenamento baseado no modelo apresentado na Figura 3.6. A interação deste componente com a base de dados é exclusivamente feita através do *schema* BMN.

Se devidamente utilizada, a *framework* BMNetwork pode ainda tirar partido das capacidades para processamento da biblioteca JUNG, apresentada na secção 2.6 (Processamento de Grafos).

3.5.1.2 Componente BioMedNet

O componente BioMedNet foi desenvolvido com funcionalidade dupla:

- Aplicativo *desktop*, uma vez que o componente implementa um *script* que é responsável pelo pré-processamento inicial da aplicação Web. Este

processo deve recolher toda a informação necessária e inserir todo o conteúdo devidamente estruturado na base de dados.

- Biblioteca de suporte ao componente BioMedNetPortal. O componente incorpora as camadas de acesso a dados (*Data Access Layer*) e a camada de lógica da aplicação (*Business Logic Layer*).

3.5.1.3 Componente BioMedNetPortal

O componente BioMedNetPortal é responsável por toda a gestão das páginas e dos serviços Web. Este componente implementa a interface REST do sistema e controla as *Managed Beans,* da tecnologia Java, necessárias ao correto funcionamento de todas as páginas Web.

3.5.2 Modelo de Dados

Na secção 2.4 (Armazenamento de dados) foram apresentadas duas ferramentas de armazenamento de dados: Microsoft SQL Server e Neo4J. Com vista a atingir os requisitos propostos, tornou-se necessário a presença de um registo permanente com a informação da rede em estudo. Uma vez que ambas as ferramentas apresentavam caraterísticas interessantes, para cada uma delas foi modelada e implementada uma solução de armazenamento. Na secção 4.1 (*Framework* de armazenamento e acesso a dados) serão apresentados os modelos de dados desenvolvidos, assim como detalhes de implementação, em ambas as tecnologias. No entanto, a escolha acabou por recair sobre a ferramenta da Microsoft uma vez que se mostrou capaz de melhor servir os interesses do projeto.

O modelo de dados implementado pela *framework* BMNetwork pode ser representado através do diagrama de base de dados da Figura 3.6. Este modelo de dados tem a capacidade de armazenar informação relativa aos vértices e arestas de um grafo. Cada vértice do grafo é caraterizado através de um identificador único, que foi, sempre que possível, o nome atribuído pela respetiva fonte de informação. O modelo consagra também a possibilidade de existirem múltiplos pesos por aresta. Esta capacidade deve ser utilizada, por exemplo, quando se verificar que uma aresta pode ser resultado de mais que um estudo ou dedução computacional. Todos os elementos do grafo (vértices, arestas e pesos múltiplos) deverão estar associados a um tipo que identifique a natureza do elemento. Caso não exista necessidade de especificar o tipo do elemento, podem ser utilizados tipos padrão. É ainda mantido um registo "chave valor" que permita controlar informação referente ao grafo, por exemplo, se o grafo é direcionado ou não e se possui tipos padrão para elementos.

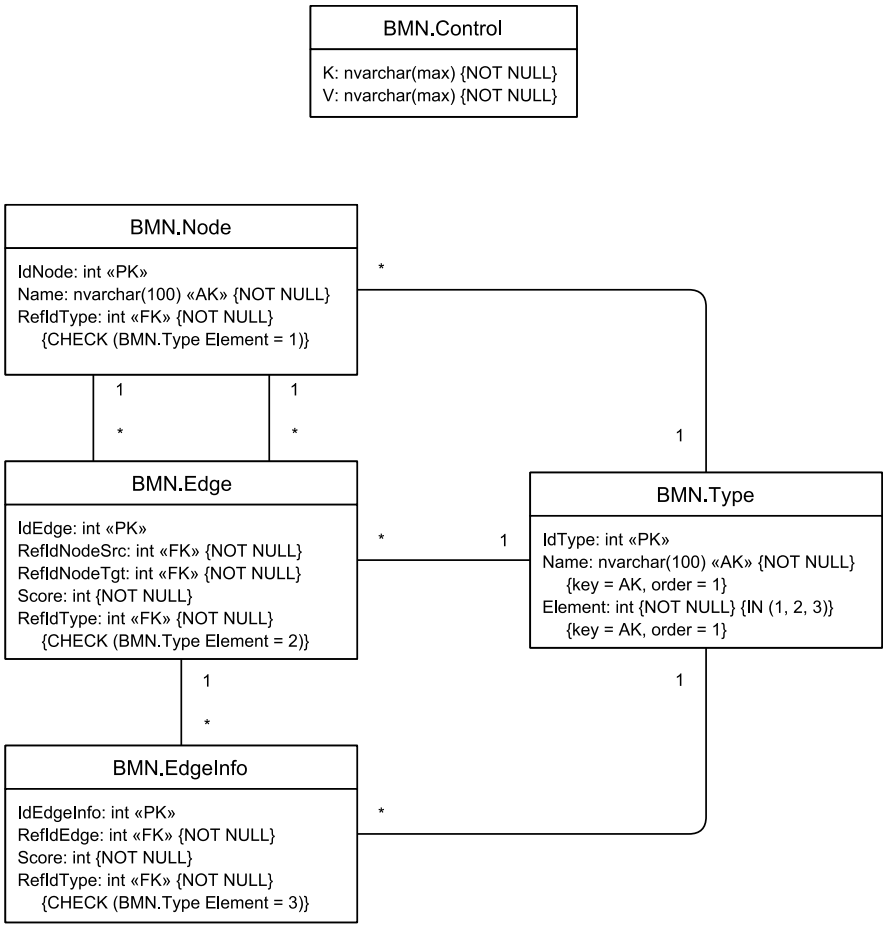


Figura 3.6 – Diagrama de Base de Dados da framework BMNetwork

Para além dos identificadores únicos de cada entidade, também foi necessário armazenar o nome completo ou o nome preferido para os vértices cujas entidades fossem comumente identificadas por tal nome. Esta representação mais amigável serve exatamente para facilitar a interpretação do utilizador. Uma vez que a *framework* BMNetwork devia ser completamente independente, a camada lógica da aplicação (componente BioMedNet) passou então a ser responsável pelo acesso aos nomes preferidos de cada vértice. De facto, excluindo os dados do conteúdo e anatomia do grafo que eram armazenados pela *framework* BMNetwork, os nomes preferidos dos vértices são a única informação restante que o sistema devia manipular (Figura 3.7).

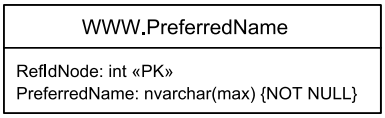


Figura 3.7 – Formato de armazenamento dos nomes dos elementos

3.5.3 Arquitetura da Solução

Considerando os três principais componentes anteriormente apresentados, foi concebido o diagrama de implementação (Figura 3.8) que servisse os requisitos do sistema. O modelo de implementação considera quatro principais elementos: o servidor da aplicação BioMedNet, o navegador Web cliente para aceder ao SaaS e os dois serviços externos usados para pesquisa.

Para aceder à aplicação Web disponibilizada pelo serviço, os utilizadores devem aceder com um navegador de Internet, às páginas Web fornecidas pelo servidor principal da aplicação BioMedNet.

O servidor BioMedNet é responsável por manter o núcleo da aplicação. O servidor detém os dados referentes ao grafo armazenado e controla o acesso a toda essa informação. Através de um servidor Web, o sistema disponibiliza um conjunto de páginas Web que permitem a interação com os utilizadores.

Os servidores da UniProt e da OMIM também fazem parte do modelo de implementação uma vez que o componente BioMedNet terá de requisitar dados para concluir determinadas ações.

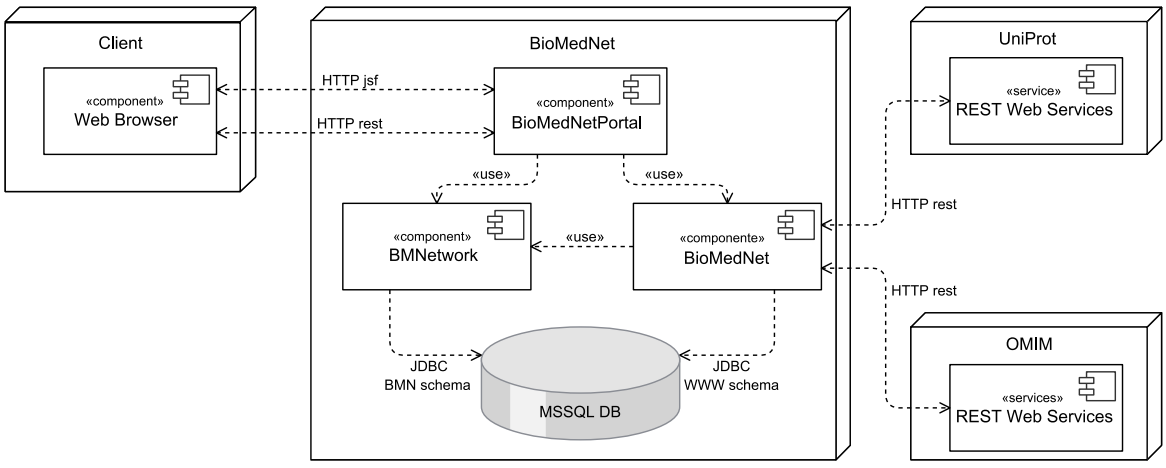


Figura 3.8 – Diagrama de Implementação

Capítulo 4
Implementação do Sistema BioMedNet

Este capítulo detalha toda a fase de desenvolvimento do sistema concebido. Serão relatados alguns dos problemas encontrados, nomeadamente a criação da *framework* de dados, os processos de filtragem da rede, a comunicação com sistemas externos, entre outros. Esta exposição faz-se acompanhar das soluções de engenharia adotadas que permitiram superar todos os obstáculos ao desenvolvimento.

Para a fase de implementação, optou por se recorrer à linguagem de programação Java, e a algumas das tecnologias associadas à mesma, para efeitos de desenvolvimento. Deste modo consegue-se garantir a independência da aplicação face a sistemas operativos ou qualquer infraestrutura na *Cloud*. Utilizando Java como ferramenta de desenvolvimento é possível criar aplicações multiplataforma, que podem ser executadas independentemente do sistema em que se encontram ser Windows ou baseado em Linux. Desta forma, qualquer gestor do sistema poderá facilmente configurar a aplicação, deixando-a pronta para distribuição, em qualquer ambiente de produção.

A fase de desenvolvimento do sistema dividiu-se em dois momentos distintos e sucessivos. O momento inicial consistiu no desenvolvimento de uma *framework* de acesso e manipulação a dados que se apresentassem sobre a forma de grafo. Essa *framework* deveria ser independente da aplicação e capaz de suportar diferentes tipos de dados em diferentes contextos. A segunda fase de desenvolvimento estava diretamente dependente da *framework* previamente concebida e resultou na criação do sistema BioMedNet. A acompanhar o sistema, foi criado um *script* de inicialização para preencher a base de dados com o grafo de uma rede de termos biomédicos.

A ferramenta de desenvolvimento NetBeans foi utilizada durante a realização de todo o projeto. No entanto, para simplificar a transição entre ambientes de desenvolvimento e facilitar o controlo de dependências, todos os projetos desenvolvidos utilizaram a ferramenta Apache Maven.

4.1 *Framework* de armazenamento e acesso a dados

Numa fase inicial, a atenção centrou-se no desenvolvimento do modelo de dados, uma vez que o desempenho do sistema final dependeria da eficiência da *framework* de acesso e manipulação de dados.

Baseado na análise de tecnologias anteriormente realizada, optou-se por desenvolver um modelo de dados utilizando duas ferramentas diferentes: a base de dados NoSQL Neo4j e a base de dados relacional SQL Server. Com os dois modelos desenvolvidos o objetivo foi perceber qual deles oferecia o melhor desempenho e permitia maior flexibilidade na realização de tarefas de filtragem, travessias ou aplicação de algoritmos de percursos mais curto, entre outros.

A escolha final acabou por incluir a combinação da base de dados SQL Server com a biblioteca JUNG, projeto ao qual se deu o nome de *BMNetwork*. Com esta solução tornou-se possível armazenar toda a informação da rede em base de dados e tirar partido das capacidades de análise e processamento da ferramenta JUNG. As secções seguintes apresentam os progressos realizados em ambos os modelos e os motivos que conduziram à decisão final adotada.

4.1.1 Modelo de Dados SQL Server

As tradicionais bases de dados relacionais armazenam toda a informação em registos de tabelas. Esta forma de arquivamento não foi projetada com o intuito de armazenar estruturas de grafos. O facto de a base de dados não ter sido criada com o fim específico de armazenar grafos não significa que seja impossível criar um modelo relacional capaz de representar essa estrutura.

Face à robustez e maturidade das bases de dados relacionais, decidiu-se investir na tecnologia e foi criado um modelo para armazenar os dados do sistema BioMedNet (Figura 3.6). Baseado nesse esquema criou-se um modelo de classes das entidades no domínio do problema (Figura 4.1). As classes ***Node***, ***Edge***, ***EdgeInfo*** e ***Type*** traduziam diretamente os elementos da base de dados, enquanto as classes ***Graph*** e ***SubGraph*** representavam coleções dos elementos anteriores. Os objetos da classe ***Graph*** contêm o grafo completo, ao passo que os objetos da classe ***SubGraph*** são subsecções

do grafo que resultaram de processos de filtragem com base em alguns nós ou arestas referência.

Para a criação de subgrafos foram desenvolvidas funções SQL que, a partir de vértices referência, iteravam pelos vários níveis de profundidade do grafo selecionando as arestas para compor o subgrafo. As funções sofreram sucessivas otimizações e acréscimos de critérios de seleção. A última versão resultou em quatro funções com utilização em cenários distintos:

- grafos não direcionados tendo como referência um único vértice (Anexo B),
- grafos não direcionados tendo como referência múltiplos vértices,
- grafos direcionados tendo como referência um único vértice,
- grafos direcionados tendo como referência múltiplos vértices (Anexo C).

Embora as funções conseguissem filtrar corretamente o grafo, o aumento do número de arestas eleitas para o subgrafo e a iteração para níveis de profundidade mais avançados aumentavam de forma exponencial o tempo de execução das funções. Um dos principais motivos que explica essa demora durante a execução é o facto do sucessivo aumento de resultados implicar a junção de tabelas com um número cada vez maior de registos. O custo da execução de *joins* de tabelas SQL têm um contributo decisivo no degradamento do desempenho das funções [51].

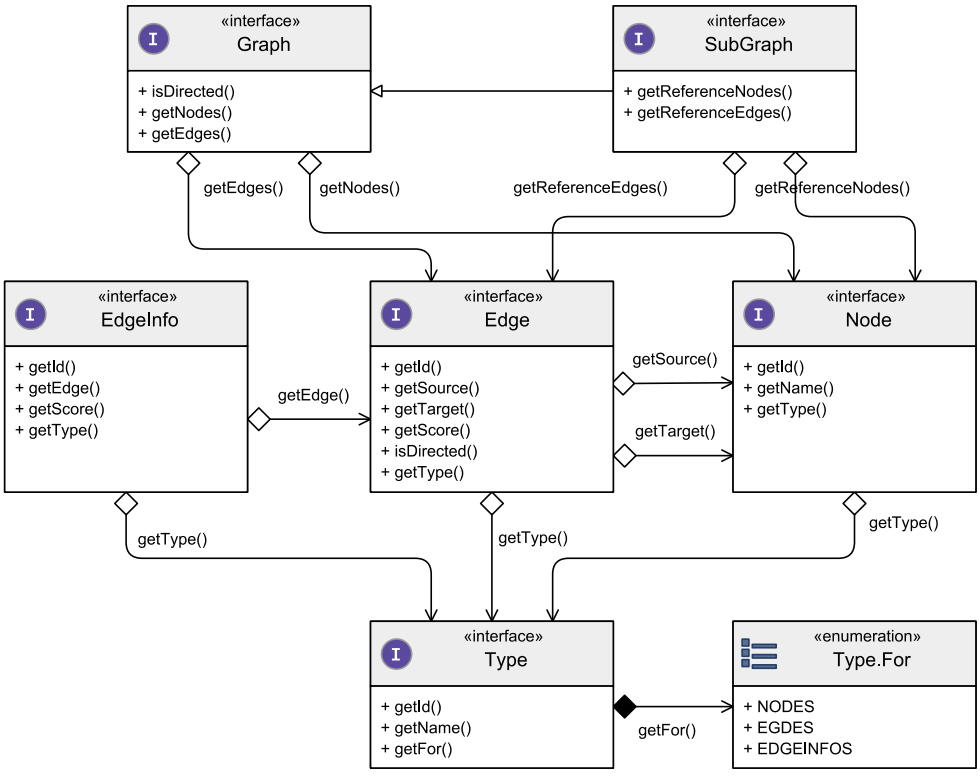


Figura 4.1 – Diagrama de Classes do Domínio do Problema da framework BMNetwork

Outro problema encontrado com esta solução residiu no facto desta não ser adequada à aplicação de algoritmos, por exemplo, de determinação de caminhos mais curtos ou de classificação de nós. Recriar esses algoritmos em linguagem SQL seria exigente e os resultados incertos, visto que também as funções já eram pouco eficientes para grafos mais extensos. Por outro lado, dependendo da dimensão do grafo e das características do equipamento onde a aplicação seria executada, poderia ser incomportável carregar toda a informação do grafo na estrutura de objetos, por limitações de memória.

4.1.2 Modelo de Dados Neo4j

A base de dados Neo4j apresenta características distintas das bases de dados relacionais. A base de dados Neo4j armazena os dados sob a forma de um grafo. São considerados três abstrações principais: os nós (os vértices), as relações entre nós (as arestas) e as propriedades que ambos podem apresentar.

A anatomia do grafo é representada pela estrutura que os nós e a relações entre eles apresentam. Cada elemento do grafo, seja ele nó ou relação, tem a capacidade de armazenar múltiplas propriedades que são identificadas através de uma chave. As propriedades são opcionais e a sua existência pode variar mesmo dentro de elementos do mesmo tipo. Para completar a estrutura dos dados armazenados, a base de dados Neo4j apresenta algumas ferramentas que facilitam a sua utilização. Entre outras funcionalidades, duas foram extensivamente utilizadas:

- A *framework* de travessias (*Traversal Framework Java API*), que permite um elevado controlo durante processos de visita a nós sucessivos de um grafo,
- A capacidade de indexamento disponibilizada pelo Apache Lucene, incluído por defeito no Neo4j.

A utilização da base de dados Neo4j em aplicações desenvolvidas em Java é bastante facilitada uma vez que é disponibilizada uma API orientada a objetos. Durante a utilização da base de dados são criadas estruturas de objeto que permitem aceder e manipular os dados de forma simples e intuitiva. De entre os principais elementos da API destacam-se as interfaces ***GraphDatabaseService***, ***Node*** e ***Relationship***, que representam respetivamente a instância da base de dados, os nós e as relações.

Seguindo o modelo esquemático de armazenamento de dados apresentado na documentação Neo4j (Figura 2.2), foi criado um esquema adaptado às características do problema deste projeto (Figura 4.2). Nesta fase de desenvolvimento ainda só era considerado um grafo contendo a rede de proteínas disponibilizada pela STRING.

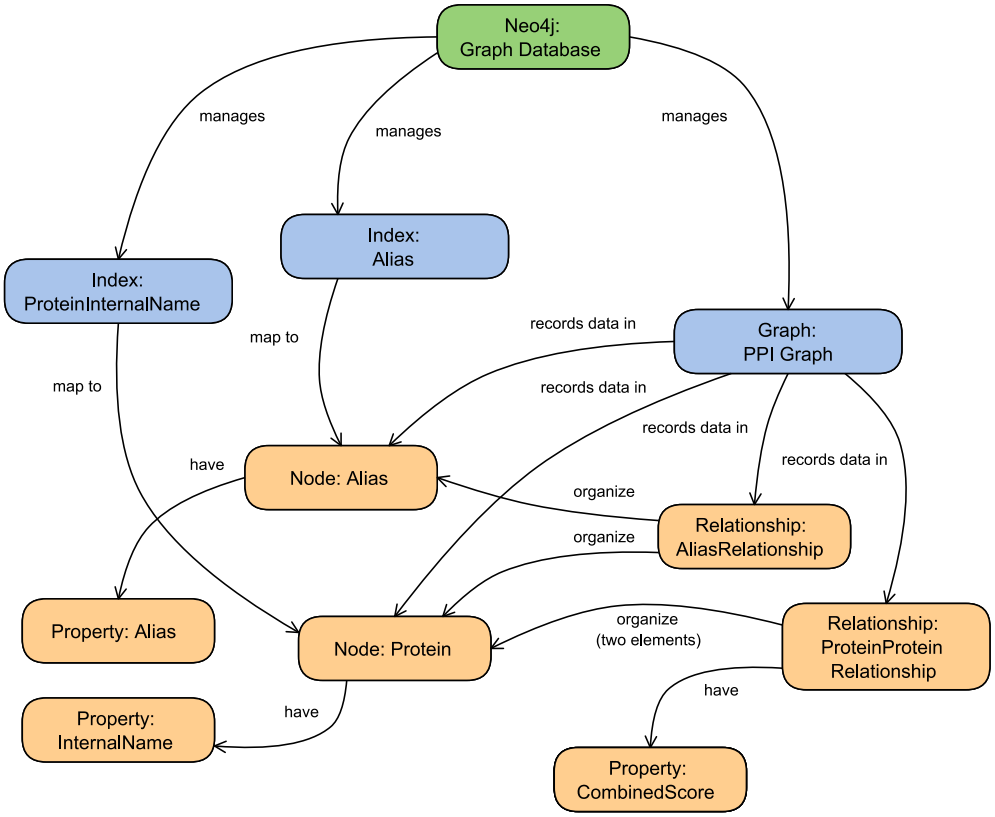


Figura 4.2 – Conceitos da base de dados Neo4j implementada segundo Figura 2.2

Um dos tipos de nós previsto pelo modelo mencionado foi o tipo ***Protein*** que representava as proteínas do grafo. Cada nó, desse tipo, devia possuir a propriedade ***InternalName*** que guardava o identificador da proteína em questão. As relações entre dois nós ***Protein*** eram do tipo ***ProteinProteinRelationship***. Estas relações teriam de ser caracterizadas por um número decimal associado à propriedade ***CombinedScore*** que representava o peso da associação. Além destes dois elementos essenciais foi considerado o tipo de nós ***Alias***, com a propriedade ***Alias***, para contemplar os nomes preferidos das diferentes proteínas. Cada nó ***Alias*** estava associado a um nó ***Protein*** através de uma relação ***AliasRelationship***. O grafo armazenado pela base de dados foi composto por nós de dois tipos que se associavam através de relações de dois tipos distintos.

Para garantir o acesso aos nós do grafo foram criados dois índices, ***ProteinInternalName*** e ***Alias*** que indexavam os nós do tipo ***Protein*** e ***Alias***, respetivamente. A indexação foi feita com base nas propriedades que caracterizavam cada nó.

Seguindo os exemplos explicativos da documentação Neo4j, foi criado um modelo do domínio (Figura 4.3) que representava os elementos do problema e interagia com os objetos gerados pela base de dados NoSQL. As classes abstratas ***NodeImplementation*** e ***RelationshipImplementation*** encapsulavam os nós e as relações da base de dados. Os objetos das classes ***AliasNode***, ***ProteinNode*** e ***ProteinProteinRelationship*** estendiam essas classes de encapsulamento e disponibilizavam interfaces minimalistas de acesso às propriedades e relações dos elementos. O ***Indexer*** era o componente responsável por interagir com a API de indexação da base de dados, para permitir o acesso a todos os nós do grafo. A última peça, a entidade ***PersistenceGraph***, exporta todos os serviços de interação com a base de dados.

Uma vez concluído o projeto para a base de dados, procedeu-se ao preenchimento da mesma com a rede de proteínas da STRING. Através da *framework* de travessias do Neo4j, implementaram-se alguns dos algoritmos de filtragem previstos aquando a análise de requisitos do sistema, nomeadamente a filtragem por profundidade máxima na rede e por confiança das arestas.

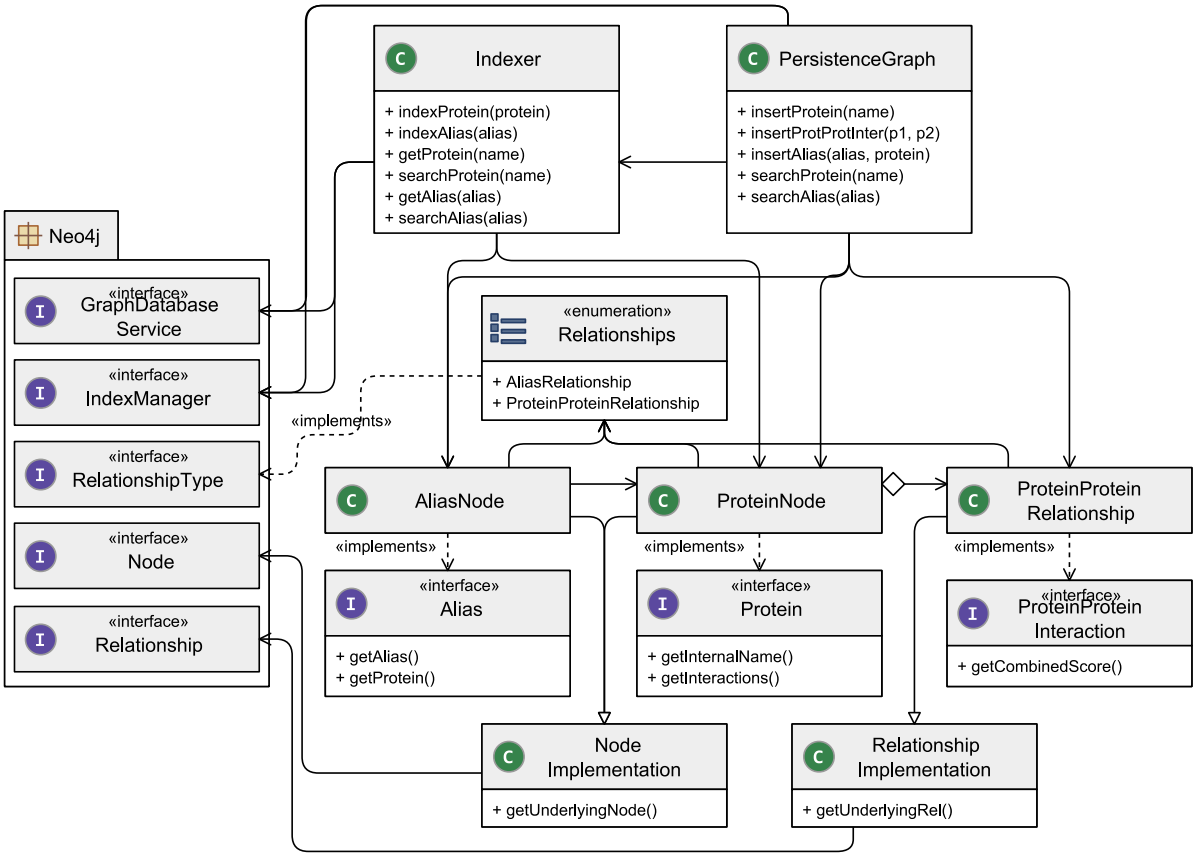


Figura 4.3 – Modelo do Domínio do Problema na Solução Neo4j

Os resultados dos processos de filtragem acabariam por não ser tão interessantes quanto se esperava. Embora de um modo global, quando comparados com as funções SQL previamente desenvolvidas, o tempo de execução da filtragem fosse ligeiramente inferior, com o aumento da profundidade da filtragem os resultados degradavam-se muito mais. Associado a este mau desempenho pode estar o facto de a verificação de determinadas condições da travessia terem de ser efetuadas sobre os objetos Neo4j. Uma vez que a verificação só acontece num nível de abstração tão alto, isso implica que a base de dados tenha que criar múltiplos objetos de dados antes que a filtragem possa efetivamente ser realizada.

4.1.3 Modelo de Dados SQL Server e JUNG

A solução para representação de dados que acabou por ser utilizada conciliou a base de dados SQL Server com a biblioteca de modelação de grafos JUNG. O objetivo desta solução combinada era tirar partido do melhor das duas ferramentas, juntando-as numa *framework* que permitisse não só as caraterísticas de armazenamento, acesso e edição de dados, mas também as funcionalidades de análise e processamento do grafo.

O desenvolvimento da ferramenta foi separado em duas camadas particulares: a camada de armazenamento e a camada de lógica. A camada de lógica está subjacente à camada de armazenamento, fazendo uso intensivo dos serviços que esse componente exporta. Ambas as camadas implementam interfaces bem definidas. Desta forma é possível criar diferentes implementações da camada de armazenamento, bastando que para isso respeitem a interface ***BMNPersistence*** (Figura 4.4).

A interface ***BMNPersistence*** reúne, em si, um conjunto de interfaces que permitem aceder e editar os vários elementos: os vértices, as arestas, os detalhes das arestas e os tipos desses três elementos. Uma interface de controlo foi também necessária, tanto para criar e eliminar a estrutura de dados, como para questionar a natureza do grafo. As possíveis implementações de armazenamento deveriam ainda disponibilizar uma listagem de todos os vértices e arestas do grafo, através de objetos ***SimpleNode*** e ***SimpleEdge***. Estas duas classes representam versões minimalistas dos vértices e das arestas do grafo e destinam-se a ser usadas pela camada de lógica para carregar somente a topologia da rede em memória, através de um grafo JUNG.

Aquando do desenvolvimento da ferramenta BMNetwork foi criada uma implementação da camada de armazenamento, baseada na base de dados SQL Server. Dada a ferramenta utilizada, a maioria dos serviços desta camada resulta na execução

de comandos SQL. O processo de criação constrói toda a estrutura de tabelas e as respetivas restrições e referências, enquanto a eliminação remove da base de dados qualquer indício dos dados e da estrutura utilizada. A maioria dos mecanismos de consulta e manipulação de dados origina a execução de simples comandos SQL. Porém, a listagem de todos os vértices e/ou arestas do grafo implica percorrer todas as entradas da tabela dos respetivos elementos. Dependendo da dimensão do grafo, esta ação pode ser computacionalmente muito exigente pois implicará a leitura de um grande volume de dados, a partir do disco rígido.

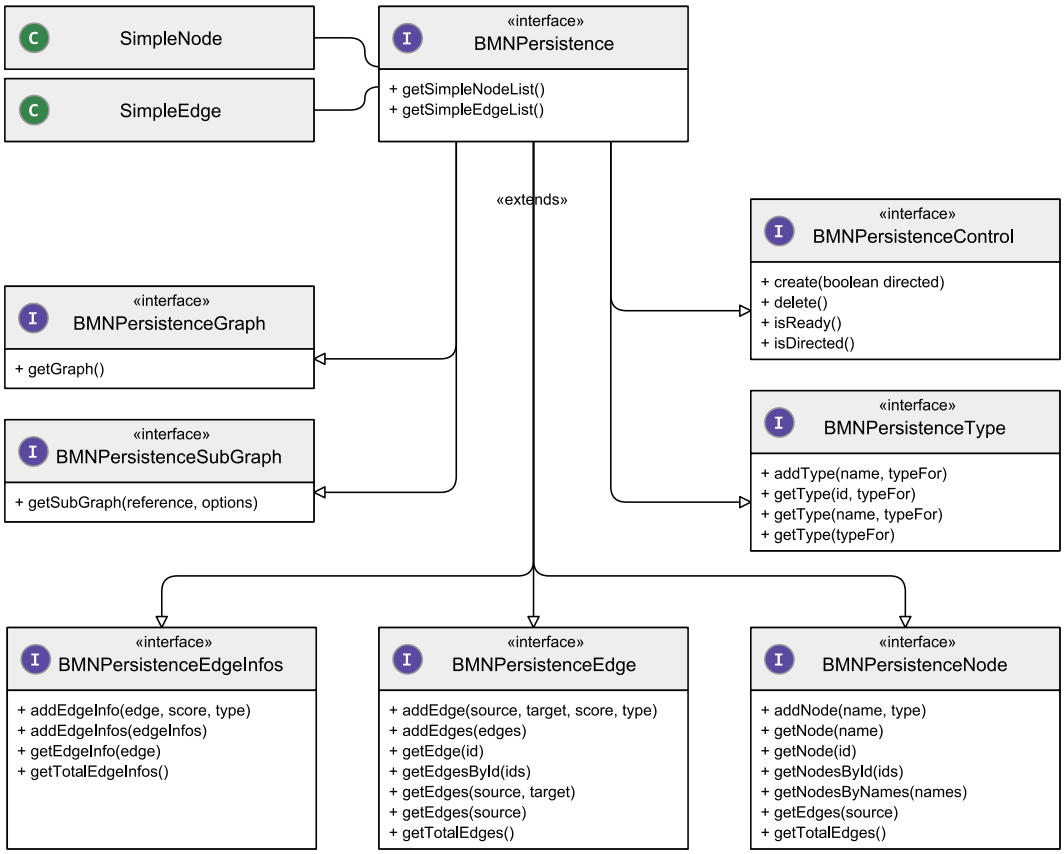


Figura 4.4 – Diagrama de Classes da Camada de Armazenamento da framework BMNetwork

A segunda camada da *framework*, a camada de lógica, foi implementada através da classe ***BMNModel***. Esta é a entidade que exporta todos os serviços desta ferramenta (Figura 4.5). Alguns dos métodos presentes nos objetos desta classe são diretamente reencaminhados para serviços da camada de persistência, o que não acontece nos serviços da interface ***BMNModelJung***. Para implementar as funcionalidades identificadas por essa interface foi necessário recorrer à biblioteca JUNG.

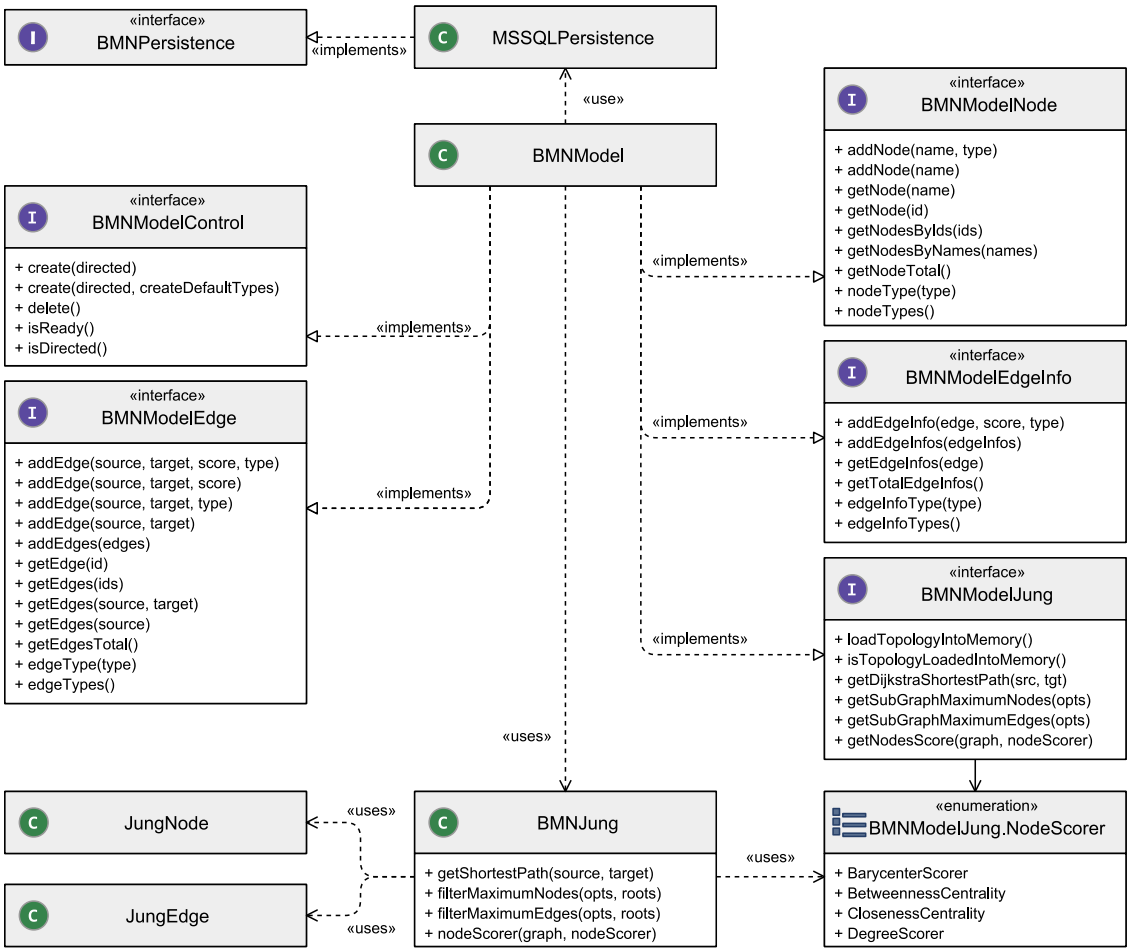


Figura 4.5 – Diagrama de Classes da Camada de Lógica da framework BMNetwork

Para ser possível efetuar processamento através da *framework* BMNetwork é indispensável carregar a topologia da rede num grafo JUNG. Quando num objeto ***BMNModel*** é invocado o método *loadTopologyIntoMemory*, dá-se início ao processo de carregamento da anatomia da rede para memória. É concebido um objeto ***BMNJung*** que será responsável por criar, preservar e aplicar ações sobre um grafo JUNG. Através dos métodos *getSimpleNodeList* e *getSimpleEdgeList* os identificadores (IDs) de todos os vértices e de todas as arestas serão importados, deste a camada de armazenamento, e será criado um grafo da biblioteca JUNG, contendo exclusivamente os IDs dos vértices e das arestas e os pesos das últimas.

Com a anatomia da rede carregada na biblioteca JUNG é possível tirar partido de muitas das capacidades de processamento que acompanham esta ferramenta. As funcionalidades de pesquisa do caminho mais curto entre dois elementos e a classificação dos elementos de um grafo são exclusivamente baseados nos algoritmos

que a biblioteca disponibiliza. No entanto, foram criados dois *plugin* que permitem fazer uma filtragem da rede baseada em diversas variáveis.

Aos *plugins* de filtragem desenvolvidos para a biblioteca JUNG foram dados os nomes **BMNFilterMaximumNodes** e **BMNFilterMaximumEdges**. A principal diferença entre estes dois processos está no primeiro critério de paragem da filtragem: enquanto o primeiro método limita a filtragem a um número máximo de vértices, o segundo termina a filtragem quando um determinado número de arestas é atingido. Em ambos os métodos, a filtragem do grafo é baseada num subconjunto de vértices e arestas referência. Esses vértices e arestas farão obrigatoriamente parte do subgrafo, resultante deste processo.

Para além do número limite de vértices ou arestas que constituirão o subgrafo filtrado, existem ainda outros parâmetros que limitam e influenciam o resultado da filtragem. Esses parâmetros são:

- Profundidade máxima – À medida que o processo de filtragem se afasta dos vértices iniciais (os vértices referência e os vértices das arestas referência) são definidos sucessivos níveis de profundidade. Este parâmetro define o nível de profundidade máxima até onde a filtragem pode chegar.
- Peso mínimo das arestas – O processo de filtragem apenas considerará arestas, cujo peso seja igual ou superior a este limite. O subgrafo resultante será exclusivamente constituído por arestas de peso superior a este limite mínimo.
- Número máximo de elementos por nível de profundidade – Este parâmetro define o número máximo de arestas que serão eleitas a cada nível de profundidade. Limitado por este valor, o processo de filtragem seleciona sempre as arestas com pesos superiores.

Esta estratégia provou-se capaz de resolver as necessidades de armazenamento e processamento de dados do sistema BioMedNet. O servidor, desde onde a aplicação é distribuída, possui capacidade de armazenamento e memória suficientes para permitir esta solução. Desde que o computador tenha capacidade de armazenamento suficiente para guardar todos os dados da rede será sempre possível consultar qualquer informação arquivada. No entanto, pode acontecer que em outras máquinas não exista espaço em memória suficiente para alocar a anatomia do grafo. Nestas circunstâncias os métodos de processamento estarão inevitavelmente comprometidos, uma vez que é impossível carregar o grafo para a biblioteca de processamento JUNG.

4.2 Portal BioMedNet

A aplicação BioMedNet depende da ferramenta apresentada na secção anterior para armazenar, num grafo, uma rede semântica de termos biomédicos. Todos os termos biomédicos presentes nessa rede de conhecimento serão representados como vértices do grafo. Para além do projeto *BMNetwork*, o sistema depende de outros projetos (Figura 4.6).

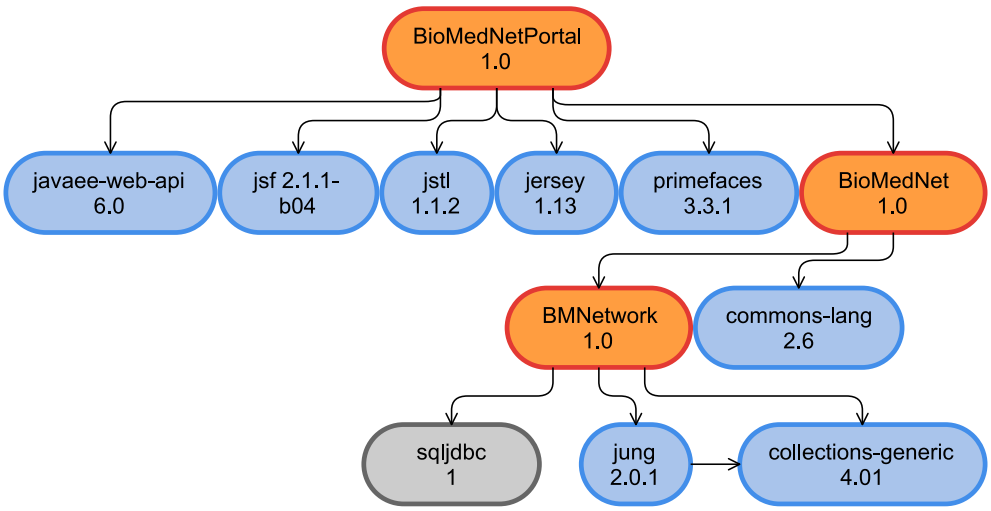


Figura 4.6 – Diagrama de dependências Maven

Projetos desenvolvidos
Dependências em repositórios públicos
Dependências em repositório local

A biblioteca JUNG e o *driver* JDBC para a base de dados Microsoft SQL Server são dependências da ferramenta BMNetwork. Já o sistema BioMedNet foi dividido em dois projetos Maven: BioMedNetPortal e BioMedNet. Enquanto o projeto BioMedNet tem função de camada lógica e *script* de inicialização, o projeto BioMedNetPortal faz a gestão da interface de utilização, tanto de páginas como de serviços Web. Do projeto BioMedNetPortal podem ser destacadas algumas das dependências essenciais:

- *Java Enterprice Web API* – plataforma Java para desenvolvimento de aplicações Web;
- *JavaServer Faces* (JSF) – *framework* Java para desenvolvimento de aplicações Web segundo um modelo MVC;
- *PrimeFaces* – biblioteca de componentes com tecnologia AJAX para JSF que simplifica a criação de páginas de maior qualidade;
- *Jersey* – implementação JAX-RS (API Java para Web Services RESTful) para a criação de Web Services RESTful.

4.2.1 Pré-Processamento para Recolha de Dados

Os dados manipulados pelo sistema têm de ser recolhidos e armazenados antes de se configurar a aplicação para distribuição. Todos os dados de natureza biomédica utilizados pelo sistema podem ser livremente acedidos a partir dos sistemas de várias instituições.

Para recolher todas a informação a utilizar foi desenvolvido um procedimento de inicialização que acompanha o projeto BioMedNet. Este método consome dados obtidos desde *flat files* (Tabela 4.1) e Web Services (Tabela 4.2) e armazena-os localmente através da *framework* BMNetwork (Figura 4.7). Para desencadear a inicialização deve ser diretamente executado o *.jar* resultado da compilação do projeto. Para que este método seja realizado com sucesso é necessário que anteriormente sejam descarregados os ficheiros que serão consumidos e que durante todo o processo seja garantida uma conexão à Internet. Cumpridos esses requisitos, é também necessário referir alguma informação extraordinária aquando a execução do método:

- Informação de base de dados – endereço e porto do servidor, nome da base de dados a utilizar, nome e *password* de um utilizador com permissões de proprietário para a base de dados utilizada;
- Localização de ficheiros a consumir - *protein.links.detailed.v9.0.txt*, *morbidmap*, *gene_association.goa_human*, *gene_ontology_ext.obo*;
- Chave de utilização do WS OMIM.

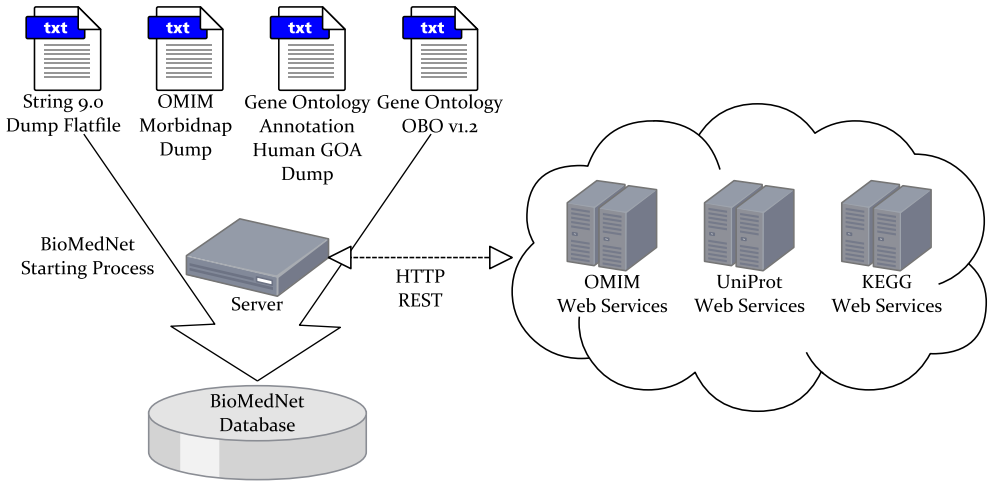


Figura 4.7 – Diagrama esquemático do pré-processamento inicial

Tabela 4.1 – Ficheiros Utilizados para Pré-Processamento

Ficheiro	Descrição
protein.links.detailed.v9.0.txt [13]	Este ficheiro, disponibilizado pela STRING, detém uma rede de proteínas que se apresentam no formato Ensembl Protein. Em cada linha pode-se identificar uma relação em duas proteínas, o valor global da confiança dessa relação e vários valores de confiança, resultado de diferentes métodos de avaliação. Desta extensa lista, apenas serão consideradas as relações que envolvam proteínas reconhecidamente associadas ao ser humano.
morbidmap [15]	Com este ficheiro, a OMIM facilita o acesso a uma lista de conhecidas associações entre fenótipos e genótipos humanos. Esses fenótipos manifestam-se através de determinadas doenças ou distúrbios, enquanto os genótipos estão diretamente associados a proteínas. Tanto os fenótipos como os genótipos estão sob o formato de identificadores MIM.
gene_association.goa_human [16]	Desde os dados recolhidos dos ficheiros da base de dados Gene Ontology Annotation é possível obter uma listagem de relações entre proteínas e ontologias. Cada proteína pode estar relacionada com múltiplas ontologias. É ainda possível obter, de entre muitas outras informações, o tipo de ontologia em questão (processos biológicos, componentes celulares ou funções moleculares). A representação das proteínas é em formato UniProt, ao passo que as ontologias são exibidas no formato GO.
gene_ontology_ext.obo [17]	Este ficheiro que a Gene Ontology disponibiliza permite a consulta de diferentes tipos de informação. Esta é a fonte que, uma vez recolhidas a associações entre ontologias e proteínas deste ficheiro *gene_association.goa_human*, permite obter, através do identificador GO, o nome completo das ontologias.

Tabela 4.2 – Web Services Utilizados para Pré-Processamento

Web Services	Descrição
UniProt WS [19]	Embora este WS não seja utilizado para obter associações entre elementos, ele tem um papel fundamental em todo o processo. Através do serviço de mapeamento é possível traduzir formatos de diferentes bases de dados, de e para a representação UniProt. Esta funcionalidade será utilizada para traduzir para formato UniProt as proteínas em Ensembl Protein da STRING e os genes MIM da OMIM.
OMIM WS [15]	Através do WS OMIM é possível obter informação detalhada sobre qualquer elemento MIM. Depois de recolhidas as relações entre doenças e genes, através do ficheiro *morbidmap*, o uso da API deste serviço permite, através de indentificadores MIM, obter os nomes das várias doenças.
KEGG WS [18]	Foram utilizados três dos muitos serviços que a KEGG oferece: • Listagem de vias metabólicas humanas e respetivos nomes; • Associações de genes humanos e vias metabólicas; • Conversão de genes humanos de representação KEGG para UniProt.

Foram recolhidos dados de quatro naturezas biomédicas diferentes. Cada um destes géneros representa um tipo distinto de vértices no grafo. No entanto, diferentes bases de dados podem utilizar diferentes identificadores para as mesmas entidades (conforme se verifica na Tabela 4.1).

Para manter a uniformidade de representação dos termos biomédicos foram utilizadas as funcionalidades de conversão dos WS UniProt e KEGG (Tabela 4.2). Para reconhecer a unicidade dos elementos de cada tipo foram utilizados quatro identificadores diferentes:

- UniProt – para proteína ou genes;
- MIM – para doenças ou distúrbios;
- GO – para ontologias (processos biológicos, componentes celulares ou funções moleculares);
- KEGG – para vias metabólicas.

Face aos elementos e associações obtidos foram identificadas quatro tipos de interações (arestas):

- Proteína – Proteína
- Doença – Proteína
- Ontologia – Proteína
- Via Metabólica – Proteína

De entre estes quatro tipos, apenas as associações entre duas proteínas eram caracterizadas por um valor de confiança variável entre 0 e 1, com os extremos exclusivos:]0,1[. Para além do valor de confiança global, relativamente a esse tipo de associações, eram ainda obtidos valores de confiança, resultado de valores experimentais, métodos de dedução computacionais, etc. Às associações dos três outros tipos foi atribuída a unidade como valor de confiança, visto estas representarem relações conhecidas e bem identificadas entre elementos.

Este pré-processamento é uma tarefa que pode levar bastante tempo a ser executada. Conforme as características do computador, da dimensão dos ficheiros e do congestionamento dos Web Services, o tempo de execução do método pode variar. No servidor onde o sistema foi implementado o tempo de pré-processamento variou entre 40 e 50 minutos.

4.2.2 Estrutura da Aplicação

A interface web da aplicação foi desenvolvida com recurso à tecnologia JavaServer Faces. Esta ferramenta é uma *framework* MVC especialmente concebida para a criação de interfaces Web.

Tirando partido de um modelo MVC é possível desenvolver a lógica e a interface da aplicação de forma completamente independente. Através da Figura 4.8 é possível compreender quais os recursos que constituem uma aplicação, segundo esse modelo:

- *View* (visão) – As páginas JSP que contêm componentes JSF são reconhecidas como a visão no modelo MVC.
- *Controller* (controlador) – O *Faces Serlevet* que acompanha a implementação do JSF é responsável por gerir todos os pedidos de páginas JSF. Esta entidade carrega as *views* apropriadas e interage com o *model* para processar cada resposta.
- *Model* (modelo) – As *ManagedBeans* correspondem ao modelo da aplicação. Estas entidades são *Javabean* cujo estado é gerido pela *framework* JSF.

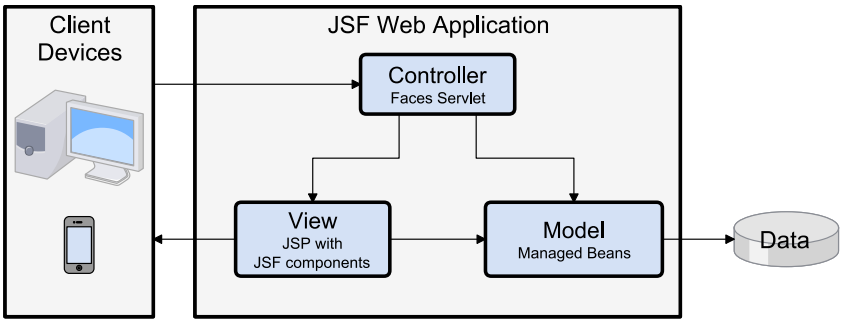


Figura 4.8 – Aplicação Web segundo o modelo MVC (adaptado de [52]*)*

Um dos componentes essenciais do sistema encontra-se na camada de serviços. Uma vez identificados os principais serviços do sistema, foi implementada uma API REST para permitir o acesso externo ao sistema (Figura 4.9). Os dados disponibilizados através desta interface são diretamente consumidos pelas páginas JSF da interface Web, mas também estão livremente acessíveis para serem consultados por quaisquer utilizadores ou aplicações externas.

O método ***export*** permite exportar, em diferentes formatos, a informação exibida no visualizador de rede. Este método é muito específico à interface Web, mais propriamente ao visualizador Cytoscape Web. Este método deve ser invocado através de um pedido HTTP POST, em que o conteúdo da mensagem corresponde à rede apresentada no visualizador.

À exceção do serviço anterior, todos os métodos implementados pela interface resultam em consulta de dados e correspondem a pedidos HTTP GET que produzem documentos XML com a informação solicitada. Estes métodos podem ser acedidos através dos respetivos caminhos:

- ***graph/node*** – Usado para obter informação relativamente a um elemento. Para se consultar esta informação é necessário conhecer o identificador ou o nome do elemento.
- ***graph/edge*** – Exporta os detalhes de determinadas associações da rede. Este serviço requer, como parâmetro de entrada, o identificador da associação que se pretende consultar. Adicionalmente pode solicitar-se os detalhes dessa associação.
- ***graph/filter*** – Permite filtrar a rede através de um completo conjunto de parâmetros. Das múltiplas variáveis de filtragem apenas três são obrigatórias: lista de *elementos base*, modo de delimitação (baseado no número total de elementos ou associações) e o número limite das entidades

definidas pelo modo de delimitação. O resultado deste método corresponde a um documento GraphML contendo a informação da sub-rede filtrada.

- ***graph/shortest_path*** – Serviço utilizado para determinar as associações no caminho mais curto entre dois elementos. Para ser efetuado, este método requer que sejam identificadas a origem e o destino do caminho.

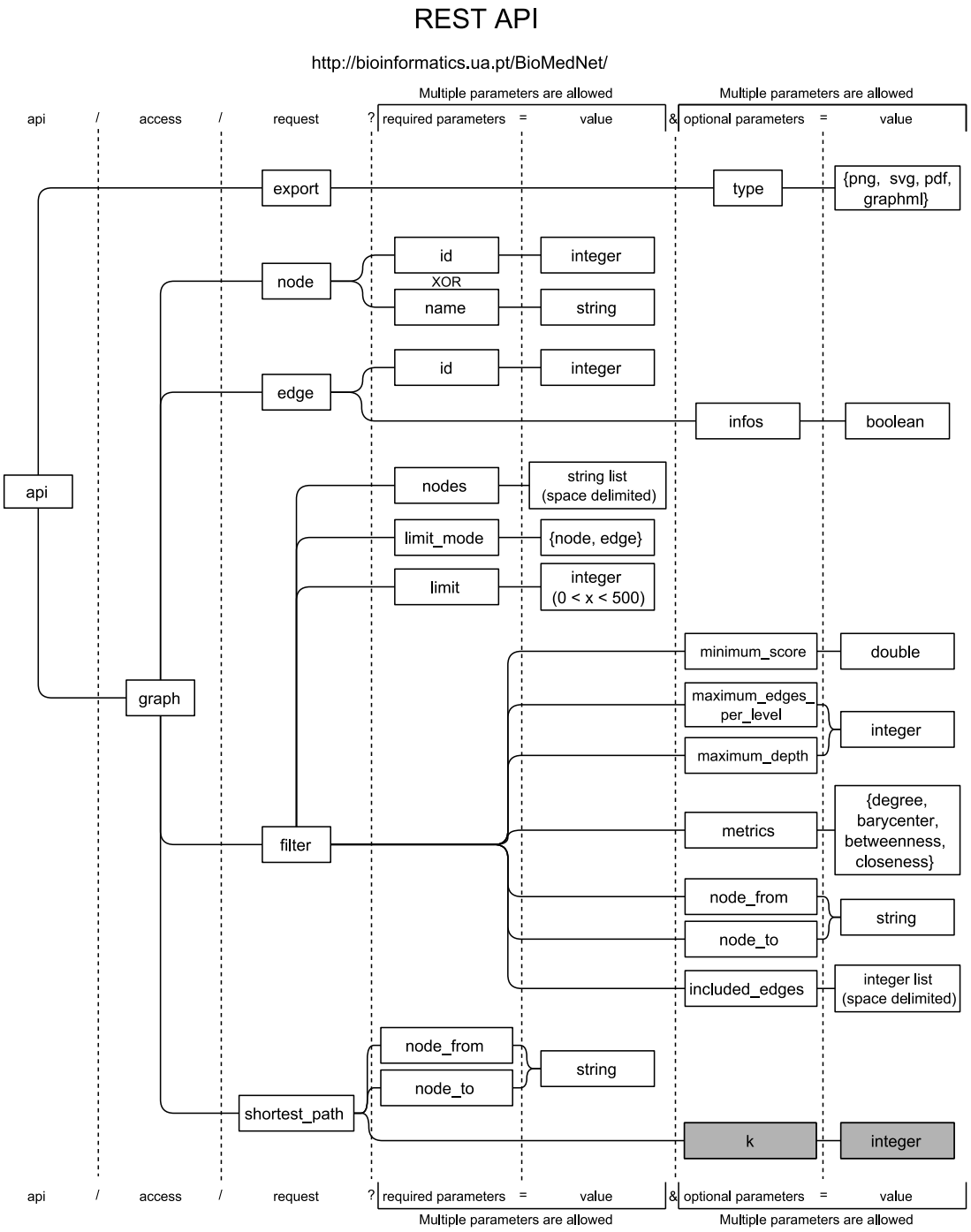


Figura 4.9 – Diagrama explicativo da interface da API REST BioMedNet

■ *Por implementar*

Se ao utilizar da interface Web, através das páginas JSF, o utilizador desejar estudar um determinado elemento, deverá começar por fazer uma pesquisa por esse termo biomédico. Através dos resultados dessa pesquisa, é possível selecionar os primeiros elementos que filtrarão a rede. Depois desse momento inicial, existem outros métodos de eleição de elementos referência. No entanto, o utilizador poderá sempre utilizar a pesquisa como forma de seleção de novos elementos.

O sistema desenvolvido permite realizar pesquisas através de dois serviços externos: o WS UniProt e o WS OMIM. A partir destes é possível pesquisar proteínas e doenças, respetivamente. A Figura 4.10 representa, de forma esquemática, o processo de pesquisa implementado pelo sistema. Ao efetuar uma pesquisa, esta é encaminhada para um serviço de pesquisa (*Object: Search*), responsável por concretizar todo o processo. Essa entidade vai efetuar, conforme o elemento que se pretende pesquisar, um pedido HTTP a um dos Web Services externo. Ambas as fontes retornam uma lista ordenada de resultados que é, de seguida, confrontada com os elementos presentes no modelo de armazenamento da aplicação. A *framework* BMNetwork devolve todos os vértices presentes no grafo cujo indentificador coincide com os resultados recolhidos. Em seguida os vértices são ordenados conforme a sequência pela qual foram enviados desde a respetiva fonte. Finalmente é apresentada a lista de elementos ao utilizador.

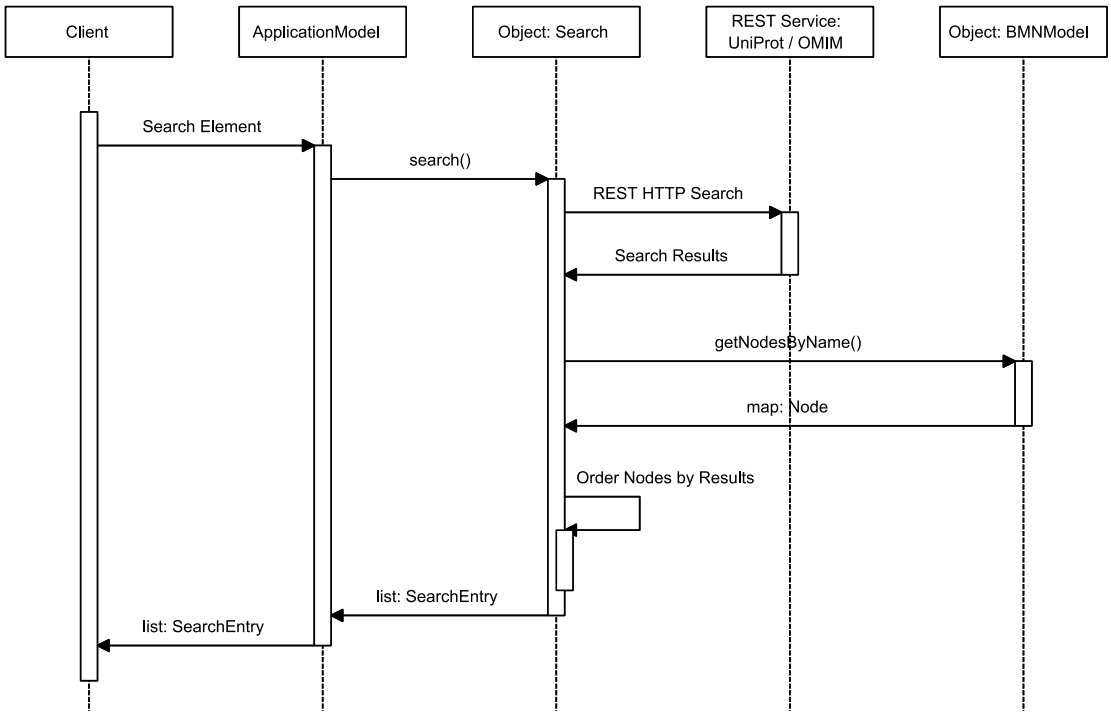


Figura 4.10 – Sequência de pesquisas de elementos

Depois do processo de seleção dos elementos iniciais, o utilizador é redirecionado para a página principal da aplicação (página *app.jsf*). Esta é a página que concentra todas as funcionalidades do sistema. O ambiente de utilização é composto por dois elementos essenciais: um visualizador de rede e um menu de controlos.

Por forma a tornar o modo de utilização da aplicação Web semelhante às tradicionais aplicações *desktop*, os controlos do menu foram configurados para produzir, mediante ação do utilizador, alterações imediatas no visualizador de redes. Este modo de funcionamento facilita muito a experiência de exploração, uma vez que o utilizador pode acompanhar de forma quase instantânea (requer um breve processo de carregamento) o resultado das alterações que aplica. O esquema da Figura 4.11 ilustra todo este processo. No primeiro acesso à aplicação, o navegador Web do utilizador irá pedir e receber a página *app.jsf*. Assim que esta seja devidamente carregada no cliente, será processado, através de *cliente-side scripting*, um novo pedido, solicitando a informação a exibir no visualizador. À medida que o utilizador vai efetuando novas alterações nos controlos da aplicação repete-se uma outra sequência de pedidos ao servidor. O primeiro desses pedidos tem o propósito de atualizar variáveis de página no servidor e proceder às respetivas alterações na interface. O segundo pedido volta a atualizar a rede do visualizador. Em todos os momentos que os dados do visualizador são alterados, os pedidos HTTP que provocam essa atualização são diretamente enviados à API REST do sistema.

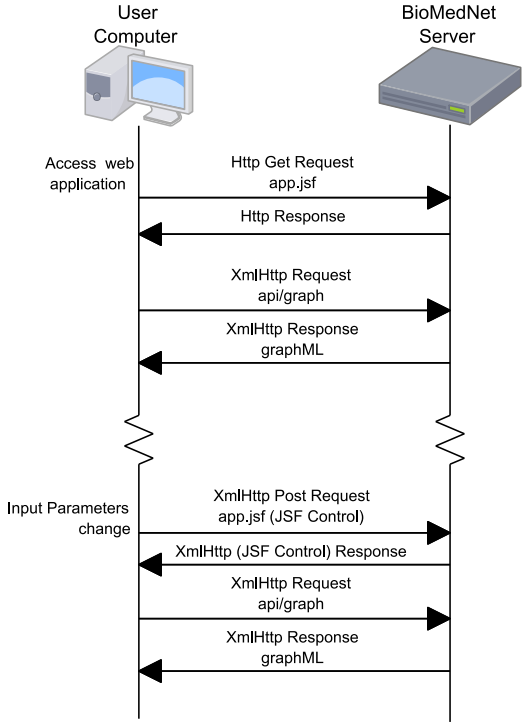


Figura 4.11 – Interação de utilizadores

4.3 Interface da Aplicação

Como contacto inicial com o sistema o utilizador necessita de efetuar uma pesquisa de um termo biomédico, podendo ser uma doença ou uma proteína (Figura 4.12). Desencadeado o processo de pesquisa o sistema irá redirecionar esse pedido para serviços externos. Usando o termo introduzido pelo utilizador será feita uma pesquisa por doenças, no serviço OMIM, e por genes, no serviço UniProt. Uma vez terminada a pesquisa, será exibida uma lista de todos os resultados obtidos, apresentando inicialmente as doenças seguidas das proteínas (Figura 4.13). Esta primeira pesquisa serve essencialmente para dar início à filtragem da rede. Visto não ser possível carregar todos os elementos e associações no visualizador de rede, é necessário que sejam identificadas referências no processo de filtragem. Aos elementos que regulam a filtragem foi atribuído o nome de *elementos base* (*root elements*). Da lista resultante da pesquisa inicial, o utilizador é solicitado a selecionar aqueles que pretenda que sejam considerados *elementos base*. Outros elementos poderão ser adicionados ao conjunto de *elementos base*, posteriormente.

Terminada a seleção de *elementos base* iniciais, o utilizador deve prosseguir para a página principal da aplicação (Figura 4.14). Baseado na recém seleção feita, é exibida através do visualizador, uma sub-rede filtrada por algumas propriedades definidas por padrão. Num painel, sobre o lado esquerdo, é apresentado um vasto conjunto de controlos que permitem a interação com a rede exibida.

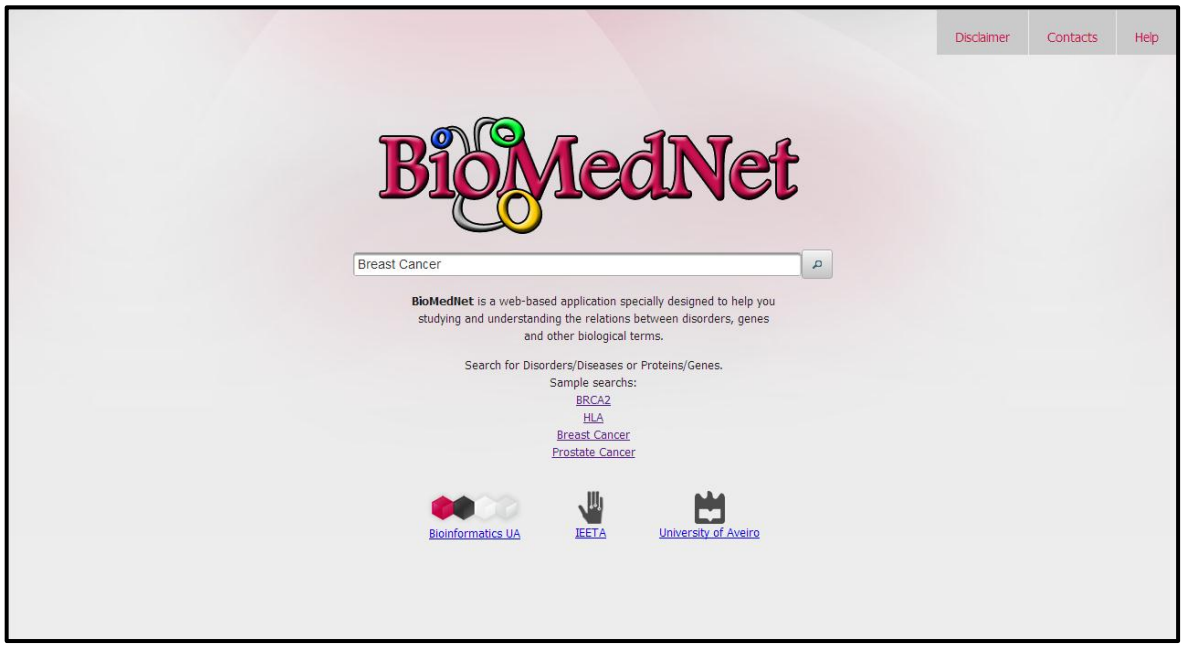


Figura 4.12 – Página inicial do sistema BioMedNet

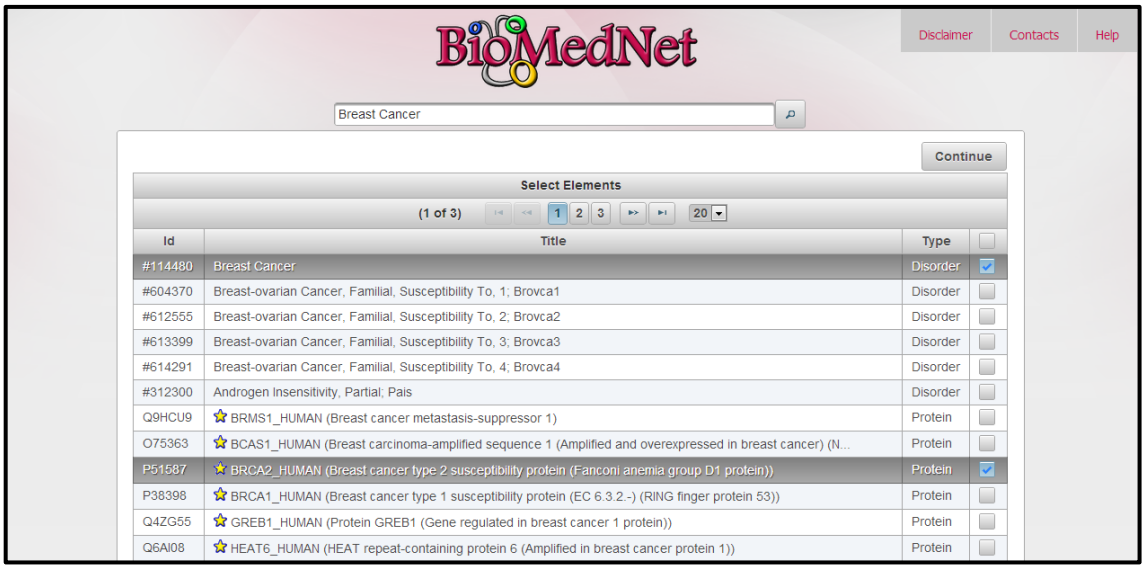


Figura 4.13 – Pesquisa inicial por doenças e genes

No visualizador da rede podem encontrar-se os elementos e as associações da rede. Todos os elementos são representados através de figuras circulares. A cor, o texto ou a imagem no interior de cada círculo, determina o tipo de elemento representado. Sob cada círculo é possível encontrar, como legenda, o identificador do elemento em questão. Para facilitar a identificação dos *elementos base,* estes apresentam uma margem mais larga e de cor diferente. O visualizador permite que o utilizador movimente a sub-rede na tela, em forma de navegação. É também possível ampliar e diminuir a representação dos elementos.

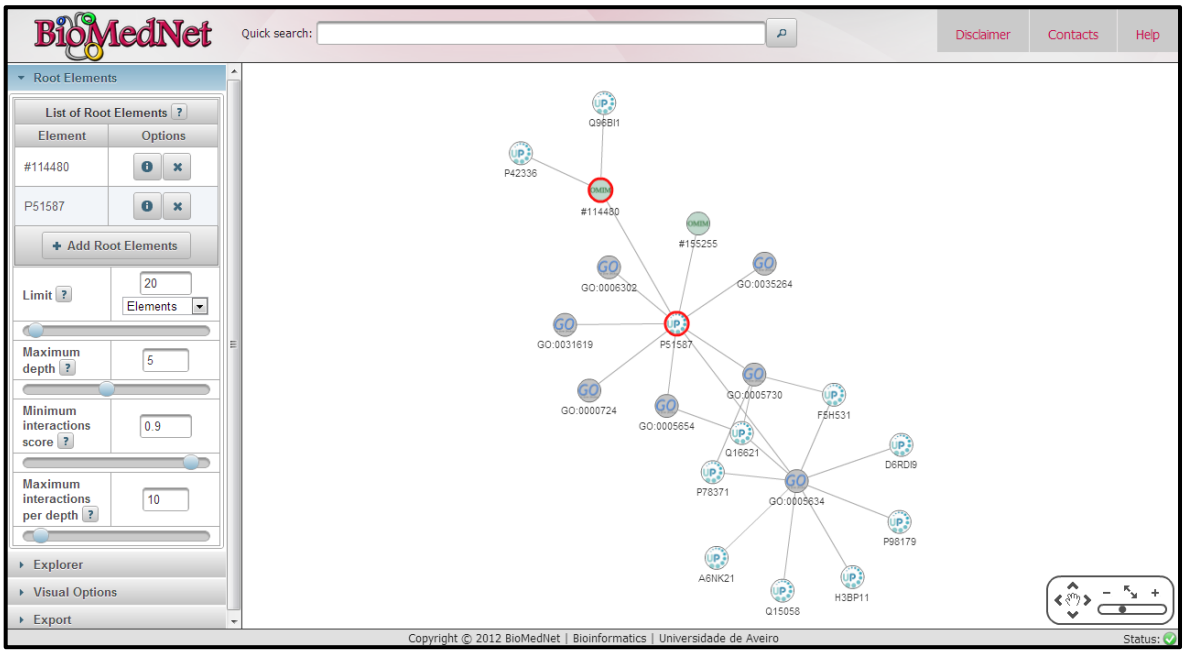


Figura 4.14 – Visualizador de rede e menu de controlos

No caso de o utilizador desejar obter mais informação sobre algum elemento ou associação exibida no visualizador pode fazer duplo clique sobre essa entidade. Uma pequena caixa de diálogo será apresentada, revelando informação mais detalhada sobre cada entidade ou possíveis ações aplicáveis sobre esse elemento. Relativamente aos elementos vértices é possível obter-se o identificador, o nome preferido, o tipo e a fonte. O utilizador pode ainda consultar a página da entidade na respetiva fonte, requerer uma listagem das associações da qual esse elemento faz parte e adicioná-lo aos *elementos base* (caso este ainda não faça parte do conjunto). No caso de se tratar de uma associação, o painel informativo apresentará os identificadores dos dois elementos nos extremos, o tipo da associação, o grau de confiança e uma lista com todos os valores de confiança intermédios.

A principal forma de interação com a aplicação é através dos controlos no menu lateral. Esse menu encontra-se dividido em quatro secções com áreas de intervenção bem definidas: Elementos Base (*Root Elements*), Explorador (*Explorer*), Opções Visuais (*Visual Options*) e Exportação (*Export*).

Na primeira secção é apresentada uma lista completa de *elementos base*. Desse conjunto de elementos é possível excluir ou adicionar elementos, sendo que esta segunda opção requer a realização de um nova pesquisa.

Todas as variáveis de filtragem da rede também se encontram nesta secção, uma vez que o resultado da filtragem depende diretamente dos elementos que determinam o início da mesma. Existem quatro variáveis de filtragem:

- Limite global de elementos ou associações: valor inteiro entre [1,300];
- Limite máximo de profundidade: valor inteiro entre [1,10];
- Limite mínimo de confiança que as associações devem apresentar: valor decimal entre [0,1];
- Limite máximo de associações a cada nível de profundidade: valor inteiro entre [1,99].

Dependendo dos *elementos base* que determinam a filtragem e alterando o valor das variáveis é possível mudar os resultados entre sub-redes muito pequenas, mas bastante específicas aos *elementos base* (Figura 4.16), ou sub-redes mais extensas, que permitem uma visão mais global da rede (Figura 4.17).

A secção de exploração possibilita a compreensão de possíveis relacionamentos entre dois elementos. De entre a lista de *elementos base*, o utilizador pode escolher dois elementos distintos para tentar compreender qual é o caminho mais curto que

une essas entidades. No visualizador, as associações que fazem parte desse caminho serão representadas com uma espessura maior e cor diferente (Figura 4.18).

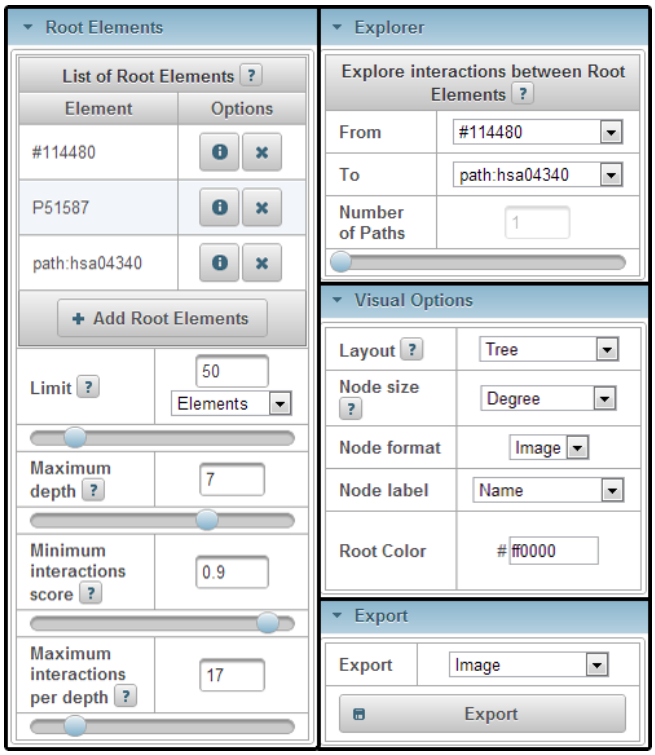


Figura 4.15 – Painel de controlos do sistema (expandido)

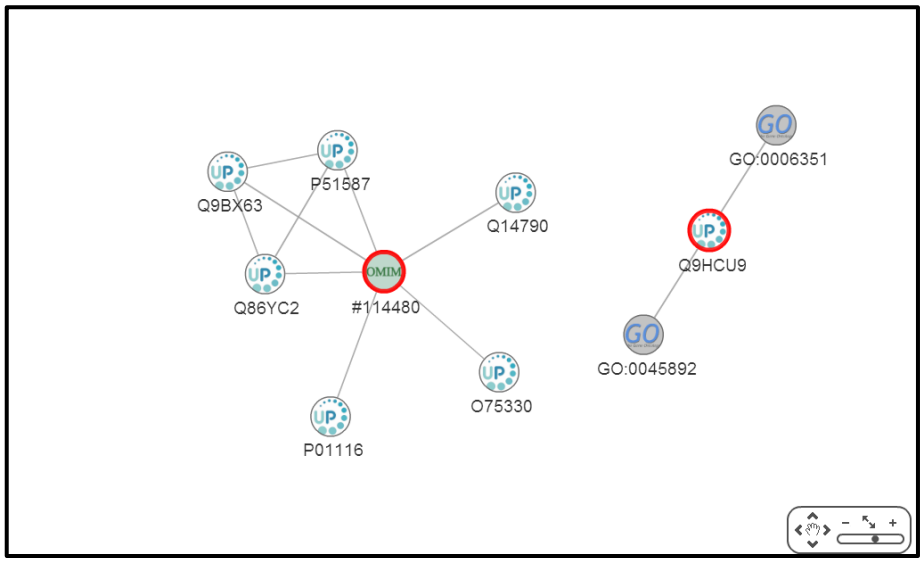


Figura 4.16 – Sub-rede com reduzido número de elementos e/ou associações

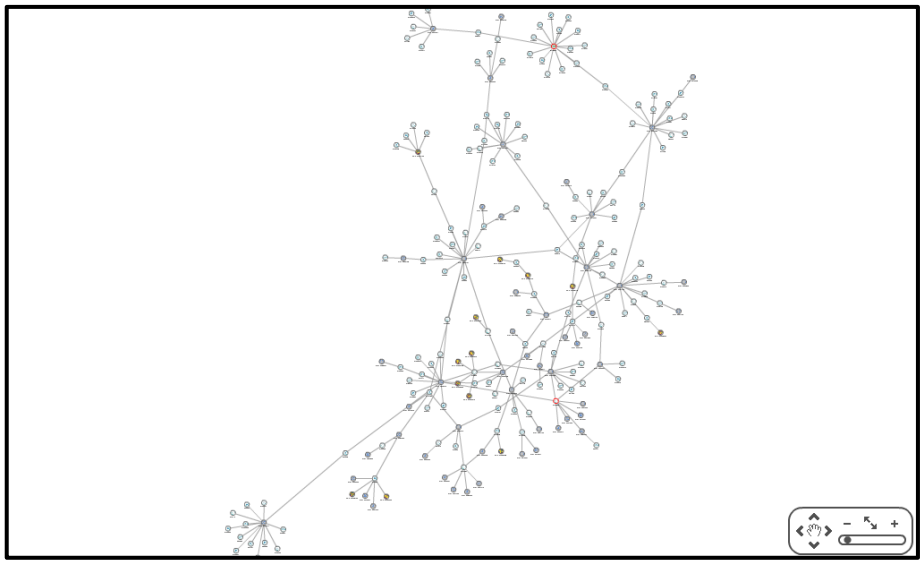


Figura 4.17 – Sub-rede com elevado número de elementos e/ou associações

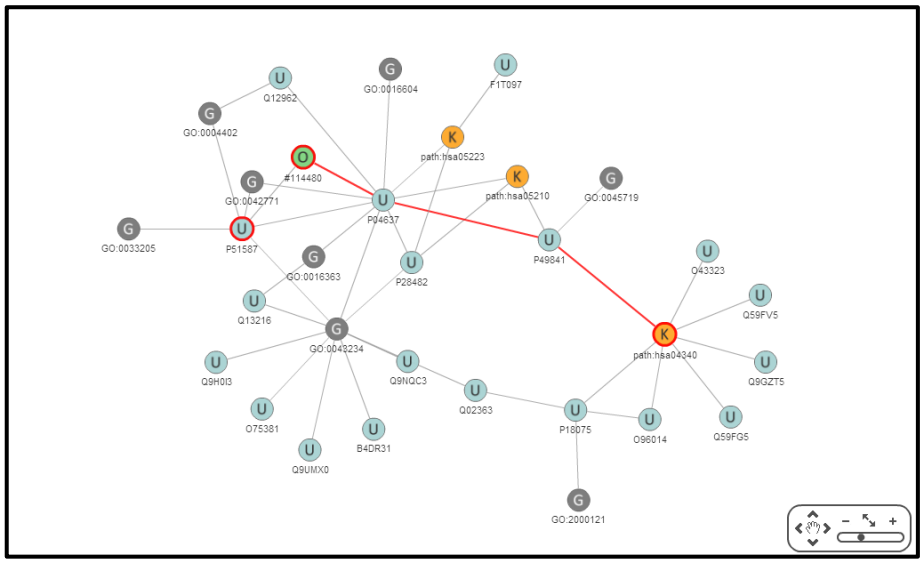


Figura 4.18 – Associações entre dois elementos

A secção de opções visuais é exclusivamente dedicada a todos os mecanismos que apenas alteram a representação da sub-rede, mas nunca o seu conteúdo. As opções disponibilizadas são:

- Alterar o esquema da rede – Permite alterar a disposição dos elementos da rede, através da aplicação de diferentes algoritmos para atribuição de coordenadas. Os modos de exibição são: *ForceDirected*, Circular, Radial e Árvore (Figura 4.19).
- Alterar o tamanho dos elementos – Através desta funcionalidade são aplicados métodos de ranking que atribuem a cada elemento um valor de

classificação. A dimensão de cada elemento pode estar associada a esse valor (Figura 4.20).

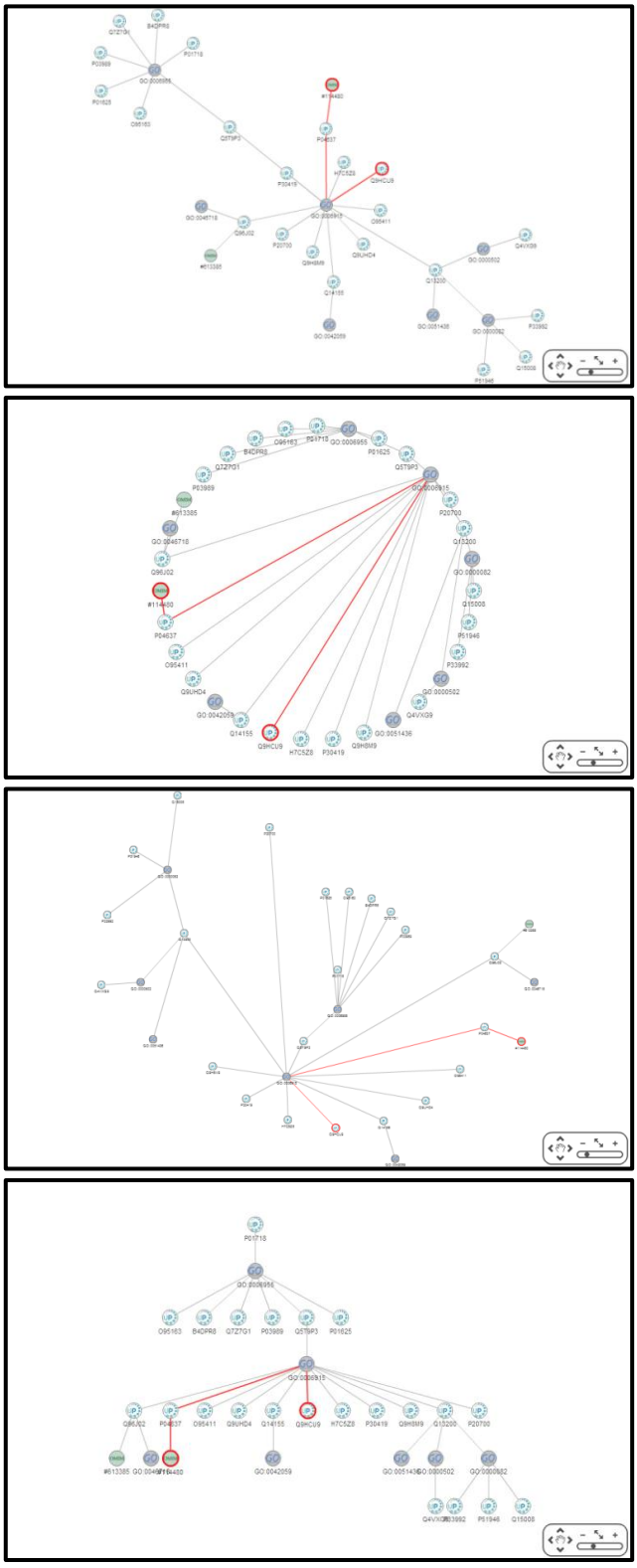


Figura 4.19 – Esquemas de representação de uma sub-rede Force Directed, Circular, Radial e Árvore

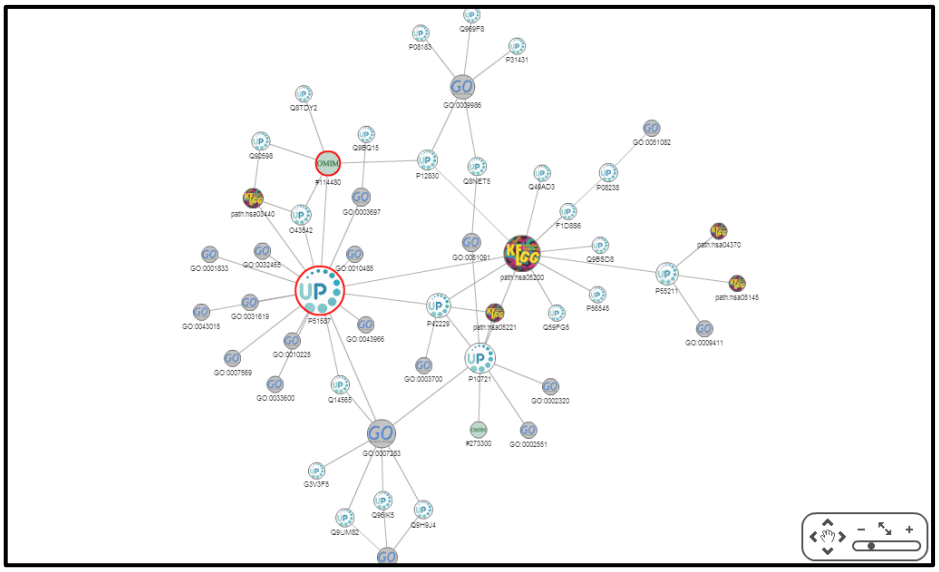


Figura 4.20 – Dimensão dos elementos influência da por resultado de métodos de classificação

- Alterar a forma dos elementos – Os elementos podem ser exibidos sob a forma de imagem ou apenas cores identificadas por letras.
- Alterar a legenda dos elementos – A legenda pode ser baseada no identificador ou no nome preferido de cada elemento.
- Alterar a cor dos *elementos base* – Permite mudar a cor da margem dos *elementos base* e das associações identificadas pelo explorador.

A última secção do menu permite exportar, através de diferentes formatos, a sub-rede que está a ser exibida.

Capítulo 5
Conclusões e Trabalho Futuro

5.1 Conclusões

Com o trabalho desenvolvido no âmbito desta dissertação foi criada uma plataforma que permite estudar e interpretar as associações entre termos biomédicos. Este documento apresenta os conhecimentos teóricos base do problema, o estudo sobre tecnologias e os desenvolvimentos práticos efetuados.

Os objetivos definidos durante a modelação do sistema foram, de modo geral, cumpridos com sucesso. As funcionalidades implementadas comprovam que o sistema final é uma ferramenta com bastante potencial nas áreas da bioinformática e biomedicina. No entanto, outras funcionalidades poderiam ter sido exploradas, naturalmente com resultados diferentes. Os mecanismos de análise da rede que acompanham o sistema podem facilmente servir diversos interesses e colaborar em estudos nas áreas referidas.

Através da integração de dados biológicos de diversas fontes externas, foi criada uma rede de termos biomédicos que permite um estudo mais abrangente das interações entre os diversos elementos. Através da representação e análise desses dados é possível extrair importantes evidências biomoleculares.

Esta aplicação foi desenvolvida segundo um paradigma de SaaS, por forma a estar mais próxima e facilmente acessível aos seus utilizadores. Através de uma aplicação Web o utilizador tem acesso a uma série de métodos e ferramentas computacionais que possibilitam a visualização e o estudo das relações entre doenças, genes e outros elementos/termos biomédicos. Certas funcionalidades também estão disponíveis através de uma API REST que pode ser livremente acedida por aplicações externas.

Representando num grafo a rede de termos biomédicos, foi possível a aplicação dos já consolidados conhecimentos em teoria de grafos e aplicá-los a esses dados de natureza biológica. Foram desenvolvidos e/ou aplicados algoritmos de filtragem, determinação de caminho mais curto e classificação de vértices que permite explorar as interações de elementos e compreender a sua relevância.

5.2 Trabalho Futuro

As decisões tomadas tanto na fase de requisitos como na fase de implementação determinaram o resultado final do projeto. Apesar dos dados e das funcionalidades que o sistema disponibiliza serem suficientemente proveitosas, várias melhorias podem ainda ser realizadas.

Relativamente aos dados da aplicação existe uma grande margem de expansão. A aplicação desenvolvida considera uma rede composta por apenas quatro tipos de termos biomédicos. Tirando partido da representação de entidades e associações é possível adicionar novas entidades de diferentes naturezas biomédicas, sem grande esforço de desenvolvimento. A título de exemplo, podem-se adicionar dados relativos a fármacos e a forma como estes se associam a determinadas doenças, ou referências de documentos que citam ontologias específicas, etc.

Também no que diz respeito a funcionalidades algumas melhorias podem ser efetuadas. No menu de exploração de associações entre dois elementos, a implementação de um mecanismo que permita a determinação, não de apenas um mas, de *n* caminhos mais curtos entre as entidades, conduziria a um estudo muito mais completo.

Os processos de filtragem da rede e classificação de elementos podem igualmente sofrer aperfeiçoamentos. A filtragem pode passar a ser influenciada pelos tipos de arestas e vértices, limitando o número de elementos de um tipo. Novos métodos de classificação podem ser adicionados, permitindo estudar a influência, não só de elementos mas também, de associações.

Bibliografia

[1] Facebook. [Online] http://www.facebook.com/. Accessed: 12-May-2012.

[2] Linkedin. [Online] http://www.linkedin.com/. Accessed: 12-May-2012.

[3] MySpace. [Online] http://www.myspace.com/. Accessed: 12-May-2012.

[4] Twitter. [Online] https://twitter.com/. Accessed: 12-May-2012.

[5] K. H. Wolfe and W.-H. Li, "Molecular evolution meets the genomics revolution", *Nature genetics*, vol. 33, pp. 255–265, Mar. 2003.

[6] M. Newman, *Networks: An Introduction*. Oxford University Press, Inc., 2010.

[7] N. L. Biggs, E. K. Lloyd, and Robin J. Wilson, *Graph Theory, 1736-1936*. Oxford University Press, 1986.

[8] R. Balakrishnan and K. Ranganathan, *A Textbook of Graph Theory*. Springer, 2000.

[9] M. V. Steen, *Graph Theory and Complex Networks*. Maarten Van Steen, 2010.

[10] G. A. Pavlopoulos, M. Secrier, C. N. Moschopoulos, T. G. Soldatos, S. Kossida, J. Aerts, R. Schneider, and P. G. Bagos, "Using graph theory to analyze biological networks", *BioData mining*, vol. 4, no. 1, p. 10, Jan. 2011.

[11] X. Wu and S. Li, "Cancer gene prediction using a network approach", *Cancer Systems Biology*, pp. 191–212, 2010.

[12] S. R. Leonard Richardson, *Restful Web Services*. O'Reilly Media, Inc., 2007.

[13] STRING 9.0. [Online] http://string-db.org/. Accessed: 25-May-2012.

[14] Ensembl. [Online] http://www.ensembl.org/. Accessed: 02-Aug-2012.

[15] Online Mendelian Inheritance in Man. [Online] http://www.omim.org/. Accessed: 29-May-2012.

[16] Gene Ontology Annotation. [Online] http://www.ebi.ac.uk/GOA/. Accessed: 06-Jul-2012.

[17] Gene Ontology. [Online] http://www.geneontology.org/. Accessed: 06-Jul-2012.

[18] Kyoto Encyclopedia of Genes and Genomes. [Online] http://www.kegg.jp/. Accessed: 06-Jul-2012.

[19] UniProt. [Online] http://www.uniprot.org/. Accessed: 06-Jul-2012.

[20] C. Vicknair, M. Macias, Z. Zhao, and X. Nan, "A comparison of a graph database and a relational database: a data provenance perspective", *Proceedings of the 48th annual Southeast regional conference*, p. 42, 2010.

[21] Microsoft SQL Server. [Online] http://www.microsoft.com/sqlserver. Accessed: 10-Jul-2012.

[22] Oracle Database. [Online] http://www.oracle.com/us/products/database. Accessed: 10-Jul-2012.

[23] MySQL. [Online] http://www.mysql.com/. Accessed: 10-Jul-2012.

[24] Amazon. [Online] http://www.amazon.com/. Accessed: 27-May-2012.

[25] N. Leavitt, "Will NoSQL databases live up to their promise?", *Computer*, vol. 43, no. 2, pp. 12–14, Feb. 2010.

[26] Google. [Online] https://www.google.com. Accessed: 10-Jul-2012.

[27] MongoDB. [Online] http://www.mongodb.org/. Accessed: 12-Jul-2012.

[28] Apache CouchDB. [Online] http://couchdb.apache.org/. Accessed: 12-Jul-2012.

[29] Neo4j. [Online] http://neo4j.org/. Accessed: 25-May-2012.

[30] The Neo4j Manual. [Online] http://docs.neo4j.org/. Accessed: 17-Jul-2012.

[31] Cytoscape Web. [Online] http://cytoscapeweb.cytoscape.org/. Accessed: 25-May-2012.

[32] JavaScript InfoVis Toolkit. [Online] http://thejit.org/. Accessed: 25-May-2012.

[33] Cobweb. [Online] http://bioinformatics.charite.de/cobweb/. Accessed: 25-May-2012.

[34] Arbor.js. [Online] http://arborjs.org/. Accessed: 25-May-2012.

[35] C. T. Lopes, M. Franz, F. Kazi, S. L. Donaldson, Q. Morris, and G. D. Bader, “Cytoscape Web: an interactive web-based network browser.”, *Bioinformatics*, vol. 26, no. 18, pp. 2347–2358, Sep. 2010.

[36] Apple. [Online] https://www.apple.com/. Accessed: 15-Jul-2012.

[37] Cytoscape Web - Github. [Online] http://cytoscape.github.com/cytoscapeweb/. Accessed: 15-Jul-2012.

[38] GraphML. [Online] http://graphml.graphdrawing.org/. Accessed: 25-May-2012.

[39] XGMML. [Online] http://www.cs.rpi.edu/~puninj/XGMML. Accessed: 25-May-2012.

[40] JGraphT. [Online] http://jgrapht.org/. Accessed: 07-Aug-2012.

[41] JDSL. [Online] http://www.cs.brown.edu/cgc/jdsl/. Accessed: 07-Aug-2012.

[42] JUNG - Java Universal Network/Graph Framework. [Online] http://jung.sourceforge.net/. Accessed: 07-Aug-2012.

[43] H. V. Westerhoff and B. O. Palsson, “The evolution of molecular biology into systems biology”, *Nature biotechnology*, vol. 22, no. 10, pp. 1249–1252, Oct. 2004.

[44] D. Hristovski, Borut Peterlin, J. A. Mitchell, and S. M. Humphrey, “Using literature-based discovery to identify disease candidate genes”, *International journal of medical informatics*, vol. 74, no. 2–4, pp. 289–298, Mar. 2005.

[45] J. P. Arrais and J. L. Oliveira, “Using biomedical networks to prioritize gene-disease associations”, *Open Access Bioinformatics*, pp. 123–130, 2011.

[46] S. H. Strogatz, “Exploring complex networks”, *Nature*, vol. 410, no. 6825, pp. 268–276, Mar. 2001.

[47] R. Buyya, J. Broberg, and A. M. Goscinski, *Cloud Computing: Principles and Paradigms*. John Wiley & Sons, 2011.

[48] S. Allamaraju, *RESTful Web Services Cookbook: Solutions for Improving Scalability and Simplicity*. O’Reilly Media, Inc., 2010.

[49] M. D. Lytras, E. Damiani, and P. O. de Pablos, *Web 2.0: The Business Model*. Springer, 2008.

[50] N. Antonopoulos and L. Gillam, *Cloud Computing: Principles, Systems and Applications*. Springer, 2010.

[51] G. Dong, L. Libkin, J. Su, and L. Wong, “Maintaining transitive closure of graphs in SQL”, *Int. J. Information Technology 5*, pp. 46–78, 1999.

[52] IBM Education Assistant - JavaServer™ Faces (JSF). [Online] http://publib.boulder.ibm.com/infocenter/ieduasst/rtnv1r0/index.jsp. Accessed: 17-Aug-2012.

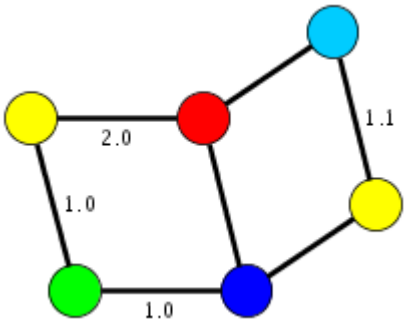
[53] U. Brandes, M. Eiglsperger, and J. Lerner, GraphML Primer. [Online] http://graphml.graphdrawing.org/primer/graphml-primer.html. Accessed: 17-Jul-2012.

Anexo A
GraphML

O exemplo seguinte mostra como o formato de descrição GraphML pode representar não só a forma mas também o estilo de um grafo [53].

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
        http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
  <key id="d0" for="node" attr.name="color" attr.type="string">
    <default>yellow</default>
  </key>
  <key id="d1" for="edge" attr.name="weight" attr.type="double"/>
  <graph id="G" edgedefault="undirected">
    <node id="n0">
      <data key="d0">green</data>
    </node>
    <node id="n1"/>
    <node id="n2">
      <data key="d0">blue</data>
    </node>
    <node id="n3">
      <data key="d0">red</data>
    </node>
    <node id="n4"/>
    <node id="n5">
      <data key="d0">turquoise</data>
    </node>
    <edge id="e0" source="n0" target="n2">
      <data key="d1">1.0</data>
    </edge>
    <edge id="e1" source="n0" target="n1">
      <data key="d1">1.0</data>
    </edge>
    <edge id="e2" source="n1" target="n3">
      <data key="d1">2.0</data>
    </edge>
    <edge id="e3" source="n3" target="n2"/>
    <edge id="e4" source="n2" target="n4"/>
    <edge id="e5" source="n3" target="n5"/>
    <edge id="e6" source="n5" target="n4">
      <data key="d1">1.1</data>
    </edge>
  </graph>
</graphml>
```

Esta descrição representará o seguinte grafo:



Anexo B

Função SQL de filtragem da rede – Grafo não orientado e baseado num só vértice central

```
create function BMN.GetEdgesUndirectedSingleNode(
    @IdNode as int,
    @MaxEdges as int,
    @MaxDepth as int,
    @MaxEdgesPerDepth as int,
    @MinScore as int
)returns @Result table(
    IdEdge int primary key,
    IdNodeReached int,
    Depth int
) as begin
    declare @ValidMaxEdges bit, @ValidMaxDepth bit,
@ValidMaxEdgesPerDepth bit;

    if @MaxEdges > 0 set @ValidMaxEdges = 1 else set @ValidMaxEdges = 0;
    if @MaxDepth > 0 set @ValidMaxDepth = 1 else set @ValidMaxDepth = 0;
    if @MaxEdgesPerDepth > 0 set @ValidMaxEdgesPerDepth = 1 else set
@ValidMaxEdgesPerDepth = 0;

    if (@ValidMaxEdges = 0 and @ValidMaxDepth = 0) return;

    declare @Inserted int, @TotalInserted int, @Depth int, @Top int;

    set @TotalInserted = 0;
    set @Depth = 1;

    set @Top = @MaxEdges - @TotalInserted;
    if @ValidMaxEdgesPerDepth = 1 and ((@ValidMaxEdges = 1 and @Top >
@MaxEdgesPerDepth) or (@ValidMaxEdges = 0))
        set @Top = @MaxEdgesPerDepth;
    else if @ValidMaxEdges = 0 and @ValidMaxEdgesPerDepth = 0
        set @Top = 0;

    if @Top > 0
        insert into @Result (IdEdge, IdNodeReached, Depth)
            select top (@Top) IdEdge, IdNodeReached, @Depth from (
                select IdEdge, RefIdNodeTgt as IdNodeReached, Score from
BMN.Edge
                    where RefIdNodeSrc = @IdNode
                    and Score >= @MinScore
```

```
                union
                select IdEdge, RefIdNodeSrc, Score from BMN.Edge
                    where RefIdNodeTgt = @IdNode
                    and Score >= @MinScore
            ) R
                order by Score desc
    else
        insert into @Result (IdEdge, IdNodeReached, Depth)
            select IdEdge, IdNodeReached, @Depth from (
                select IdEdge, RefIdNodeTgt as IdNodeReached, Score from
BMN.Edge
                    where RefIdNodeSrc = @IdNode
                    and Score >= @MinScore
                union
                select IdEdge, RefIdNodeSrc, Score from BMN.Edge
                    where RefIdNodeTgt = @IdNode
                    and Score >= @MinScore
            ) R
                order by Score desc

    set @Inserted = @@ROWCOUNT;
    set @TotalInserted = @Inserted;

    while @Inserted > 0 and (@ValidMaxEdges = 0 or @TotalInserted <
@MaxEdges) and
    (@ValidMaxDepth = 0 or @Depth < @MaxDepth) begin
        set @Depth += 1;

        set @Top = @MaxEdges - @TotalInserted;
        if @ValidMaxEdgesPerDepth = 1 and ((@ValidMaxEdges = 1 and @Top >
@MaxEdgesPerDepth) or (@ValidMaxEdges = 0))
            set @Top = @MaxEdgesPerDepth;
        else if @ValidMaxEdges = 0 and @ValidMaxEdgesPerDepth = 0
            set @Top = 0;

        if @Top > 0
            insert into @Result (IdEdge, IdNodeReached, Depth)
                select IdEdge, IdNodeReached, @Depth from(
                    select top (@Top) IdEdge, MIN(IdNodeReached) as
IdNodeReached, MIN(Score) as Score  from (
                        select IdEdge, RefIdNodeTgt as IdNodeReached,
Score from BMN.Edge
                            where RefIdNodeSrc in (select IdNodeReached
from @Result where Depth = @Depth - 1)
                            and IdEdge not in (select IdEdge from
@Result)
                            and Score >= @MinScore
                        union
                        select IdEdge, RefIdNodeSrc, Score from BMN.Edge
                            where RefIdNodeTgt in (select IdNodeReached
from @Result where Depth = @Depth - 1)
                            and IdEdge not in (select IdEdge from
@Result)
                            and Score >= @MinScore
                    ) R
                        group by IdEdge
                ) R
        else
            insert into @Result (IdEdge, IdNodeReached, Depth)
```

```
                select IdEdge, MIN(IdNodeReached), @Depth from (
                    select IdEdge, RefIdNodeTgt as IdNodeReached from
BMN.Edge
                        where RefIdNodeSrc in (select IdNodeReached from
@Result where Depth = @Depth - 1)
                        and IdEdge not in (select IdEdge from @Result)
                        and Score >= @MinScore
                    union
                    select IdEdge, RefIdNodeSrc from BMN.Edge
                        where RefIdNodeTgt in (select IdNodeReached from
@Result where Depth = @Depth - 1)
                        and IdEdge not in (select IdEdge from @Result)
                        and Score >= @MinScore
                ) R
                    group by IdEdge

        set @Inserted = @@ROWCOUNT;
        set @TotalInserted += @Inserted;
    end;
    return;
end;
go
```

Anexo C

Função SQL de filtragem da rede – Grafo orientado e baseado em múltiplos vértices centrais

```
create function BMN.GetEdgesDirectedMultiNodes(
    @IdNodes as BMN.ListOfNodes readonly,
    @MaxEdges as int,
    @MaxDepth as int,
    @MaxEdgesPerDepth as int,
    @MinScore as int
)returns @Result table(
    IdEdge int primary key,
    IdNodeReached int,
    Depth int
) as begin
    declare @ValidMaxEdges bit, @ValidMaxDepth bit,
@ValidMaxEdgesPerDepth bit;

    if @MaxEdges > 0 set @ValidMaxEdges = 1 else set @ValidMaxEdges = 0;
    if @MaxDepth > 0 set @ValidMaxDepth = 1 else set @ValidMaxDepth = 0;
    if @MaxEdgesPerDepth > 0 set @ValidMaxEdgesPerDepth = 1 else set
@ValidMaxEdgesPerDepth = 0;

    if (@ValidMaxEdges = 0 and @ValidMaxDepth = 0) return;

    declare @Inserted int, @TotalInserted int, @Depth int, @Top int;

    set @TotalInserted = 0;
    set @Depth = 1;

    set @Top = @MaxEdges - @TotalInserted;
    if @ValidMaxEdgesPerDepth = 1 and ((@ValidMaxEdges = 1 and @Top >
@MaxEdgesPerDepth) or (@ValidMaxEdges = 0))
        set @Top = @MaxEdgesPerDepth;
    else if @ValidMaxEdges = 0 and @ValidMaxEdgesPerDepth = 0
        set @Top = 0;

    if @Top > 0
        insert into @Result (IdEdge, IdNodeReached, Depth)
            select top (@Top) IdEdge, RefIdNodeTgt, @Depth from BMN.Edge
                where RefIdNodeSrc in (select IdNode from @IdNodes)
                and Score >= @MinScore
                order by Score desc
    else
        insert into @Result (IdEdge, IdNodeReached, Depth)
```

```
            select IdEdge, RefIdNodeTgt as IdNodeReached, @Depth from
BMN.Edge
                where RefIdNodeSrc in (select IdNode from @IdNodes)
                and Score >= @MinScore
                order by Score desc

    set @Inserted = @@ROWCOUNT;
    set @TotalInserted = @Inserted;

    while @Inserted > 0 and (@ValidMaxEdges = 0 or @TotalInserted <
@MaxEdges) and
    (@ValidMaxDepth = 0 or @Depth < @MaxDepth) begin
        set @Depth += 1;

        set @Top = @MaxEdges - @TotalInserted;
        if @ValidMaxEdgesPerDepth = 1 and ((@ValidMaxEdges = 1 and @Top >
@MaxEdgesPerDepth) or (@ValidMaxEdges = 0))
            set @Top = @MaxEdgesPerDepth;
        else if @ValidMaxEdges = 0 and @ValidMaxEdgesPerDepth = 0
            set @Top = 0;

        if @Top > 0
            insert into @Result (IdEdge, IdNodeReached, Depth)
                select top (@Top) IdEdge, RefIdNodeTgt as IdNodeReached,
@Depth from BMN.Edge
                    where RefIdNodeSrc in (select IdNodeReached from
@Result where Depth = @Depth - 1)
                    and IdEdge not in (select IdEdge from @Result)
                    and Score >= @MinScore
        else
            insert into @Result (IdEdge, IdNodeReached, Depth)
                select IdEdge, RefIdNodeTgt, @Depth from BMN.Edge
                    where RefIdNodeSrc in (select IdNodeReached from
@Result where Depth = @Depth - 1)
                    and IdEdge not in (select IdEdge from @Result)
                    and Score >= @MinScore

        set @Inserted = @@ROWCOUNT;
        set @TotalInserted += @Inserted;
    end;
    return;
end;
go
```