Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Graph2Vis: Artefato de Visualização 3D para Banco de Dados Neo4j

Lucas Correa Lemos

Monografia apresentada como requisito parcial
para conclusão do Curso de Computação — Licenciatura

Orientadora
Prof.a Dr.a Célia Ghedini Ralha

Brasília
2022

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Graph2Vis: Artefato de Visualização 3D para Banco de Dados Neo4j

Lucas Correa Lemos

Monografia apresentada como requisito parcial
para conclusão do Curso de Computação — Licenciatura

Prof.a Dr.a Célia Ghedini Ralha (Orientadora)
CIC/UnB

Prof.a Dr.a Maristela Terto de Holanda
CIC/UnB

Prof.a Dr.a Vanessa Tavares Nunes
CIC/UnB

Prof. Dr. Wilson Henrique Veneziano
Coordenador do Curso de Computação — Licenciatura

Brasília, 06 de maio de 2022

Dedicatória

A todos os leitores deste trabalho.

Agradecimentos

Agradeço aos meus pais Ana e Guilherme, irmão Caio e meu padastro Felipe, que me apoiaram incondicionalmente neste trabalho e em todas as etapas da minha passagem pela UnB, que estão sempre ao meu lado em cada novo desafio que enfrento, acreditando sempre no meu sucesso.

À minha orientadora, Prof.a Dr.a Célia Ghedini Ralha, que ofereceu a devida orientação, atenção e paciência ao desenvolvimento deste trabalho.

Também agradeço aos amigos e colegas de curso que contribuíram para que este trabalho tivesse sucesso.

Aos colegas Aurélio Ribeiro Costa, doutorando do Programa de Pós-Graduação em Informática da UnB e Mariana Alencar do Vale, estudante de graduação em Ciência da Computação da UnB pela contribuição à este trabalho.

Em especial agradeço ao Prof. Me. Natan de Souza Rodrigues, doutorando do Programa de Pós-Graduação em Informática da UnB, o qual me acompanha desde a infância, me proporcionando momentos de imenso aprendizado durante esses anos.

Resumo

O grande volume de dados abertos a respeito de integrantes de redes sociais aumenta o desafio para realização de análises de cunho político, financeiro, educacional, entre outras aplicações, assumindo um papel de destaque na sociedade. O grande desafio se estende aos aspectos de visualização da informação para viabilizar a compreensão quantitativa e qualitativa de aspectos relevantes. Desta forma, o objetivo deste trabalho é apresentar o Graph2Vis, uma solução para visualização 3D de banco de dados online orientadas a grafo utilizando Neo4J. A construção do artefato Graph2Vis passou pelas etapas de modelagem da solução, construção arquitetural utilizando o padrão MVC e a implementação com *framework* Angular e biblioteca *3d-force-graph*. Foram realizados três estudos de caso com discussão de resultado utilizando uma rede social científica vinculada ao Projeto *SCI-synergy*[1], um substrato do *SCI-synergy* relacionado a aspectos de gênero, e uma rede formada pelo projeto *ADAN*[2] relacionada à desambiguação de nomes em repositórios bibliográficos. O artefato permite extensões futuras, principalmente quanto à funcionalidades relacionadas à cálculo de medidas de centralidade em grafos, as quais foram utilizadas no Projeto *SCI-synergy*. Também é possível implementar outras técnicas de visualização bidimensional, como por exemplo utilizando matrizes para resumir aspectos de conectividade da rede, além de listas para apresentar os resultados das medidas de centralidade.

Palavras-chave: grafos, Neo4j, redes sociais, visualização de grafos

[1] `http://165.227.113.212`
[2] `https://gitlab.com/InfoKnow/SocialNetwork/sci_clan/adan`

Abstract

The large volume of open data about members of social networks increases the challenge to carry out analyzes of a political, financial, educational nature, among other applications, assuming a prominent role in society. The great challenge extends to the information visualization aspects to enable the quantitative and qualitative understanding of relevant aspects. Thus, the objective of this work is to present Graph2Vis, a solution for graph-oriented 3D online database visualization using Neo4J. The Graph2Vis artifact construction went through the stages of solution modeling, architectural construction using the MVC pattern, and implementation with Angular framework and 3d-force-graph library. Three case studies with discussion of results were carried out using a scientific social network linked to the *SCI-synergy*[3] Project, a substrate of *SCI-synergy* related to gender aspects, and a network formed by the project *ADAN*[4], related to disambiguation of names in bibliographic repositories. The artifact allows future extensions, mainly regarding functionalities related to the calculation of measures of centrality in graphs, which were used in the *SCI-synergy* Project. It is also possible to implement other two-dimensional visualization techniques, such as using matrices to summarize aspects of network connectivity, as well as lists to present the results of centrality measures.

Keywords: graphs, Neo4j, social networks, graph visualization

[3] `http://165.227.113.212`
[4] `https://gitlab.com/InfoKnow/SocialNetwork/sci_clan/adan`

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Contextualização

Durante o ano de 2021, a estimativa da quantidade de dados criados todos os dias foi em torno de 1,145 trilhões de *megabytes* [6]. Além da grande quantidade de dados disponíveis, há de se considerar a sua complexidade. É um problema de *design* atemporal - o que incluir versus o que cortar na busca para se comunicar com clareza. A visualização de dados não está isenta, especialmente quando os dados são abundantes. O erro presente na maioria dos casos é o de gerar em uma única visualização a representação de quantidade significativa de dados. Os humanos não estão aptos a calcular o significado de vários valores abstraídos de forma visual. Desta forma, a solução para visualização de uma quantidade significativa de dados é incluir diversas formas de visualizar os dados para que a comunicação seja mais efetiva.

De modo geral, os dados que são disponibilizados pela Internet são de difícil manipulação. Neste sentido, a visualização de dados em forma de grafos pode ser favorável e simples aos olhos de quem vê, seja ele leigo ou especialista. Já o uso de um banco de dados não relacional (*Not Only Structured Query Language* - NoSQL) baseado em grafos pode oferecer recursos interessantes como: boa performance, flexibilidade e agilidade [7].

Segundo a *Neo Technology*® [8], criadora do banco de dados NoSQL orientado a grafo, há vantagens no uso do Neo4j [9]. Como pontos fortes pode-se citar a facilidade de uso, alta performance de escrita e leitura em consultas, e a maior comunidade de usuários dentre os banco de dados NoSQL orientado a grafo. O Neo4j possui uma área para visualização 2D dos dados de consultas denominado *Neo4j Browser* [10]. Porém Lanum [11] aponta que a oclusão influencia negativamente a experiência de visualização e análise dos grafos, especialmente em contextos 2D. Neste sentido, em consonância com o problema da oclusão existem outros aspectos que demandam soluções, por exemplo, a viabilização de uma

maior interatividade nas visualizações 2D apresentadas pelo *Neo4j Browser*, indicando a necessidade de um artefato 3D.

Neste contexto, esta monografia aborda o problema da dificuldade de visualização de variados volumes de dados textuais armazenados em banco de dados orientado a grafo Neo4j. A importância desse trabalho é válido para a área de visualização de dados, permitindo análises tanto do ponto de vista qualitativo como quantitativo.

1.2 Problema e Hipótese

De acordo com a platorma *DB-engine*, o uso de Banco de dados NoSQL têm ganhado popularidade [12]. Desde leigos, pesquisadores, analistas e técnicos de tecnologia da informação e comunicação vem utilizando várias formas de visualização desses dados [13].

Esta monografia aborda o problema relativo a dificuldade de visualização de dados textuais armazenados em banco de dados NoSQL orientado a grafo usando Neo4j. Alguns problemas acessórios podem ser citados, tais como a minimização do problema de oclusão à visualização 2D de dados manipulados, o que dificulta as análises visuais de redes baseadas em grafo.

O contexto desse trabalho é gerar uma solução de visualização 3D aliada ao banco de dados Neo4j adequada ao tratamento de volumes diferentes de dados com a viabilização de interatividade através de diferentes visões dos grafos reduzindo as possíveis oclusões. Desta forma, a hipótese adotada no trabalho é que um artefato de visualização 3D para bancos de dados Neo4j online seja adequado para apresentar dados textuais viabilizando interpretações dos grafos com diferentes visões. Esta hipótese está justificada na detecção de uma lacuna de pesquisa, a qual foi identificada durante a revisão da literatura apresentada no Capítulo 3, onde os trabalhos selecionados presentes na Tabela 3.1 apontam que somente um artigo apresenta artefato de visualização com técnica 3D ([14]).

1.3 Questões de Pesquisa

Durante a confecção deste trabalho algumas questões investigativas foram definidas para direcionar o escopo da pesquisa. Responder as questões investigativas tem como finalidade aumentar o conhecimento a cerca da representação gráfica dos dados manipuladas nos estudos de caso, além de serem úteis para visualização e análise de grafos das redes sociais envolvidas. As questões definidas foram:

1. Quais são as soluções teóricas e de artefatos implementados existentes na literatura para visualização 3D de bases de dados orientadas a grafo utilizando Neo4j?

2. Como desenvolver um artefato de visualização 3D de bases de dados orientadas a grafo utilizando Neo4j?

3. Quais estudos de caso podem ilustrar o artefato desenvolvido durante a execução deste trabalho?

1.4 Objetivos

O trabalho tem como objetivo principal o desenvolvimento de uma solução que simplifique o acesso e uso do Neo4j potencializando a capacidade de visualização dos dados armazenados. Neste sentido, o objetivo é propor e desenvolver um artefato de visualização 3D para bases de dados orientadas a grafo usando Neo4j. Como objetivos secundários podemos citar:

1. Validar a solução implementada utilizando estudos de caso com redes sociais armazenadas em banco de dados Neo4j online.

2. Gerar informação quantitativa das redes sociais armazenadas no Neo4j utilizando padrões de coloração e tamanho dos nós para identificação de comunidades e nós específicos da rede.

1.5 Estrutura do Documento

O restante da monografia inclui os seguintes capítulos:

- o Capítulo 2 apresenta uma revisão dos fundamentos teóricos utilizados nesse trabalho incluindo aspectos de análise visual, conceitos de teoria de grafos, conceitos e técnicas de banco de dados NoSQL, redes sociais e arquitetura de software padão MVC (*Model* - modelo, *View* - visão e *Controller* - controle).

- o Capítulo 3 aborda uma revisão da literatura com foco em técnicas, funções e artefatos de visualização de grafos.

- no Capítulo 4 é apresentada a proposta de solução incluindo as fases de *design* até o desenvolvimento.

- os experimentos são apresentados no Capítulo 5 através de três estudos de caso incluindo redes sociais científicas.

- finalmente, as conclusões e publicações são apresentadas no Capítulo 6, bem como sugestões de trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Neste capítulo serão apresentados os conceitos que serviram como fundamentos para o trabalho realizado. Na Seção 2.1 é apresentado o conceito de análise visual com diferentes formas de interpretação. A Seção 2.2 aborda conceitualização relacionada à teoria de grafos. A Seção 2.3 refere-se aos aspectos de redes sociais. A Seção 2.4 trata de bancos de dados NoSQL. Por fim, a Seção 2.5 apresenta uma visão geral sobre arquitetura de software com padrão MVC.

2.1 Análise Visual

O termo *Visual Analytics* foi introduzido por Jim Thomas et al. [15], definindo como uma solução de problemas de análise de dados usando uma interface visual interativa e facilitadora. O mesmo termo também é usado para nomear a disciplina científica sobre esta atividade [16].

Atualmente, análises visuais em forma de grafos são amplamente utilizadas em várias áreas de ações humanas, tais como, o monitoramento de doenças em tempos de pandemia [17], dados de programas sociais do governo [18], entre outras áreas. A chave para o amplo uso com sucesso de análises visuais em vários campos e circunstâncias é a pré-disposição das pessoas ao pensamento espacial e reconhecimento de imagens [19].

Interpretação das Análises Visuais da Representação Gráfica

Segundo Keim et al. [20], o problema relativo a interpretação do que é representado graficamente é resolvido diretamente por quem está analisando. Os resultados podem e irão variar dependendo da pessoa em questão, sua formação e objetivos. Keim et al. definem a solução do problema de interpretação de representações gráficas em duas etapas genéricas, a saber:

1. *Interpretação visual* – Fundamentalmente importante para quem está realizando a análise visual para entender que resultados são naturais e convenientes a partir dos dados originais. Nesta etapa, o analista olha para a representação gráfica e percebe algumas regularidades ou irregularidades. Esta etapa não precisa ser estritamente formalizada, porém pode-se definir o foco de análise, como por exemplo:

   - As formas de objetos no espaço da cena: "Um objeto número 1 é um cubo, enquanto o Objeto 2 é uma esfera".
   - A posição relativa dos objetos: "Objetos 1 a 6 estão próximos, enquanto o Objeto 7 está distante".
   - Parametros Óticos: "um objeto é vermelho"ou "outro objeto é opaco".

   É importante ressaltar que os resultados da Etapa 1 irão diferir a respeito do analista e suas habilidades de pensamento espacial, percepção de detalhes, foco, entre outros.

2. *Interpretação dos resultados com respeito aos dados iniciais* – Essa etapa é crucial, pois oferece a visão geral dos resultados e isso geralmente não pode ser formalizado. O analista tem que entender a conexão e transformação dos dados iniciais para a cena espacial e assim ter suas conclusões. A efetividade desta etapa depende fortemente do entendimento do analista, ao mesmo tempo que possibilita o conhecimento de novas análises.

A Figura 2.1 apresenta a análise visual incluindo as Etapas 1 e 2 descritas.

Figura 2.1: Ilustração de uma análise visual simples (autoria própria).

2.2 Teoria de Grafos

A cada dia o volume de dados gerados é cada vez maior e mais complexo. Desta forma, aumentam os desafios de análise organizada das redes geradas a partir desses dados. Também faz-se necessário tornar as etapas do processo mais compactas e leves do ponto de vista computacional, aliado a capacidade de suporte dos *hardwares* disponíveis.

Segundo Sipser [1], um grafo é um conjunto de pontos com linhas conectando outros pontos. Os pontos são chamados de *nós* ou *vértices* e as linhas são chamadas *arestas*. A Figura 2.2 apresenta a estrutura de dois grafos não direcionados e totalmente conexos.

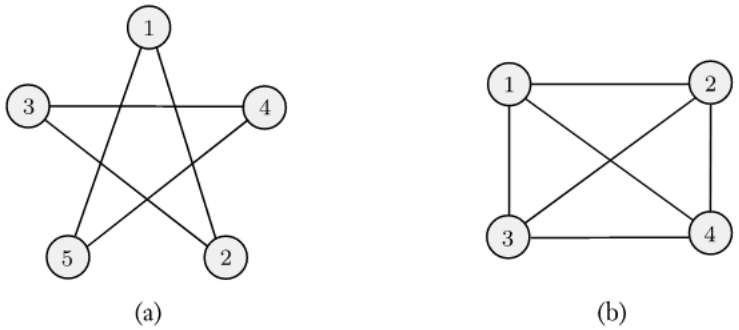


Figura 2.2: Exemplos de grafos não direcionados e conexos (Fonte: [1]).

Através da notação de conjuntos, um grafo G que contém nós i e j, o par (i,j) representa a aresta que conecta i e j. A ordem de i e j não importam neste tipo de grafo não direcionado. Se V é o conjunto de nós do grafo G e E é o conjunto de arestas, dizemos que $G = (V, E)$. A descrição formal do grafo da Figura 2.2(b) utilizando a notação de conjuntos é ({1,2,3,4}, {(1,2), (1,3), (1,4), (2,3), (2,4), (3,4)}).

Sipser [1] aponta que os grafos enquanto estrutura são frequentemente usados para representar dados. Os nós podem representar pessoas e as arestas relações de amizade entre elas, ou os nós podem ser cidades e arestas as estradas entre elas.

Medidas de Centralidade

Em teoria de grafos e análise de redes, indicadores de centralidade assumem números ou níveis para nós dentro de um grafo, representando as suas posições na rede. Em aplicações, por exemplo, identificar pessoas mais influentes em uma rede social pode-se valer dessas medidas para uma análise mais detalhada.

Degree centrality O grau do nó de um grafo é definido como o número de arestas incidentes sobre o nó. Aplicado a um grafo de uma rede social, por exemplo, o grau de um nó pode representar a quantidade de relações de uma pessoa com outras, como

coautorias em uma rede de pesquisadores, definindo assim diferentes níveis de interação entre as pessoas. O grau de centralidade de um vértice v para um dado grafo $G := (V, E)$ com $|V|$ vértices e $|E|$ arestas é dado pela Equação 2.1.

$$C_d(v) = deg(v) \tag{2.1}$$

O cálculo da centralidade de grau para todos os nós em um grafo leva $\theta(V^2)$ em uma representação de matriz de adjacência densa do grafo, e para as arestas leva $\theta(E)$ em uma representação de matriz esparsa.

A definição de centralidade do grau do nó pode ser estendida para todo o grafo. Neste caso estamos falando de centralização de grafos [21]. Seja $v*$ o nó com maior grau de centralidade em um grafo G. Seja $X := (Y, Z)$ sendo $|Y|$ -ésimo nó conectado do grafo que maximiza a seguinte quantidade (com $y*$ sendo o nó com maior grau de centralidade em X), tal como apresentado na Equação 2.2.

$$H = \sum_{j=1}^{|Y|} [C_D(y*) - C_D(y_i)] \tag{2.2}$$

Correspondentemente, o grau de centralidade do grafo G é dado pela Equação 2.3.

$$C_D(G) = \frac{\sum_{i=1}^{|V|} [C_D(v*) - C_D(v_i)]}{H} \tag{2.3}$$

Louvain Centrality A medida de *Louvain* para detecção de comunidades utiliza um método para extrair comunidades de redes criadas por Blondel et al. [22] da Universidade de Louvain. O método utiliza otimização gulosa que é executado no tempo $O(n.log(n))$ onde n é o número de nós na rede. [23]. A idéia para este método tem como foco o uso de otimização de modularidade a medida que algoritmo é executado. Esse valor de modularidade varia em uma escala entre -0.5 (baixa modularidade) e 1 (alta modularidade) que mede a densidade relativa de arestas dentro dessas comunidades em relação às arestas das comunidades mais periféricas. Otimizando esse valor de modularidade resultam no melhor agrupamento de nós de uma rede. Mas passar por todas as iterações possíveis dos nós em grupos é impraticável, algoritmos heurísticos são usados.No método de detecção de comunidades de Lo*Louvain*, primeiro pequenas comunidades são encontradas otimizando a modularidade localmente em todos os nós, então cada pequena comunidade é agrupada em um nó e o primeiro passo é repetido.

Algoritmo Louvain A modularidade a ser otimizada, é definida como valor na faixa $[-1/2, 1]$ que mensura a densidade das arestas dentro das comunidades comparadas com as arestas de outras comunidades [22]. Para um grafo, a modularidade é definida como:

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j), \tag{2.4}$$

Onde:

- A_{ij} representa o peso da aresta entre os nós i e j
- k_i e k_j são a soma dos pesos das arestas anexadas aos nós i e j, respectivamente;
- m é a soma de todos os pesos das arestas no grafo;
- c_i e c_j são as comunidades dos nós;
- δ é a função delta de Kronecker $\delta\ (x, y) = 1$ se $x = y$, 0 caso contrário).

Onde Σ_{in} é a soma de todos os pesos dos links dentro da comunidade i que está entrando, Σ_{tot} é a soma de todos os pesos dos links para nós na comunidade i que está se movendo, k_i é o grau ponderado de i, in é a soma dos pesos dos links entre i e outros nós da comunidade para a qual i está se movendo, e m é a soma dos pesos de todos os links na rede. Então, uma vez que este valor é calculado para todas as comunidades que i está conectado, i é colocado na comunidade que resultou no maior aumento de modularidade. Se nenhum aumento for possível, i permanece em sua comunidade original. Este processo é aplicado repetidamente e sequencialmente a todos os nós até que nenhum aumento de modularidade possa ocorrer. Uma vez atingido este máximo local de modularidade, a primeira fase terminou.

Na segunda fase do algoritmo, ele agrupa todos os nós da mesma comunidade e constrói uma nova rede onde os nós são as comunidades da fase anterior. Quaisquer links entre nós da mesma comunidade agora são representados por auto-loops no novo nó da comunidade e links de vários nós na mesma comunidade para um nó em uma comunidade diferente são representados por arestas ponderadas entre comunidades. Uma vez que a nova rede é criada, a segunda fase terminou e a primeira fase pode ser reaplicada à uma nova rede.

2.3 Redes Sociais

A perspectiva das redes sociais engloba teorias, modelos e aplicações que são expressadas em termos de relações entre os componentes dessas redes. Mais especificamente, as relações são definidas como *links* ou ligações, que identificam algum processo de interação entre as unidades de uma rede [24].

No âmbito de comunicação social, Recuero [25] conceitua redes sociais como:

> Uma rede social é definida como um conjunto de dois elementos: atores (pessoas, instituições ou grupos; os nós da rede) e suas conexões (interações ou laços sociais). Uma rede, assim, é uma metáfora para observar os padrões de conexão de um grupo social, a partir das conexões estabelecidas entre os diversos atores. A abordagem de rede tem, assim, seu foco na estrutura social, onde não é impossível isolar os atores sociais e nem suas conexões (RECUERO, 2009, p. 24).

Assim, um primeiro fator para compreender as redes sociais é analisar os atores e suas interações. Porém essas interações têm suas peculiaridades, principalmente porque esses laços são construídos tendo como mediador um dispositivo conectado à Internet. Essa observação é necessária, pois as conexões podem ocorrer de diferentes maneiras. Portanto, um ator, e consequentemente seus laços sociais, podem ser representados por um *blog*, *site* ou uma conta com perfil no *Twitter*, por exemplo [25].

Em termos computacionais, uma rede social é um grafo que tem como vértices as pessoas e as arestas são as interações sociais entre essas pessoas, como por exemplo relações de amizade ou trabalho em equipe em algum projeto. Esses grafos podem ser direcionados ou não direcionados. Por exemplo, o *Facebook* pode ser descrido como um grafo não direcionado, pois a relação de amizade entre as pessoas ali é bidirecional, isto é, João e Maria serem amigos é o mesmo que Maria e João serem amigos. Já a rede social formada pelo *Twitter* pode ser descrita como um grafo direcionado: João pode seguir Maria sem que Maria siga João. A Figura 2.3 apresenta um modelo de uma rede de *tweets*.

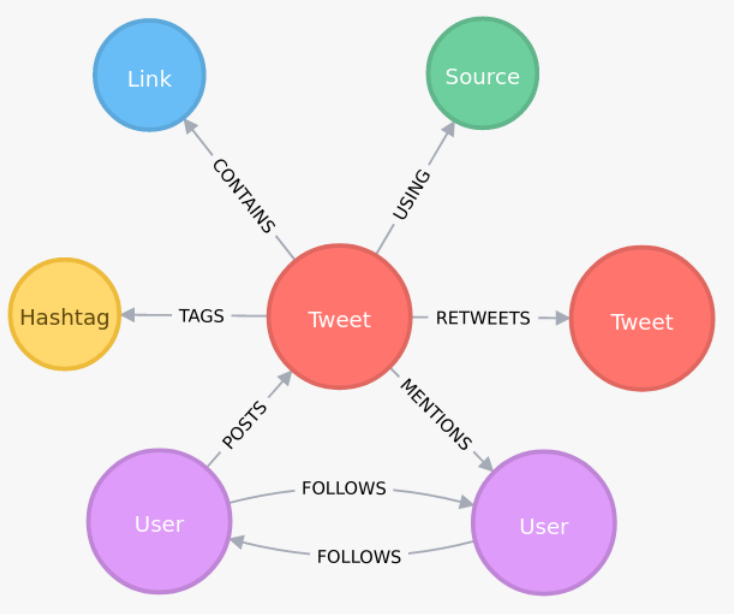


Figura 2.3: Um modelo de rede de Tweets (Fonte: [2]).

Redes sociais tendem a ter características e propriedades de grafos. Por exemplo, há a tendência de se ter um caminho mínimo de distância entre quaisquer dois nós de uma rede (teoria dos seis graus de separação), a qual diz que qualquer pessoa está a no mínimo seis passos de conhecer outra pessoa no mundo [26]. Há também a tendência a se formarem *triângulos* na rede. Ou seja, se João é amigo de Maria e Carol, então Maria e Carol são propensas a serem amigas entre si.

2.4 Banco de Dados NoSQL

Esta seção trata sobre o banco de dados NoSQL, com ênfase no modelo com base nos grafos de redes sociais utilizados neste trabalho.

Strozzi [27] popularizou em 1998 o termo "NoSQL", que tem como significado *Not Only Structured Query Language*, através de uma solução de banco de dados que não fornecia uma interface SQL. Porém, o sistema em questão tem por base a arquitetura relacional. Os líderes no uso de bancos de dados NoSQL são grandes empresas da web, como Google, Amazon e Facebook para promovê-los, construir e apoiar seus negócios. Depois de tornarem o NoSQL público e de código aberto, outras empresas gigantes da web, como Twitter, Instagram e Apple começaram a usá-los [28].

Algumas situações tornaram-se um grande desafio para quem até então utiliza os bancos de dados relacionais, como por exemplo [29]:

- Aplicações que criam alto volume de mudanças em dados estruturados, semi-estruturados e não estruturados.
- O ciclo de desenvolvimento de aplicações não tem mais períodos longos de meses ou até anos, com a geração de novo código sendo feita em questão de 1-2 semanas.
- Aplicações atualmente requerem que estejam sempre *online* e acessíveis de diferentes dispositivos e que escale o acesso para milhões de usuários.

Neste contexto, existem alguns motivos para utilização de bancos de dados NoSQL em relação aos bancos relacionais, tais como [30]:

- O crescimento de *Big Data* demandando alta velocidade de processamento, com grande variedade e complexidade de dados.
- Dados sempre disponíveis.
- Arquitetura de dados flexível e de alta performance.
- Altas capacidades transacionais.

Nesse sentido, o NoSQL se encaixa em banco de dados distribuídos, oferecendo assim uma escalabilidade horizontal, que resulta em um número crescente de máquinas disponíveis sem prejudicar a performance do banco de dados. Com a grande quantidade de instâncias, há um armazenamento e processamento com um maior desempenho, oferencendo assim uma baixa latência [30]

2.4.1 Neo4j

O livro elaborado por Vukotic et al. [31] apresenta um manual compreensivo sobre o Neo4j e suas funcionalidades. Neste sentido, os autores apontam que o Neo4j é eficiente e performático em relação aos outros bancos de dados Relacionais e Não - Relacionais para a solução de problemas específicos, como por exemplo, aplicações em redes sociais. Os autores ilustram a performance superior de pesquisas e escalabilidade do Neo4j em relação aos bancos de dados relacionais.

Já Fowler et al.[30] ressaltam mais uma das vantagens do Neo4J relacionada a garantia das propriedades *ACID* (*Atomicity, Consistency, Isolation, Durability*) [30], a saber:

1. *A*tomicidade: Cada operação move o banco de dados de um estado válido para outro;

2. *C*onsistência: Todos os usuários têm a mesma visão dos dados a qualquer momento;

3. *I*solamento: As operações no banco de dados não interferem umas nas outras;

4. *D*isponibilidade: Quando um banco de dados diz que salvou dados, você sabe que os dados estão seguros.

Segundo o CEO da empresa, Emil Eifrem, o Neo4j utiliza o conceito altamente cognitivo e poderoso para desenvolvimento de relações entre diversos dados, o conceito do *whiteboard* [32]. Quando uma pessoa tenta passar uma ideia em um quadro branco, não há limites para desenhar grafos, por exemplo. O grafo é implementado de forma íntegra no Neo4j. Assim sendo, o banco de dados possui a total capacidade de criação e manipulação de grafos, conseguindo resultados também através deles. A Figura 2.4 apresenta o resultado de uma consulta no Neo4j com dados de usuários do *Facebook*.

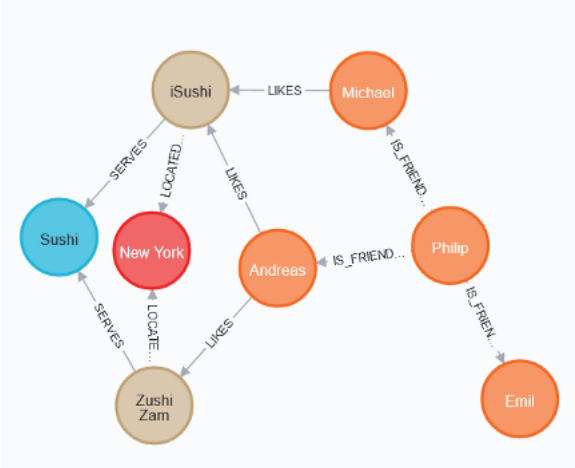


Figura 2.4: Rede de usuários do *Facebook* (Fonte: [3]).

O Neo4j possui versões disponíveis para vários sistemas operacionais. Como linguagem para realizar as consultas ao banco de dados, há o uso do *Cypher* [33]. Como é baseado em grafos, o Neo4j tem um forte uso em análise de redes sociais e sistemas de recomendação para os usuários.

A principal ferramenta utilizada pelo Neo4j é o *Neo4j Browser* [10]. O *Neo4j Browser* possui uma linha de comando para realizar as pesquisas em CYPHER e então apresentar os resultados em forma de grafos em duas dimensões. Todas as edições e versões do Neo4j possuem o *Neo4j Browser* instalado. A visualização apresentada pela Figura 2.4 ilustra um exemplo de uso.

Um *plugin* muito útil para usar no Neo4j é o *Graph Data Science Library* (GDS) [34]. Há uma variedade de algoritmos para computar métricas relacionadas a grafos, nós ou relações. Esses algoritmos podem prover *insights* em entidades do grafo (centralidades, *ranking*), ou estruturas como comunidades (detecção de comunidades, particionamento de grafos, clusterização). Muitos algoritmos presentes no GDS possuem alta complexidade algorítmica. Há também outros algoritmos otimizados, que utilizam certas estruturas do grafo, memorizam partes exploradas e paralelizam operações.

Com as aplicações e diversidades aqui apresentadas, o Neo4j se torna uma aplicação de NoSQL altamente viável de forma íntegra, sendo direta para a maioria das aplicações que envolvam grafos, e percorrendo uma grande quantidade de melhorias para os desenvolvedores e usuários.

2.4.2 Linguagem CYPHER

Nesta seção é apresentada a linguagem CYPHER utilizada em algumas implementações de banco de dados orientado a grafos e no Neo4j utilizado neste trabalho.

A linguagem CYPHER é utilizada para pesquisa em banco de dados orientado a grafos, de forma muito intuitiva e permite a manipulação eficiente do banco de dados Neo4j. Construída para ser uma linguagem de alto nível, a linguagem CYPHER foi desenvolvida baseada na língua inglesa, assim tornando as queries auto-explicativas [33].

O foco principal da linguagem CYPHER é *o quê* e não *como* será pesquisado. Para chegar a esse objetivo o CYPHER foi criado como uma linguagem declarativa, inspirada em uma série de abordagens e em práticas estabelecidas para realizar suas consultas. Assim, muitas das palavras e comandos utilizados no CYPHER foram inspirados em outras linguagens declarativas de pesquisas em bancos de dados relacionais. Como citado, o CYPHER utiliza algumas estruturas que encontramos também no SQL, tais como:

- *MATCH* - comando para encontrar dados, sendo equivalente ao comando SELECT no MySQL;
- *WHERE* - cláusula condicional, para melhor filtragem dos dados pesquisados;
- *RETURN* - estrutura para especificar o que será retornado;
- *CREATE* - comando para criação de um nó;
- *DELETE* - comando para a exclusão de um nó;
- *SET* - define valores para as propriedades dos nós que os nomeiam;
- *REMOVE* - remove os valores e propriedades dos nós;
- *MERGE* - atualiza ou cria um novo nó, de acordo com sua existência prévia;
- *[:relationship]* - indica o relacionamento a ser pesquisado na consulta.

A seguir serão apresentadas algumas consultas CYPHER para ilustrar exemplos de uso, a saber:

- Pesquisar os nós de um grafo com a *label* (tipo) pessoa;
- Retornar os filmes em que uma determinada pessoa atuou;
- Criar um atributo 'genero' em um nó com a *label* pessoa;
- Excluir as pessoas com idade menor que dezoito anos.

Listing 2.1: Pesquisa em CYPHER que retorna os nós com a *label* Person

```
MATCH (p:Person) RETURN p
```

Listing 2.2: Pesquisa em CYPHER que retorna os filmes que uma pessoa atuou

```
MATCH (:Person {name: 'Tom Hanks'}) -[:ACTED_IN]->
(movie:Movie) RETURN movie
```

Listing 2.3: Pesquisa em CYPHER que cria o atributo 'genero' aos nós com a *label* Person

```
MATCH (p:Person) MERGE p.genero='F'
```

Listing 2.4: Pesquisa em CYPHER que deleta as pessoas com a idade menor de 18 anos

```
MATCH (p:Person) DELETE p WHERE p.idade < 18
```

2.5 Arquitetura MVC

O *Model-View-Controller* (MVC) ou Modelo-Visão-Controlador [35] é um padrão de projeto (*design pattern*), também conhecido como arquitetura, frequentemente utilizado para aplicações que necessitam manter múltiplas visões de um mesmo conjunto de dados. O MVC provê uma separação clara de objetos em três partes: modelo, visão e controle. Por causa dessa separação, múltiplas visões e controles podem interagir com um mesmo modelo. E novos tipos de visões e controles podem interagir com o modelo sem necessidade de realizar alterações no modelo definido no projeto. Nesse sentido, tem-se as definções de cada camada aplicadas a solução:

- Modelo (*Model*) - O modelo define quais dados a solução deve conter. Se o estado desses dados for alterado, o modelo geralmente notificará a Visão (para que a Visão possa mudar conforme necessário) e, às vezes, o controlador (se for necessária uma lógica diferente para controlar a camada de Visão atualizada);

- Visão (*View*) - responsável por gerenciar as informações inseridas pelo usuário (login, consultas, formulários) e repassar para a camada *Controller*. Também é responsável por apresentar os dados após ser atualizada pela camada Controladora;

- Controlador (*Controller*) - responsável pela interação com a camada *Model*, permitindo após as regras de negócio serem aplicadas, o envio da requisição para acessar o banco de dados. O *Controller* também é responsável por interagir com a camada *View*, tanto validando os dados vindos desta camada como atualizando-a para apresentação ao usuário.

A Figura 2.5 apresenta o fluxo de funcionamento da arquitetura MVC, incluindo cada camada e suas atribuições dentro da implementação de uma solução genérica.

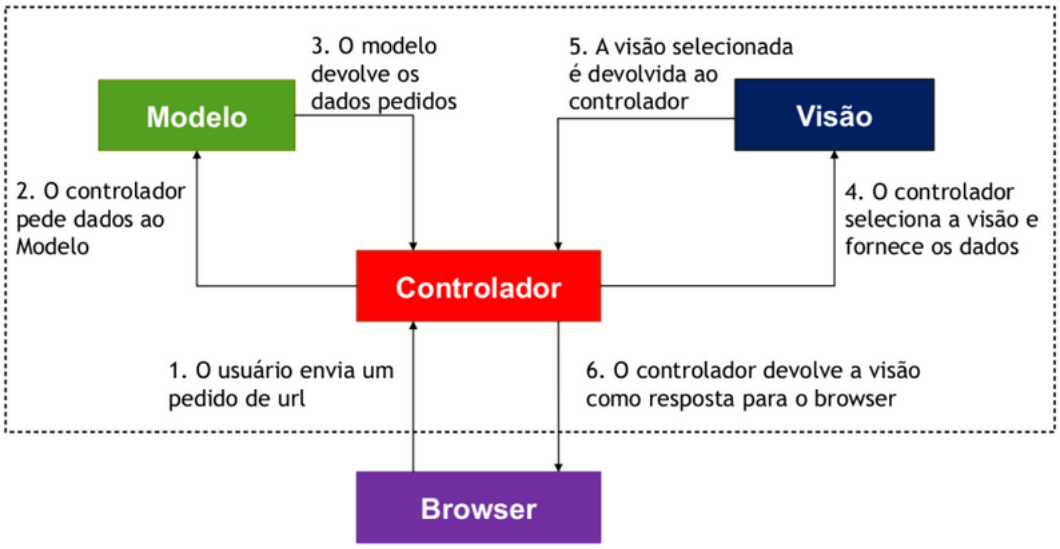


Figura 2.5: Esquema da arquitetura MVC (Fonte: [4]).

Este capítulo apresentou a fundamentação teórica realizada para auxiliar no entendimento da solução proposta. Na sequência, o Capítulo 3 apresenta uma descrição da revisão de literatura.

Capítulo 3

Revisão da Literatura

Visando responder a primeira questão de pesquisa definida na Seção 1.3 do Capítulo 1 - Quais são as soluções teóricas e de artefatos implementados existentes na literatura para visualização 3D de bases de dados orientadas a grafo utilizando Neo4j? - foi realizado uma revisão da literatura com o foco em técnicas, funções, artefatos e bibliotecas relacionadas à área de visualização de grafos.

Durante a realização da revisão foram encontrados artigos que apresentam revisões de trabalhos (*reviews*) sendo estes interessantes para a identificação de trabalhos relacionados. Neste sentido, a revisão de Chen et al. (2019) [36] apresenta um modelo de visualização de grafos dividido em três etapas (i) modelagem de relações, (ii) técnicas de visualização, (iii) funções de simplificação e interação do grafo, conforme a Figura 3.1.

Note que a primeira etapa de modelagem de relações inclui o pré-processamento (I) dos dados não tratados e transformados em ficha de dados (*datasheets*) na modelagem (II). Após a modelagem será feita a geração do grafo que representa as relações entre as entidades que formam a rede na fase de mineração (III). A Etapa II define as técnicas empregadas para visualização do grafo gerado. Nesta etapa são apresentadas sete técnicas, incluindo nós e links, matriz de adjacências, hipergrafo, diagrama de fluxo, grafo com informações geoespaciais, multi-atributos e preenchimento de espaços. É possível utilizar diversas técnicas em conjunto com o objetivo de alcançar proveito de cada uma. A Etapa III apresenta funções de simplificação e interação para auxiliar os usuários na análise de grafos grandes e complexos, desde uma visão geral até análises mais detalhadas. As técnicas empregadas incluem clusterização de nós, filtragem do grafo, empacotamento de links, transformação baseada na topologia do grafo e limitação de dimensões. Novamente, é possível utilizar diversas técnicas em conjunto dependendo do objetivo de simplificação e interação definidos no modelo de solução.

O modelo proposto por Chen et al.(2019) [36] será utilizado para classificar os trabalhos encontrados na revisão da literatura. A revisão foi realizada durante o mês de outubro de

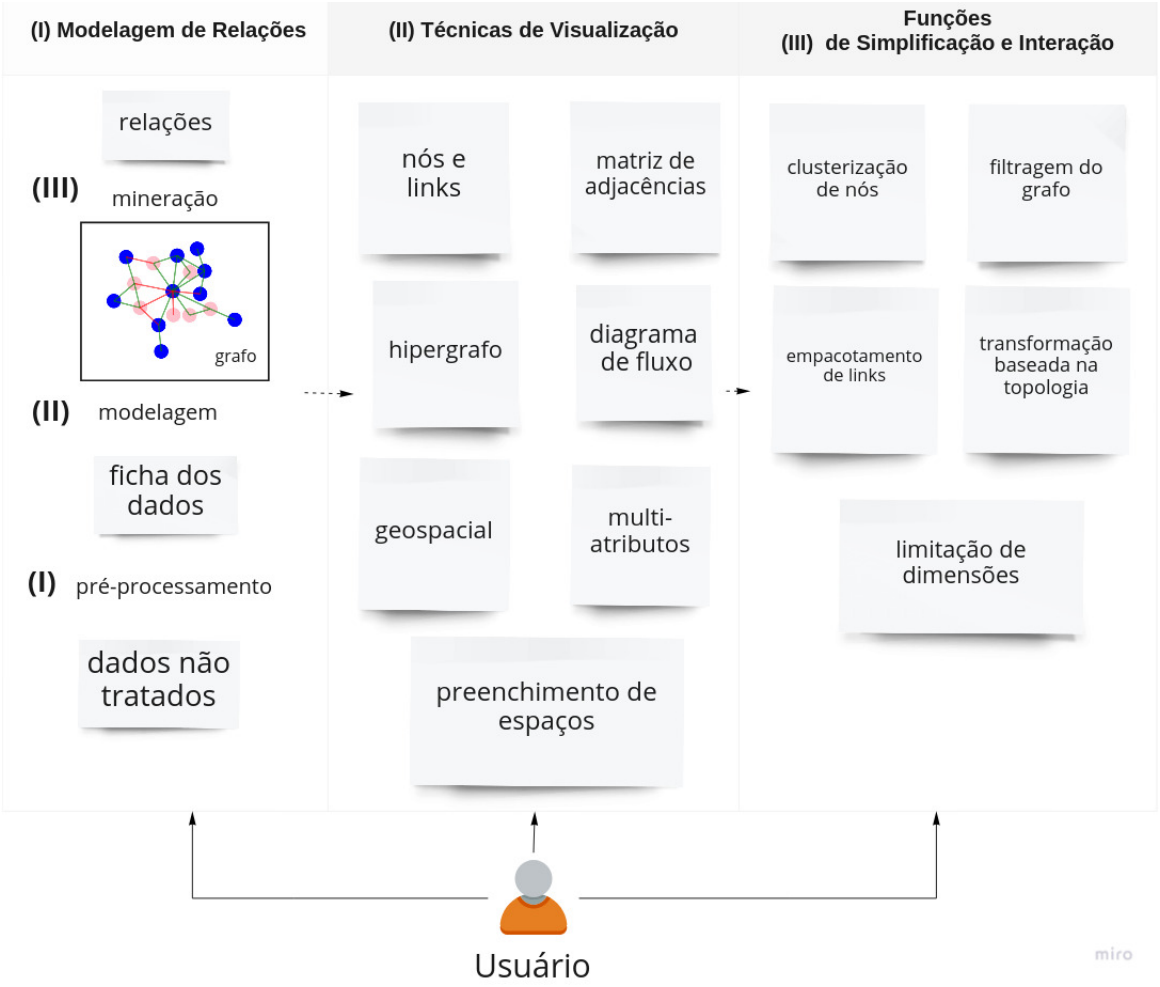


Figura 3.1: Modelo proposto por Chen et al. (2019) (adaptado e traduzido pelo autor deste trabalho).

2021 utilizando a plataforma Google Acadêmico[1] com as *strings* de busca *Visualization, Neo4j, Social Networks*. A plataforma Google Acadêmico se apresentou interessante para a revisão da literatura uma vez que inclui trabalhos de diversos repositórios importantes na área da Computação, tais como, Springer, Elsevier, IEEE, ACM, *Multidisciplinary Digital Publishing Institute* (MDPI), *American National Institutes of Health* (NIH), Mit-press, Scitepress, Easychair, Researchgate, core.ac.uk e arxiv.org. Filtrando o período de publicação de 2015 a 2021. Analisando o título e abstract dos trabalhos foram selecionados os mais relacionados ao objetivo deste trabalho, que necessariamente incluíssem técnicas, funções e artefatos de visualizações de grafos utilizando banco de dados Neo4j. Neste contexto, foram selecionados 19 trabalhos os quais serão apresentados na Tabela 3.1.

[1] `https://scholar.google.com.br/?hl=pt`

Tabela 3.1: Trabalhos selecionados durante a revisão literatura

	Referência	**Técnicas de Visualização**	**Funções Exploradas**	**Domínio de Aplicação**
Técnicas e Funções de Visualização	Bludau, Dörk e Tominski (2021) [5]	nós e links 2D	empacotamento de links	Geral
	Rahman e Karim (2016) [37]	nós e links, esférico, clusterizado 2D	filtragem do grafo, clusterização de nós	Redes Sociais
	Tallat et al. (2019)[38]	nós e links, hipergrafo 2D	filtragem do grafo	Medicina
	Farooq et al. (2018)[39]	nós e links, hipergrafo 2D	filtragem do grafo	Criminal
	Kumar e Teo (2018)[40]	nós e links 2D	filtragem do grafo	Eng. Civil
	Summer et al.(2015)[41]	nós e links 2D	filtragem do grafo	Geral
Artefatos de Visualização	Antweiler et al. (2021)[17]	nós e links 2D	filtragem do grafo	Medicina
	Yokoyama et al.(2021)[42]	nós e links 2D	filtragem grafo, clusterização nós	RH
	Garcia et al. (2021)[43]	nós e links 2D	filtragem do grafo	Medicina
	Jo et al. (2021)[44]	nós e links 2D	filtragem do grafo	Redes Sociais
	Keywan et al (2021)[45]	nós e links 2D	filtragem do grafo	Medicina
	Syed et al. (2021)[46]	nós e links, matriz, preenchimento espaços 2D	filtragem do grafo	Medicina
	Carnaz, Nogueira e Antunes (2021)[47]	nós e links 2D	filtragem do grafo, pacotamento de links	Criminal
	Kohalmi,Sheils e Oprea (2020)[48]	nós e links 2D	filtragem grafo, pacotamento de links	Medicina
	Muller et al. (2018)[49]	hipergrafo, preenchimento de espaços 2D	filtragem do grafo	Desenv. de Software
	Huhne et al. (2018)[14]	nós e links 3D	filtragem do grafo, clusterização de nós	Medicina
	Kerzner et al. (2017)[50]	nós e links, preenchimento de espaços 2D	filtragem do grafo	Aviação
	Partl et al. (2016)[51]	nós e links, preenchimento de espaços 2D	filtragem do grafo, clusterização de nós	Redes Sociais
	Rodrigues et al. (2015)[52]	nós e links 2D	filtragem do grafo, clusterização de nós	Geral

Na sequência serão apresentados os principais trabalhos selecionados na literatura agrupados por técnicas e funções de visualização de grafos (Seção 3.1) e artefatos de visualização (Seção 3.2). Observou-se que os artefatos de visualização estudados incluem as etapas defindas por [36] e apresentados na Figura 3.1.

3.1 Visualização de Grafos: Técnicas e Funções

O trabalho de Bludau, Dörk e Tominski [5] apresenta o *unfolding edges* para expandir a capacidade de exploração visual interativa das arestas de redes em formato de nós e *links*. A visualização de atributos de arestas em uma rede muitas vezes é limitada por problemas de oclusão e espaço disponível em tela, não explorando suficientemente o potencial de interação. A abordagem proposta pelos autores foi inspirada por *focus+next* apresentada em [53]. Para aprimorar as possibilidades de visualização de vários atributos de arestas, as capacidades interativas são focadas ao *unfolding* dessas arestas sobre demanda. Com isso, a complexidade geral da visualização pode ser mantida baixa, sem desordenar o fluxo de análise. A abordagem é apresentada em duas variantes: (a) ao selecionar uma aresta entre dois nós da rede um gatilho é disparado e então o *zoom* é aplicado a visualização dessa aresta apresentando todas as outras informações existentes na aresta; (b) o grafo selecionado é rotacionado na posição horizontal e então a aresta sofre o *unfold*, apresentando informações dentro desse espaço gerado por um espaço temporal. A Figura 3.2 apresenta as duas variantes da técnica *unfolding edges*.

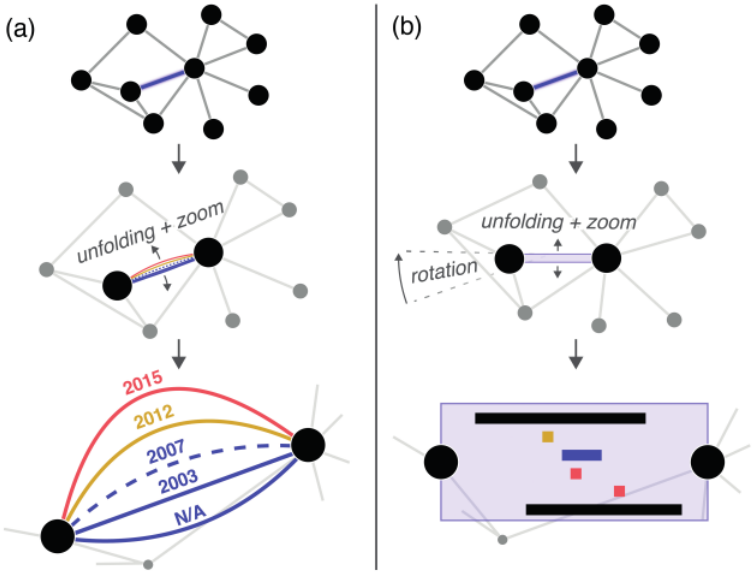


Figura 3.2: Variantes do *unfolding edges* proposto por [5].

As primeiras entrevistas e discussões com pessoas da área de análise de redes apontaram resultados promissores, porém validações formais estão pendentes. A técnica do *Unfolding Edges* deixou algumas questões em aberto: Como facilitar a interação com arestas? Como prenunciar detalhes realçáveis? Como poderia a visualização *on-demand* ser usada para tarefas globais?

No trabalho de Rahman e Karim [37] é abordado um estudo comparativo de técnicas de visualização e análises em redes sociais. Três técnicas de visualização foram consideradas: (i) nós e links, (ii) esférico e (iii) clusterizado. Como conjunto de dados é utilizado um arquivo de *log* a respeito de um dia de funcionamento de um servidor de *e-mail* formando uma rede com 369 vértices e 905 arestas. O estudo consistiu em analisar os seguintes requisitos:

- tipo de visualização - 2D, 3D, hierárquico;
- critério da visualização - apresenta toda a rede ou parte da rede;
- grafo de saída - se é possível ter interpretações claras sobre o grafo apresentado;
- número de nós manipulados - quantidade de vértices e arestas;
- complexidade - quanto tempo um método leva para completar a tarefa da visualização.

A Tabela 3.2 apresenta os resultados comparativos feitos entre as três técnicas de visualização, considerando os requisitos citados.

Tabela 3.2: Análise entre as três técnicas de visualização

Critério	**Nós e links**	**Esférico**	**Clusterizado**
Tipo de visualização	Visualização 2D no plano	Visualização 3D no plano da esfera	Visão hierárquica da rede
Critério de visualização	Oferece uma visão geral da rede	Entrega as relações entre diferentes grupos da rede	Mostra a estrutura clusterizada da rede de acordo com um atributo
Grafo gerado	Não oferece informação clara	Visualização clara dos grupos na rede	O melhor para visualizar grandes redes e grupos de relações
Número de nós	Adequado para grafos de tamanho entre 50-100 vértices	Adequado para um número pequeno de grupos	Visualiza grandes números de nós de forma eficiente e compacta
Complexidade	O(n^3) - cúbica	O(n^2) - quadrática	O(n^2) - quadrática

A partir da análise dos resultados os autores concluiram que a técnica de nós e *links* é usada para visualizar a rede total estrutura (visão geral) e é simples e fácil de implementar. Essa técnica é adequada para grafos de tamanho médio (50-100 vértices). Os algoritmos utilizados para disposição da rede não exigem forte conhecimento sobre a teoria de grafos.

Esta técnica possui complexidade cúbica. O *layout* esférico é usado para revelar padrões de comunicação, relações entre diferentes grupos. Os nós são distribuídos de maneira mais uniforme na superfície da esfera, em vez de colapsarem no centro da rede. Também é adequado para grafos de tamanho médio com complexidade quadrática. Já a técnica de clusterização permite visualização de uma grande quantidade de dados de forma eficiente e compacta. Oferece uma visão hierárquica e relações de grupo. Portanto, entre os três métodos é o melhor para lidar com grandes redes e grupos de relacionamentos possuindo tempo de execução quadrático.

Em Tallat et al. [38] é conduzida uma comparação de visualizações e análises de dados biológicos utilizando diferentes ferramentas e técnicas. São explorados duas ferramentas para visualização - Neo4j e o CIRCOS [54]. O CIRCOS é um software de visualização que utiliza a técnica circular de dados. Com as duas ferramentas foram explorados procedimentos para análise dos dados. O *dataset* denominado MYBIOGRID foi formado a partir da coleta de oito fontes contendo dados biológicos heterogêneos.

As funções exploradas com o Neo4j foram:

- visualização - explorando a visão em forma de grafos das interações genes-genes e genes-proteínas.
- análises dos caminhos presentes na rede - consultas para determinar o caminho mais curto entre dois genes indicando a proximidade de interação entre eles e o triângulo no grafo, indicando a quantidade de *clusters* existentes na rede.
- análises das medidas de centralidade - calculado o grau de um nó em relação a um símbolo de proteína, que representa o número de proteínas ligadas a determinado gene.

As relações entre genes e proteínas presentes no conjunto de dados do MYBIOGRID foram carregadas no CIRCOS e geradas as visualizações de relações de peso molecular, coordenadas gênicas, classe molecular e método de interação. Os resultados apontaram que o Neo4j e o CIRCOS atuam de forma diferente, com parâmetros distintos. Cada ferramenta possui características candidatas para diversas análises. Em uma rede com muitos relacionamentos a visualização no Neo4j fica prejudicada, porém o uso de algoritmos para cálculo de centralidades mostrou-se favorável. Novos algoritmos podem ser incluídos ao Neo4j no sentido de permitir mais cálculos. Já o CIRCOS provê um maior entendimento dos relacionamentos da rede potencializando a visão geral de grafos densos.

O trabalho de Farooq et al. [39] foca na visualização e análise de uma rede de pessoas influentes em uma rede social, através do cálculo de diversas medidas de centralidade. São exploradas as métricas de *Degree, Eigenvector Centrality (EC), Closeness Centrality*

(CC), Betweenness Centrality (BC), Clustering Coefficient (CC) e Page Rank. O conjunto de dados utilizado é de uma rede de ações terroristas na Indonésia. Com o cálculo das medidas de centralidade foi possível identificar visualmente e quantitativamente nós influentes na rede, ou seja as pessoas que tem influência.

Em Kumar e Teo [40] é discutida a dificuldade de usuários procurarem informações relevantes dentro de documentos do tipo *Construction Operations Building Information Exchange* (COBie). Tais documentos são utilizados no ramo de construção civil. Vários dados de partes distintas de uma construção são coletados e juntos formam um único documento. Os autores discutem os desafios associados ao aspecto da visualização de documentos COBie e propoêm uma solução para mitigá-los. O artigo demonstra que a técnica baseada em uma rede de nós e links pode ser usada para representar de forma efetiva as relações presentes em documentos COBie. Mas do que isso, a pesquisa também evidencia que as mudanças ao longo do tempo nesses documentos podem ajudar a entender as mudanças envolvidas durante a execução de obras na construção civil.

O trabalho de Summer et al. [41] apresenta o *cyNeo4j*, um *framework* que integra o Neo4j ao sistema do *Cytoscape*, uma biblioteca de visualização de redes em formato *nós e links*. Os pesquisadores exploraram dois algoritmos: (i) *NetworkAnalyzer*: computa medidas de centralidade *betweenness centrality, average paths, shortest paths*, (ii) *ForceAtlas2*: algoritmo usado para buscar a melhor disposição dos grafos, com o intuito de evitar sobreposições e/ou oclusões. Em uma rede com 4.436 vértices and 93.286 arestas o *NetworkAnalyzer* implementado no *cyNeo4j* atingiu performance em segundos quatro vezes melhor do que a implementação feita pelo *Cytoscape*. Já o *ForceAtlas2* foi executado via Neo4j e visualizado pelo *Cytoscape*, permitindo a organização dos componentes da rede gerada com menos sobreposições dos nós e relações.

Conforme descrito, os trabalhos de técnicas e funções de visualização majoritariamente ujtilizam nós e links 2D com filtragem do grafo, sendo que somente dois deles foram aplicados em domínio geral. Na sequência serão apresentados os trabalhos com foto em artefatos de visualização.

3.2 Artefatos de Visualização

Em Antweiler et al. [17] é apresentado o uso de aprendizado de máquina para predição de *links* e visualização de clusters na rede de infecção por SARS-CoV2 (COVID-19) de pessoas da cidade de Colônia, na Alemanha. O conjunto de dados incluiu 44.634 pessoas, dos quais 11.652 são os *index cases* que testaram positivo para a doença. Para 7.846 (67.3%) desses casos índice, a origem da infecção é desconhecida ou não está presente no conjunto de dados. Foi adicionado também 40.000 amostras negativas de pares de

pessoas, as quais não compartilham conexões dentro do conjunto original. O modelo pode facilmente ser escalado para uma quantidade maior de nós e gerar classificação de resultados do conjunto de 44.000 pacientes usado nos experimentos, pois demorou pouco mais que milissegundos para executar. Foi possível através do sistema de predição de *links* calcular rotas de infecção de pessoas ao longo do tempo. A visualização dos dados em forma de grafos proporcionou mapear a distribuição do vírus entre os *clusters* da rede, o que é muito importante especialmente para identificar novas mutações do SARS-CoV-2.

O sistema denominado *Polarizing Attributes for Network Analysis of Correlation on Entities Association* (Panacea) foi desenvolvida por Yokoyama et al. [42] para integrar um modelo de gráfico variável no tempo e visualização dinâmica de grafos para dados tabulares heterogêneos. Panacea foi proposto para ser utilizado por especialistas de recursos humanos (RH) para inspecionar interativamente as relações entre dois atributos de candidatos potenciais a funcionários ao longo de vários anos. Os autores demonstraram a usabilidade do Panacea com exemplos representativos para encontrar tendências ocultas em conjuntos de dados do mundo real e discutiram o *feedback* de especialistas de RH obtidos durante o desenvolvimento do Panacea. O sistema Panacea permite que os especialistas explorem visualmente o recrutamento anual de recém graduados, se apresentando útil para os profissionais da área de RH.

Em Garcia et al. [43] é desenvolvido um artefato *online* denominado DisMaNET para visualização do mapamento de vocábulos de doenças. O conjunto de dados utilizado contempla oito repositórios de área de medicina totalizando mais de 500.000 nós e 900.000 arestas. O mapeamento dos vocabulários é feito pelo cálculo da métrica *relevance score*, que indica o nível de similaridade entre dois vocábulos médicos. Partindo de uma lista de doenças selecionadas, por exemplo, AIDS (*Acquired Immunodeficiency Syndrome*) e Lupus (*Systemic Lupus Erythematosus*) é possível visualizar em forma de grafos os sintomas em comum entre essas duas doenças. O caminho inverso também é possível, ou seja, gerar visualizações de doenças possíveis a partir de uma lista de sintomas informados. Foi realizado uma análise comparativa entre o DisMaNET e outros artefatos de mapeamento de vocabulários médicos. O DisMaNET foi capaz de obter mais de 80% dos mapemamentos de vocábulos presentes no MonDO[2] e identificando também novos mapementos em relação ao projeto DisGeNET[3].

O artefato desenvolvido por Jo et al. [44] denominado OLGAVis trata da visualização de dados dos seguintes repositórios bibliográficos digitais: DBLP, ACM, Microsoft Academic Graph (MAG)[4]. Partindo da modelagem desses dados em um banco de dados Neo4j, a ferramenta faz uma exploração intuitiva com um volume de dados incluindo 630.000

[2]http://obofoundry.org/ontology/mondo
[3]https://www.disgenet.org/
[4]https://makg.org/

artigos. Sumarização de relacionamentos, criação de sub-grafos e cálculos de agregações são funções exploradas neste trabalho. As análises das redes sociais formadas propiciaram resultados convenientes, contribuindo para o desenvolvimento de pesquisa de vários pesquisadores, por exemplo, uma análise para visualizar a rede de colaboração formada entre dois pesquisadores co-autores de um mesmo artigo.

O artefato apresentado por Keywan et al.[45] denominado *KnetMiner* tem foco no domínio médico, com a visualização de redes em formato nós e links de interações entre genes. A plataforma proposta pelos autores disponibiliza diversos algoritmos para gerar redes gênicas e interatividade com a visualização gerada. Como estudo de caso foi utilizado o contexto biológico de plantas, sendo possivel identificar genes diversos e suas interações. O *KnetMiner* foi capaz de auxiliar tanto leigos como pesquisadores da área médica a entender melhor informações antes presentes em forma textual de uma forma diferente e interativa.

Em Syed et al. [46] é apresentado o *LinkedImm*, um artefato para integração e visualização de dados médicos, relativos a imunologia. Partindo da coleta de dados em diversas fontes médicas foi formada uma rede com 37.992 nós e 72.229 relações. Os autores utilizaram técnicas de processamento de linguagem natural (*Natural Language Processing* - NLP) para simplificar as manipulações da base de dados armazenada em Neo4j. Os resultados obtidos apontaram que a manipulação e visualização de dados complexos se tornou mais viável e flexível.

O trabalho de Carnaz, Nogueira e Antunes [47] apresenta o *SEMCrime*, um artefato usado para extrair, classificar e visualizar nomes, entidades e relações em documentos e boletins de crimes em cidades Portuguesas. Na etapa de extração dos dados os pesquisadores utilizaram um método denominado *5W1H - Who, What, Why, Where, When, and How*, que utiliza *Semantic Role Labelling* (SRL) e Named Entity Recognition (NER). Após a etapa de extração o processamento dos dados a cerca dos crimes se dá pelo uso de NLP, formando assim as relações semânticas. Por fim, o Neo4j é utilizado para armazenamento e visualização. Na avaliação de NER foram verificados a Precisão (P), Revocação (R) e *F-Measure* atingidos pelo *SEMCrime* em três grupos de documentos criminalísticos (*Criminal News, PGdLisboa News, Criminal Investigation reports*). A média alcançada pelo *SEMCrime* nesses três grupos ficou com P = 0.808, R = 0.722 e *F-Measure* = 0.733. O método 5W1H foi avaliado obtendo P = 0.732, R = 0.634 e *F-Measure* = 0.653. No estudo de caso o *SEMCrime* foi comparado com outra aplicação denominada IBM™i2. A avaliação do estudo de caso obteve resultados semelhantes em análises feitas por investigadores de polícia com ambas as aplicações. No geral, a abordagem dos pesquisadores funciona para representar um agrupamento de relações e entidades que povoam os documentos criminais utilizando consulta e visualização dos resultados na forma de nós e

links.

Em Kőhalmi, Sheils e Oprea [48] é apresentado o artefato *SmartGraph*, uma plataforma para investigação de relações entre compostos de remédios e proteínas. O conjunto de dados utilizado possui 1.732.553 componentes e 4.583 proteínas. Além do cálculo de *shortest path*, o *SmartGraph* realiza a predição de *links*, i.e., relações entre os nós que formam rede através de métodos de similaridade de atributos. Os autores demonstraram a usabilidade do *SmartGraph* com exemplos de interações entre vários componentes e proteínas, bem como a predição de relações entre eles. O uso do Neo4j permitiu a fácil integração de informações médicas através de consultas intutitivas e a integração com a biblioteca de visualização D3[5]

O trabalho de Muller et al. [49] tem foco nos desafios da aquisição, análise e visualização de dados a cerca de artefatos de sistemas de *software*. É proposto um *stack* de ferramentas código aberto para resolver esses desafios. Foi utilizado o *jQAssistant*, uma ferramenta extensível que verifica diferentes tipos de artefatos de software e armazena os dados em um banco Neo4j. Além do armazenamento de dados, o Neo4j foi utilizado para realizar filtros, cálculos e funções para que sejam geradas visualizações utilizando as bibliotecas *React* e *D3*.

O artefato proposto funciona com as seguintes etapas:

1. o *jQAssistant* coleta os dados acerca dos artefatos do *software*, incluindo o código fonte, resultados de testes, resultados de análise de código e *logs* para controle de versionamento;

2. a unificação dos dados coletados é feita e eles são armazenados no Neo4j;

3. funções e filtros são aplicados ao conjunto de dados permitindo visualização dos dados em diversas formas.

Na Etapa 3 são realizados:

- análise de dependência - retorna o grau de dependência que uma parte ou módulo do *software* possui com outro, indicando o nível de acoplamento e coesão de um sistema. A visualização da análise de dependências é dada por um diagrama circular.

- análise de *hotspots* - permite avaliar o risco de um *software*. Os *hotspots* representam partes do código que muda frequentemente. Eles geralmente são candidatos a melhorias ou refatoração. Para visualizar esses pontos, dados temporais são necessários, como os *logs* de ferramentas de versionamento de código (*GitHub, GitLab*). A visualização é apresentada na forma de mapa de calor.

[5]https://d3js.org/

- análise de cobertura de testes - permite avaliar a qualidade de um *software*. Para visualizar a cobertura do teste são utilizados dados comportamentais e estruturais. A visualização é dada por meio do *treemapping*.

- análise de código estático - outro meio de avaliar a qualidade de um *software*. Aqui, o código-fonte é verificado contra regras predefinidas. Melhores práticas, estilo de código e documentação são analisadas. A visualização dessa análise é feita por meio do mapa de radar.

Em Huhne et al. [14] é apresentado o *Javascript AgeFactDB Network-viewer* (JANet) que foca na visualização em três dimensões de relações entre genes, componentes químicos e outros fatores que influenciam no envelhecimento. Os autores utilizaram os dados do repositório *JenAge Ageing Factor Database* (AgeFactDB)[6] que inclui o conjunto de 16599 fatores de envelhecimento e 9611 outras observações. Diferentes tipos de redes e estratégias de visualização são propostas, partindo de uma interface customizada. É possível analisar uma lista de genes em combinação com o AgeFactDB em uma visualização interativa dos resultados. Com o JANet foi possível ter maiores *insights* em padrões de fatores de envelhecimento que não são facilmente acessíveis. Esses conceitos são podem ser utilizados em muitos outros campos de pesquisa.

O trabalho de Kerzner et al. [50] apresenta o Graffinity, uma ferramenta *open-source* para análise do fluxo de voôs de Estados Norte-Americanos. O conjunto de dados utilizado consiste de 308 aeroportos (nós) e 13.000 voôs (arestas). O objetivo da análise se dá por duas técnicas de visualização que funcionam em conjunto para resumir a conectividade das sub-redes geradas: (i) matriz de conectividade - utiliza a técnica de matriz de adjacências para apresentar a quantidade total de caminhos formados entre aeroportos dos Estados de origem e destino, bem como a quantidade de empresas de aviação presentes nesses caminhos; (ii) tabela de nós intermediários - identifica a posição dos aeroportos que estão entre os caminhos dos Estados de origem e destino. Como visualização suplementar, partindo da matriz de conectividade gerada é possível selecionar e visualizar sob demanda os caminhos na forma de grafos em duas dimensões. Os resultados dos testes realizados pelo Graffinity apontam que as duas técnicas apresentadas para resumir a conectividade das sub-redes geradas funcionam em tempo linear para matrizes com 100.000 caminhos. Em relação aos resultados qualitativos de pessoas do ramo de aviação que utilizaram o Graffinity pelo periodo de seis horas de entrevistas informais e demonstrações, afirmaram que "Figuras que eu nem imaginei que fossem possiveis de serem geradas", "É exatamente o que eu precisava".

[6] http://agefactdb.jenage.de/

O trabalho de Partl [51] apresenta o *PathFinder*, um artefato com foco na análise dos caminhos de um grafo no formato nós e *links*. Os autores mostram que buscando os caminhos da rede pode-se verificar a escalabilidade a respeito da visualização, permitindo analistas explorar de forma clara grandes redes. Uma série de funções são exploradas, tais como listas de propriedades dos caminhos e ranqueamento de nós mais conectados em um contexto topológico. Os autores utilizaram cenários de redes de co-autoria e biológico como exemplos de demonstração.

O trabalho de Rodrigues et al. [52] apresenta o artefato denominado *GMine*, um sistema escalável e interativo de visualização e mineração de grafos no formato nós e *links*. Os pesquisadores dão foco em dois problemas: a performance das interações em grafos muito grandes, e a capacidade de leitura e análise da visualização apresentada devido a limitações da tela. O *GMine* utiliza o conjunto de dados proveniente do DBLP formando uma rede de coautoria com 315.688 nós e 1.659.853 arestas. As técnicas utilizadas pelos autores consistem no uso de sumarização e multi-resolução, para que seja criada a clusterização dos nós, particionando a rede em comunidades distintas e armazenando-as em uma estrutura hierarquizada de árvore, denominada *G-Tree*. Foram calculadas ao todo 626 comunidades distintas com uma média de 500 nós por comunidade. Com o *GMine* foi possível visualizar grandes grafos permitindo a navegação por diferentes comunidades de uma forma hierarquizada. A escalabilidade se deu pelo fato de pequenas partes do grafo serem processadas uma a uma, ao invés de todo o grafo.

Bibliotecas

Existem diversas bibliotecas para a visualização de redes baseadas em grafos. Nesta seção são apresentadas seis bibliotecas voltadas para aplicações em páginas Web com uma pequena ilustração de uso. Para as ilustrações foi utilizada uma rede de coautoria dos pesquisadores vinculados ao Programa de Pós-Graduação em Informática da Universidade de Brasília (PPGI/UnB). O conjunto de dados utilizado contempla os 31 pesquisadores atualmente vinculados à UnB, totalizando uma de rede com 7.617 nós e 92.287 relações. O texto a seguir inclui aspectos de avaliação das bilbiotecas baseados na experiência do autor desta monografia.

- a biblioteca *Vis* [55] foi projetada para ser fácil de usar, permitindo a manipulação e interação de quantidades significativas de dados. O formato de visualização baseado em grafos do *Vis* oferece suporte a formas, estilos, cores, tamanhos, imagens e muito mais personalizações da rede. A visualização da rede funciona em qualquer navegador Web para até alguns milhares de nós e arestas. Há a possibilidade de uso

de algoritmos para organização espacial da rede, como o *Barnes-Hut* [56] e *ForceAtlas2* [57], com seus parâmetros editáveis. Para lidar com uma quantidade maior de nós, o *Vis* tem suporte para *clustering*. Não há uma conexão integrada ao banco de dados Neo4j, necessitando uma implementação externa. A Figura 3.3 ilustra o uso do *Vis* com uso do algoritmo *ForceAtlas2* para organização dos componentes da rede.

Figura 3.3: Vis apresentando a rede de coautoria do PPGI/UnB.

- a biblioteca *Neovis* [58] foi projetada para combinar a visualização da biblioteca *Vis* e o banco de dados Neo4j. A conexão é simples e direta através da construção de modelo modelo de grafo de propriedades do Neo4j com um formato de dados que o Neovis alinha com o banco de dados. A personalização de cores e formas de nós e relacionamentos da rede baseado em rótulos e propriedades são definidos em um único objeto de configuração. O Neovis pode ser usado sem utilizar a linguagem *Cypher* para as *queries* e com JavaScript mínimo para integração em um projeto. O algoritmo *Barnes-Hut* é utilizado pelo *Neovis*, porém os parâmetros desse algoritmo não são editáveis. A Figura 3.4 apresenta o uso do *Barnes-Hut*.

Figura 3.4: Neovis apresentando autor da rede social PPGI/UnB.

- o *Popoto* [59] é baseado na linguagem *JavaScript* em conjunto com outra biblioteca de visualização de redes D3 [60]. O *Popoto* funciona como um *query builder* visual para *queries* no Neo4j. Ou seja, ao selecionar uma rede o *Popoto* retorna como resultado a consulta em linguagem *Cypher* da rede projetada. Com o Popoto é possível focar na funcionalidade de filtragem do grafo, devido ao caminho que gera a consulta *CYPHER* partir do grafo projetado. A Figura 3.5 apresenta um exemplo de filtragem de grafo.

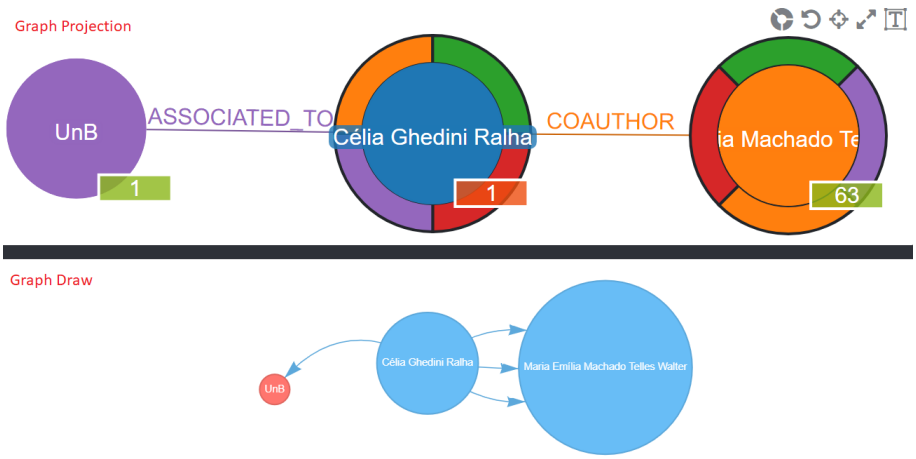


Figura 3.5: Popoto apresentando a rede de coautoria de um autor do PPGI/UnB.

- enquanto algumas bibliotecas tem como foco incluir todos os recursos em um pacote, o *Sigma* [61] apresenta um ambiente altamente extensível que permite adicionar bibliotecas de extensão ou *plugins* para fornecer recursos adicionais. Esta biblioteca suporta dados exportados nos formatos *JSON* ou *GEXF*. Partindo de uma visualização básica da rede com pouca customização de cores e formas tem a possibilidade de adicionar módulos personalizados para funcionalidades específicas, como customização da rede, uso de algoritmos para organização do *layout* e a realização de cálculos de medidas de centralidade. O *Sigma* utiliza o algoritmo *ForceAtlas2* para a organização de sua rede, com seus parâmetros editáveis. A Figura 3.6 apresenta o uso do *ForceAtlas2*.

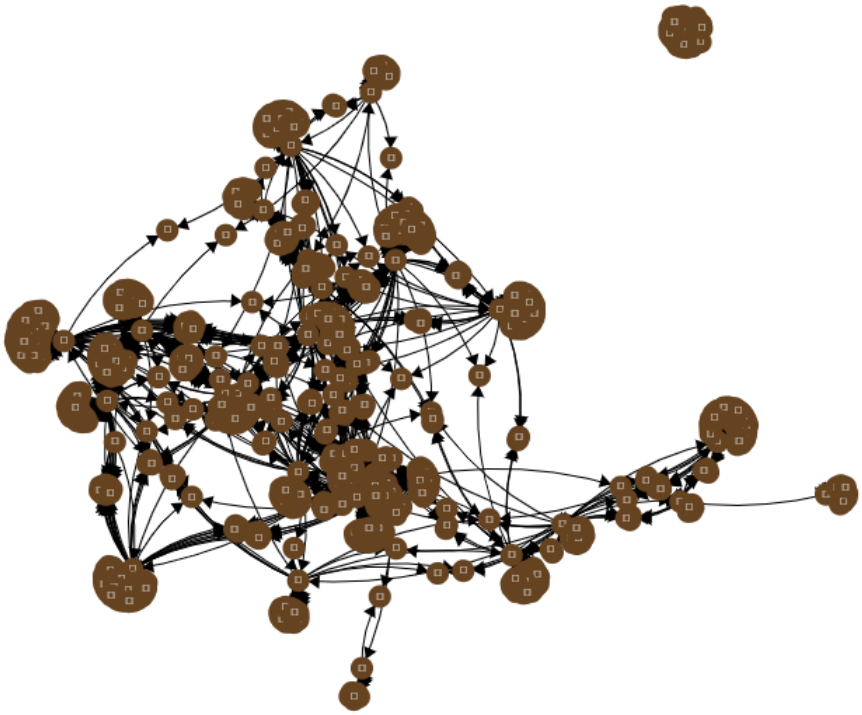


Figura 3.6: Sigma apresentando a rede de coautoria de um autor do PPGI/UnB.

- o *Vivagraph* [62] foi construído com o foco para tratar diferentes tipos de algoritmos de *layout* para organizar nós e arestas, contando com pouca customização inicial já implementada. As customizações da rede estão disponíveis por meio de modificações CSS e bibliotecas de extensão. Ele também pode rastrear alterações na estrutura da rede e atualizar a visualização correspondente. O Sigma utiliza o algoritmo *ForceAtlas2* para a organização de sua rede, com seus parâmetros editáveis. A Figura 3.7 apresenta o uso do *ForceAtlas2*.

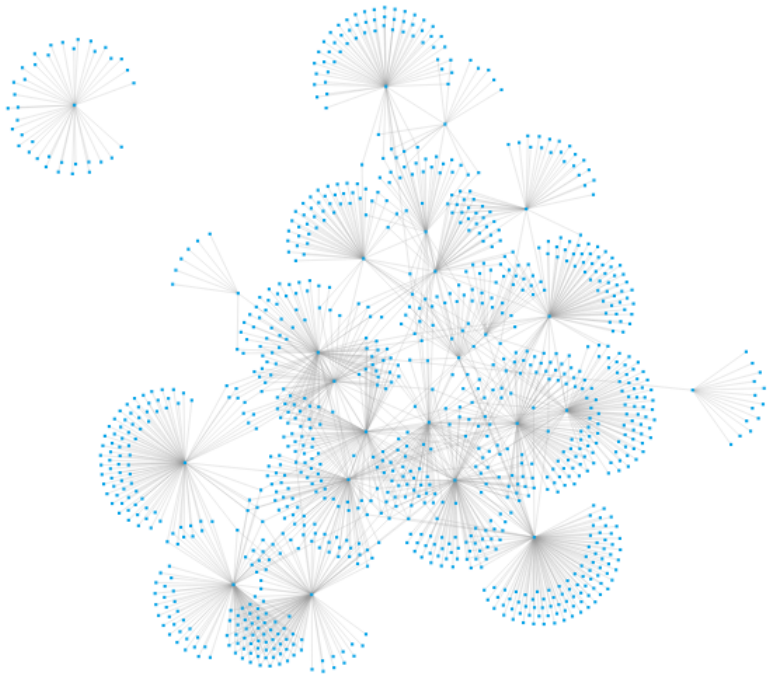


Figura 3.7: Vivagraph apresentando a rede de coautoria do PPGI/UnB.

- o *3d-force-graph* [63] tem foco na visualização em três dimensões de redes baseadas em grafos, permitindo uma ampla customização de cores e formas de vértices e arestas da rede. O algoritmo *d3-force-3d* é utilizado para a organização espacial dos componentes da rede. A conexão ao banco de dados Neo4j fica a cargo de implementação externa a biblioteca. A biblioteca *3d-force-graph* será descrita e ilustrada nos Capítulos 4 e 5 presentes neste documento.

3.3 Discussão

Para dar foco aos estudos realizados durante a revisão da literatura buscou-se responder as seguintes perguntas investigativas (PI):

- PI 1 - Quais são as técnicas e funções presentes na literatura para visualização de grafos?

  De acordo com Chen et al. (2019) [36] as principais técnicas de visualização de grafos incluem: (i) nós e links, (ii) matriz de adjacências, (iii) hipergrafos, (iv) diagrama de fluxos, (v) geoespacial, (vi) multiatributos e (vii) preenchimento de espaços. As principais funções exploradas incluem: (i) clusterização de nós, (ii) filtragem do grafo, (iii) empacotamento de links, (iv) transformação baseada na topologia, (v) limitação de dimensões.

- PI 2: Quais soluções para visualização de redes armazenadas em Neo4j são de domínio específico? Os 19 trabalhos estudados, apresentados na Tabela 3.1 e descritos nas Seções 3.1 e 3.2, incluem aplicações em diversos domínios específicos, além de domínio geral, conforme apresentado na Figura 3.8. Note que a área de medicina e bioinformática apresenta a maior concentração de trabalhos (total 7), seguido pelas áreas de redes sociais (total 3) empatada com domínio geral (total 3), criminal (total 2) e finalmente engenharia civil, RH, aviação, e desenvolvimento de *software* com um trabalho cada.

Em resposta a PI 1 pode-se notar algumas soluções teóricas e artefatos implementados existentes na literatura para visualização de bases de dados utilizando Neo4j. A Tabela 3.1 apresenta as técnicas de visualização e funções exploradas por cada um. Pode-se notar que a técnica nós e links é a mais utilizada. Já a função mais explorada é a de filtragem do grafo.

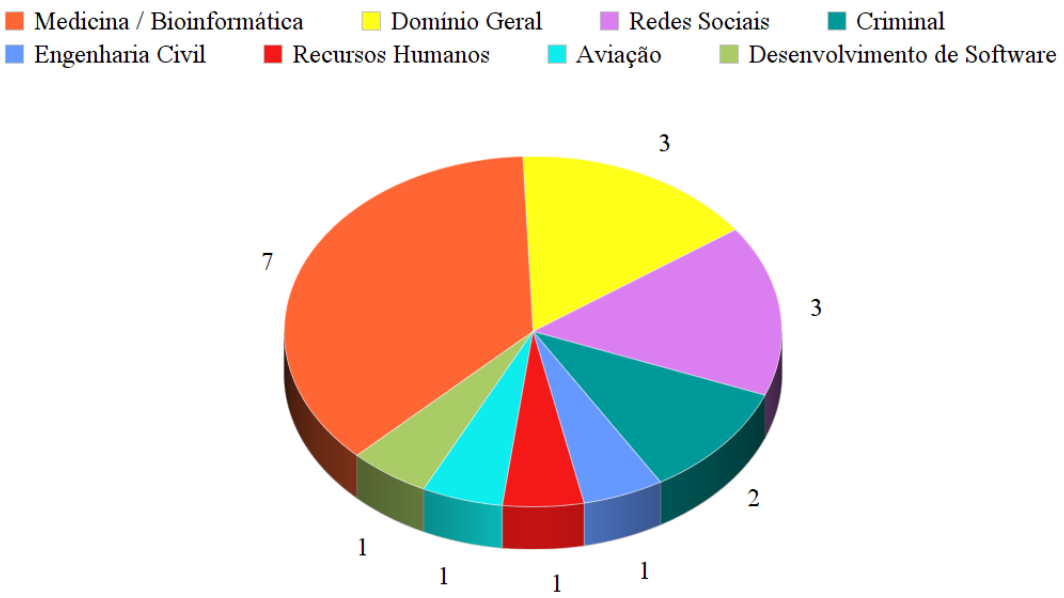


Figura 3.8: Domínios de aplicação dos trabalhos da literatura.

Este capítulo apresentou a revisão de literatura realizada para auxiliar no desenvolviento da solução. Considerando as perguntas investigativas PI 1 e 2, ressaltamos que os trabalhos classificados como técnicas, funções e artefatos de visualização apresentam majoritariamente nós e links em um contexto 2D com filtragem do grafo aplicados em domínios específicos. Portanto foi detectado uma lacuna de trabalhos com visualização 3D aplicado a domínio geral o qual será apresentada no Capítulo 4 de descrição da solução proposta.

Capítulo 4

Descrição da Proposta

Visando responder a segunda questão de pesquisa definida na Seção 1.3 do Capítulo 1 - Como desenvolver um artefato de visualização 3D de bases de dados orientadas a grafo utilizando Neo4j? - neste capítulo será descrita a solução proposta. Conforme apresentado no Capítulo 1, o objetivo principal desse trabalho é desenvolver uma solução que simplifique o acesso e uso do Neo4j potencializando a capacidade de visualização dos dados armazenados. Para esse propósito foi desenvolvido o artefato Graph2Vis capaz de gerar um conjunto de visualizações 3D de redes em formato nós e links. Em relação ao Graph2Vis será apresentado o ciclo de processamento, detalhamento da arquitetura e aspectos implementacionais.

Pela disponibilidade, versatilidade e abrangência, foram escolhidas tecnologias e linguagens de programação de característica *open-source* para o desenvolvimento da solução. Para a construção dos componentes da interface gráfica foi utilizada a ferramenta baseada na linguagem de programação *TypeScript* denominada *Angular*[1]. Já a escolha das técnicas de visualização justifica-se a partir dos trabalhos estudados durante a revisão de literatura (Capítulo 3). A técnica nós elinks foi a mais utilizada e mostrou-se vantajosa para visualização de grandes grafos. Para a construção das visualizações foi utilizado a biblioteca *3d-force-graph*[2]. Durante a revisão de literatura constatou-se que esta bilioteca apresentou características importantes para o desenvolvimento da solução, i.e., código aberto, ampla documentação de uso, facilidade de integração ao *Angular*, e a capacidade de gerar visualizações em três dimensões. Na sequência será detalhado o ciclo de processamento, a arquitetura e aspectos implementacionais do Graph2Vis.

[1] `https://angular.io/`

[2] `https://github.com/vasturiano/3d-force-graph`

4.1 Ciclo de Processamento

O ciclo de processamento construído neste trabalho segue o modelo definido por Chen et al. (2019) apresentado no Capítulo 3. O Graph2Vis possui como componente principal a interface web que faz a conexão a um banco de dados Neo4j e realiza a interação do usuário. Incluindo desde o acesso ao banco de dados até a saída final com as visualizações geradas. A Figura 4.1 apresenta o ciclo de processamento formado por cinco etapas. De forma geral, após o usuário conectar ao banco de dados Neo4j, o mesmo poderá informar a consulta a ser feita e configurar os parâmetros de visualização dos resultados. Na sequencia, a filtragem dos dados é feita e serão executadas as funções para gerar a visualização dos resultados em formato nós e links.

Figura 4.1: Ciclo de processamento do Graph2Vis.

A descrição das etapas apresentadas na Figura 4.1 inclui:

1. Login/Senha – o usuário especifica as credenciais de acesso ao banco de dados Neo4j. Nesta etapa a interface verifica junto ao banco de dados as credenciais de acesso (login/senha) para a autenticação. A Figura 4.2 ilustra esta etapa no Graph2Vis;

2. Consulta e Parâmetros de Visualização – nessa etapa é possível que o usuário queira informar apenas a *query* a ser executada no banco de dados. Dessa forma, serão

aplicados parâmetros *default* ao grafo resultante da pesquisa. A Figura 4.3 apresenta o exemplo de uma pesquisa em modo padrão. É possível também o usuário especificar no formulário a *query* de pesquisa ao banco de dados e os parâmetros de estilo da visualização dos nós do grafo, i.e, *label* a ser mostrado ao se passar o ponteiro do *mouse*, tamanho e cor. No campo *Based On* do formulário é possível escolher uma das duas opções: (i) *property*: indica que os parâmetros de visualização *label, size e color* serão atributos presentes no banco de dados a ser consultado; (ii) *custom*: os parâmetros *label, size e color* serão os informados pelo usuário. As Figuras 4.4 e 4.5 mostram um exemplo de uso dessas opções.

3. Visualização dos grafos – os resultados da consulta ao banco de dados são modelados para seram usados na visualização dos grafos em formato *node-link*, utilizando os parâmetros de estilo informados na Etapa 2.

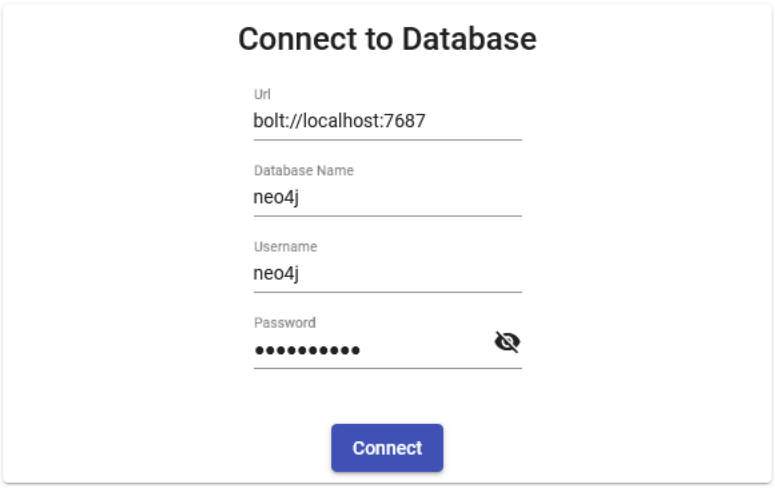


Figura 4.2: Captura de tela da etapa de login/senha.

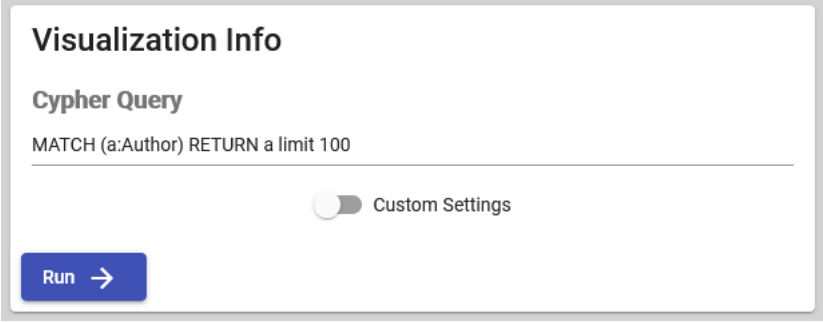


Figura 4.3: Captura de tela de uma pesquisa em modo *default*.

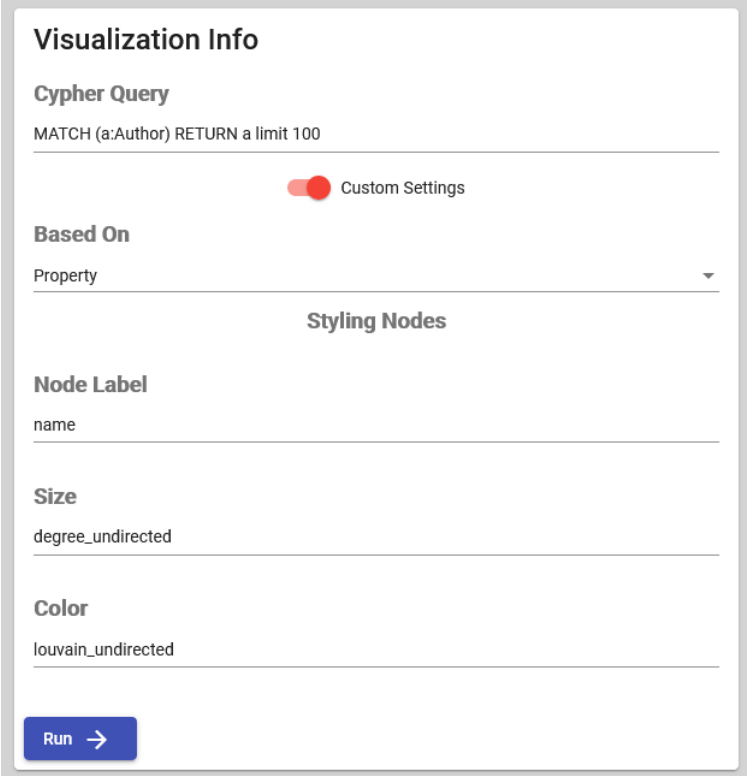


Figura 4.4: Captura de tela de uma pesquisa com a opção *property* selecionada.

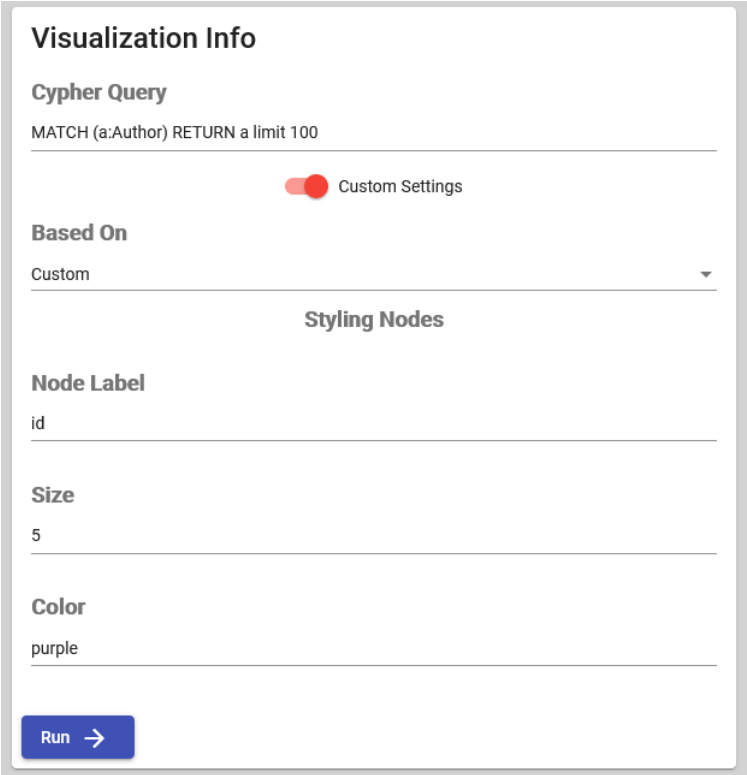


Figura 4.5: Captura de tela de uma pesquisa com a opção *custom* selecionada.

4.2 Arquitetura MVC

Esta seção aborda a arquitetura utilizada para desenvolvimento da interface web apresenta na Figura 4.1. Foi escolhida a arquitetura MVC para implementação. Como vantagens dessa arquitetura podemos citar: alta coesão, baixo acoplamento e fácil desenvolvimento de novas funcionalidades e manutenção [64]. Na arquitetura MVC, cada camada é responsável por uma série de ações que em conjunto trabalham no funcionamento da solução. De modo geral, essas camadas desempenham as seguintes funções:

- Modelo - responsável por definir a estrutura dos dados e seus relacionamentos e também interagir com o banco de dados.
- Controladora - camada intermediária que envia requisições a camada *Modelo* e atualiza a camada *Visão* com os resultados retornados pela camada *Modelo*.
- Visão - camada que apresenta os dados retornados pela *Controladora*.

Aplicando a arquitetura MVC teremos etapas que usarão camadas específicas. A Figura 4.6 apresenta a arquitetura construída neste trabalho. A descrição das etapas apresentadas na Figura 4.6 inclui:

1. parâmetros visualização - o Usuário específica a consulta ao banco de dados com os parâmetros de visualização (*label*, cor e tamanho) dos nós.
2. uma requisação HTTP é formada e enviada com as entradas informadas na Etapa 1 para a camada *controladora*.
3. a camada *controladora* gera e envia a requisição HTTP no formato Neo4j para a camada *modelo*.
4. é feita a coleta dos dados no banco Neo4j, de acordo com a requisição recebida.
5. a camada *modelo* retorna o resultado da pesquisa ao banco de dados em um objeto *JavaScript*, contendo a lista de vértices e arestas que representam os grafos.
6. a partir do objeto gerado na camada *controladora* são construídos os grafos resultantes para visualização.
7. a camada *visão* é atualizada e a visualização dos grafos é apresentada ao usuário no *browser*.

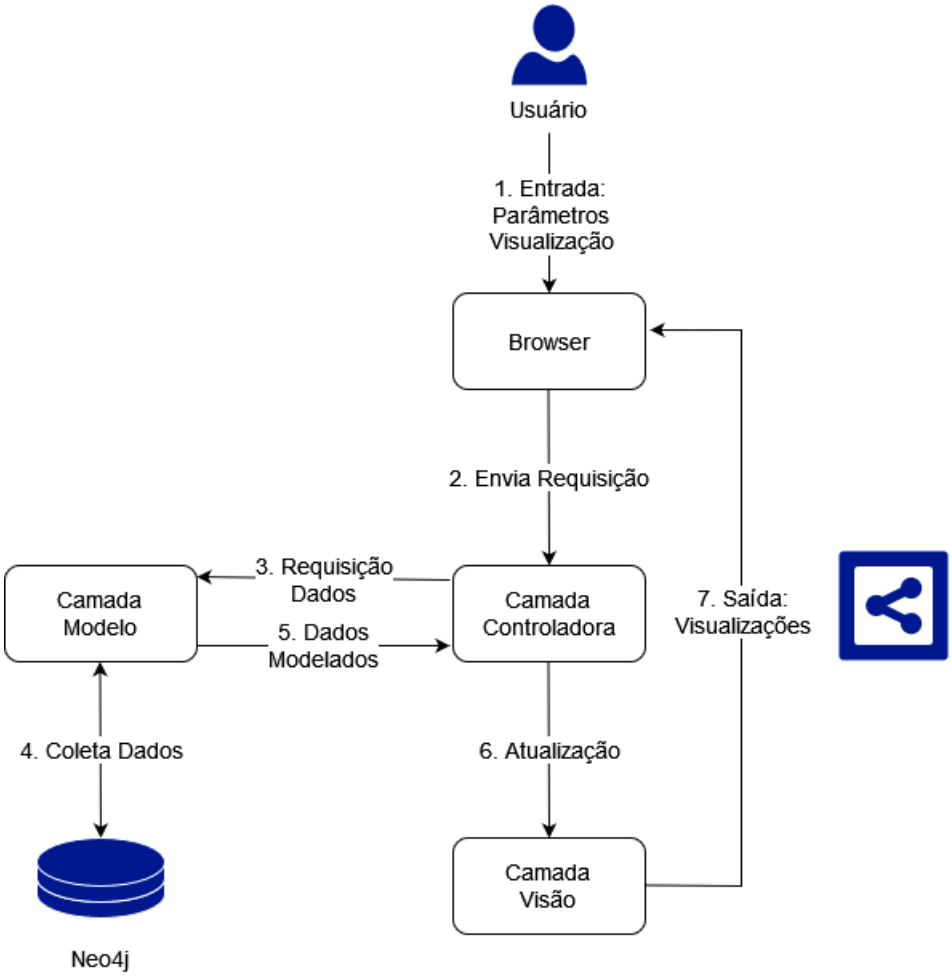


Figura 4.6: Arquitertura MVC do Graph2Vis.

4.3 Aspectos Implementacionais

Nesta seção serão apresentados os detalhes da implementação do protótipo do artefato Graph2Vis. Para a modelagem foi utilizado o paradigma de orientação a objetos utilizando o *Unified Modeling Language* (UML) [65] para a construção do diagrama classes com seus respectivos atributos e métodos. Para a construção dos componentes da interface web foram utilizadas diferentes tecnologias. O *framework Angular*[3], o qual permite a construção de aplicações de forma simples, dinâmica e segura [66], utilizando a linguagem de programação *TypeScript*. A etapa de visualizações dos grafos em formato *node-link* foi construída com o uso da biblioteca *3d-force-graph*[4].

[3] `https://angular.io/`
[4] `https://github.com/vasturiano/3d-force-graph`

Os arquivos de implementação do protótipo, bem como a documentação a cerca do *Graph2Vis* encontram-se disponíveis no repositório *GitHub*[5] disponibilizado sobre a licença MIT de código aberto. Foi também disponibilizado o *deploy* do protótipo em um servidor acessível pelo link `http://graph2vis.herokuapp.com/`

4.3.1 Diagrama de Classes

Com o desenvolvimento do diagrama de classes há uma maior organização e definição clara sobre os componentes que formam o Graph2Vis. A Figura 4.7 representa as classes utilizadas para classificar os objetos envolvidos na solução bem como suas características, por meio dos atributos e métodos. O relacionamento entre as classes pode ser visto através das associações. Na solução foram utilizadas cinco classes, a saber:

A Figura 4.7 representa as classes utilizadas para classificar os objetos envolvidos na solução bem como suas características, por meio dos atributos e métodos. O relacionamento entre as classes pode ser visto através das associações. Na solução foram utilizadas cinco classes, a saber:

- *ConnectionFormComponent* – representa o formulário que recebe as informações de autenticação ao banco de dados Neo4j.
- *VisualizationFormComponent* – representa o formulário que coleta a consulta a ser executada no banco de dados, bem como os parâmetros de visualização dos resultados.
- *Neo4j Service* – classe que realiza as operações de autenticação e consultas ao banco de dados.
- *DataService* – classe que realiza as funções de troca de informações entre as classes *ConnectionFormComponent–Neo4jService* e *VisualizationFormComponent–Neo4jService*.
- *GraphComponent* – modela dos resultados vindos da classe *Neo4jService*, para a apresentação dos grafos em formato *node-link*.

Cada classe descrita possui atributos que definem as características de cada componente desenvolvido. Para um melhor entendimento desses atributos foi elaborado um dicionário de atributos representado pelas Tabelas A.1, A.2, A.3, A.4, A.5 incluídas no Apêndice A.

[5] `https://github.com/lucaslemos94/graph2vis`

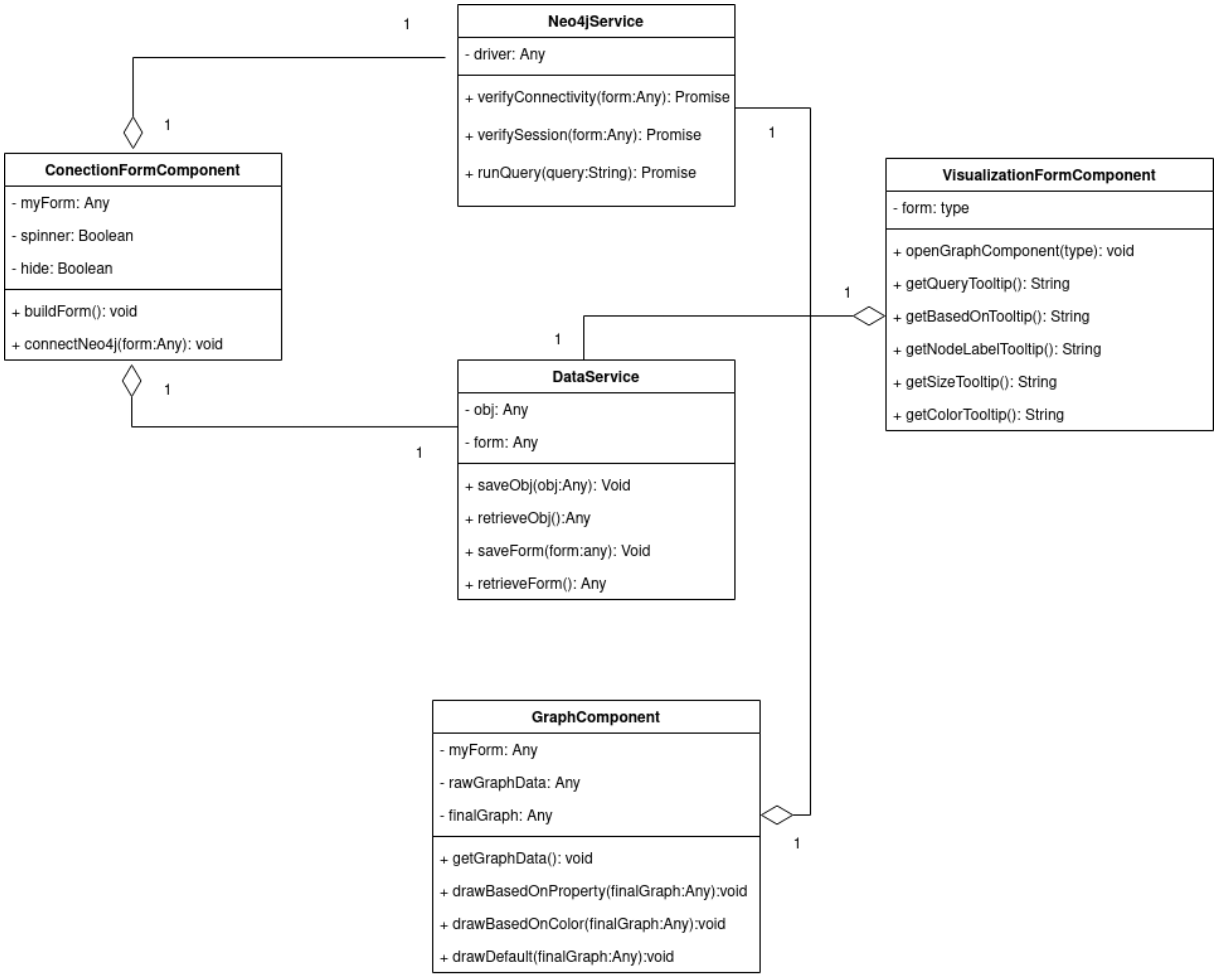


Figura 4.7: Diagrama de classes do Graph2Vis.

4.3.2 *Framework Angular*

O *Angular* é uma ferramenta baseada em *TypeScript* e desenvolvida por engenheiros do Google, que decidiram criá-lo com intuito de facilitar manutenção de alguns de seus serviços Web. Em funcionamento, o *Angular* estrutura a aplicação de forma próxima ao MVC. Com o desenvolvimento de soluções sendo divida em módulos, a usabilidade e manutenibilidade de código é feita de forma mais produtiva. A estrutura do *Angular* é baseada em *Components*. Como pode ser visto na Figura 4.8, cada componente de uma aplicação possui três arquivos com extensões distintas: (i) .html: parte visual do componente, como cores e formas; (ii) .ts: onde ficam os códigos correspondentes as ações do componente, e.g., enviar requisições para acessar um banco de dados; (iii) .service: código que atuará como o *back-end* do componente, e.g., receber requisições vindas dos arquivos .ts. Fazendo paralelo de forma mais direta ao padrão MVC, os arquivos .html, .ts e .service correspondem as camadas de visão, modelo e controladora, respectivamente. A Figura 4.9 apresenta os componentes do Graph2Vis implementados no *Angular*, incluindo

formulários para conexão, consulta ao banco de dados e configuração de parâmetros de visualização, bem como processamento dos resultados de consulta para a construção dos grafos de saída.

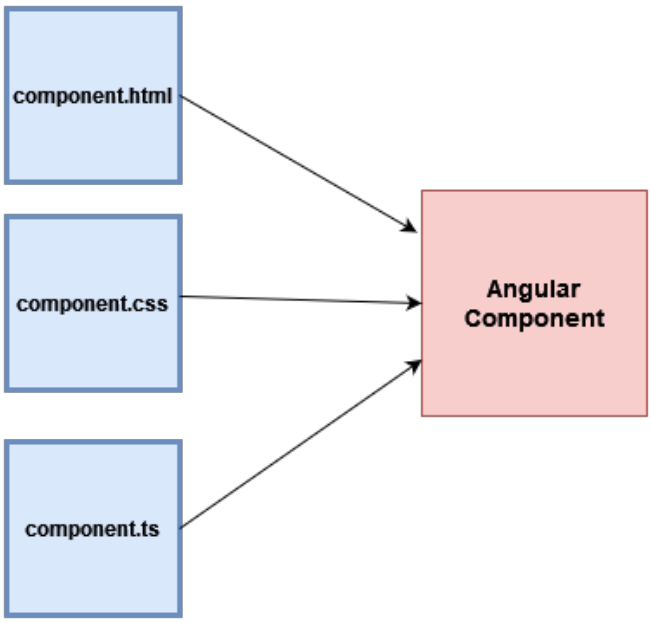


Figura 4.8: Estrutura de um *component Angular*.

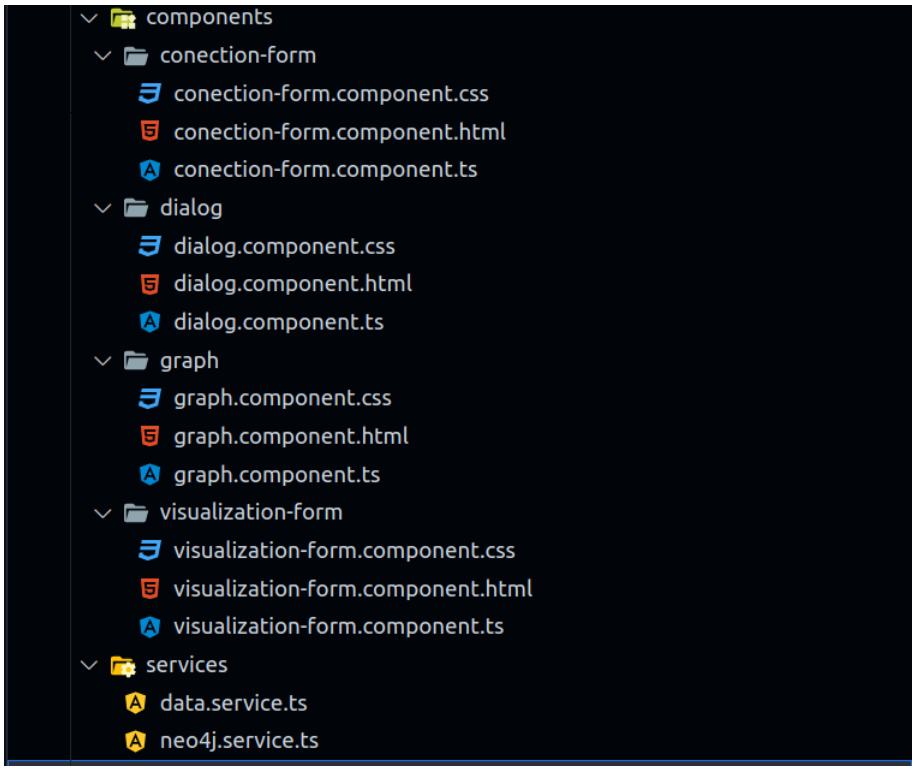


Figura 4.9: Estrutura dos componentes que formam o Graph2Vis.

4.3.3 Biblioteca *3d-force-graph*

Para a visualização dos grafos gerados pelo Graph2Vis foi utilizado a biblioteca de visualização *3d-force-graph*. Com o uso dessa biblioteca em aplicações web é possível visualizar grafos em três dimensões, utilizando o algoritmo *force-directed* [67] para a organização do *layout* das redes formadas. Tal algoritmo utiliza conceitos relacionado a física de objetos sendo útil no contexto de visualização de grafos, procurando a melhor disposição dos elementos apresentados na tela.

É possível também a configuração de diversos parâmetros de visualização, como cores, tamanhos e formas dos grafos gerados, potencializando as análises visuais. Há também uso do *3d-force-gragh* em contextos de aplicações 2D, *Virtual Reality* (VR) e *Augmented Reality* (AR). Como os *browsers* entendem apenas JavaScript em seu funcionamento, o *Angular* faz uma transpilação dos arquivos *TypeScript* dos componentes para *JavaScript*, i.e, há uma tradução de código de uma linguagem de programação para outra [68]. Desse modo, a integração do *3d-force-gragh* ao *Angular* se dá de forma simplificada e direta. Ao acessar a pasta raiz do projeto no *Angular*, basta executar um comando via *prompt* de comando do sistema operacional que irá fazer o *download* e instalação. A Listing 4.1 apresenta esta etapa de *download* e instalação do *3d-force-graph* junto ao *Angular*.

Listing 4.1: Comando para *download* e instalação do *3d-force-graph*

```
C:\Users\lclem\OneDrive\Documents\graph2vis> npm i 3d-force-
    graph
> installed 1 package, and audited 1408 packages in 9s.
C:\Users\lclem\OneDrive\Documents\graph2vis>
```

Foi apresentado nesse capítulo os aspectos de desenvolvimento do artefato de visualização Graph2Vis, incluindo o ciclo de processamento, arquitetura e aspectos implementacionais. O Capítulo 5 contém uma descrição dos estudos de caso utilizados com o artefato apresentado neste capítulo.

Capítulo 5

Estudos de Caso

Visando responder a terceira questão de pesquisa definida na Seção 1.3 do Capítulo 1 - Quais estudos de caso podem ilustrar o artefato desenvolvido durante a execução deste trabalho? - neste capítulo são apresentados três estudos de caso de redes sociais científicas. O projeto *SCI-synergy*[1], um substrato do banco Neo4J do SCI-synergy relacionando aos aspectos de gênero e o *Agent Disambiguation Author Name* (ADAN)[2]. Os estudos serão apresentados em seções separadas incluindo entradas, processamento e saídas ilustradas.

5.1 Projeto *SCI-synergy*

O projeto SCI-synergy apresenta uma rede social científica formada por cinco programas de pós-graduação da área de Ciência da Computação, vinculadas à cinco universidades públicas brasileiras: Universidade Federal do Amazonas (UFAM), Universidade Federal de Minas Gerais (UFMG), Universidade Federal do Rio Grande do Norte (UFRN), Universidade de Brasília (UnB), Universidade de São Paulo (USP). O projeto é vinculado ao Grupo de Pesquisa *InfoKnow - Computer Systems for Information and Knowledge Treatment* do CNPq[3]. Atualmente a rede formada no *SCI-synergy* ilustra relações de coautoria e publicações entre autores, conforme apresenta a Figura 5.1. Neste sentido, há um *nó* e relação denominada *ALIAS*, indicando os *aliases* que cada pesquisador possui. Há também o nó *Institution* e a relação *ASSOCIATED_TO*, indicando a Universidade que cada pesquisador está associado.

A extração dos dados feita pelo projeto se dá pelo uso do repositório *Digital Bibliography and Library Project* (DBLP)[4], com a construção de uma base de dados NoSQL orientada a grafo, implementada no sistema Neo4j. Conforme a atualização da base de

[1] `http://165.227.113.212/`
[2] `https://gitlab.com/InfoKnow/SocialNetwork/sci_clan/adan`
[3] `http://dgp.cnpq.br/dgp/espelhogrupo/34202`
[4] `https://dblp.org/`

dados do projeto realizada em 03/02/2021, a base de dados do SCI-synergy conta com o total de 194 pesquisadores distribuídos entre as cinco universidades federais, formando uma rede com 17.617 nós e 92.287 relações.

Figura 5.1: Esquema da base de dados SCI-synergy

5.2 Aspectos da Rede Social do PPGI/UnB

Dos 194 pesquisadores presentes no banco de dados do SCI-synergy, 31 pertencem ao Programa de Pós-Graduação da Universidade de Brasília (PPGI/UnB), que em suas redes totalizam 1.621 autores e 3.619 relações de coautoria. Além das informações da rede social já disponíveis, foram executadas no Neo4j duas rotinas na linguagem CYPHER para computar e armazenar em cada nó duas medidas de centralidade, a saber:

1. *degree* - representa o número de relações que são incidentes em determinado nó, indicando o grau de conectividade direta desse nó com outros na rede. Neste caso, o *degree* indica o grau de coautoria de um pesquisador (vide Listing 5.1).

2. *louvain* - representa qual comunidade determinado nó pertence a rede, indicando grupos de nós que tem grande probabilidade de compartilhar propriedades comuns ou terem papéis semelhantes no grafo. Neste caso, pesquisadores que possuem muitas relações de coautoria entre si são propensos a serem de uma mesma comunidade. (vide Listing 5.2).

As medidas de centralidade descritas serão utizadas como parâmetros de entrada ao fazer as consultas e visualizações no protótipo do Graph2Vis. A medida *degree* será

usada para determinar o tamanho de cada nó. Enquanto a medida *louvain* indicará a sua respectiva cor. Além das medidas relacionadas aos nós, foi executado no Neo4j uma rotina em linguagem CYPHER para calcular uma nova relação no banco de dados denominada *COAUTHOR_WEIGHT* (vide Listing 5.3). Essa relação representa a contagem de coautorias que um autor tem com outro. A relação *COAUTHOR_WEIGHT* será útil para resumir a quantidade de relações de coautoria na etapa de visualização dos grafos, proporcionando uma visualização mais coesa.

Listing 5.1: Consulta CYPHER para o grau de coautoria de cada pesquisador do PPGI/UnB.

```
MATCH (a:Author)-[r:COAUTHOR]-(b:Author)
WITH a,b,count(r) as dg
SET a.degree_undirected = dg
```

Listing 5.2: Consulta CYPHER para calcular a comunidade de cada pesquisador do PPGI/UnB.

```
CALL algo.louvain(
MATCH (a:Author) where (a)-[:COAUTHOR]->() RETURN id(a) as
    id,
MATCH (a)-[:COAUTHOR]->()<-[:COAUTHOR]-(b) return id(a) as
    source, id(b) AS target, count(*) as `louvain_undirected
    `, { graph: `neo4j`, iterations: 1})
```

Listing 5.3: Consulta CYPHER que cria a relação *COAUTHOR_WEIGHT* de cada pesquisador do PPGI/UnB.

```
MATCH (a:Author)-[r:COAUTHOR]->(b:Author) WITH a, b ,count(r
    ) as W
CREATE (a)-[O:COAUTHOR_WEIGHT{weight:W}]->(b)
```

Após o cálculo das medidas *degree* e *louvain* e a relação *COAUTHOR_WEIGHT*, foram realizadas algumas consultas usando o protótipo do Graph2Vis para verificar características da rede de coautoria do PPGI/UnB, a saber:

1. visualizar a rede de coautoria de um pesquisador específico do PPGI/UnB (vide Listing 5.4 e Figuras 5.2, 5.3 e 5.4).

2. visualizar a rede dos 31 pesquisadores do PPGI/UnB (vide Listing 5.5 e Figuras 5.5 e 5.6).

3. visualizar a rede de coautoria dos cinco Programas de Pós-Graduação em Computação (vide Listing 5.6 e Figura 5.7).

Listing 5.4: Consulta CYPHER para a rede de coautoria de um pesquisador do PPGI/UnB.

```
MATCH
(a1:Author)-[co:COAUTHOR_WEIGHT]-(a2:Author{name:'Celia
    Ghedini Ralha'})-[ass:ASSOCIATED_TO]-(i:Institution{name
    :'UnB'})
RETURN a1,co,a2
```

Listing 5.5: Consulta CYPHER para a rede de coautoria dos 31 pesquisadores vinculados ao PPGI/UnB.

```
MATCH (a1:Author)-[co:COAUTHOR_WEIGHT]-(a2:Author})-[ass:
    ASSOCIATED_TO]-(i:Institution{name:'UnB'})
RETURN a1,co,a2
```

Listing 5.6: Consulta CYPHER para a rede de coautoria dos 5 programas de pós-graduação em Computação.

```
MATCH (a1:Author)-[co:COAUTHOR_WEIGHT]-(a2:Author))
RETURN a1,co,a2
```

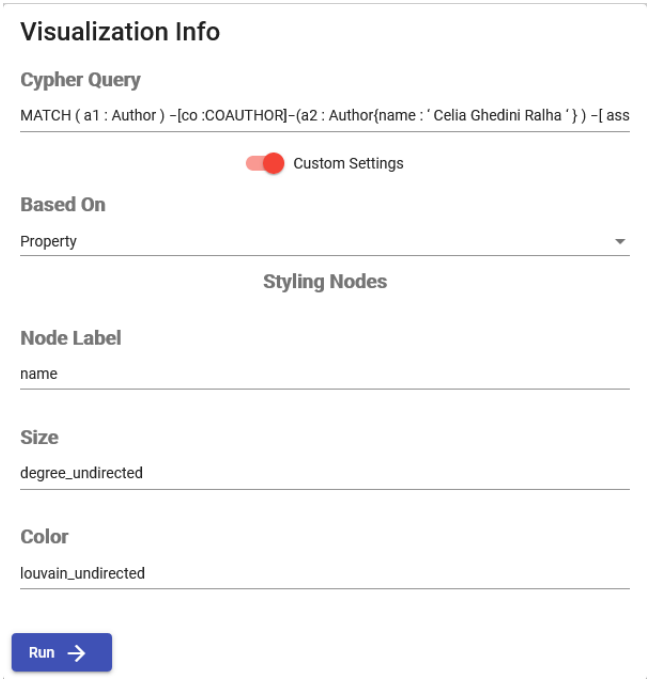


Figura 5.2: Captura de tela da consulta de um pesquisador específico.

Figura 5.3: Informações a cerca de um pesquisador específico do PPGI/UnB.

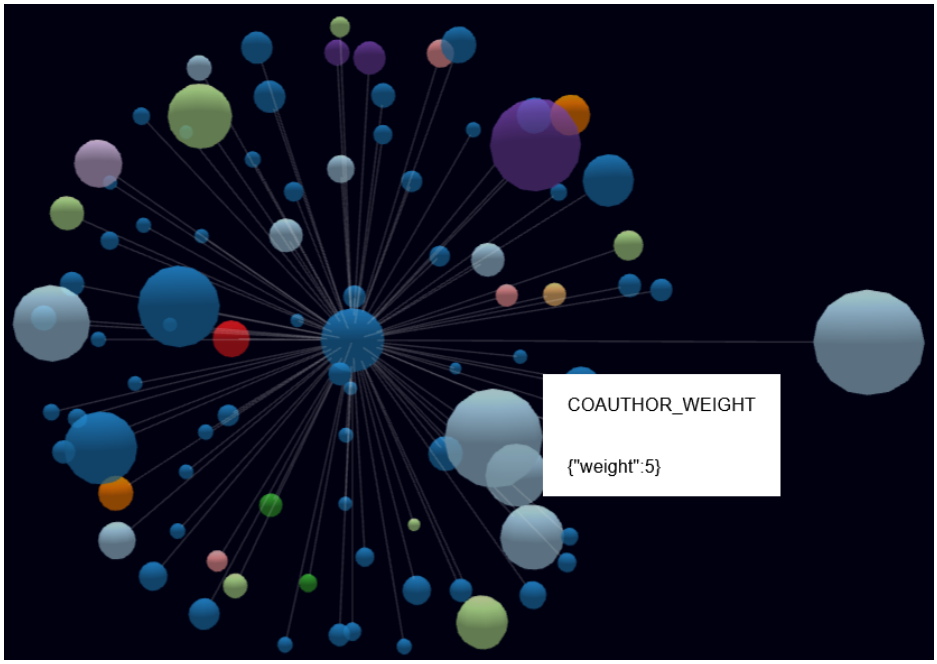


Figura 5.4: Informações a cerca do peso de coautoria entre dois autores.

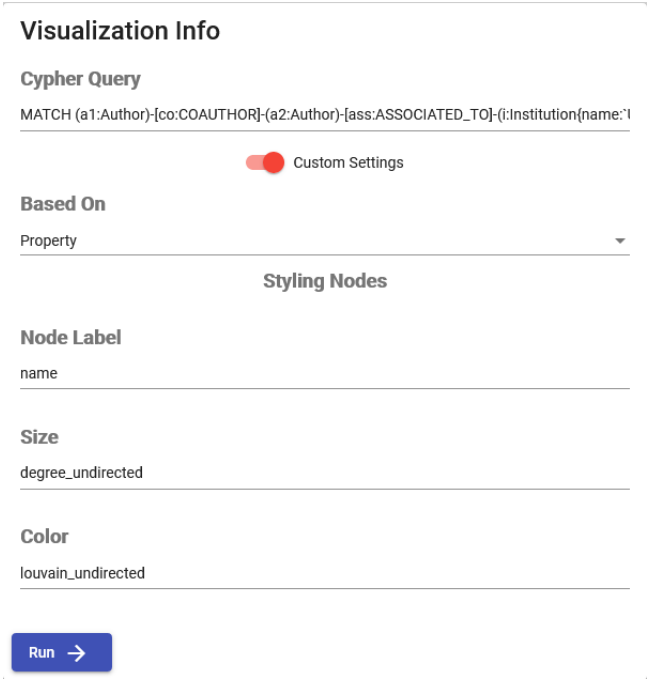


Figura 5.5: Captura de tela da etapa de consulta dos 31 pesquisadores do PPGI/UnB.

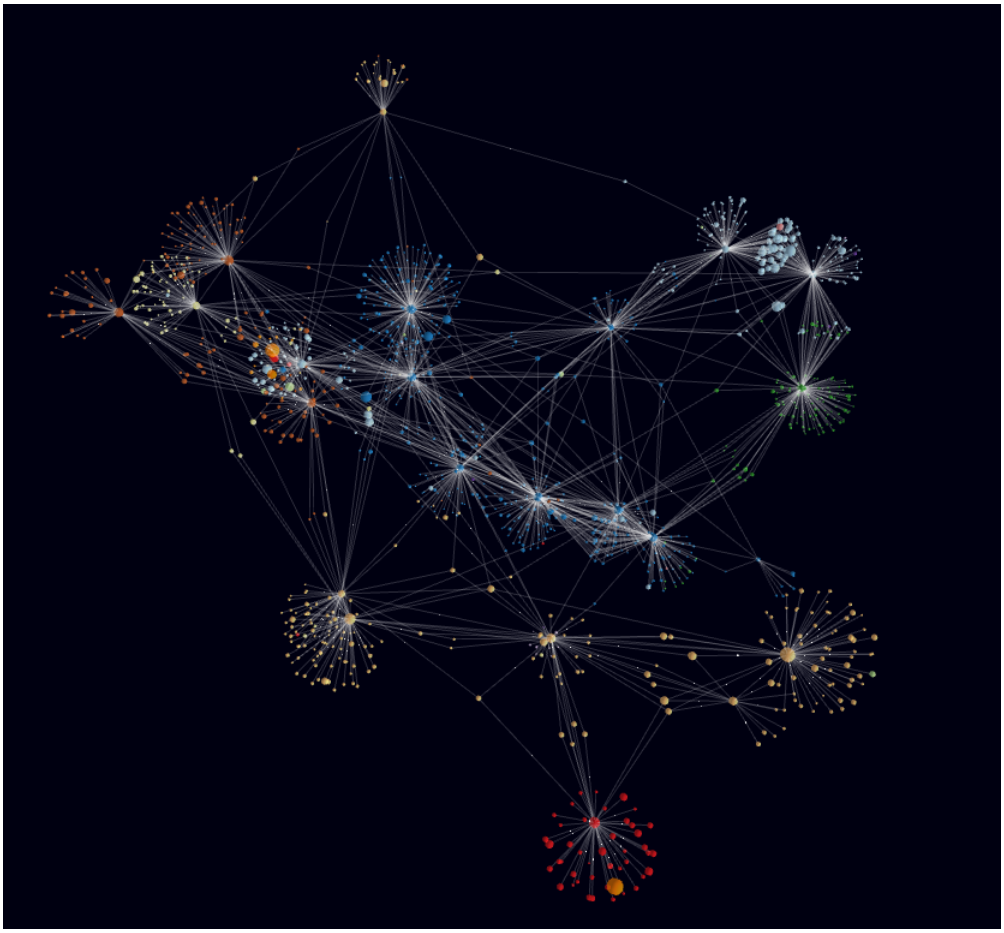


Figura 5.6: Rede de coautoria dos 31 pesquisadores do PPGI/UnB.

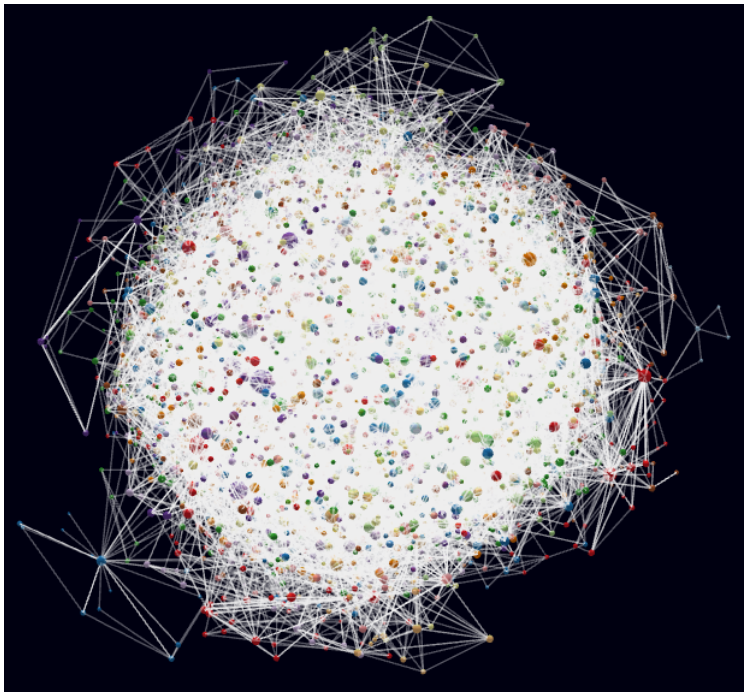


Figura 5.7: Rede de coautoria dos cinco programas de pós-graduação em Computação.

Observando as visualizações apresentadas é possível identificar a rede de coautoria de um pesquisador específico do PPGI/UnB, bem como a rede dos 31 pesquisadores, e a rede dos programas de pós-graduação em computação das cinco Universidades públicas Brasileiras: Universidade Federal do Amazonas (UFAM), Universidade Federal de Minas Gerais (UFMG), Universidade Federal do Rio Grande do Norte (UFRN), Universidade de Brasília (UnB), e Universidade de São Paulo (USP).

Nos grafos apresentados foram identificadas algumas características na rede de coautoria do PPGI/UnB. Na Figura 5.3 pode-se visualizar as informações de um pesquisador ao passar o ponteiro do *mouse* sobre ele. Note que na Figura 5.4 na mesma rede pode-se visualizar variados pesquisadores com suas cores, indicando a comunidade que cada um pertence (perfis de pesquisa distintos na rede) através de suas cores, tamanhos variados de cada nó representando o grau de conectividade entre cada pesquisador (o grau de colaboração entre os mesmos). É possível também visualizar o grau de coautoria entre dois pesquisadores através das informações contidas em suas relações.

Já em um cenário maior, como na Figura 5.6 é possível identificar *clusters* da rede, indicando pesquisadores com alto grau de colaboração científica. Pode-se notar também pesquisadores intermediários indicando uma *ponte* de colaboração que ligam esses *clusters*. Utilizando a visualização em três dimensões (3D) é possível rotacionar e/ou aplicar *zoom* para melhor exploração dos componentes da rede. Por fim, na Figura 5.7 torna-se difícil a visualização da rede dos cinco programas de pós-graduação devido a grande quantidade de vértices e arestas presentes.

5.3 Aspectos de Gênero do PPGI/UnB

Adicionando um novo olhar sobre a rede de colaboração científica formada dentro do projeto *SCI-synergy*, é possível identificar características relacionadas ao gênero dos pesquisadores vinculados ao PPGI/UnB. Dos 31 pesquisadores que integram o PPGI/UnB, 21 são homens e 10 são mulheres. Já toda a rede de coautoria dos 31 pesquisadores contempla 988 homens e 286 mulheres. Como aponta o trabalho em [69], as áreas de Ciências Exatas possuem presença majoritariamente masculina e isso se torna ainda mais evidente na pós-graduação. Neste sentido, foram feitas consultas no Neo4j para identificar características relacionadas a gênero, a saber:

1. a quantidade de artigos científicos publicados pelos autores do PPGI/UnB considerando o gênero feminino ou masculino (vide pesquisas CYPHER 5.7 e 5.8).

2. a quantidade de coautoria das mulheres em relação aos homens (vide pesquisas CYPHER 5.9 e 5.10).

Listing 5.7: Consulta CYPHER que retorna as autorias por gênero feminino.

```
MATCH (a:Author)-[au:AUTHORING]-(p:Paper) WHERE a.gender='F'
    return count (au)
```

Listing 5.8: Consulta CYPHER que retorna as autorias por gênero masculino.

```
MATCH (a:Author)-[au:AUTHORING]-(p:Paper) WHERE a.gender='M
    ' return count (au)
```

Listing 5.9: Consulta CYPHER que retorna coautoria do gênero feminino.

```
MATCH (a1:Author)-[co:COAUTHOR]->(a2:Author) WHERE a1.gender
    ='F' return count(co)
```

Listing 5.10: Consulta CYPHER que retorna a coautoria do gênero masculino.

```
MATCH (a1:Author)-[co:COAUTHOR]->(a2:Author) WHERE a1.
    gender='M' return count(co)
```

Com resultado dessas consultas foi possível notar algumas diferenças entre os dois gêneros. A Figura 5.8 apresenta a quantidade de artigos publicados, sendo 2.326 masculino e 735 feminino. Pode-se notar que homens possuem uma maior quantidade de publicações. Em relação a quantidade de coautorias por gênero masculino e feminino, pode-se notar também diferenças entre os integrantes da rede do PPGI/UnB. As coautorias masculinas totalizam 2.164, enquanto as femininas 629.

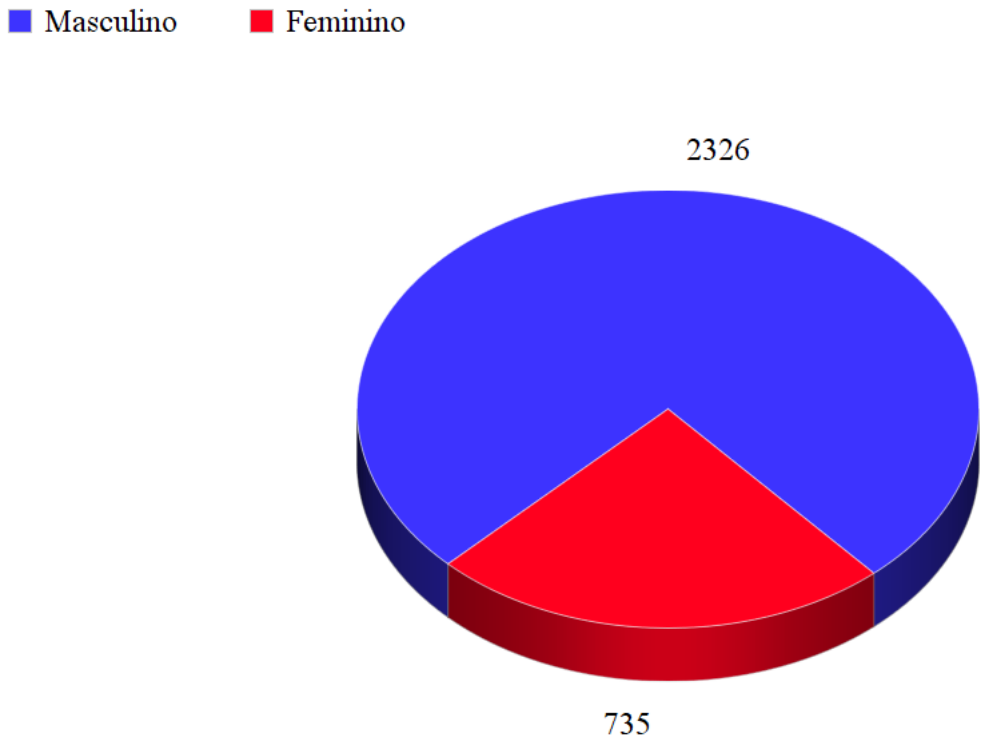


Figura 5.8: Relações de autoria por gênero

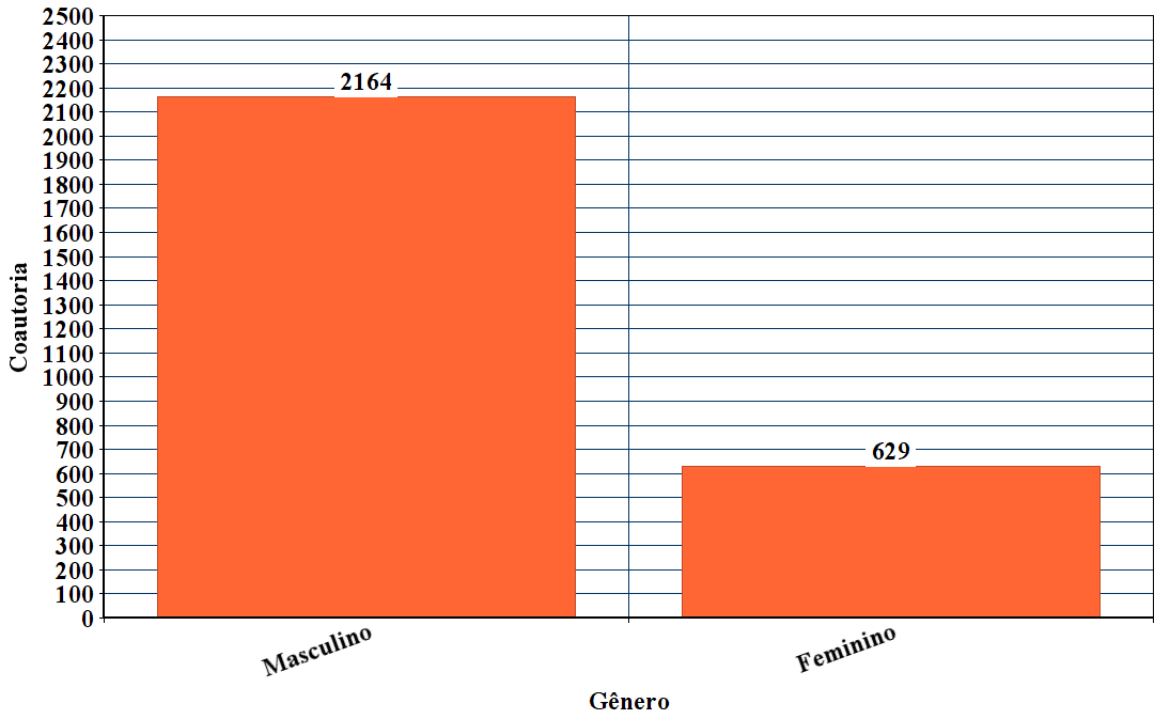


Figura 5.9: Coautorias por gênero.

Com o intuito de gerar visualizações em grafos dos resultados no Graph2Vis, foram realizadas as consultas CYPHER 5.7, 5.8, 5.9 e 5.10 em modo *default* (vide Figuras 5.10, 5.11). A Figura 5.12 apresenta a visualização dos grafos dessas consultas, incluindo os artigos publicados por gênero. Note que a rede de artigos por gênero feminino conta com nós mais desconexos e com menos *clusters* formados. Já a rede formada pelo gênero masculino apresenta o contrário, mais *clusters* formados e menos grafos desconexos. Pode-se notar também que em comparação com as mulheres, os homens tendem a publicar mais artigos entre si. Neste sentido, os grafos elucidam algumas características a cerca dos artigos publicados por cada gênero.

Foram realizadas consultas no Neo4j para verificar três perfis de coautorias: homem/homem, homem/mulher - mulher/homem e mulher-mulher (vide pesquisas CYPHER 5.11, 5.12,5.13). Como apresentado na Figura 5.13, pode-se notar que os homens possuem mais coautorias no perfil homem-homem, enquanto as mulheres possuem maior coautoria no perfil mulher-homem. Executando as pesquisas no Graph2Vis, foi possível gerar os grafos correspondentes (vide Figuras 5.14, 5.15).

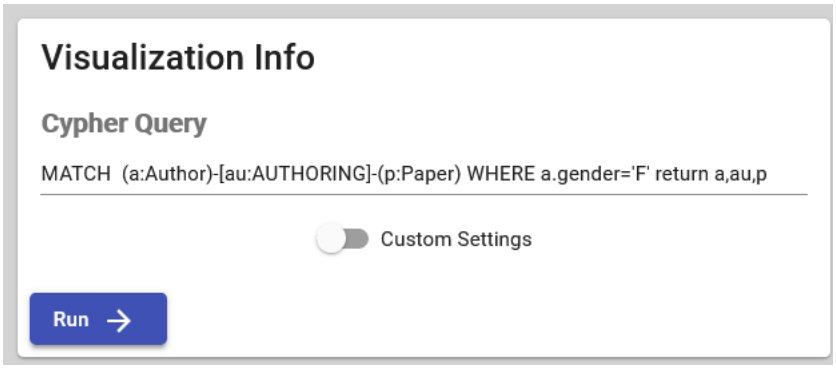


Figura 5.10: Captura de tela da etapa de pesquisa de artigos publicados pelo gênero feminino.

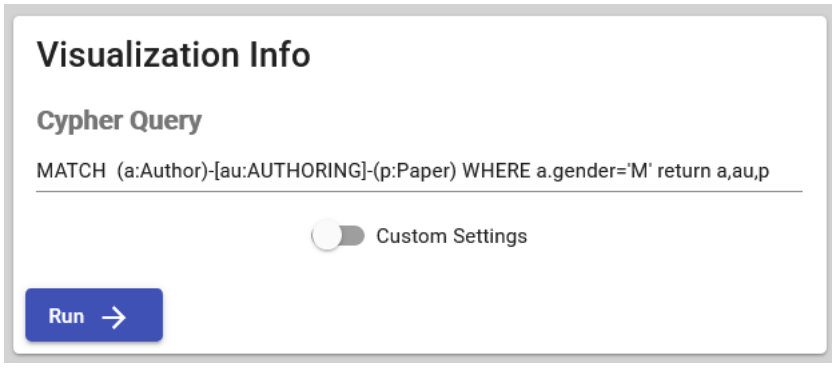
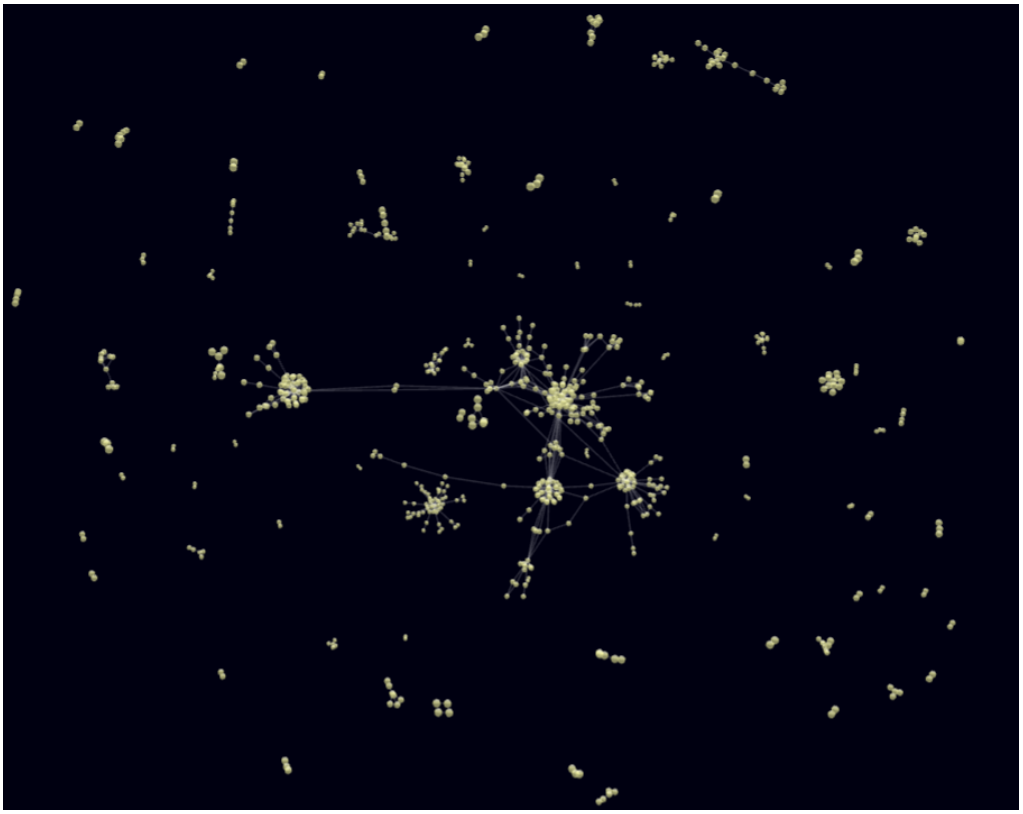
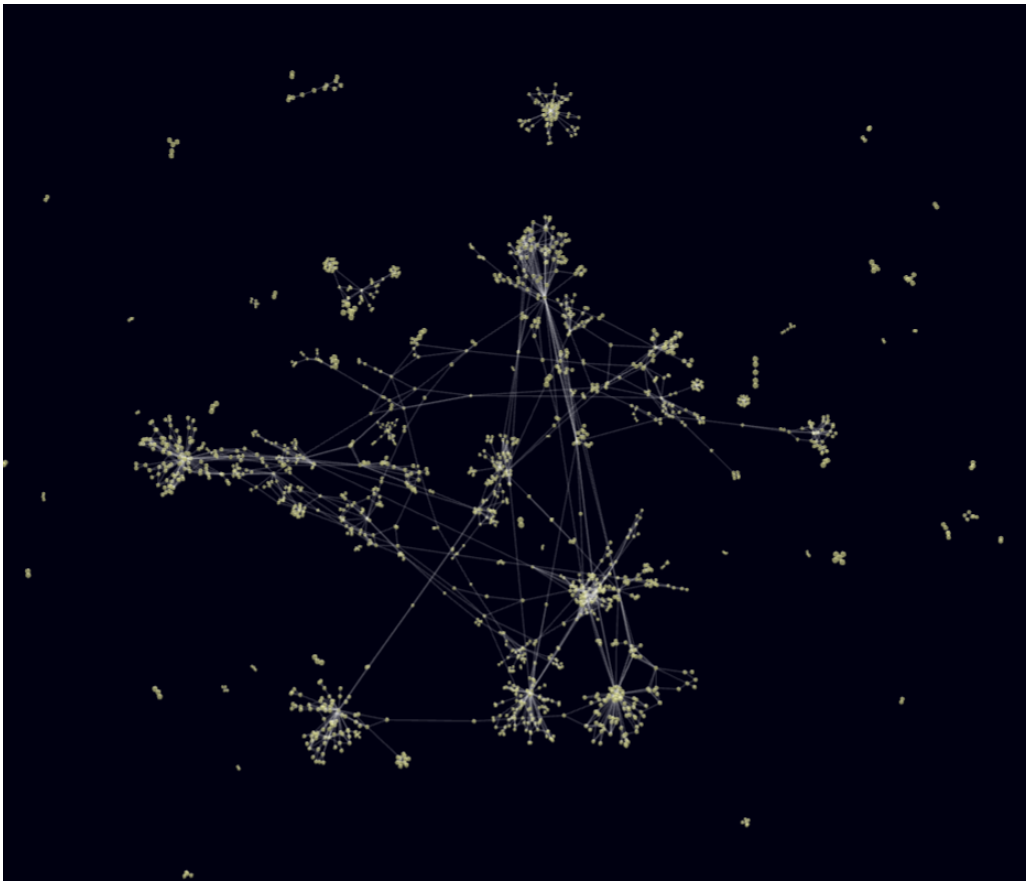


Figura 5.11: Captura de tela da etapa de pesquisa de artigos publicados pelo gênero masculino.



(a) Artigos publicados pelos integrantes do gênero feminino.



(b) Artigos publicados pelos integrantes do gênero masculino.

Figura 5.12: Visualização dos artigos publicados por gênero

Listing 5.11: Consulta CYPHER para a coautoria homem/homem.

```
MATCH (a1:Author)-[co:COAUTHOR]->(a2:Author) WHERE a1.
    gender='M' and a2.gender='M' return count(co)
```

Listing 5.12: Consutla CYPHER para a coautoria mulher/mulher.

```
MATCH (a1:Author)-[co:COAUTHOR]->(a2:Author) WHERE a1.gender
    ='F' and a2.gender='F' return count(co)
```

Listing 5.13: Consulta CYPHER para a coautoria homem/mulher - mulher/homem

```
MATCH (a1:Author)-[co:COAUTHOR]-(a2:Author)
WHERE a1.gender ='M' and a2.gender='F' return count(co)
```

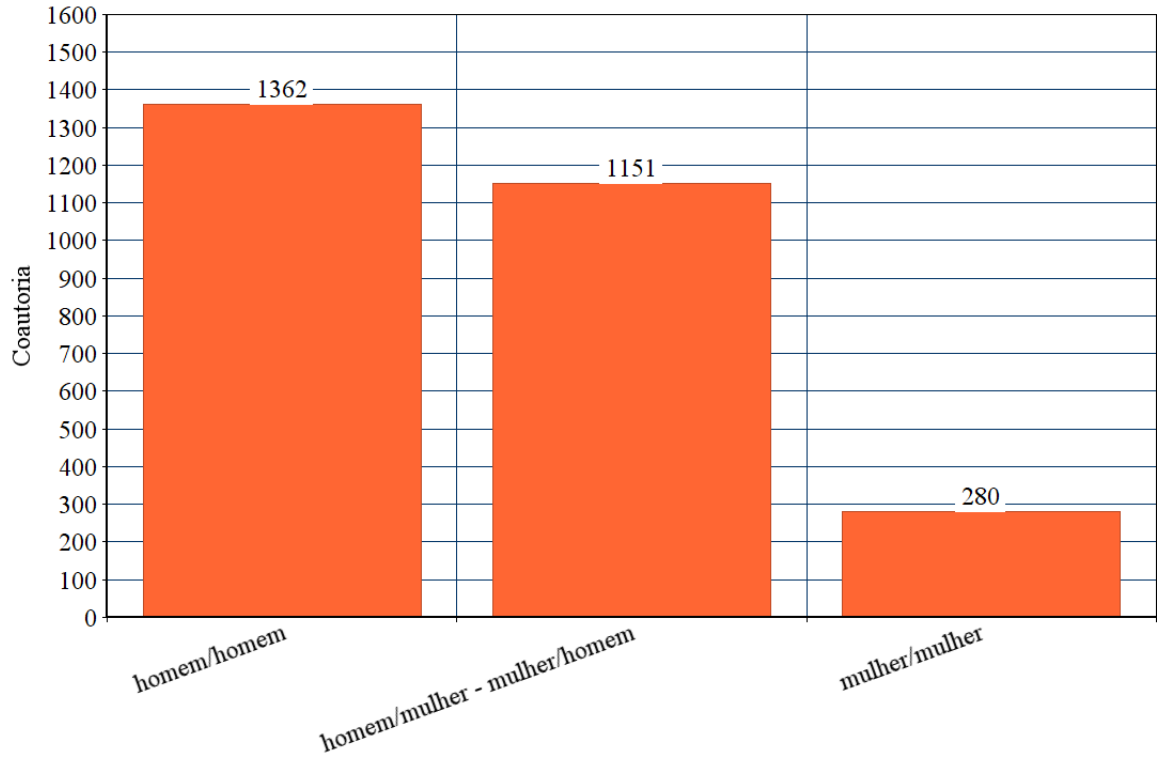


Figura 5.13: Coautorias por perfil de gênero.

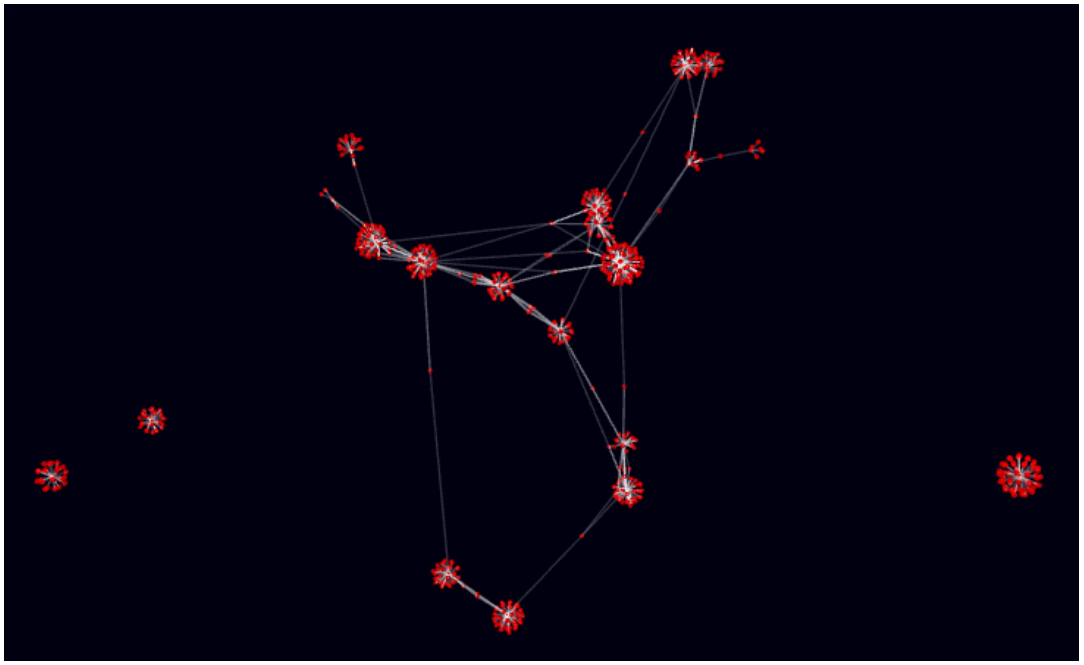


Figura 5.14: Perfil homem/homem.

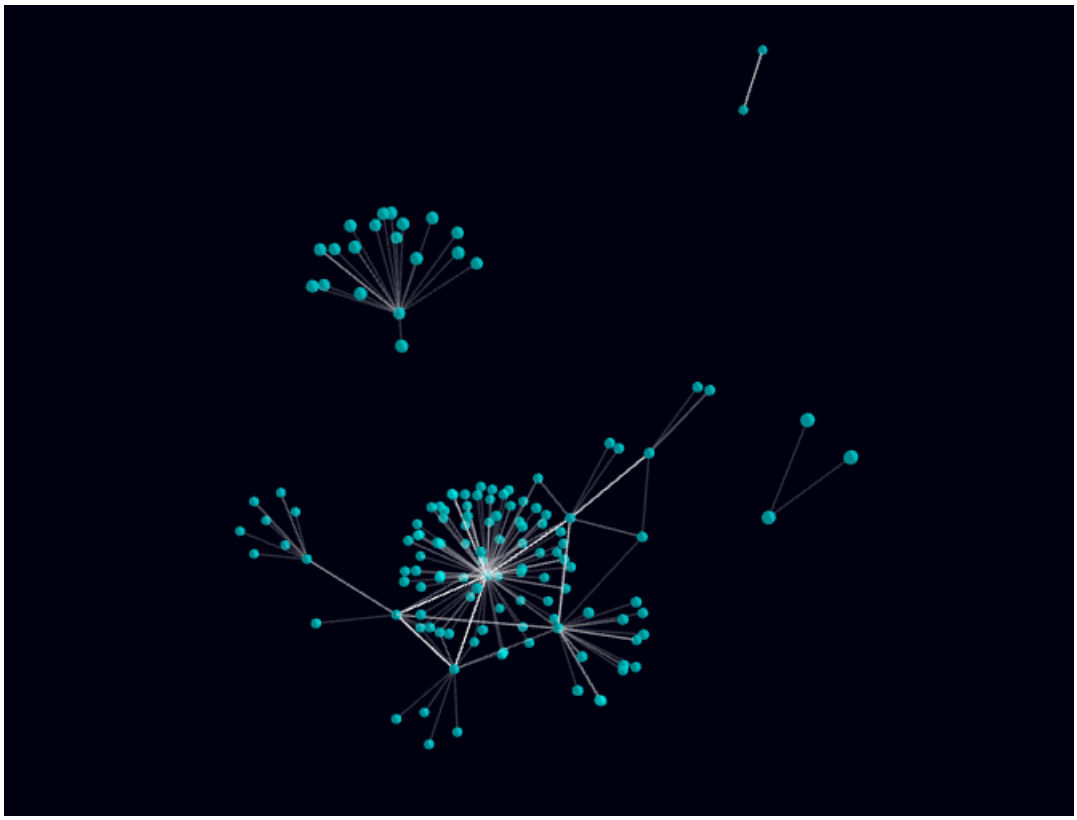


Figura 5.15: Perfil mulher/mulher.

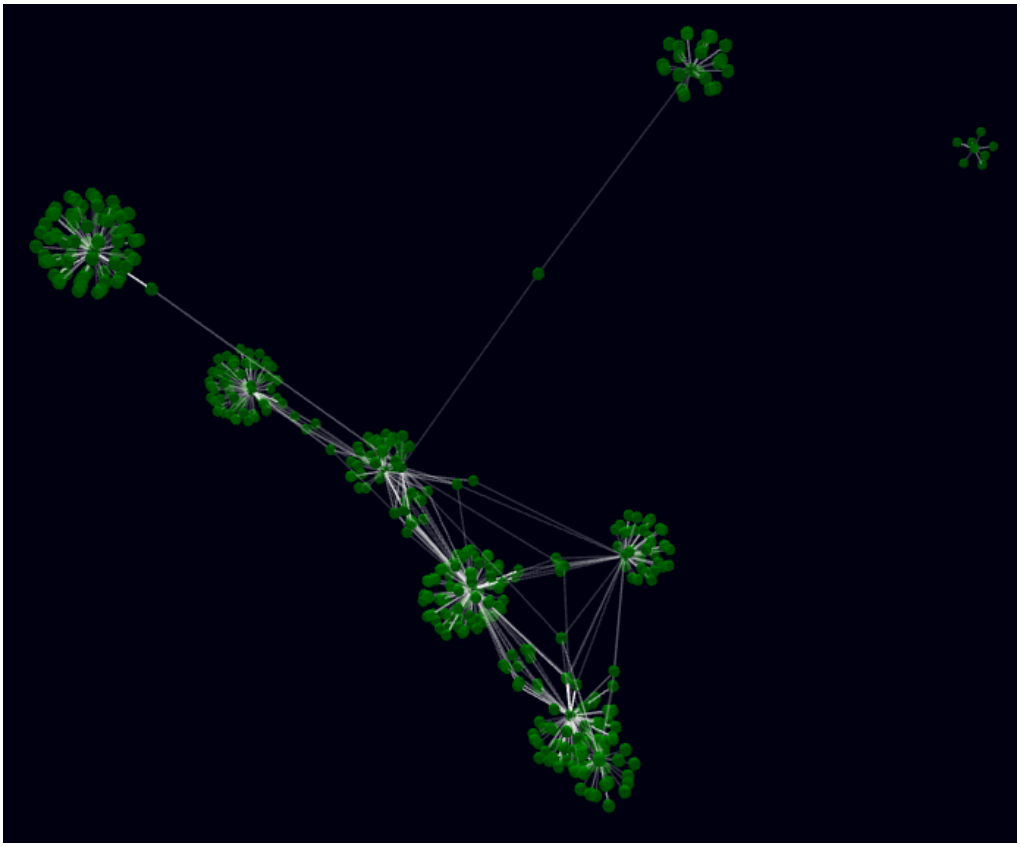


Figura 5.16: Perfil homem/mulher - mulher/homem.

Conforme apresentado, foi possível visualizar alguns aspectos relacionados a gênero do PPGI/UnB, incluindo artigos publicados e coautorias. Nota-se que a presença masculina é maior tanto em publicações quanto coautorias, além que homens publicam mais entre si. Já as mulheres possuem menos relações de autoria e coautoria que os homens, porém possuem um perfil mais colaborativo.

5.4 Projeto ADAN

Os repositórios bibliográficos digitais são uma importante fonte de informações relacionadas às comunidades científicas, tais como, publicações, autores e suas áreas de pesquisa. Essa funcionalidade proporciona uma centralidade de informações e gera confiabilidade nos dados acadêmicos disponibilizados. Em contrapartida, a ambiguidade de nomes de autores afeta de forma significante a busca por informações nesses repositórios sendo também um problema estudado por décadas. A ambiguidade de nomes de autores ocorre quando diferentes autores têm o mesmo registro de nome ou quando um autor tem uma variedade de nomes registrados no repositório. O projeto denominado *ADAN (**A**gent **D**isambiguation **A**uthor **N**ame*) tem como proposta uma solução que utiliza a abordagem

multiagente distribuída para desambiguar nomes de autores em repositórios bibliográficos digitais.

Com base no problema da ambiguidade de nomes de autores em bases digitais, foi coletada a rede de coautoria de 139 pesquisadores que estão no banco de dados do *SCI-synergy* provenientes dos cinco programas de pós-graduação em Ciência da Computação de cinco universidades públicas brasileiras - UnB, UFAM, USP, UFMG e UFRN. O intuito da coleta dessa amostra foi o de comparar os resultados de desambiguação do ADAN com o *SCI-synergy*. A Figura 5.17 apresenta a diferença de desambiguação das abordagens comparadas. Considerando os 139 autores, o *ADAN* conseguiu desambiguar 123 autores e manteve 16 ambíguos. O *SCI-synergy* apresentou 110 autores ambíguos e somente 29 foram desambiguados. Foi possível verificar que do número total de autores utilizados o *ADAN* obteve resultados de desambiguação superiores ao *SCI-synergy*.

O número de registros de sete autores bastante ambíguos do conjunto de dados foi verificado em outros repositórios bibliográficos digitais considerando também os resultados gerados pelo *SCI-synergy* e o *ADAN*. O primeiro repositório bibliografico verificado foi o *Arnetminer* [70] que é definido como um sistema para integrar publicações de bancos de dados da *Web* on-line e que utiliza uma estrutura probabilística para lidar com o problema de ambiguidade de nomes. O segundo repositório bibliográfico utilizado foi o *Microsoft Academic Search* que é definido como um mecanismo de busca acadêmico com mais de 257 milhões de autores e 242 milhões de publicações [71]. A quantidade de registros de nomes de autores comparados entre os repositórios bibliográficos são apresentados na Tabela 5.1. É possível observar que há uma diminuição nos registros de nomes de autores no conjunto de dados do ADAN em comparação com outras abordagens.

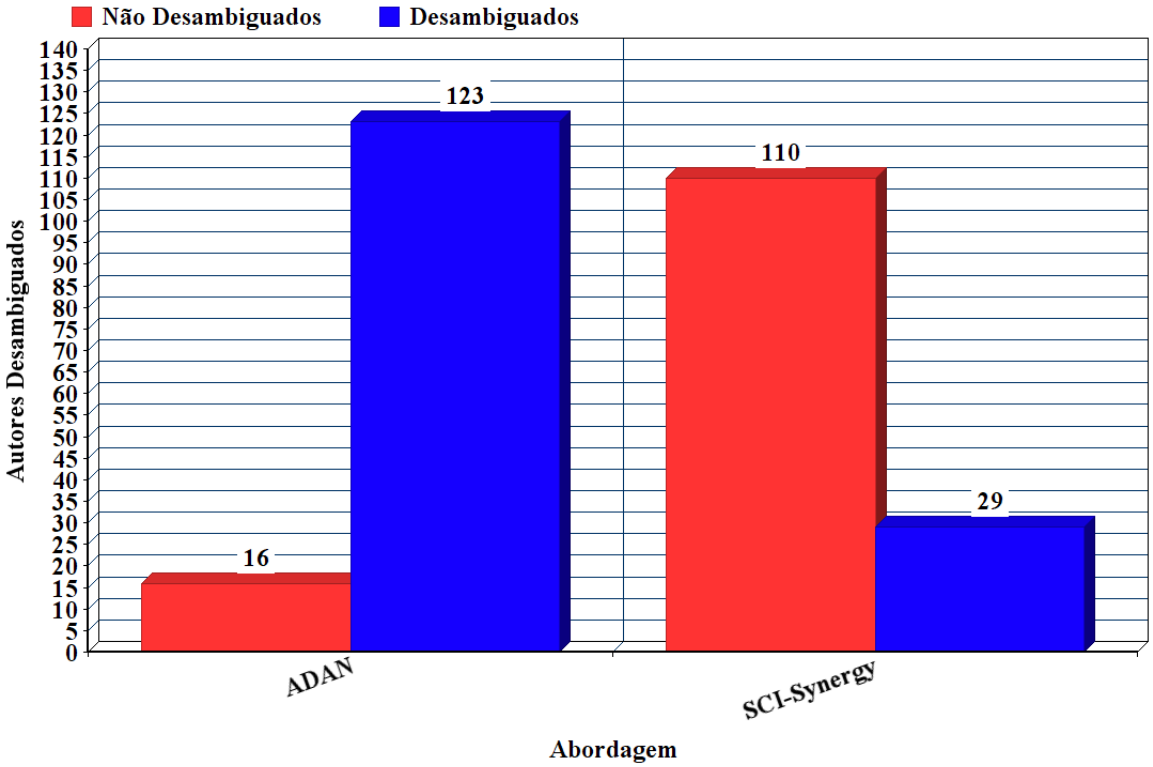


Figura 5.17: Comparação das abordagens de desambiguação do *ADAN* e *SCI-synergy*.

Tabela 5.1: Quantidade de nomes de autores ambíguos em diferentes repositórios.

Nome de autor	**AMiner**	**Microsoft Academic**	***SCI-synergy***	**ADAN**
1 - Virgílio A. F. Almeida	4	4	2	1
2 - Cícero R. F. de Almeida	4	4	2	1
3 - Marcílio C. P. Souto	8	2	2	1
4 - Raquel Minardi	7	3	2	1
5 - Mônica Matzenauer	1	2	4	1
6 - Pedro H. F. Holanda	1	3	2	1
7 - Renato A. C. Ferreira	2	3	2	2
TOTAL	27	21	16	8

5.5 Aspectos da Rede Social do ADAN

Com as informações dos autores utilizados no ADAN armazenadas no banco de dados Neo4j foram feitas consultas e análises que possibilitaram verificar características da rede. Com o intuito de visualizar os resultados apresentados na Tabela 5.1 foi pesquisado no Graph2Vis as redes sociais do ADAN e SCI-synergy com os parâmetros de visualização em modo *custom* (vide Listing 5.14 e Figura 5.20). Como resultado da pesquisa foram apresentadas as redes sociais dos autores e suas respectivas relações de co-autoria, conforme apresentam as Figuras 5.18 e 5.19. Na rede formada pelo ADAN é possível notar que os Autores 1 e 4 possuem maior densidade de relacionamento, enquanto os Autores

2, 3 e 5 estão desconectados. Já na rede do *SCI-synergy* os Autores 1, 4 e 7 são os mais densos, enquanto os Autores 2, 3 e 5 estão desconectados. Porém, todos os autores nessa rede encontram-se repetidos, ou seja, ambíguos. Dessa forma, a visualização dos resultados em forma de grafo reforça os resultados de desambiguação obtida pelo ADAN em relação ao *SCI-synergy*.

Listing 5.14: Consulta CYPHER para a rede de coautoria dos 7 pesquisadores da Tabela 5.1.

```
MATCH (a1:Author)-[co:COAUTHOR]-(a2:Author) WHERE a1.name
    IN ['Virgilio A. F. Almeida', 'Cicero R. F. de Almeida',
     'Marcilio C. P. Souto', 'Raquel Minardi','Monica
    Martzenauer','Pedro H. F. Holanda','Renato A. C.
    Ferrreira'] RETURN a1,co,a2
```

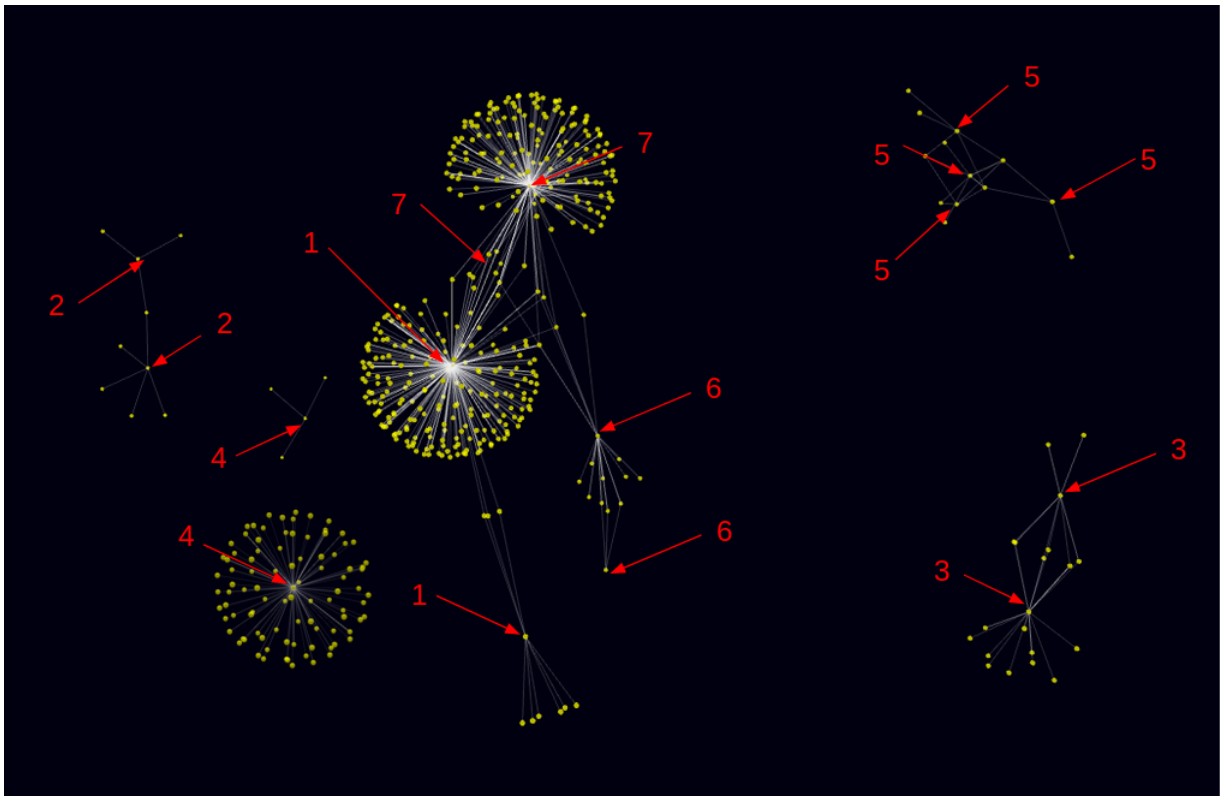


Figura 5.18: SCI-Synergy.

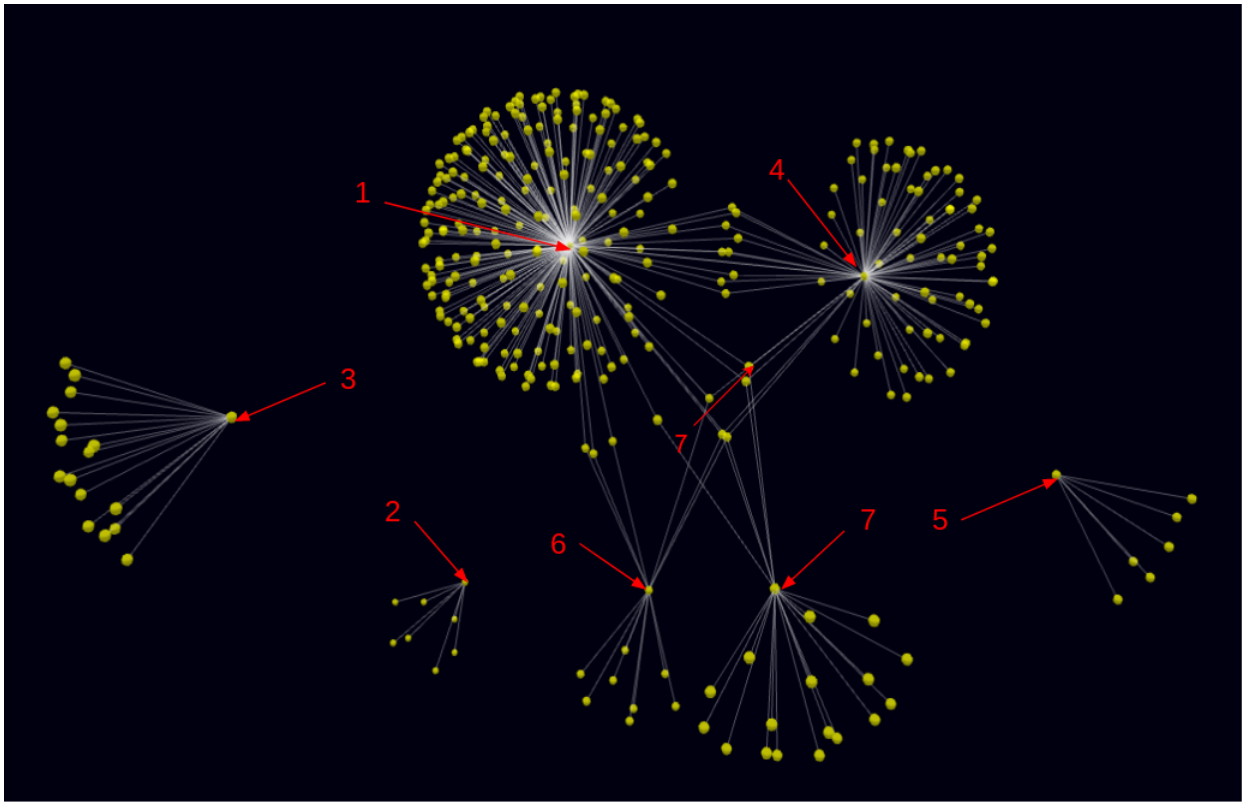


Figura 5.19: ADAN.

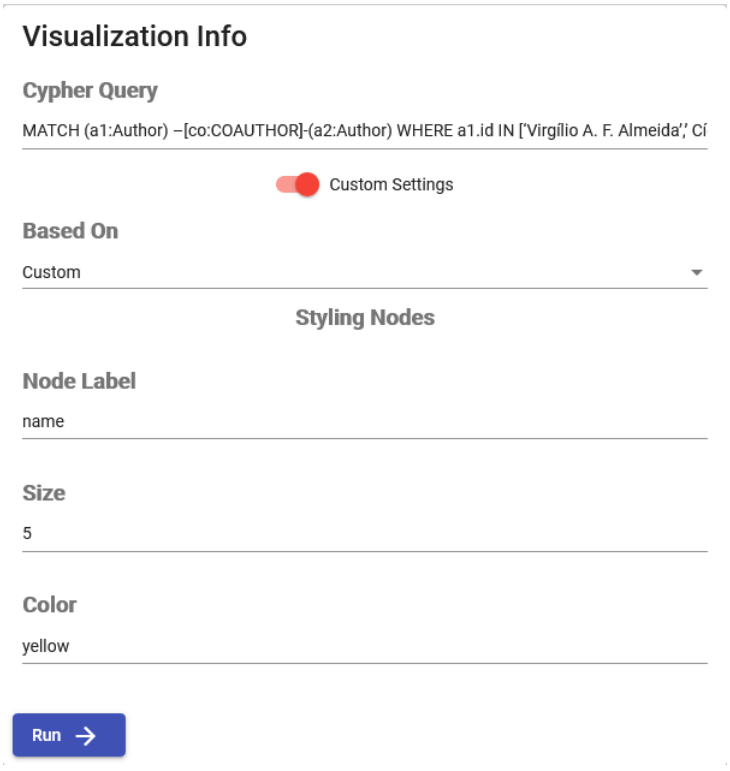


Figura 5.20: Captura de tela da etapa de consulta dos 7 pesquisadores.

Com a visualização da rede social científica do *SCI-synergy* foi possível explorar conceitos de comunidades, graus de relacionamentos entre nós e *clusters* ao longo da rede. Com o olhar dos aspectos de gênero do PPGI/Unb foi possível identificar algumas características sobre o perfil de publicação e colaboração de homens e mulheres. Já a visualização da rede social do *ADAN* explorou um aspecto simples, como a identificação da quantidade de nós presentes. Este capítulo apresentou três estudos de caso e discussões sobre os aspectos das redes aplicadas à solução proposta. O Capítulo 6 contém as conclusões, publicações e trabalhos futuros.

Capítulo 6

Conclusões

A dificuldade de visualização de grande volume de dados é um desafio a ser discutido e estudado. A visualização de dados em forma de grafos também apresenta problemas tal como a dificuldade de análise em redes densas. Em visualizações de grafos em duas dimensões, a sobreposição e/ou oclusão de componentes da rede é uma dificuldade a ser atacada. Neste sentido, este trabalho tratou do projeto, construção e validação da solução *Web* denominada Graph2Vis para atacar o problema da visualização de dados armazenados em banco de dados Neo4j. A solução foi projetada para ser uma ferramenta de auxílio tanto para leigos como pesquisadores que necessitem uma forma de visualização de dados além da textual.

As questões de pesquisa definidas no Capítulo 1 foram respondidas ao longo deste trabalho. A respeito da primeira questão, durante a revisão de literatura, pôde-se notar que a técnica de visualização *node-link* foi a mais utilizada. A escolha dessa técnica deve-se ao poder de visualização de uma quantidade significativa de dados. Já a função de filtragem do grafo foi a mais explorada, pois grafos muito densos requerem esse método para que análises visuais continuem proveitosas. A segunda questão de pesquisa foi respondida durante a descrição da proposta, onde foi descrito o desenvolvimento do Graph2Vis. Já com a aplicação dos três estudos de casos, foi possível responder a terceira questão de pesquisa. Cada estudo de caso ilustrou uma funcionalidade desenvolvida no Grap2Vis.

Foi possível com a realização desse trabalho a construção um artefato de visualização 3D para bases de dados orientadas a grafo usando Neo4j. Neste sentido, o objetivo geral foi atingido. Já com os três estudos de casos aplicados, foi possível validar o uso do Graph2Vis, atingindo assim os objetivos secundários.

Foi possível compreender com a aplicação dos estudos de caso à solução proposta, que o objetivo geral de visualização de dados armazenados em bancos de dados Neo4j e os objetivos secundários 1 e 2 definidos na Seção 1.3 do Capítulo 1 foram alcançados.

6.1 Publicações

Durante a realização deste trabalho, diversas inicitivas de participação vinculadas aos Projetos *SCI-synergy* e *ADAN* foram executadas resultando em publicação científica conforme listadas abaixo. Também foi realizado um projeto de Iniciação Científica através do Edital ProIC 2020/2021 com bolsa CNPq vigente no período de 01/02/2020 a 31/08/2021, com a publicação descrita abaixo.

- de Souza Rodrigues, N., Costa, A.R., Lemos, L.C., Ralha, C.G. (2021). Multi-strategic Approach for Author Name Disambiguation in Bibliography Repositories. In: Lossio-Ventura, J.A., Valverde-Rebaza, J.C., Díaz, E., Alatrista-Salas, H. (eds) Information Management and Big Data. SIMBig 2020. Communications in Computer and Information Science, vol 1410. Springer, Cham. https://doi.org/10.1007/978-3-030-76228-5_5

- Lemos, L.C. e Ralha, C.G. (2021). Estudo, Análise e Implementação de Interfaces Visuais para Redes Sociais Científicas. In: 27^o Congresso de Iniciação Científica da UnB e do 18^o Congresso de Iniciação Científica do DF. 27 setembro a 1^o outubro, Brasília, DF, Brasil. Disponível em: `https://conferencias.unb.br/index.php/iniciacaocientifica/index/schedConfs/current`

Além das publicações científicas acima listadas foram apresentados dois artigos em evento local organizados pelo Programa de Pós-Graduação em Informática (PPGI) e Programa de Computação Aplicada (PPCA) do Departamento de Ciência da Computação da UnB:

- Lemos, L.C. e Ralha, C.G. WPOS/WCOMP 2020. Estudo Experimental de Ferramentas de Visualização para Rede Social Científica. Universidade de Brasília, Brasília, DF, Brasil. Disponível em: `http://wpos.unb.br/`

- Lemos, L.C. e Ralha, C.G. WPOS/WCOMP 2021. Estudo, Análise e Implementação de Interfaces Visuais Para Rede Social Científica. Universidade de Brasília, Brasília, DF, Brasil. Disponível em: `http://wpos.unb.br/`

6.2 Trabalhos Futuros

No atual estado do protótipo Graph2Vis especificamente no modo de visualização *property* é possível inserir os parâmetros de tamanho e cores de nós utilizando medidas de centralidade previamente calculadas e armazenadas no banco de dados Neo4j (i.e., *degree*, *louvain*). Entretanto, seria de grande valia a possibilidade desses cálculos serem feitos

pela próprio Graph2Vis. Tal funcionalidade poderia ser útil tanto para usuários leigos quanto pesquisadores.

A solução desenvolvida permite tanto evoluções relacionadas a adição e/ou manutenção de funcionalidades, quanto na criação de novas estruturas e/ou aplicações que utilizem o Graph2Vis. Com foco nas funcionalidades, foram desenvolvidas as básicas para que a solução pudesse resolver o problema de visualização de dados no Neo4j.

Funcionalidades de uso pela integração de outras bibliotecas de visualização ao Graph2Vis é um aspecto a ser trabalhado futuramente. Cada biblioteca requer um modelo específico de dados a ser usado como entrada para gerar seus grafos, considerando as bibliotecas estudadas (*Vis, Neovis, Popoto, Sigma, Vivagraph*). Além disso, um conjunto de visualizações suplementares como matrizes e listas seriam úteis no sentido de potencializar as análises a respeito dos grafos gerados.

Por fim, outras funcionalidades que permitam melhorar o desempenho da solução podem ser implementadas de maneira evolutiva. Neste sentido, pode-se citar a pesquisa e/ou desenvolvimento de um algoritmo específico para melhor organização dos elementos em grafos densos. Na atual solução, o algoritmo implementado pelo *3d-force-graph* tem mostrado dificuldade na disposição dos elementos em cenários de grafos densos.

Referências

[1] Sipser, Michael: *Introduction to the Theory of Computation*. International Thomson Publishing, 2a edição, 2006, ISBN 053494728X. ix, 6

[2] Neo4j Inc.: *Graph your twitter activity in neo4j!* Disponível em: `http://network.graphdemos.com/`, 2022. ix, 9

[3] Foong, Ng Wai: *The beginner's guide to the neo4j graph platform*. Disponível em: `https://betterprogramming.pub/the-beginners-guide-to-the-neo4j-graph-platform-a39858ccdeaa`, 2020. ix, 12

[4] Lopes, Bruno: *Classes de domínio: Entendendo o modelo mvc*. Disponível em: `https://vaidegrails.wordpress.com/2015/06/19/classes-de-dominio-entendendo-o-modelo-mvc/`, 2015. ix, 15

[5] Bludau, Mark Jan, Marian Dörk e Christian Tominski: *Unfolding Edges for Exploring Multivariate Edge Attributes in Graphs*. Em Byška, Jan, Stefan Jänicke e Johanna Schmidt (editores): *EuroVis 2021 - Posters*. The Eurographics Association, 2021, ISBN 978-3-03868-144-1. ix, 18, 19

[6] Bulao, Jacquelyn: *How much data is created every day in 2021?* Disponível em: `https://techjury.net/blog/how-much-data-is-created-every-day/#gref`. Acessado em: 9 Novembro de 2021. 1

[7] Ian Robinson, Jim Webber, Emil Eifrem: *Graph Databases*. O Reilly Media, 1ª edição, 2013. `http://gen.lib.rus.ec/book/index.php?md5=552688af09a579aa8a88705c25dc4321`. 1

[8] Inc, Neo4j: *Top ten reasons for choosing neo4j*. `https://neo4j.com/top-ten-reasons/`. Acessado em: 12 Novembro de 2021. 1

[9] Inc., Neo4j: *Neo4j - a nosql a relational database management system*. Disponível em: `https://neo4j.com`. Acessado em: 7 Novembro de 2021. 1

[10] Inc., Neo4j: *Neo4j browser user interface guide*. `https://neo4j.com/developer/neo4j-browser/`. Acessado em: 12 Novembro de 2021. 1, 12

[11] Lanum, Corey: *Visualizing Graph Data*. Manning Publications, 1ª edição, 2016, ISBN 1617293075,9781617293078. 1

[12] Salesforce: *Db-engines: Knowledge base of relational and nosql database management systems*. Disponível em: `https://db-engines.com/en/ranking`, 2022. 2

[13] Battista Giuseppe di, Peter Eades e Roberto Tamassia: *What is data visualization? definition, examples, and learning resources*. Disponível em: `https://www.tableau.com/learn/articles/data-visualization`, 2022. 2

[14] Hühne, Rolf, Viktor Kessler, Axel Fürstberger, Silke Kühlwein, Matthias Platzer, Jürgen Sühnel, Ludwig Lausser e Hans Kestler: *3D Network exploration and visualisation for lifespan data*. BMC Bioinformatics, 19(390), 2018. 2, 18, 26

[15] Thomas, Jim, Kris Cook, Vern Crow, Beth Hetzler, Richard May, Dennis McQuerry, Renie McVeety, Nancy Miller, Grant Nakamura, Lucy Nowell, Paul Whitney e Pak Chung Wong: *Human—Computer Interaction with Global Information Spaces — Beyond Data Mining*, páginas 32–46. Springer London, London, 2000, ISBN 978-1-4471-3646-0. `https://doi.org/10.1007/978-1-4471-3646-0_3`. 4

[16] Thomas, Jim e Kris Cook: *A visual analytics agenda*. IEEE Computer Graphics and Applications, 26(1):10–13, 2006. 4

[17] Antweiler, Dario, David Sessler, Sebastian Ginzel e Jörn Kohlhammer: *Towards the detection and visual analysis of covid-19 infection clusters*. The Eurographics Association, janeiro 2021. 4, 18, 22

[18] Cardoso, J. N. D. O.: *Visualizando dados do bolsa família em forma de grafo*. Em *semanticscholar*, 2018. 4

[19] Khine, Myint Swe: *Spatial cognition: Key to STEM success*, páginas 3–8. Visual-spatial Ability in STEM Education: Transforming Research into Practice. Springer, Cham, October 2017, ISBN 978-3-319-44384-3. Doi 10.1007/978-3-319-44385-0_1. 4

[20] Keim, Daniel A., Florian Mansmann, Jörn Schneidewind, James J. Thomas e Hartmut Ziegler: *Visual analytics: Scope and challenges*. Em *Visual Data Mining*, 2008. 4

[21] Freeman, Linton C.: *Centrality in social networks conceptual clarification*. Social Networks, 1(3):215–239, 1978, ISSN 0378-8733. `https://www.sciencedirect.com/science/article/pii/0378873378900217`. 7

[22] Blondel, Vincent D, Jean Loup Guillaume, Renaud Lambiotte e Etienne Lefebvre: *Fast unfolding of communities in large networks*. Journal of Statistical Mechanics: Theory and Experiment, 2008(10):P10008, oct 2008. 7

[23] Lancichinetti, Andrea e Santo Fortunato: *Community detection algorithms: A comparative analysis*. Physical Review E, 80(5), nov 2009. 7

[24] Wasserman, Stanley e Katherine Faust: *Social Network Analysis: Methods and Applications*. Structural Analysis in the Social Sciences. Cambridge University Press, 1994. 8

[25] Cunha Recuero, Raquel da: *Redes sociais na internet: Considerações iniciais*. E-Compós, 2, 2009. 8, 9

[26] Zhang, Lei e Wanqing Tu: *Six degrees of separation in online society*. Em *2009 Web Science, Greece*, 2009. 10

[27] Strozzi, Carlo: *Nosql a relational database management system*. Disponível em: `http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/NoSQL/Home%20Page`. Acessado em: 7 Novembro de 2021. 10

[28] Hauger, W. e M. Olivier: *Nosql databases: Forensic attribution implications*. SAIEE Africa Research Journal, 109:119–132, 2018. 10

[29] Haseeb, Abdul e Geeta Pattun: *A review on nosql: Applications and challenges*. International Journal of Advanced Research in Computer Science, 8:203–207, 2017. 10

[30] Fowler, Adam: *NoSQL For Dummies*. 1ª edição, 2015. `http://gen.lib.rus.ec/book/index.php?md5=2afea990c282e97644b1fafcb6223852`. 10, 11

[31] Vukotic, Aleksa, Nicki Watt, Tareq Abedrabbo, Dominic Fox e Jonas Partner: *Neo4j in Action*. Manning, 2014, ISBN 978-1617290763. `http://gen.lib.rus.ec/book/index.php?md5=a417ec90148e5b8453b1f01f1bc15fa0`. 11

[32] Ivan, P.: *Wayblazer cognitive computing application powered by ibm watson e neo4j*. Disponível em: `https://neo4j.com/blog/wayblazer-watson-neo4j/`. Acessado em: 8 Novembro de 2021. 11

[33] Neo Technology: *Cypher query language*. Disponível em: `hhttps://neo4j.com/developer/cypher/`. Acessado em: 8 Novembro de 2021. 12, 13

[34] Neo4j Community: *The neo4j graph data science library manual*. Disponível em: `https://neo4j.com/docs/graph-data-science/current/`, 2022. 12

[35] Corporative, Mozilla: *Mvc*. Disponível em: `https://developer.mozilla.org/en-US/docs/Glossary/MVC`, 2022. 14

[36] Chen, Yi, Zeli Guan, Rong Zhang, Xiaomin Du e Yunhai Wang: *A survey on visualization approaches for exploring association relationships in graph data*. Journal of Visualization, 22, abril 2019. 16, 18, 31

[37] Rahman, Marufur e Rezaul Karim: *Comparative study of different methods of social network analysis and visualization*. Em *2016 International Conference on Networking Systems and Security (NSysS)*, páginas 1–7, 2016. 18, 20

[38] Tallat, Raiha, Rana M. Amir Latif, Ghazanfar Ali, Ahmad Nawaz Zaheer, Muhammad Farhan e Syed Umair Aslam Shah: *Visualization and analytics of biological data by using different tools and techniques*. Em *2019 16th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, páginas 291–303, 2019. 18, 21

[39] Farooq, Aftab, Gulraiz Javaid Joyia, Muhammad Uzair e Usman Akram: *Detection of influential nodes using social networks analysis based on network metrics*. Em *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, 2018. 18, 21

[40] Kumar, Vishal e Evelyn Ai Lin Evelyn Teo: *Exploring the application of property graph model in visualizing cobie data*. Journal of facilities management, 19(4):500–526, 2021, ISSN 1472-5967. 18, 22

[41] Summer, Georg, Thomas Kelder, Keiichiro Ono, Marijana Radonjic, Stephane Heymans e Barry Demchak: *cyNeo4j: connecting Neo4j and Cytoscape*. Bioinformatics, 31(23):3868–3869, 2015, ISSN 1367-4803. 18, 22

[42] Yokoyama, Toshiyuki T., Masashi Okada e Tadahiro Taniguchi: *Panacea: Visual exploration system for analyzing trends in annual recruitment using time-varying graphs*. PLOS ONE, 16(3):1–22, 2021, ISSN 1932-6203. `http://dx.doi.org/10.1371/journal.pone.0247587`. 18, 23

[43] García del Valle, Eduardo P., Gerardo Lagunes García, Lucía Prieto Santamaría, Massimiliano Zanin, Ernestina Menasalvas Ruiz e Alejandro Rodríguez-González: *DisMaNET: A network-based tool to cross map disease vocabularies*. Computer Methods and Programs in Biomedicine, 207:106233, 2021, ISSN 0169-2607. `https://www.sciencedirect.com/science/article/pii/S0169260721003072`. 18, 23

[44] Jo, Sunhwa, Beomjun Park, Suan Lee e Jinho Kim: *OLGAVis: On-Line Graph Analysis and Visualization for Bibliographic Information Network*. Applied Sciences, 11(9), 2021, ISSN 2076-3417. `https://www.mdpi.com/2076-3417/11/9/3862`. 18, 23

[45] Hassani-Pak, Keywan, Ajit Singh, Marco Brandizi, Joseph Hearnshaw, Jeremy D Parsons, Sandeep Amberkar, Andrew L Phillips, John H Doonan e Chris Rawlings: *KnetMiner: a comprehensive approach for supporting evidence-based gene discovery and complex trait analysis across species*. Plant Biotechnology Journal, 19(8):1670–1678, 2021, ISSN 1467-7644. 18, 24

[46] Bukhari, Syed Ahmad Chan, Shrikant Pawar, Jeff Mandell, Steven H Kleinstein e Kei Hoi Cheung: *LinkedImm: a linked data graph database for integrating immunological data*. BMC bioinformatics, 22(S9):105–105, 2021, ISSN 1471-2105. 18, 24

[47] Carnaz, Gonçalo, Vitor Beires Nogueira e Mário Antunes: *A graph database representation of portuguese criminal-related documents*. Informatics, 8(2), 2021, ISSN 2227-9709. `https://www.mdpi.com/2227-9709/8/2/37`. 18, 24

[48] Zahoránszky-Kőhalmi, Gergely, Timothy Sheils e Tudor Oprea: *SmartGraph: a Network Pharmacology Investigation Platform*. Journal of Cheminformatics, 12(5), 2020. 18, 25

[49] Müller, Richard, Dirk Mahler, Michael Hunger, Jens Nerche e Markus Harrer: *Towards an open source stack to create a unified data source for software analysis and visualization*. Em *2018 IEEE Working Conference on Software Visualization (VISSOFT)*, páginas 107–111, 2018. 18, 25

[50] Kerzner, Ethan, Alexander Lex, Crystal Lynn Sigulinsky, Timothy Urness, Bryan William Jones, Robert E. Marc e Miriah Meyer: *Graffinity: Visualizing connectivity in large graphs*. Computer Graphics Forum (EuroVis), 36(3):251–260, 2017, ISSN 0167-7055. `https://doi.org/10.1111/cgf.13184`. 18, 26

[51] Partl, Christian, Samuel Gratzl, Marc Streit, Anne Mai Wassermann, Hanspeter Pfister, Dieter Schmalstieg e Alexander Lex: *Pathfinder: Visual analysis of paths in graphs.* Computer Graphics Forum (EuroVis '16), 35(3):71–80, 2016. 18, 27

[52] Rodrigues, Jose, Hanghang Tong, Agma Traina, Christos Faloutsos e Jure Leskovec: *GMine: A System for Scalable, Interactive Graph Visualization and Mining.* 2015. 18, 27

[53] Cockburn, Andy, Amy Karlson e Benjamin B. Bederson: *A review of overview+detail, zooming, and focus+context interfaces.* ACM Comput. Surv., 41(1), janeiro 2009, ISSN 0360-0300. https://doi.org/10.1145/1456650.1456652. 19

[54] Krzywinski, Martin I, Jacqueline E Schein, Inanc Birol, Joseph Connors, Randy Gascoyne, Doug Horsman, Steven J Jones e Marco A Marra: *Circos: An information aesthetic for comparative genomics.* Genome Research, 2009. http://genome.cshlp.org/content/early/2009/06/15/gr.092759.109.abstract. 21

[55] Vis Community: *Vis js - community edition.* Disponível em: https://visjs.org/, 2021. 27

[56] Ventimiglia, Tom e Kevin Wayne: *The barnes-hut algorithm.* Disponível em: http://arborjs.org/docs/barnes-hut, 2011. 28

[57] Jacomy, Mathieu, Tommaso Venturini, Sebastien Heymann e Mathieu Bastian: *Forceatlas2, a continuous graph layout algorithm for handy network visualization designed for the gephi software.* Disponível em: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0098679, 2014. 28

[58] Neo4j Community: *Neovis js.* Disponível em: https://github.com/neo4j-contrib/neovis.js/, 2021. 28

[59] Interactive, NHOGS: *Popoto js.* Disponível em: http://www.popotojs.com/, 2021. 29

[60] Bostock, Mike: *D3 js.* Disponível em: https://d3js.org, 2021. 29

[61] Jacomy, Alexis e Guillaume Plique.: *Sigma js.* Disponível em: http://www.sigmajs.org/, 2021. 30

[62] Kashcha, Andrei: *Vivagraph js.* Disponível em: https://github.com/anvaka/VivaGraphJS, 2021. 30

[63] Asturiano, Vasco: *3d-force, a web component to represent a graph data structure in a 3-dimensional space.* Disponível em: https://github.com/vasturiano/3d-force-graph, 2021. 31

[64] Gamma, Erich, Richard Helm e Ralph E. Johnson: *Design Patterns. Elements of Reusable Object-Oriented Software.* Addison-Wesley Longman, Amsterdam, 1st ed., reprint. edição, 1994, ISBN 0201633612. http://www.javier8a.com/itc/bd1/articulo.pdf. 37

[65] Object Management Group®, Inc: *What is uml*. Disponível em: https://www.uml.org/what-is-uml.htm, 2022. 38

[66] Angular University: *Angular for beginners guide: Why angular? understanding the top benefits*. Disponível em: https://blog.angular-university.io/why-angular-angular-vs-jquery-a-beginner-friendly-explanation-on-the-advantages-of-angular-and-mvc/, 2022. 38

[67] Battista Giuseppe di, Peter Eades e Roberto Tamassia: *Graph drawing: Algorithms for the visualization of graphs*. Disponível em: https://visjs.org/, 1999. 42

[68] Angular University: *Typescript configuration*. Disponível em: https://angular.io/guide/typescript-configuration, 2022. 42

[69] Nunes, Thayanna Klysnney, Aleteia Araujo e Maristela Holanda: *Mulheres na pós-graduação nas Áreas de exatas: Um estudo de caso na universidade de brasília*. Em *Anais do XIV Women in Information Technology*, páginas 244–248, Porto Alegre, RS, Brasil, 2020. SBC. https://sol.sbc.org.br/index.php/wit/article/view/11303. 50

[70] AMiner: *Search and mining of academic social networks*. https://www.aminer.org/, 2005-2021. Tsinghua University, Beijing, 100084. China. 57

[71] Ortega, J. L. e I. F. Aguillo: *Microsoft academic search and google scholar citations: Comparative analysis of author profiles*. Journal of the Association for Information Science and Technology, 65(6):1149–1156, 2014. 57

Apêndice A

Tabela de atributos das classes do Graph2Vis

As Tabelas A.1, A.2, A.3, A.4, A.5 apresentam os atributos contidos nas classes *ConnectionFormComponent, Neo4jService, VisualizationFormComponent, DataService e GraphComponent* referentes ao diagrama de classes do Graph2Vis apresentado na Figura 4.7.

Atributo	**Significado**
myForm	objeto que representa as informações inseridas no formulário de conexão banco de dados.
spinner	objeto que representa a interface de *loading* ao acessar o banco de dados
hide	variável booleano utilizado para aplicar máscara a credencial de acesso.

Tabela A.1: Atributos da classe *ConnectionFormComponent*

Atributo	**Significado**
driver	objeto que realiza a conexão ao banco de dados Neo4j

Tabela A.2: Atributos da classe *Neo4jService*

Atributo	**Significado**
myForm	objeto que representa as informações inseridas no formulário de consulta ao banco de dados e parâmetros de visualização

Tabela A.3: Atributos da classe *VisualizationFormComponent*

Atributo	Significado
form	objeto que representa as informações inseridas no formulário de consulta ao banco de dados e parâmetros de visualização
obj	objeto genérico a ser salvo na classe.

Tabela A.4: Atributos da classe *DataService*

Atributo	Significado
form	objeto que representa as informações inseridas no formulário de consulta ao banco de dados e parâmetros de visualização
rawGraphData	objeto que representa os dados não processados provenientes da consulta ao banco de dados
finalGraph	objeto que representa os dados processados após consulta realizada ao banco de dados

Tabela A.5: Atributos da classe *GraphComponent*