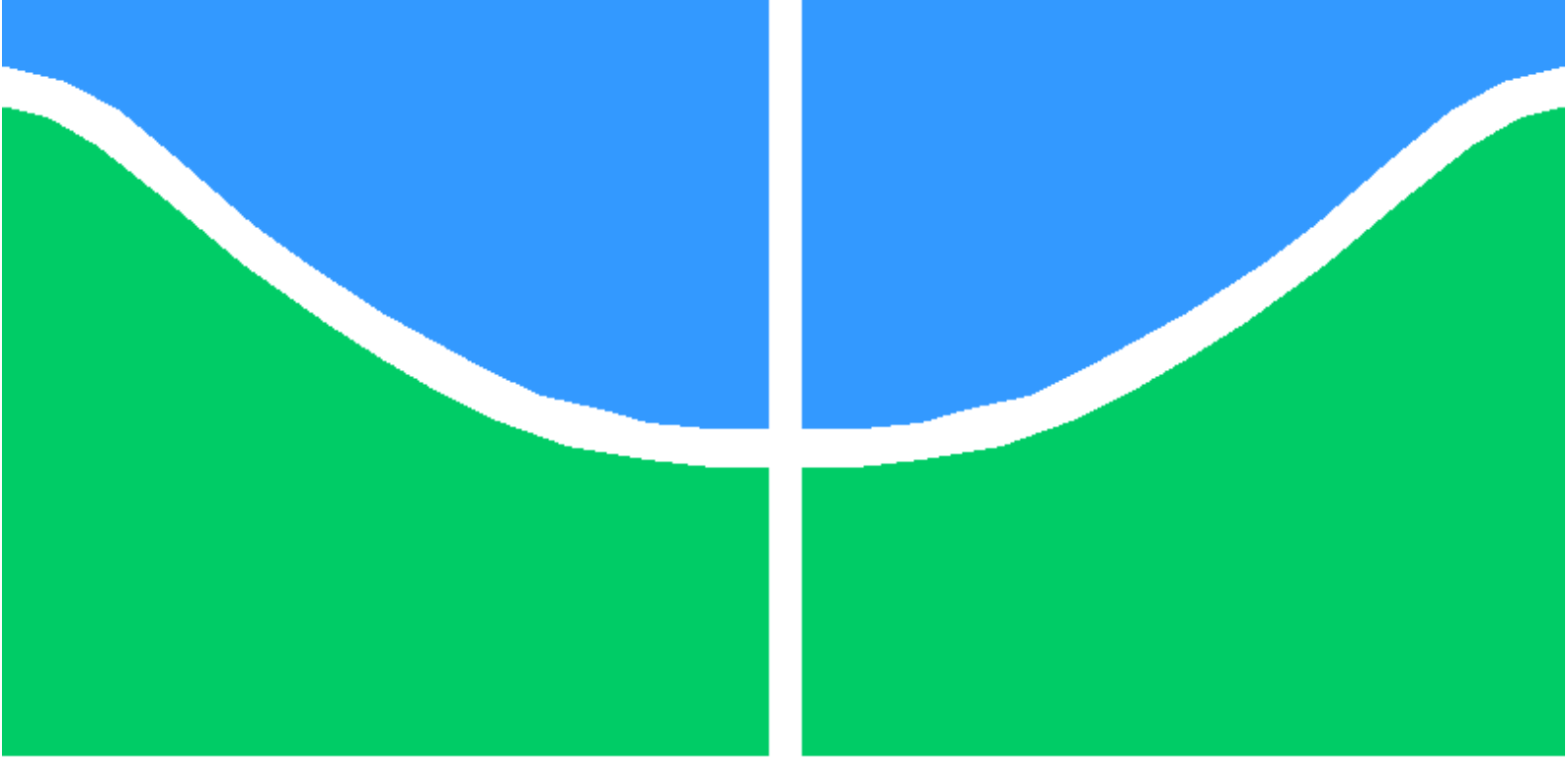

TRABALHO DE GRADUAÇÃO

PROPOSTA DE APLICAÇÃO WEB PARA ANÁLISE DE DADOS ABERTOS USANDO UM BANCO DE DADOS ORIENTADO A GRAFOS

LUIS FILIPE CAMPOS CARDOSO

Brasília, 07 de julho de 2017

UNIVERSIDADE DE BRASILIA

Faculdade de Tecnologia

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia de Redes de Comunicação

TRABALHO DE GRADUAÇÃO

PROPOSTA DE APLICAÇÃO WEB PARA ANÁLISE DE DADOS ABERTOS USANDO UM BANCO DE DADOS ORIENTADO A GRAFOS

Luis Filipe Campos Cardoso

Relatório submetido ao Departamento de Engenharia
Elétrica como requisito parcial para obtenção do grau de
Engenheiro de Redes de Comunicação.

Banca Examinadora

Prof. Georges Daniel Amvame Nze, Dr, UnB/ ENE (Orientador)	
Prof. Rafael Timóteo de Sousa Júnior, Dr, UnB/ ENE (Examinador Interno)	
Robson de Oliveira Albuquerque, Dr, (Examinador Externo)	

Brasília, 07 de julho de 2017

FICHA CATALOGRÁFICA

Campos Cardoso, Luis Filipe

Proposta de Aplicação Web para Análise de Dados Abertos Usando um Banco de Dados Orientado a Grafos / Luis Filipe Campos Cardoso. -- Brasília, 2017.

55 f.

Orientador: Prof. Georges Daniel Amvame Nze, Dr.

TCC (Graduação - Engenharia de Redes de Comunicação) -- Universidade de Brasília, Faculdade de Tecnologia (Departamento de Engenharia Elétrica), 2017.

1. Visualização de Informação. 2. Banco de Dados Orientado a Grafos. 3. Aplicação Web Cliente. 4. Usabilidade. I. Prof. Georges Daniel Amvame Nze, Dr., orientador. II. Proposta de Aplicação Web para Análise de Dados Abertos Usando um Banco de Dados Orientado a Grafos.

REFERÊNCIA BIBLIOGRÁFICA

CAMPOS CARDOSO, L. F., (2017). Proposta de Aplicação Web para Análise de Dados Abertos Usando um Banco de Dados Orientado a Grafos. Trabalho de Graduação em Engenharia de Rede de Comunicação, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 55p.

CESSÃO DE DIREITOS

AUTOR: Luis Filipe Campos Cardoso.

TÍTULO DO TRABALHO DE GRADUAÇÃO: Proposta de Aplicação Web para Análise de Dados Abertos Usando um Banco de Dados Orientado a Grafos.

GRAU: Engenheiro ANO: 2017

_____

Luis Filipe Campos Cardoso

Brasília – DF - Brasil.

AGRADECIMENTOS

Aos meus pais, por me propiciar um lar com amor e de bons valores e por me dar apoio incondicional em todas as minhas decisões. Além de estarem sempre presentes.

A minha família, por ser acolhedora e unida.

Aos meus amigos, por serem pessoas fantásticas, as quais pude compartilhar grandes momentos da minha vida.

Ao meu orientador, o Professor Georges Daniel Amvame Nze, por sempre se dispor a me auxiliar e a me guiar para a conclusão desse projeto e por me incentivar nos momentos de dificuldade.

Ao Robson de Oliveira, por me apresentar o banco de dados orientado a grafos e a visualização de dados.

Ao Leonardo Hideki, por fazer a logo da Ana Lisa.

A Universidade de Brasília por me propiciar experiências incríveis tanto academicamente como pessoal que me enriqueceram muito como ser humano. Além disso, esse ambiente diverso e plural me encantou muito e me dará muita saudade.

A Capes, por ter financiado uma das melhores experiências da minha vida que foi meu intercâmbio para os EUA.

“A informação é a resolução da incerteza”.

Claude Shannon

RESUMO

O presente trabalho apresenta o desenvolvimento de uma aplicação Web cliente, a Ana Lisa, para inserir e visualizar dados em um formato de grafo. Seu principal objetivo é trazer um modo de interpretar os dados de forma mais clara, a fim de criar relações entre as entidades executando um processo de investigação para o melhoramento do entendimento dos dados. Esse software permite, a partir de um arquivo CSV (como uma forma de estruturar os dados de entrada), inserir as entidades em um banco de dados orientado a grafos e criar os relacionamentos entre elas, utilizando a ontologia como uma maneira de se estabelecer a semântica do que se está sendo visualizado. Para saber qual seria o banco de dados orientado a grafo escolhido no trabalho, foi feita uma comparação entre o Neo4j e o OrientDB, onde se estabeleceu suas vantagens e suas características para guiar a opção de qual SGBD usar. No seu desenvolvimento, utilizou-se bibliotecas baseadas em JavaScript e CSS de visualização, de construção de interface e de transformação de dados. Por fim, a Usabilidade foi empregada na implantação do software para garantir uma experiência do usuário adequada aos objetivos da Ana Lisa.

Palavras Chave: Visualização de Informação, Banco de Dados Orientado a Grafos, Aplicação Web Cliente, Usabilidade.

ABSTRACT

The present work shows the development of a client-side Web application to upload and visualize data in a graph format. The name of the software is Ana Lisa. Its main purpose is to provide a way of interpreting data more clearly to create relationships between the input entities performing a process of research to improve the data understanding. This software allows, from a CSV file (as a way of structuring the input data), to insert entities in a graph database and create the relationships between them, using ontology concepts to establish the Semantics of what is being visualized. To choose a graph database that best fits this job, a comparison was made between Neo4j and OrientDB, to determine their advantages and their characteristics. In its development, this work used libraries based on JavaScript and CSS for visualization, interface construction and data transformation. Finally, Usability played an important role in the software development process to ensure a user experience appropriate to Ana Lisa's goals.

Keywords: Information Visualization; Graph Database; Client Web Application; Usability.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Unidades de Medida

ºC	Medida de Temperatura	Graus Celsius

Siglas

ABNT	Associação Brasileira de Normas Técnicas
JSON	*JavaScript Object Notation* - Notação de Objetos JavaScript
CSV	*Comma-separated Values* – Valores Separados por Vírgula
RDF	*Resource Description Framework* - Framework de Descrição de Recursos
IoT	*Internet of Things* – Internet das Coisas
API	*Application Programming Interface* - Interface de Programação de Aplicação
BD	Banco de Dados
SVG	*Scalable Vector Graphics* - Gráficos Vetoriais Escaláveis
TCC	Trabalho de Conclusão de Curso
HTML	*HyperText Markup Language* - Linguagem de Marcação de Hipertexto
NoSQL	*Not Only SQL* - Não Somente SQL
SGBDR	Sistema de Gerenciamento de Banco de Dados Relacional
ACID	*Atomicity, Consistency, Isolation and Durability* - Atomicidade, Consistência, Isolamento e Durabilidade
SQL	*Structured Query Language* - Linguagem de Consulta Estruturada
BASE	*Basically Available, Soft state and Eventual consistency* - Disponibilidade Básica, Estado Maleável e Consistência Eventual
CRUD	*Create, Read, Update and Delete* - Criar, Ler, Atualizar e Apagar
OLTP	*Online Transaction Processing* - Processamento de Transações em Tempo Real
SGBD	Sistema de Gerenciamento de Banco de Dados
SPARQL	*SPARQL Protocol and RDF Query Language* - Protocolo e linguagem de consulta RDF
Web	*World Wide Web* – Rede Mundial de Computadores
XML	*eXtensible Markup Language* - Linguagem de Marcação Extensiva
ISO	*International Organization for Standardization* - Organização Internacional para Padronização
IEC	*International Electrotechnical Commission* - Comissão Eletrotécnica Internacional
NBR	Norma da ABNT

HCI	*Human–Computer Interaction* – Interação Humano-Computador
CSS	*Cascading Style Sheets* - Folhas de Estilo em Cascata
AJAX	*Asynchronous Javascript and XML* - Javascript Assíncrono e XML
DOM	*Document Object Model* - Modelo de Objeto de Documentos
ID	*Identification* – Identificação
HTML5	*HyperText Markup Language (Version 5)* - Linguagem de Marcação de Hipertexto (Versão 5)
REST	*Representational State Transfer* - Transferência de Estado Representacional

1 - INTRODUÇÃO

Este capítulo apresenta as considerações gerais preliminares desse trabalho sobre análise de dados e banco de dados orientados a grafos, destacando os problemas a serem solucionados e os aspectos gerais. Além disso, serão explicitados os objetivos e a estrutura do trabalho em questão.

1.1 ASPECTOS GERAIS E PROBLEMAS

Na Era da Informação, extrair dados úteis e interpretá-los a fim de gerar informação e, por sua vez, conhecimento é um grande desafio [28]. Ainda mais quando se está tratando de grandes massas de dados que são atualizadas constantemente [1]. As redes sociais são um excelente exemplo de aplicação que necessitam um gerenciamento de uma grande quantidade de dados não estruturados gerados continuamente [39]. Uma alternativa viável para a manipulação e para o armazenamento dessas massas é o uso dos bancos de dados orientados a grafos. Eles, por sua vez, são interessantes por gerenciarem os dados em um modelo que cria relacionamento entre eles. Dessa forma, o banco de dados orientado a grafos é uma opção por apresentar um modelo que implementa contextos complexos definindo naturalmente relações existentes entre as entidades de uma base de dados [1]. Para fazer o processo de estruturação desses dados, é fundamental utilizar alguma padronização que dê significado a eles na forma de metadados (dados sobre dados) [27]. Pode-se destacar o JSON (*JavaScript Object Notation*) e o CSV (*Comma-separated values*), como exemplo de padrões de formatação de texto. Estes padrões, por sua vez, são muito úteis por serem bastante difundidos, o que pode facilitar na universalização da aplicação a ser criada ao usá-los.

Com a mudança de paradigmas da Web na contemporaneidade, foi requisitado um maior volume de dados, um alto grau de disponibilidade e uma escalabilidade na demanda. Além disso, com o volume de dados (*Big Data*), se faz necessário o uso de ferramentas que busquem uma melhor apresentação das massas e alocação dessas informações em uma arquitetura de banco de dados mais flexível e rápida. À vista disso, esse banco traz uma maneira de enxergar os dados de forma mais clara para que seja possível fazer investigações de seu conteúdo de forma mais amigável [6]. Pode-se ver também que para criar uma aplicação que atenda a esses requisitos, seria necessário a implementação de um outro modelo arquitetural de banco de dados fora do Modelo Relacional. Os bancos de

dados relacionais – propostos na década de 70 - trabalhavam com aplicações de estrutura fixa e bem definida [2], com isso, esse tipo de banco não seria capaz de funcionar com eficiência em aplicações que exijam escalabilidade, alto grau de disponibilidade, alto volume de dados e clusterização [2]. Dessa forma, tomando o exemplo dos sistemas de buscas – como o Google, dificilmente o modelo relacional conseguiria tratar o volume de dados dessas aplicações, por se tratarem de um uso dinâmico e escalável. Além do mais, há outro problema: a estruturação de um padrão para a troca de informações. Com o advento dos conceitos da Web Semântica, novos padrões surgiram buscando atender os requisitos ontológicos e construir metadados, como o RDF (*Resource Description Framework)*, e padrões de formatação de texto, como o CSV e o JSON [27]. Ainda assim, unir todos esses conceitos e soluções para criar uma aplicação única de visualização de dados abertos ainda é um desafio, quando é vista a dificuldade em se manipular esses dados e em se transmitir as informações [28].

Na contemporaneidade, a cultura dos dados está cada vez mais presente e as tecnologias digitais proporcionam uma grande produção, coleta e circulação dos dados, ocasionando um grande volume (*Big Data*) [38]. Desse modo, o emprego de técnicas de visualização de dados (*DataViz*) é uma estratégia pujante e inovadora que pode ser explorada por diversas áreas do conhecimento [38]. Para que os dados coletados tenham um real valor de informação, deve-se criar esquemas ontológicos para descrever conceitos, relações e termos do domínio de conhecimento a ser analisado [8]. Nesse âmbito, há uma grande dificuldade de se administrar e explorar os dados de maneira efetiva [40] e, por isso, criar formas de visualização é tão essencial para uma maior compreensão do ser humano aos dados na sua análise [28].

Por ter características que são indispensáveis para o tratamento de aplicações contemporâneas que visam escalabilidade, alta disponibilidade e grande volume de dados; o grande desafio deste trabalho é criar uma aplicação Web cliente que consiga gerar uma visualização de relações em forma de grafo [28], a partir da inserção de dados em um banco, a fim de criar uma ferramenta de administração dos dados para uma melhor investigação de suas informações pelo usuário. Para realizar essa tarefa, serão usados padrões da Web Semântica - como o CSV, que estrutura os dados em forma de tabela usando um separador (como, por exemplo, a virgula), - para criar um esqueleto padrão que consolide o processo de entrada dos dados. Para armazenar os dados, será empregado o Banco de Dados Orientado a Grafos como uma forma eficiente de provimento das relações e das entidades. Por fim, foram usados conceitos de Usabilidade para assegurar a satisfação dos usuários quanto ao uso da aplicação e garantir sua eficiência e eficácia.

1.2 OBJETIVOS

1.2.1 OBJETIVO GERAL

Esse trabalho tem como objetivo criar uma aplicação Web cliente de fácil uso capaz de inserir dados, através de um arquivo CSV, e visualizá-los em forma de grafo usando um banco de dados orientado a grafos na composição da aplicação com a finalidade de analisar os dados de forma mais eficiente em busca de informações.

1.2.2 OBJETIVOS ESPECÍFICOS

- Comparar dois bancos de dados orientados a grafo para uma melhor escolha de qual solução usar;
- Desenvolver uma página Web que possibilite a visualização dos dados armazenados no banco em forma de grafo com o intuito de administrar os dados em busca de uma visualização mais eficiente que investigue as informações explicitadas pela estrutura;
- Configurar o banco de dados e interligar seu serviço com a aplicação Web cliente por meio da API do banco;
- Desenvolver um software que utilize os conceitos da Usabilidade, como atributo da qualidade, para facilitar o processo de aprendizagem quanto ao uso da aplicação por parte do(a) usuário(a);
- Fazer com que a aplicação consiga inserir dados no banco por meio de um arquivo CSV.

1.3 ESTRUTURA DO TRABALHO

Este trabalho foi organizado em seis capítulos. O capítulo 1, introdução, faz uma exposição dos aspectos gerais e dos objetivos do trabalho realizado. O capítulo 2 apresenta os trabalhos correlatos que motivaram esse projeto e exibe os conceitos e ferramentas essenciais para a compreensão da implementação da aplicação, dando devido embasamento teórico. No capítulo 3, foi realizada uma comparação entre dois bancos de dados orientados a grafo. No capítulo 4, será mostrado o processo de desenvolvimento do software e os resultados da implementação. E por fim, no capítulo 5, fez-se as conclusões do trabalho e as propostas de melhorias da ferramenta.

2 – TRABALHOS CORRELATOS E FUNDAMENTAÇÃO TEÓRICA

Nesse capítulo, será abordado os trabalhos correlatos a esse, assim como as bases teóricas e as ferramentas necessárias para a execução desse trabalho sobre a análise de dados e o banco de dados orientado a grafos.

2.1 TRABALHOS CORRELATOS

Para realizar esse trabalho de conclusão de curso, baseou-se em trabalhos de comparação de banco de dados orientado a grafos e de visualização de dados e de informação. Dessa forma, foram relacionados abaixo, através da Tabela 2.1, trabalhos sobre esses assuntos.

Tabela 2.1 Trabalhos Correlatos realizado em diversas partes do globo

Categoria do Trabalho	*Brasil*	*América*	*Ásia / Pacífico*
Visualização de Dados e de Informação	1 - Um modelo de visualização de dados utilizando banco de dados orientado a grafo suportado por Big Data Brasil (2016)	5 - D3: Data-Driven Documents EUA (2011)	7 - Visual framework for Big Data in d3.js China (2014)
	2 - Visualização de redes gênicas a partir da integração De dados biológicos Brasil (2016)	-	-

Categoria do Trabalho	*Brasil*	*América*	*Ásia / Pacífico*
Comparação e Implementação de Banco de Dados Orientado a Grafos	3 - Análise comparativa dos bancos orientados a grafos de primeira e segunda geração – Uma aplicação na análise social Brasil (2016)	6 - Graph Database applications and concepts with Neo4j EUA (2013)	8 - Analysis Of Distribution Network Outrage Region Based On Graph Database China (2016)
	4 - Bancos de dados não-relacionais: um novo paradigma para armazenamento de dados em sistemas de ensino colaborativo Brasil (2014)	-	9 - Graph Database: A Contemporary Storage Mechanism for Connected Data Índia (2016)

Analisando a tabela 1, percebe-se que o trabalho 1 tem, como semelhança a esse, uma proposta de visualização de dados utilizando o Neo4j como BD e, como diferença, o uso do Hadoop para clusterização e não houve uma preocupação com a construção de uma aplicação Web própria para o processo de visualização. Já no segundo, houve a construção de uma aplicação Web cliente e a utilização do banco de dados orientado a grafos em sua construção; no entanto, aplicou-se outras bibliotecas para realizar algoritmos específicos para serem usados na área da Genética, como o DBpedia e o Bio4j. O trabalho 5, por sua vez, mostra o processo de construção da visualização de informação a partir do emprego de *frameworks* de visualização (como o D3.js), do CSV (como padrão de estrutura dos dados), do SVG (para criar os vetores gráficos), e das linguagens JavaScript e HTML, para construção das páginas. Todas essas ferramentas fizeram parte da concepção desse TCC; no entanto, não se realizou um *benchmarking* para analisar a performance do D3.js e um estudo aprofundado sobre o comportamento dessa biblioteca. Para finalizar a categoria de *Visualização de Dados e de Informação*, o sétimo trabalho faz uma pesquisa semelhante ao quinto, mas alia esse estudo do D3.js ao seu uso na Computação em Nuvem, no Data

Warehouse e em aplicações de Big Data. Essa parte não compreende esse Trabalho de Conclusão de Curso: o que se buscou aqui foi criar uma aplicação Web cliente monolítica que pudesse fazer a visualização de dados em um banco de dados orientado a grafos para que servisse de base para softwares futuros.

No trabalho 3, fez-se uma comparação de dois bancos de dados orientados a grafo, também feito nesse trabalho, onde foi feito um comparativo dos resultados de ambos; porém, não foi feita uma aplicação de visualização desses dados. Já o quarto trabalho explorou as características de diversos bancos de dados NoSQL e suas vantagens sob o modelo relacional. Nesse trabalho só se ateve ao modelo orientado a grafo. O trabalho 6, no que lhe diz respeito, é bem semelhante ao anterior, mas, assim como esse trabalho, se falou apenas sobre o banco de dados orientado a grafos, mais especificamente sobre o Neo4j. No entanto, houve uma investigação mais aprofundada, o que não foi explorado aqui, sobre certos aspectos desse tipo de BD, como os diferentes tipos de aplicação que podem ser construídas e conceitos de segurança e de núcleo do banco. Já o oitavo conta com uma aplicação que usa o banco de dados orientado a grafos (no caso, o GraphSQL) em uma aplicação de controle de potência de rede elétrica. Comparando com o trabalho atual, tentou-se elaborar uma aplicação prática ao uso do BD orientado a grafos, usando suas estruturas para realizar relações; entretanto, seus objetivos foram diferentes: enquanto trabalho 8 ateve-se ao uso estrito do gerenciador GraphSQL, este desenvolveu uma aplicação Web de inserção e visualização de dados e de informações de qualquer SGBD Neo4j com serviço ativo. Para finalizar a categoria de *Comparação e Implementação de Banco de Dados Orientado a Grafos*, tem-se o nono artigo que se constituiu de uma análise sobre os bancos de dados orientados a grafos, relacionando suas gerações, principais características e propriedades, suas diferenças com o modelo relacional, sua relação com padrões ontológicos e diferentes aplicações que esse tipo de BD pode prover. Nesse trabalho, tem-se um capítulo (o terceiro) que relaciona todas essas questões; entretanto, não se apoiou em mostrar as diversas aplicações.

2.2 VISUALIZAÇÃO DE INFORMAÇÃO

Visualizar é simplesmente tornar algo visual ou visível e o conceito de visualização é definido como a transformação de uma ideia abstrata em algo real ou mentalmente visível em um sistema computacional de fácil compreensão por parte do usuário [28]. A visualização é uma atividade naturalmente humana, o que torna muito difícil sua exteriorização, transformando e sintetizando esses conceitos e informações em uma imagem ou em um gráfico [28].

A Visualização de Informação pode ser definida como “o uso de representações visuais, interativas e suportadas por computador, de dados abstratos para ampliar a cognição” [28]. Não só dados científicos podem ser visualizados, como também outros tipos de dados, como os financeiros, organizacionais e governamentais. Por isso, essa área tem muita relevância nas tarefas humanas, que se intensificaram depois da modernização e da expansão da computação e das tecnologias da informação [28].

As informações descrevem fenômenos, processos e entidades ao se estudar um objeto ou realizar uma análise; dessa forma, elas trazem atributos que as caracterizam [28]. Para avaliar esses atributos (classe, categoria e qualidade, por exemplo) e seus critérios, deve-se saber seu domínio de aplicação e seu tipo de informação. Os atributos representam uma propriedade indicada por meio de uma grandeza, a qual possui uma escala, uma unidade e um intervalo de atuação [28]. Além disso, eles podem indicar relacionamentos e ligações. Por exemplo, a temperatura é de grandeza escalar, de tipo numérico, contínuo e com a unidade ºC e ela pode ser relacionada a uma cidade ou município.

Uma outra abordagem que se pode fazer ao analisar um dado é de acordo com sua estrutura, ou seja, como ele está organizado [28]. Realizando essa estruturação dos dados, eles ficam melhor formatados, consolidados e mais compreensíveis pela aplicação fim. Dentre essas estruturas, estão [28]:

- Listas e tabelas – Conjunto de dados que se relacionam linearmente entre seus componentes, cujos elementos são constituídos por dados primitivos ou estruturados (metadados). Dentre eles estão: textos, mapas, imagens, dados relacionais e estatísticos [28].
- Árvores – São estruturas de dados hierárquicas de subordinação. Em outras palavras, um dado é subordinado a outro dado verticalmente [28]. São exemplos de árvores: catálogos de bibliotecas, diretórios de arquivos, diagramas organizacionais e manuais.
- Grafos (será explicado no item abaixo).

2.3 BANCO DE DADOS ORIENTADO A GRAFOS

Os primeiros gerenciadores de banco de dados relacionais (SGBDRs) foram criados no início dos anos 70 com a finalidade de realizar controle de concorrência, segurança, recuperação de falhas, gerenciamento dos mecanismos de armazenamento de dados e controle das restrições de integridade do BD, além de fazer gerenciamento de transações [2]. Em síntese, uma transação pode ser definida como um conjunto de operações de leitura

ou escrita que são realizadas no banco de dados que seguem algumas propriedades. Estas propriedades são chamadas de ACID e são definidas a seguir [2]:

- Atomicidade: a transação deve ser executada por completo, ou nada é executado;
- Consistência: após uma transação ter sido concluída, o banco de dados deve permanecer em um estado consistente e integro igual ao estado anterior;
- Isolamento: as transações devem ser independentes e ocorrer em concorrência. O SGBD deve fazer essa gestão;
- Durabilidade: uma vez que uma transação ocorreu com sucesso, seu efeito não poderá mais ser desfeito, mesmo em caso de falha. Esta propriedade está relacionada a capacidade de recuperação de falhas do SGBD [2].

São conceitos básicos do modelo relacional: a relação, caracterizado por uma tabela; o atributo, ilustrado pela coluna; e a tupla, formado pela linha. Intrinsicamente, uma tabela representa um conceito – América, Brasil - ou uma associação entre conceitos – Brasil fica na América -, além disso, as colunas definem as propriedades destes conceitos – Nome e Sigla, no caso de um país -, já as linhas compreendam as instâncias propriamente ditas (Brasil, BRA) [2]. O grande sucesso desse tipo de banco de dados ocorreu devido a sua simplicidade, a sua padronização de conceitos, a sua base formal e a sua facilidade de uso da linguagem SQL (*Structured Query Language*), que é a linguagem padrão para consultas e manipulação de dados relacionais [2].

Em oposição ao termo ACID, visto previamente, usado por bancos de dados relacionais, houve um crescimento do termo BASE para expressar as bases do NoSQL de forma otimista ao mostrar as formas de armazenamento, de escalabilidade e resiliência [5]. As propriedades BASE são consideradas mais fracas que a ACID, pois não oferece garantia de consistência nas réplicas no momento de escrita, mesmo tendo o selo de disponibilidade. Essas propriedades podem ser detalhadas como:

- Disponibilidade Básica (*Basic Availability*): O banco busca estar disponível no maior tempo possível;
- Estado Maleável (*Soft-state*): O banco não precisa ter sua escrita consistente, nem toda replica precisa estar mutualmente consistente em todo o período;
- Consistência Eventual (*Eventual Consistency*): A base pode mostrar consistência em algum ponto posterior (Exemplo: maior tempo na hora de leitura).

A modelagem de um sistema de gerenciador de banco de dados orientado a grafos é realizada baseado na teoria de grafos estruturando os dados na forma de grafo. A estrutura básica conta com três elementos: os nós (ou vértices), as arestas (ou entidades) e as propriedades. Um exemplo de grafo de propriedade (onde vértices e arestas possuem atributos para descrever suas propriedades) é apresentado na Figura 2.1. Os nós expressam as entidades, as arestas representam as relações entre os nós e as propriedades mostram características das entidades e relacionamentos [4]. Uma topologia de grafo G pode ser representada como G=(V,E), no qual V é o conjunto de vértices (também chamados de nós) e E é o conjunto de arestas. Cada aresta representa uma relação entre dois nós e um conjunto de vértices conectados por meio de arestas definem um caminho no grafo [1].

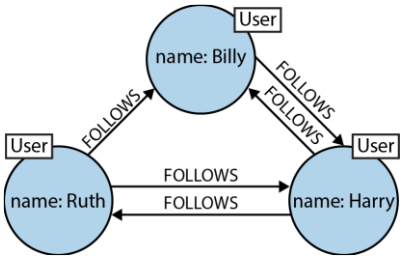

Figura 2.1 Exemplo de grafo de rede social [5].

Nesse exemplo da Figura 2.1, os nós (ou vértices) são representados pelos usuários e as arestas pela linha (relacionamento, no caso, “ato de seguir”). As arestas expressam um sentido de orientação [5].

Um conceito importante para entender como é realizada a recuperação da informação é a travessia. Ela pode ser classificada como uma operação localizada, onde cada vértice e aresta no grafo armazenam um índice dos objetos conectados a ele [4]. Isso faz com que não haja perda de desempenho ao se aumentar o tamanho do grafo; por isso, esse tipo de base de dados permite processar com eficiência densos conjuntos de dados, num modelo de chave-valor. Seu design permite a construção de modelos preditivos e análise de correlações e padrões de dados [4]. Como todos os nós estão ligados por relações, a travessia entre os diversos vértices e arestas ocorrem rapidamente.

Os sistemas de gerenciamento de banco de dados usam os métodos CRUD (Criar, *Create*; Ler, *Read*; Atualizar, *Update* e Deletar, *Delete*) e são construídos sob sistemas transacionais OLTP (Processamento de Transações em Tempo Real) [5]. Eles podem usar armazenamento nativo ou não e não possuem um índice global igual ao modelo relacional. Dessa forma, esses sistemas são muito ágeis e flexíveis por permitir alterações em nós, relações e propriedades sem afetar a estrutura existente e por utilizar novas formas de

manutenção como o REST API (Transferência de Estado Representacional/Interface de Programação de Aplicativos) para as aplicações [5].

Pode-se classificar o mapeamento lógico-físico dos SGBDGs como não-nativos e nativos. Os não-nativos modelam logicamente seus dados como grafos; no entanto, os armazenam por meio de outros modelos como tabelas relacionais e estrutura chave-valor. Já os nativos, usualmente, usam listas de adjacência, onde cada vértice mantém referências diretas para seus vértices adjacentes formando uma espécie de micro-índice para os vértices próximos [1]. Esse mecanismo é conhecido como adjacência livre de índices. Um bom exemplo da utilização desse tipo de BD é o Twitter: sua base é modelada como um grafo, onde os usuários são os vértices e os relacionamentos entre eles são as arestas [1].

Esse tipo de banco de dados é utilizado muitas vezes em aplicações de redes sociais, por conta de seu modelo que cria relacionamento entre as entidades. Dessa forma, vê-se que é bem interessante formar grafos de relacionamentos entre pessoas, por exemplo. Em muitos casos, as aplicações desse tipo usam uma característica interessante desse modelo: buscar os nós mais próximos. Isso pode ser muito útil, quando se quer analisar os relacionamentos. Em redes sociais, pode-se usar para descobrir amigos em comum, por exemplo [6]. Dessa forma, esse tipo de banco de dados torna-se uma excelente ferramenta para visualização dos dados para facilitar o entendimento dessas relações. Nesse caso das redes sociais, os usuários dessa rede são nós e as relações entre eles são as arestas. Para realizar a travessia, o banco acessa apenas as arestas formadas pelos seus relacionamentos, com isso, não é necessário verificar o repositório por completo.

2.4 ESTUDO DE DOIS BANCOS DE DADOS ORIENTADO A GRAFOS

2.4.1 NEO4J

Neo4j é um banco de dados em forma de grafo nativo, projetado para armazenar e processar grafos de baixo para cima. Ele é baseado em disco, embarcado, totalmente transacional, de motor Java que armazena dados estruturados em grafos [2]. Esse SGBD possui rendimentos altos, se comparados a banco de dados relacionais, e usa o hardware com mais eficiência. Isso se deve pelo buffer de atualização rápido e um log de transações unificado [2]. O Neo4j tem versões aberta e proprietária e oferece suporte ao Tinkerpop. O carregamento e a consulta podem ser feitos pela linguagem Gremlin, Cypher (proprietária) e

SPARQL (SQL *for linked data*) [1]. O armazenamento físico pode tratado tanto em memoria quanto em disco e o modelo é baseado em repositórios chave-valor [1].

2.4.2 ORIENTDB

Esse SGBDG é um banco de dados em grafo de código aberto e também com versão comercial. Ele foi implementado em Java, pode ser utilizado tanto de forma centralizada quando distribuída com replicação e, além disso, é transacional. O OrientDB oferece grafos de propriedades para a modelagem lógica e dá suporte ao Tinkerpop [1]. Sua manipulação da base é realizada com as linguagens Gremlin, Java e uma adaptação do SQL. Já o armazenamento físico dos dados pode ser feito em memória e em disco [1]. Para viabilizar o processamento nativo das consultas, ele utiliza a lista livre de adjacências; no entanto, esse sistema usa recursos de banco de dados de documentos e de orientado a objetos para o armazenamento físico dos vértices para garantir uma maior flexibilidade na estrutura dos dados a serem armazenados na base [1].

2.5 ONTOLOGIA

Apesar da ontologia ter seu surgimento na Filosofia grega, como a teoria sobre a existência da natureza e tudo que a ela pertence, a Inteligência Artificial usa essa expressão para representar o que existe. Resumidamente, a Ontologia pode ser descrita como uma especificação explícita e formal de uma conceitualização compartilhada [34].

A Web se tornou uma excelente fonte de pesquisa em razão da sua grande base de informação, porém a grande maioria de seus dados não estão devidamente estruturados e padronizados, dificultando ainda mais o processo de busca [8]. Nesse contexto, surge o conceito de Web Semântica que tem por objetivo construir uma rede capaz de reconhecer o significado dos documentos e inferir novos conhecimentos, a partir do uso de metadados [8]. Dessa forma, a Web Semântica poderia ter a seguinte arquitetura expressada na Figura 2.2.

Assim como na Internet, sua implementação também é feita em camadas. A camada de estrutura, nesse contexto, é responsável por estruturar os dados e dar-lhe significado. Uma das possíveis implementações se dá no uso de tecnologias como o XML (*eXtensible Markup Language*), o RDF (*Resource Description Framework*), o RDF Schema e o CSV. Já a camada de esquema é composta pelas ontologias que representam os conceitos por meio de uma taxonomia e um conjunto de regras de inferência. A taxonomia, por sua vez,

estabelece as classes de objetos e as relações previstas [8]. A camada subsequente, lógica, é responsável pela definição dos mecanismos para se fazer inferência sobre os dados.

Pode-se definir uma ontologia como um conjunto comum de termos usados para descrever e representar um domínio (como medicina, biblioteca, matemática). Estas definições são expressas em linguagens baseadas em lógica e seus conceitos básicos de domínios e seus relacionamentos são processáveis computacionalmente [8].

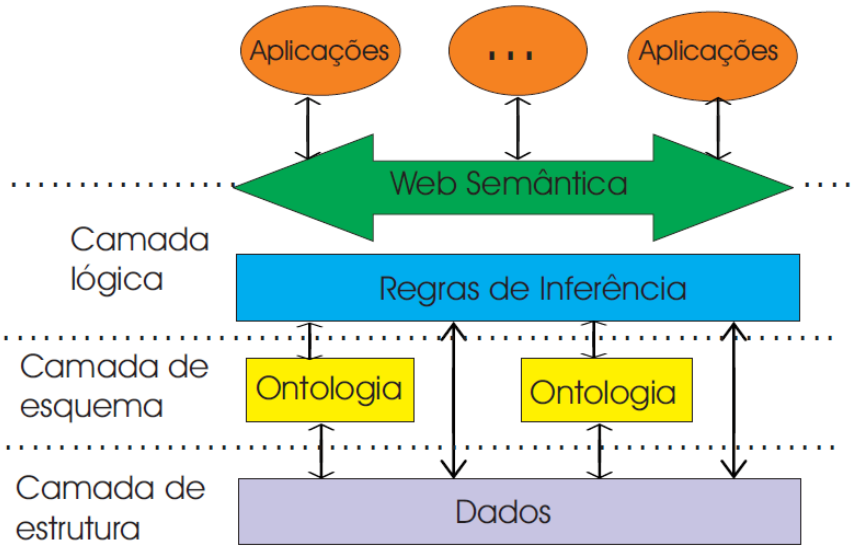

Figura 2.2 Possível arquitetura da Web Semântica [8].

2.6 USABILIDADE

Essa expressão começou a ser utilizado na década de 1980, atuando nas áreas da Psicologia, da Ciência Cognitiva e da Ergonomia, como um substituto do termo *amigável* [36]. A usabilidade, então, pode ser entendida como o conjunto de propriedades de uma interface que abrange os itens: capacidade de memorização, eficiência, fácil aprendizado, satisfação, baixo índice de erros e prazer ao uso [35]. Ou seja, ela é um atributo de qualidade que traduz a capacidade do usuário de aprender a usar algo de forma fácil, eficiente, com pouco erro e agradável de usar [35]. A usabilidade pode ser trabalhada tanto em produtos físicos (tais como, embalagens, móveis, manuais e guias), quanto em sistemas computacionais (que focam na interação com o ser humano) [36].

Dessa forma, a Qualidade em seu uso mais divulgado é o de usabilidade que engloba os conceitos vistos no parágrafo anterior [36]. Esse conceito e esses componentes são defendidos na norma ISO/IEC 9126-1 (2003), a qual estabelece a usabilidade como o potencial do produto de software ser compreendido, aprendido, operado e atraente ao

usuário, quando usado sob uma certa condição [35]. Por conseguinte, a norma ISO/IEC 9241-11 (2003) trata a usabilidade como a capacidade de um produto ser usado por um usuário pertinente para atingir objetivos específicos com eficiência, eficácia e satisfação em um contexto de uso inerente [35]. A ISO, com isso, define expressões que são empregadas quando se fala em usabilidade [35]:

- Eficácia – capacidade dos sistemas em conferir diferentes tipos de usuários a almejar seus objetivos em número e com a qualidade esperada;
- Satisfação – sentimento que o software proporciona quando o usuário utiliza o sistema e como ele reage aos recursos e aos resultados exibidos a ele, levando em consideração os objetivos idealizados no projeto;
- Usuário – pessoa que usa alguma instância do software;
- Tarefa – objetivo a ser alcançado ou resultado a ser atingido;
- Eficiência – quantidade de recurso que o software solicita ao usuário para que o sistema seja observado de acordo com os objetivos;
- Contexto – conjunto de elementos, os quais se destacam: o ambiente físico, a tecnologia utilizada e a motivação.

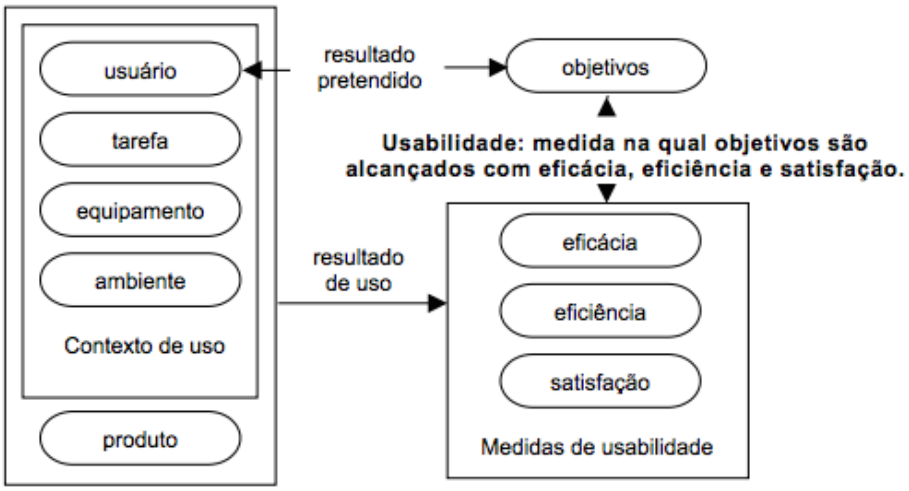

Figura 2.3 Estrutura da usabilidade de acordo com a NBR 9241-11 [37].

No Brasil, a ABNT, responsável por normatizações técnicas no país, produziu a norma NBR 9241-11 (ABNT, 202), a qual estabelece requisitos ergonômicos para trabalhos de escritório com computadores [37]. A Figura 2.3 mostra a estrutura da usabilidade, onde são apresentados os conceitos vistos acima na lista. Para que se tenha uma maior compreensão dessa imagem, vê-se que ela apresenta o contexto de uso do produto (ou seja, quais são os usuários, as tarefas que ele executa, os equipamentos necessários, o ambiente ao qual se situa, entre outros) e as medidas de usabilidade que devem ser

mensuradas para definir como foi a usabilidade (tais como, a eficiência, a eficácia e a satisfação). Unindo os dois, tem-se os resultados do uso (a percepção do usuário, que serão avaliadas pelas medidas de usabilidade) e os resultados pretendidos (que tem como base os objetivos, os quais entraram na avaliação das medidas de usabilidade).

2.6.1 METAS DE USABILIDADE

As metas de usabilidade assistem o desenvolvimento de produtos fáceis de usar, agradáveis e eficientes [35]. Dentre elas, estão [35]:

1. Eficácia;
2. Utilidade;
3. Eficiência;
4. Facilidade de lembrar como se usa;
5. Segurança;
6. Facilidade de aprendizado.

2.6.2 ÁREAS DA USABILIDADE

A usabilidade é atendida em várias áreas, tais como [35]:

- Interação Humano Computador (HCI) – teve seu início na década de 1980 e descreve como uma área de estudo voltada à preocupação de como o uso dos computadores poderia enriquecer a vida pessoal e profissional do ser humano [35]. Seu principal objetivo é melhorar a eficácia e a satisfação do usuário, a partir do aprimoramento dos sistemas computacionais, nos quais os usuários possam realizar as tarefas eficientemente, com segurança e com satisfação [35].
- Arquitetura da Informação – A informação é muito presente na contemporaneidade e ela pode ser definida como aquilo que leva o entendimento [35]. Com o excesso de informação dos meios de comunicação, fica mais difícil realizar sistemas que atentam todas as metas da usabilidade, como a facilidade de lembrar como usa. A arquitetura da informação, por sua vez, é a arte e a ciência de se estruturar e organizar os ambientes informacionais que auxiliem o usuário a gerenciar e manipular as informações [35]. Com isso, houve a necessidade de se acentuar a pesquisa na área de usabilidade para essas aplicações. Neste ponto, a presença do arquiteto de

informação é essencial para que se possa balancear as necessidades do usuário com as regras de negócios e seus objetivos. Esse profissional deve ser multidisciplinar com conhecimentos em Design Gráfico, Ciência da Computação, Marketing, Ciência da Informação e Engenharia de Usabilidade [35].

2.7 BIBLIOTECAS E FERRAMENTAS USADAS NO TRABALHO

2.7.1 BOOTSTRAP

O Bootstrap é um *framework front-end* que usa o CSS para desenvolver projetos Web responsivos, ou seja, que se adaptam a diferentes tamanhos de tela; de acordo com o dispositivo do(a) usuário(a) [21]. Para fazer esse processo de adaptação de tela, ele usa um sistema de 12 *grids* responsivos. Além disso, com ele, é capaz de criar uma grande quantidade de componentes na tela: como botões, ícones, componentes de formulário, entre outros. Com todo esse arsenal, é possível poupar muito tempo de desenvolvimento ao escrever os estilos da página de forma fácil e intuitiva.

É uma solução bem completa que com uma vasta documentação e centenas de exemplos funcionais que resolvem os mais diversos problemas de *layout* e responsividade em todos os tipos de dispositivos.

2.7.2 JQUERY

O jQuery é uma biblioteca de código aberto criada a partir do JavaScript, a qual tem como principal intuito, simplificar a programação dos *scripts* que compõem uma aplicação Web no lado do cliente [22] e reduzir as linhas de código de um documento original do JavaScript. Ela é bem intuitiva na hora de tratar o documento HTML e um excelente recurso na criação de aplicação do tipo AJAX (*Asynchronous JavaScript and XML* – técnica para acessar servidores Web, a partir da página na parte do cliente de forma assíncrona), além de ter uma seleção de elementos eficiente no DOM (*Document Object Model* – um meio de acessar dinamicamente elementos, estruturas e estilos de um documento) para a construção de eventos (como, cliques de mouse e de teclado) e de animações [22]. Na Figura 2.4, tem-se um exemplo de função realizada no projeto que se usou essa biblioteca. Nesse caso, há um evento criado por meio do clique do botão *#editEntProp* que executa a função *carEditEntProp*, o qual libera a visualização de uma *div* com o *id #passo4*.

```
function carEditEntProp(evt) {
    $("#passo4").show();
}
$("#editEntProp").click(carEditEntProp);
```

Figura 2.4 Função *carEditEntProp* que é acionada no clique do botão com o id *editEntProp* para visualização de uma div com id *passo4*

2.7.3 POPOTO.JS

O Popoto.js é uma biblioteca JavaScript construída, a partir do D3.js (uma biblioteca JavaScript de visualização de dados) [24] e de desenhos SVG (*Scalable Vector Graphics* – uma linguagem XML que descreve uma forma vetorial de desenhos e gráficos bidimensionais), a qual fornece uma interface para a realização de pesquisas de entidades e relacionamentos armazenados no Neo4j na forma de grafo [23]. Ela tem suporte a todos os navegadores modernos (exceto o IE8 e versões anteriores) e ao Neo4j (após versão 2.0). Com ele, é possível criar blocos de visualização de grafo de maneira amigável e dinâmica, além de ser editável por se tratar de uma solução de código aberto.

2.7.4 PAPAPARSE

O Papaparse é uma biblioteca de leitura de CSVs para o JavaScript, a qual é bastante eficiente em termos de acertos e performance. Ela usa uma estrutura *multi-thread* eficaz para rodar em páginas Web, tendo a capacidade de rodar *gigabytes* de conteúdo sem que congele o navegador, além de possuir um sistema que lida com falhas no CSV para reduzir qualquer impacto [25]. Um dos seus pontos positivos é que, como ele pode ser rodado do lado do cliente, muito se ganha no contexto de privacidade, já que não é necessário passar todas as informações pela Internet para processar; dessa forma, grande parte desse processo é feito pelo cliente. Além de atuar no lado do cliente, ele pode atuar em servidores com a ajuda do Node.js, com algumas restrições.

Além de ser compatível com todos os navegadores modernos, ele é de código aberto e bem conhecido. Entre os *players* do mercado, tem-se seu uso no VisualEditor [26] do Wikipedia para auxiliar na construção de tabelas, através de arquivos de texto. Dentre suas funcionalidades, destacam-se: a conversão de CSV para JSON (e vice-e-versa) e a extração desses arquivos para uma *string*, para um arquivo local, para um arquivo remoto e para um objeto do jQuery [25].

3 – COMPARAÇÃO ENTRE DOIS BANCOS DE DADOS ORIENTADOS A GRAFOS

Este capítulo apresenta uma comparação entre dois grandes bancos de dados orientados a grafo: o Neo4j e o OrientDB. Foi feita uma relação da semelhanças e diferenças de cada sistema de gerenciamento de banco de dados, além de um experimento de performance.

3.1 EXPERIMENTANDO DOIS BANCOS DE DADOS ORIENTADOS A GRAFOS

Nesse experimento inicial, foi construído uma mesma arquitetura de dados usando os gerenciadores Neo4j e OrientDB. Foram coletados dados públicos de três presidentes do Brasil e seus ministros da Fazenda, Educação e Saúde, assim como de servidores aposentados desses ministérios e seus cargos [7]. Buscou-se, então, criar um ambiente de rede social, onde se pudesse relacionar políticos com outros políticos e seus respectivos cargos, além de ligar os ministérios com seus chefes e servidores aposentados. Para descrever melhor a arquitetura montada, a Tabela 3.1 apresenta os nós usados informando seus atributos e número de nós daquele tipo. Já na Tabela 3.2, enumerou-se as relações feitas entre os nós. Para exemplificar, exibiu-se um exemplo de grafo da arquitetura montada no Neo4j (Figura 3.1) e um exemplo feito no OrientDB (Figura 3.2).

Para fazer a inserção dos dados, foram usadas *queries* na linguagem Cypher para o Neo4j e uma adaptação do SQL para o OrientDB. Em ambos os casos, não houve problemas no processo de adição e, com o auxílio do de planilhas, as *queries* foram montadas. A adição dos dados no OrientDB ocorreu via *console*, onde as *queries* eram informadas por linha de comando. Usou-se essa abordagem, pois a interface gráfica do OrientDB não suporta a execução de múltiplas *queries* simultâneas.

Tabela 3.1 Tipos de nós, seus atributos e quantidade de dados

Nó	*Atributos*	*Quantidade*
Políticos	Nome e ID	035
Cargo Político	Nome e ID	004

Nó	*Atributos*	*Quantidade*
Cargo	Nome e ID	004
Ministério	Nome, Sigla e ID	003
Servidor(a)	Nome, Data de Aposentadoria e ID	138

Tabela 3.2 Relações (arestas), seus atributos e quantidades

Relações (Arestas)	*Nós relacionados*	*Atributos*	*Quantidade*
Foi do Governo	Político -> Político	Nome, Data de Início e Data de Fim	35
Teve Ministro	Político -> Político	Nome, Data de Início e Data de Fim	35
Chefe do	Cargo Político -> Ministério	-	3
Ocupou Cargo Político	Político -> Cargo Político	Nome do Cargo, Data de Início e Data de Fim	40
Se Aposentou do	Servidor -> Ministério	Data da Aposentadoria	138
Teve esse Cargo	Servidor -> Cargo	-	138
Possui Alguem Nesse Cargo	Ministério -> Cargo	-	10

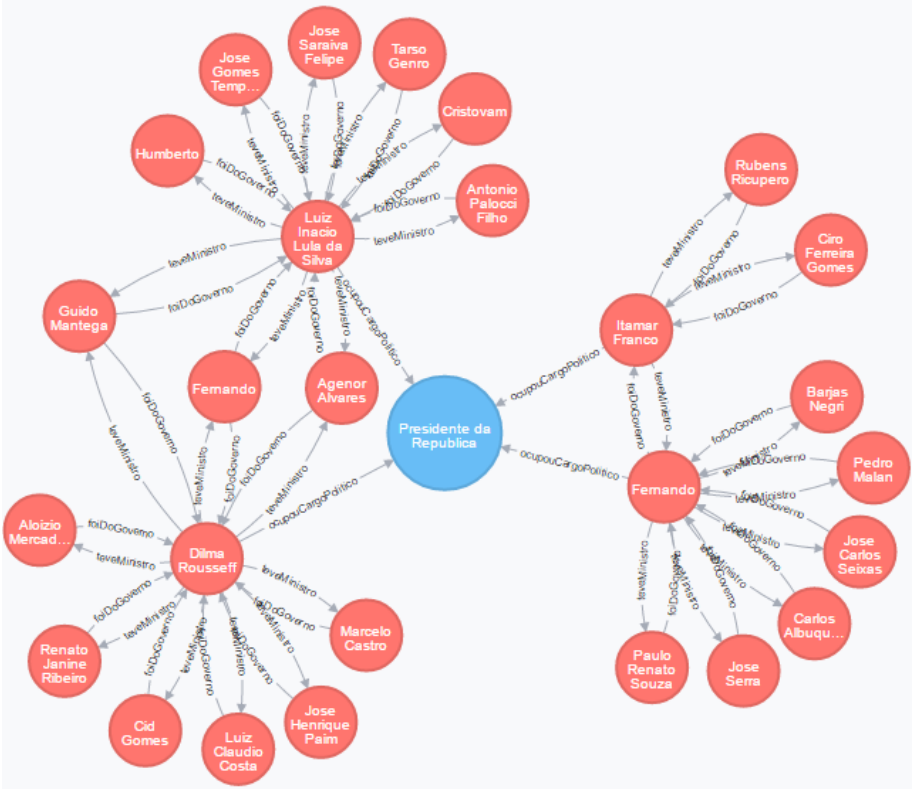

Figura 3.1 Grafo que liga os políticos através das relações *Foi do Governo* e *Teve Ministro*. E ainda, relaciona os presidentes entre si. Realizado no Neo4j.

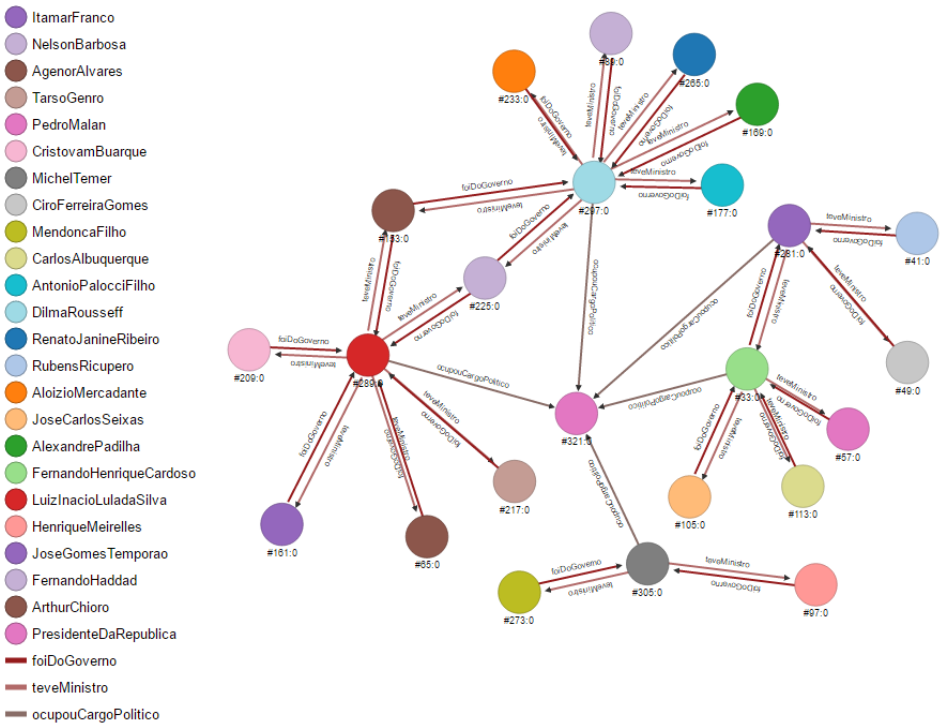

Figura 3.2 Grafo que liga os políticos através das relações *Foi do Governo* e *Teve Ministro*. E ainda, relaciona os presidentes entre si. Realizado no OrientDB.

3.2 COMPARAÇÃO ENTRE ESSES DOIS BANCOS DE DADOS ORIENTADOS A GRAFOS

Ao realizar o trabalho foi percebido que o Neo4j possui um número maior de conteúdo disponível se comparado ao OrientDB. Pode-se destacar também que ambos os SGBDGs constroem grafos com atributos, ou seja, os nós ou relacionamentos podem possuir atributos que o classificam. Para fins de comparação, foi criada a Tabela 3.3 para mostrar a performance de ambos os bancos em certos aspectos. Para encontrar a média, foi feito o procedimento 10 vezes, retirou-se os dois maiores e os dois menores valores e calculou-se a média dos valores restante.

Tabela 3.3 Aspectos Mensurados sobre a Performance de Ambos os Bancos

Aspectos mensurados	*Detalhe do Aspecto*	*Média (em segundos)*	
		Neo4j	*OrientDB*
Criar um nó (Para OrientDB: [Classe]/[Nó])	Político	0.0318	1.6970/0.0390
	Cargo Político	0.0487	1.8590/0.0160
	Servidor	0.0028	1.8210/0.0030

Aspectos mensurados	*Detalhe do Aspecto*	*Média (em segundos)*	
		Neo4j	*OrientDB*
Criar um enlace	Foi do Governo	0.0175	0.0285
	Ocupou Cargo Político	0.0063	0.05035
	Se Aposentou do	0.0041	0.0453
Leitura de enlaces com no máximo 20 nós	Foi do Governo	0.0188	0.0247
Leitura de enlaces com no máximo 100 nós	Teve esse Cargo	0.0247	0.0770
Uso do algoritmo de *shortest path* do banco	Profundidade = 2	0.0168	0.0178
	Profundidade = 3	0.0177	0.0180
	Profundidade = 4	0.0195	0.0181
	Profundidade = 5	0.0243	0.0250

No caso do OrientDB, é necessário criar uma classe antes do nó em si. Isso é interessante por fazer com que uma classe possa ter uma forma padrão, dando mais consistência na hora de montar o grafo. No entanto, é um passo a mais a ser dado ao inserir um novo nó, que custa processamento e um tempo maior de execução. Mesmo porque, o tempo de inserção de uma classe é dezenas de vezes mais demorada que a de um nó em si.

Ficou claro na Tabela 3.3, que o Neo4j se destacou quanto ao desempenho da criação de enlaces. Quanto à criação do nó em si, o OrientDB tem um tempo menor de processamento; no entanto, ele esbarra no procedimento a mais a ser feito: a criação de uma classe para o nó. Percebe-se também que a performance dos algoritmos é bastante similar nos dois gerenciadores de banco de dados. Fazendo uma comparação com [4], viu-se que foi obtido um resultado semelhante para a profundidade igual a 2. Nesse trabalho em questão, os autores não conseguiram fazer o experimento usando o OrientDB para profundidades superiores a 4. No procedimento feito nesse trabalho, conseguiu-se realizar todas as profundidades que o modelo cabia para ambos os bancos. Para concluir, o Neo4j foi ligeiramente vantajoso se comparado ao OrientDB nos aspectos mensurados.

4 – METODOLOGIA E DESENVOLVIMENTO DA APLICAÇÃO WEB – ANA LISA

Este capítulo explorará o processo de desenvolvimento e a metodologia para a realização da aplicação Web de visualização. Para isso, usou-se certas linguagens, como o Javascript e HTML, e certas bibliotecas, como o JQuery, o Bootstrap e o Popotojs. Toda a aplicação roda no lado do cliente através de um navegador Web.

4.1 ESCOPO DA APLICAÇÃO

O trabalho em questão tem como principal objetivo fazer a visualização, em uma página Web, de dados armazenados em um banco de dados orientado a grafo. Esses dados devem estar dispostos na forma de um grafo, onde se possa verificar as relações entre as entidades, a fim de iniciar uma investigação das informações inerente a esses dados. Essa aplicação deve entregar uma solução para inserir novos dados ao banco através de um arquivo CSV para que sejam visualizados posteriormente.

Não é objetivo do trabalho elaborar uma forma de automatizar o processo de coleta e inserção de dados, assim como de limpeza do banco através da aplicação. Por ser uma aplicação cliente, não haverá nenhum tipo de automação nem orquestração em um servidor para gerir a criação de serviços e novos bancos de dados.

4.2 REQUISITOS FUNCIONAIS E NÃO-FUNCIONAIS

Serão apresentados os requisitos funcionais [RF] e os não funcionais [NF] da aplicação Web na seção 4.2.1 e 4.2.2, respectivamente:

4.2.1 REQUISITOS FUNCIONAIS

Os requisitos funcionais do software, ou seja, o que a aplicação Web deve oferecer ao usuário em termos de funcionalidade, são:

- [RF01] Escolha das entidades a serem visualizadas:
    - Descrição do caso de uso: O(a) usuário(a) poderá se conectar ao banco de dados, através da aplicação e ele(a) poderá escolher quais

serão as entidades e formar a ontologia que deseja visualizar para fazer as relações na forma de grafos.

- Pré-condições: O(a) usuário(a) deve possuir o local, a porta, o nome de usuário(a) e senha do Neo4j disponível.
- Entradas: Nome de usuário(a), senha, local, porta, entidades e suas características.
- Saídas e pós-condição: O local, a porta, o nome de usuário(a) e a senha serão usados para se conectar ao banco de dados usado e as entidades e suas características serão utilizadas para montar os grafos no visualizador.

- [RF02] Visualização do grafo a partir das entidades escolhidas:
  - Descrição do caso de uso: Depois de escolhidas as entidades (RF01), o(a) usuário(a) verá as entidades em forma de nós. Ele(a) terá acesso a uma lista ontológicas com as disponíveis para que ele possa clicar naquela que ele deseja ver as relações. Ao mostrar as relações, o(a) usuário(a) terá a possibilidade de ver outros nós (entidades) que ele se relaciona, de acordo com a relação indicada.
  - Pré-condições: O(a) usuário(a) deve realizar os passos do caso de uso da RF01 com sucesso.
  - Entradas: Entidade.
  - Saídas e pós-condição: Ao clicar na entidade, ela será apresentada como um nó, onde poderá ser visualizado suas relações, que são os vértices. Dessa forma, o(a) usuário(a) verá outros nós(entidades), a partir das relações exploradas.
- [RF03] Visualização dos resultados das relações:
  - Descrição do caso de uso: Depois de feita as relações entre as entidades (RF02), o(a) usuário(a) verá os resultados desses relacionamentos, ou seja, será criada uma lista de entidades filtradas, a partir das relações escolhidas, para que ele(a) possa identificar quais nós tem aquele tipo de relacionamento.
  - Pré-condições: O(a) usuário(a) deve realizar os passos do caso de uso da RF02 com sucesso.
  - Entradas: Entidades (nós) e relações (vértices).
  - Saídas e pós-condição: Será apresentada uma lista com as entidades e suas entidades que caibam nas relações configuradas no visualizador.

- [RF04] Inserção de novas entidades e relacionamentos a partir de um arquivo CSV:
    - Descrição do caso de uso: O(a) usuário(a) deve ser capaz de inserir entidades e relacionamentos, através de um arquivo CSV estruturado, sendo realizado previamente sua configuração, a partir de um formulário.
    - Pré-condições: O(a) usuário(a) deve ter se autenticado no Neo4j. e o(a) usuário(a) deve possuir um CSV com as informações de entidades a serem inseridas e o usuário e senha do banco.
    - Entradas: porta, local, usuário e senha do Neo4j e arquivo CSV, entidades e relacionamentos.
    - Saídas e pós-condição: Confirmação da inserção das entidades e relacionamentos escolhidos.
- [RF05] Formatação e apresentação da página Web:
    - Descrição do caso de uso: O(a) usuário(a) deve abrir uma página que contenha uma parte principal que descreva o conceito e as funcionalidades da aplicação. Em seguida, deve visualizar uma parte autenticação. uma de inserção dos dados, outra de visualização de grafos e uma outra de contato, assim como uma de visualização do autor e bibliotecas usadas.
    - Pré-condições: O(a) usuário(a) deve possuir um navegador Web compatível.
    - Entradas: Nenhuma. Será só visualização por parte do(a) usuário(a).
    - Saídas e pós-condição: Visualização da aplicação com as partes descritas por parte do(a) usuário(a).

4.2.2 REQUISITOS NÃO FUNCIONAIS

Os requisitos funcionais do software, ou seja, as características técnicas e comportamentos do software que, não necessariamente, serão visíveis aos(às) usuários(as) finais, são:

- [NF01] Usabilidade:
    - A interface com o usuário é de vital importância para o sucesso do sistema. Nesse contexto, deve-se levar em consideração os conceitos da Usabilidade para que se atenda a metas de Usabilidade.

    - O software terá uma interface amigável ao usuário primário sem se tornar cansativa aos usuários mais experientes e ainda deve ser responsiva para que seja possível sua visualização também em diversos dispositivos com dimensões de tela variáveis.
- [NF02] Tratamento de entradas pelo usuário:
    - Os formulários devem tratar possíveis entradas equivocadas e maliciosas para que o dinamismo da aplicação não seja afetado. Por ser uma aplicação assíncrona, assegurada por bibliotecas do JavaScript, as possíveis entradas devem ser pensadas para que a aplicação não caia em estados que não estão em conformidade com o objetivo do sistema.
- [NF03] Conectividade com o banco de dados:
    - A aplicação deve ser capaz de se conectar ao banco de dados de forma eficiente o suficiente para fazer inserção de novos parâmetros e registros a eles, quando exigido pelos requisitos funcionais, e extrair informações para realizar a visualização dos grafos e listas de entidades, relacionamentos e resultados das relações.

4.3 ESCOLHA DAS FERRAMENTAS, DAS BIBLIOTECAS E PROPOSTA DE ARQUITETURA

Para a construção da aplicação Web, usou-se as seguintes linguagens: HTML5, CSS e JavaScript, para que se pudesse fazer tanto a parte de marcação das páginas, quanto a criação de estilos de formatação e funções para execução de rotinas e eventos dinâmicos do sistema. Para uma maior velocidade de desenvolvimento, escolheu-se certas bibliotecas que pudessem auxiliar na performance e no desenvolvimento mais ágil da aplicação. Dentre eles estão: jQuery, Popoto.js, Bootstrap, Papaparse e D3.js. O jQuery foi usado para tornar o processo de desenvolvimento mais eficiente na construção de aplicações Web cliente. Já o Popoto.js e o D3.js foram utilizados no processo de visualização do grafo. Por sua vez, o Bootstrap auxiliou no desenvolvimento das interfaces para o(a) usuário(a) e o Papaparse atuou como ferramenta para trabalhar com a leitura do CSV.

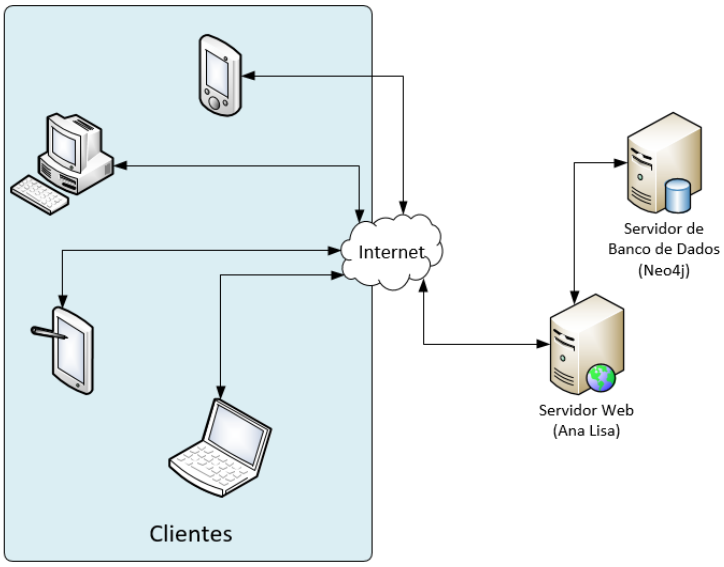

Figura 4.1 Proposta de arquitetura cliente-servidor da aplicação Web.

Na Figura 4.1, pode-se ver como a arquitetura cliente-servidor foi criada para aplicação. Apesar de ser um software que tem o processamento, em grande parte, no cliente (Navegador Web), tem-se a presença do Servidor Web (um Apache, por exemplo) para fornecer as páginas e os conteúdos da aplicação e do Servidor de Banco de Dados (onde foi instalado o serviço de banco de dados, que, no caso, o Neo4j). A escolha do Neo4j, ao invés do OrientDB, se deu pelo fato de sua performance ter sido ligeiramente maior na comparação realizada na Seção 3, por sua rapidez na instalação do serviço no servidor e por possuir uma excelente API para a realização da aplicação. Há ainda uma vasta documentação sobre esse SGBD, o que facilita no processo de solução de problemas e de integração com outros serviços.

4.4 DIAGRAMAS DE CASO DE USO

Para que se pudesse entender com clareza as funcionalidades do software a ser criado, elaborou-se o diagrama de caso de uso para detalhar as funções que estão disponíveis para o usuário. Esse diagrama é visto na Figura 4.2, onde são mostrados os casos de uso de todo o programa. Como ele, vê-se que o usuário tem a capacidade de visualizar a interface da aplicação, inserir entidades e relacionamentos no banco (a partir de um carregamento de um CSV e do preenchimento de um formulário), visualizar seus dados em forma de grafos (escolhendo as relações entre eles e vendo os resultados dos relacionamentos) e obter informação de contato.

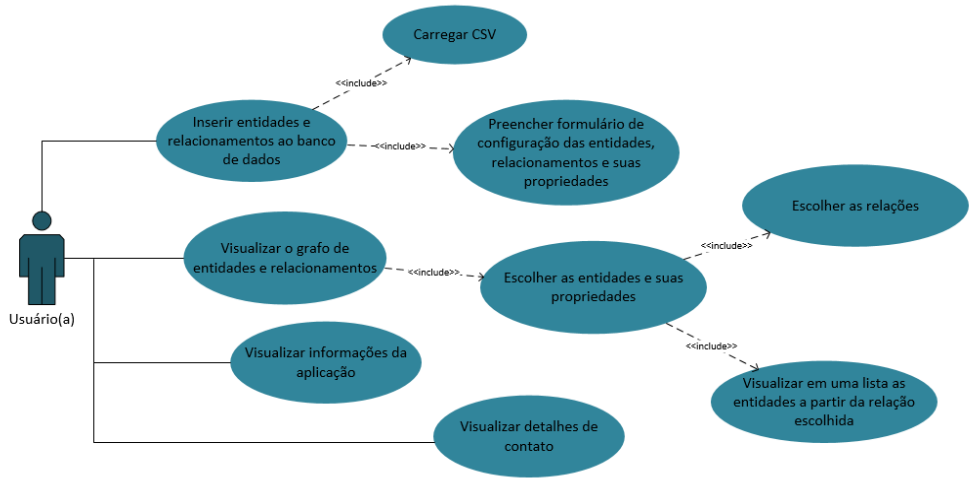

Figura 4.2 Diagrama de Caso de Uso da Ana Lisa.

4.5 DIAGRAMAS DE SEQUÊNCIA

Os diagramas de sequências foram montados com a finalidade de expressar a sequência dos processos nessa aplicação e mostrar como as mensagens são trocadas ao decorrer do tempo. Foram elaborados dois deles para as duas principais sequências: a Inserção de Entidades e Relacionamentos (Figura 4.3) e a Visualização de Grafos (Figura 4.4).

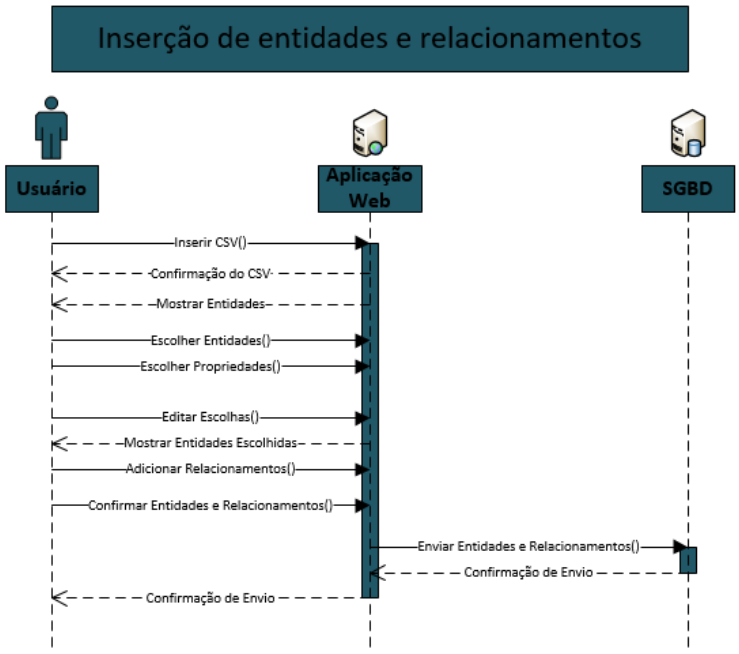

Figura 4.3 Diagrama de Sequência do processo: Inserção de entidades e relacionamentos.

Figura 4.4 Diagrama de Sequência do processo: Visualização de grafos.

Através do diagrama de sequência do processo de inserção de entidades e relacionamentos, percebe-se que os três autores envolvidos são o usuário, a aplicação em um servidor Web e o SGBD (Neo4j). Fazendo uma descrição das sequências, tem-se que primeiro o usuário envia um arquivo CSV para a aplicação que confirma seu recebimento, processa os dados e envia as entidades para o cliente. O usuário, por sua vez, escolhe as entidades e propriedades que quer inserir e a aplicação faz esse gerenciamento e possibilita o usuário a editar suas escolhas. A partir disso, a aplicação mostra as entidades marcadas e o usuário nomeia o relacionamento entre elas. Ele envia esses relacionamentos e a aplicação acrescenta todas essas informações no SGBD (usando o REST API do Neo4j). Por fim, a aplicação retorna uma mensagem de confirmação do processo. O segundo diagrama possui os mesmos autores da primeira e descreve o método de visualização dos dados. Inicialmente o usuário realiza a autenticação com o SGBD, usando a aplicação como intermediário. Depois da confirmação da conexão, a aplicação solicita as entidades do banco que são exibidas pela aplicação para o usuário. Este, por seu turno, elege as entidades usando a aplicação que comunica as escolhas ao banco. O banco envia as entidades e os relacionamentos para a aplicação e ela gera uma página HTML com os

grafos para o usuário visualizar e escolher as entidades e relacionamentos, a fim de obter os resultados dessas relações.

4.6 MAPA DO SITE

O Mapa do Site é um interessante diagrama para explorar as páginas, parte das páginas e os documentos envolvidos em uma aplicação Web. Com ele, consegue-se visualizar os arquivos, fazendo um elo entre as páginas, as estruturas e os arquivos dentro do contexto do software criado para a Web. Dessa forma, foi elaborado um mapa do site para a aplicação Web – Ana Lisa – Figura 4.5. Nesse mapa, nota-se a construção do software Web baseado em uma página inicial com seis partes: Principal, Autenticação, Visualizador de Grafos, Inserir Dados, Contato e Rodapé. As partes Visualizador de Grafos e Inserir Dados fazem o uso do banco de dados Neo4j e a primeira parte mencionada gera uma nova página, a Grafos e Resultados, que mostra a visualização dos dados em forma de grafos. Para finalizar, todas essas páginas usam, principalmente, duas folhas de estilo e cinco *scripts* em JavaScript.

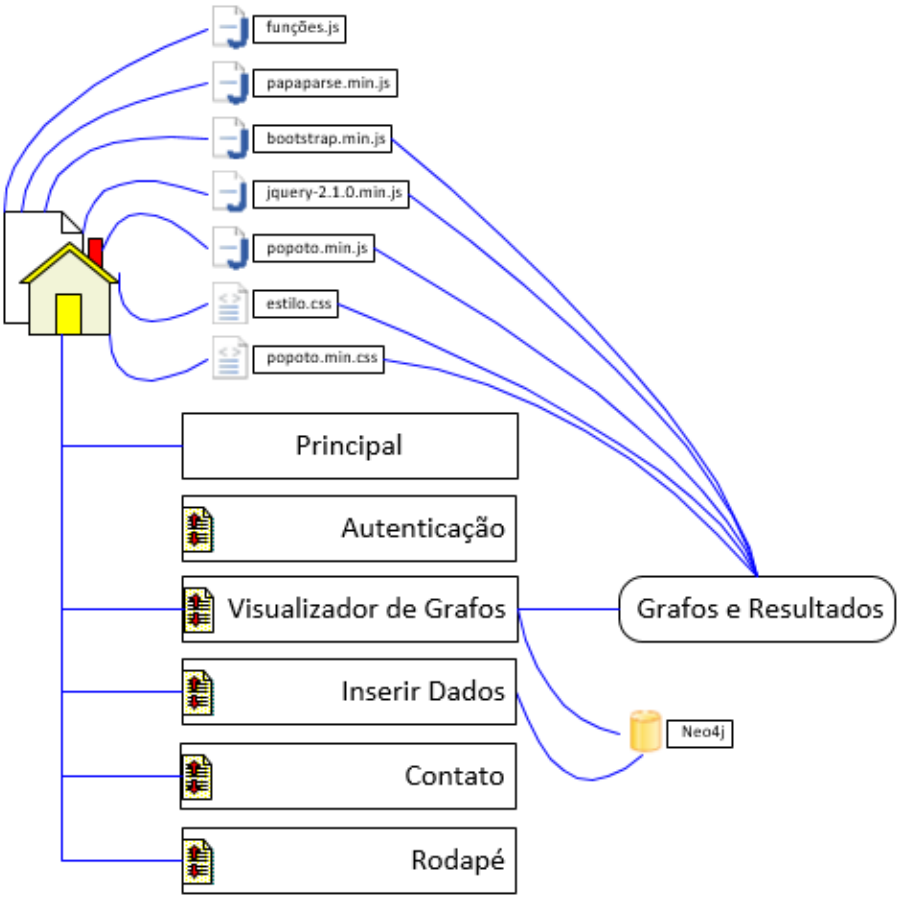

Figura 4.5 Mapa do Site da Aplicação Web Ana Lisa.

4.7 RESULTADO FINAL E O USO FINAL DA APLICAÇÃO

Depois de realizada todas as fases da construção do software: Análise, Projeto, Desenvolvimento e Teste; tem-se o resultado final apresentado pelas figuras (Figura 4.6), (Figura 4.7), (Figura 4.8), (Figura 4.9), (Figura 4.10), (Figura 4.11), (Figura 4.12) e (Figura 4.13); sendo que cada uma delas expressa uma parte da aplicação de acordo com o Mapa do Site (Figura 4.5).

Como se pode ver nas figuras a seguir, realizou-se uma aplicação que pudesse atender os requisitos de usabilidade de acordo com as metas de qualidade da seção 2.6.1. Com isso, vê-se que a aplicação é bem simples de usar, pois todas as ações são seguidas por passos muito bem explicados e os formulários são dinâmicos: os elementos aparecem e somem de acordo com a ação do usuário, como se pode ver nos formulários da Figura 4.8 e da Figura 4.9. A Ana Lisa é eficiente e eficaz, porque ela é capaz de executar os recursos e de atingir os objetivos que foram propostos, ou seja, foi possível inserir dados ao Neo4j e visualizá-los em forma de grafos de maneira adequada e conforme os requisitos funcionais. Pode-se observar isso na Figura 4.11 e na Figura 4.12, onde são mostrados os dados do banco de dados no formato de grafo com a finalidade de visualizar as informações de forma mais clara e compreensível ao ser humano, segundo conhecido na seção 2.2, a qual trata da Visualização de Informação e administração e estruturação dos dados. A aplicação também é útil, por fornecer um serviço que insere dados de entidades e relacionamentos, estruturados em um arquivo CSV, no Neo4j para que eles possam ser visualizados em forma de grafo pela Ana Lisa.

Figura 4.6 Página principal da aplicação Ana Lisa.

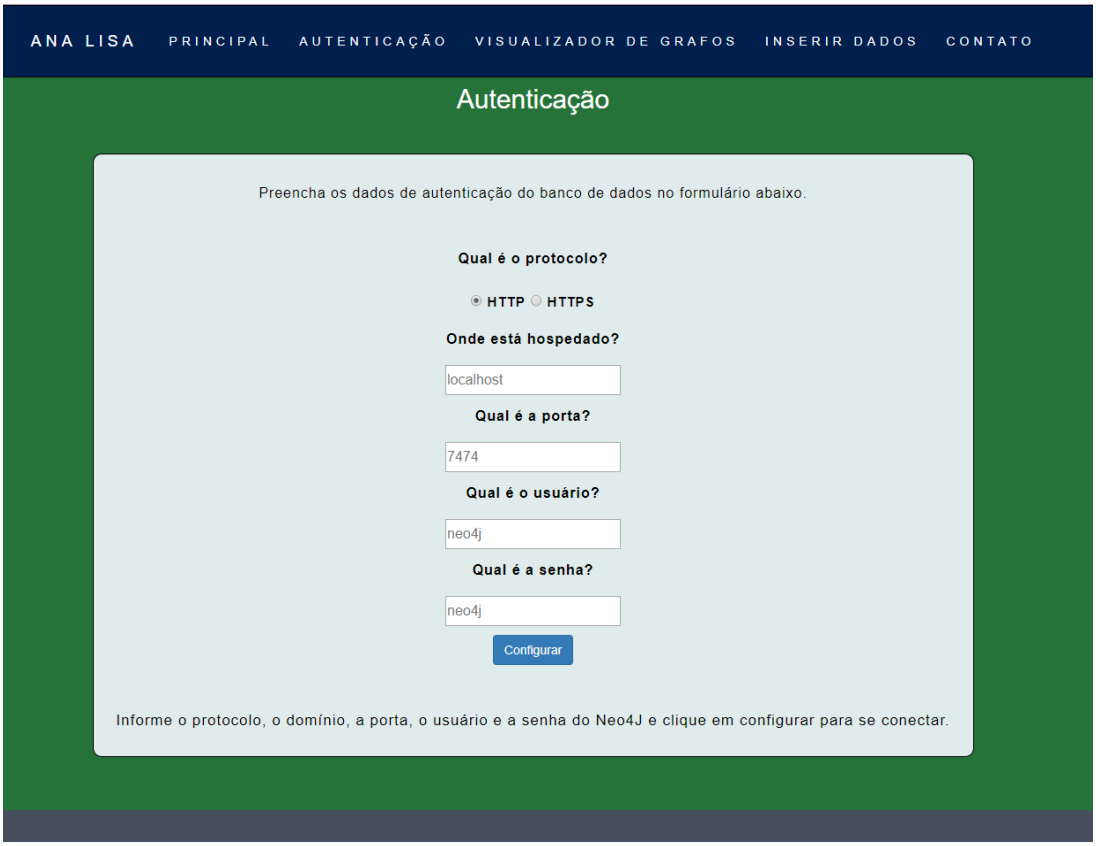

Figura 4.7 Formulário de autenticação para liberar o uso das funcionalidades da Ana Lisa

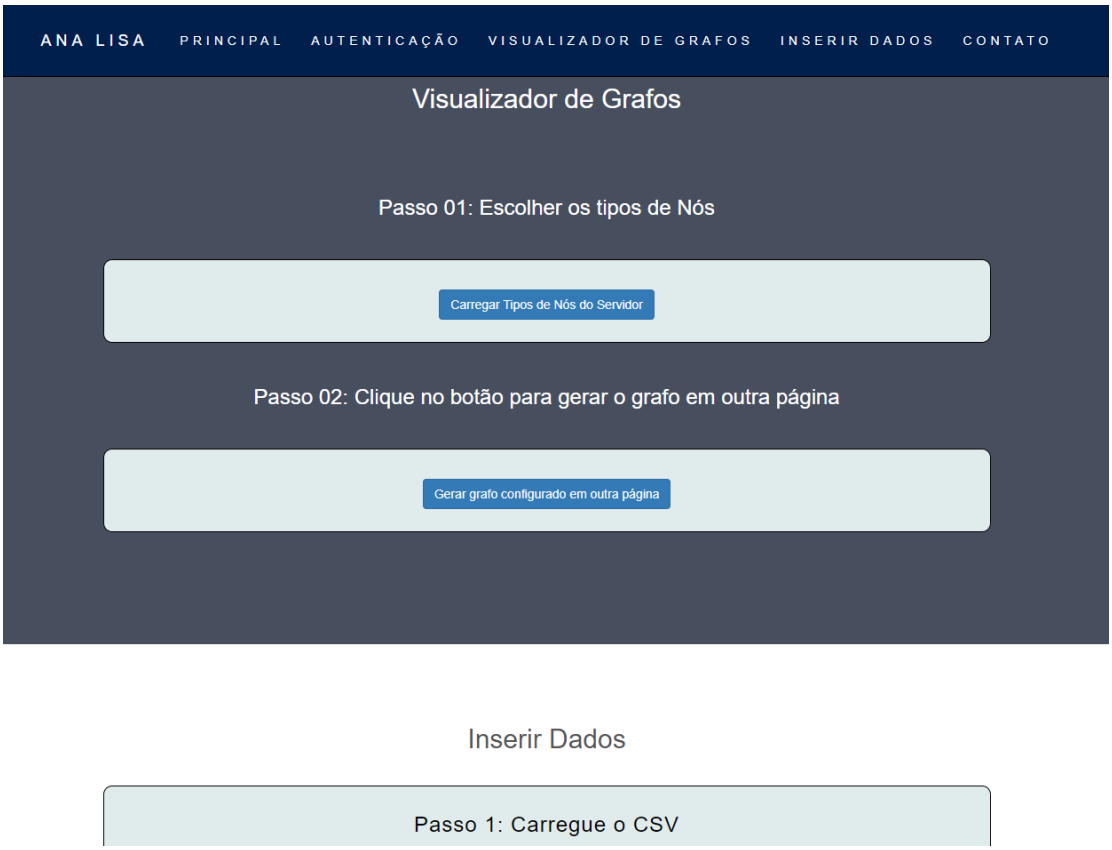

Figura 4.8 Formulário do Visualizador de Grafos.

Figura 4.9 Formulário de Inserção de Entidades e Relacionamentos.

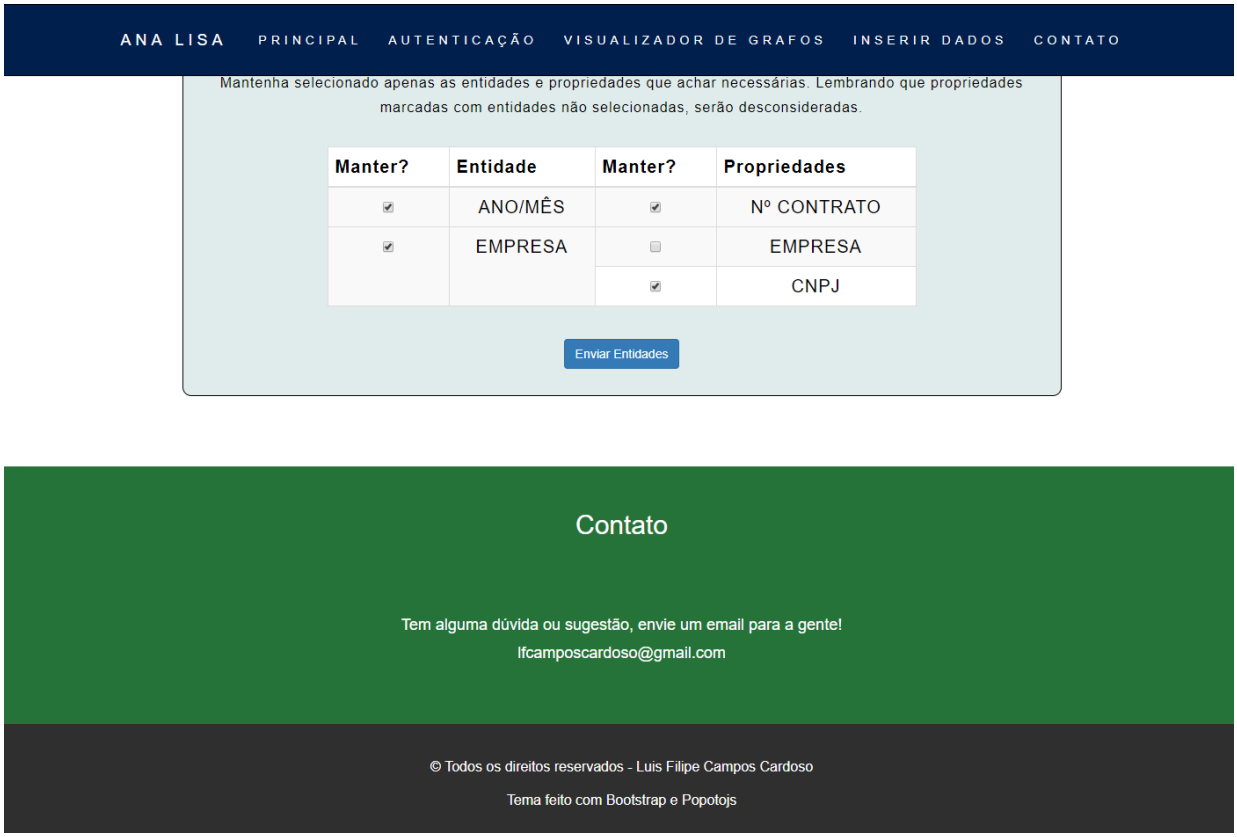

Figura 4.10 Parte de contato e rodapé.

Figura 4.11 Página de Visualização do Grafo.

Figura 4.12 Parte de resultados da página de Visualização do Grafo.

Para fazer essa aplicação Web, foi fundamental o uso do banco de dados Neo4j, pois ele gerencia e armazena todos os dados disponíveis para visualização; conforme a arquitetura da Figura 4.1. Também foi essencial o uso das bibliotecas de desenvolvimento, pois elas aceleraram o processo de desenvolvimento e fizeram com que a página Web ficasse dinâmica. Nesse contexto, o Bootstrap serviu como ferramenta na criação de toda a interface, a fim de que ficasse agradável sob a visão do usuário e funcionasse de forma responsiva de acordo com a tela. Na Figura 4.13, por exemplo, tem-se como ficou a tela para ser visualizada por celulares e *smartphones*. Percebe-se seu uso na parte principal da aplicação (Figura 4.6), onde foi definido o formato do *layout* em painéis retangulares com o conteúdo em seu interior. O que faz a separação das partes é justamente a mudança da cor de seu fundo. Essa separação é muito visível na Figura 5.10, onde se vê a transição entre o

formulário de inserção de dados da parte de contatos e do rodapé através das cores empregadas no fundo.

Figura 4.13 Tela responsiva capaz de se adaptar a telas menores, como as de *smartphones*.

A biblioteca Papaparse tem um papel importantíssimo para ler o conteúdo dos arquivos CSV que serão carregados e, com a ajuda do JavaScript e do jQuery, foi possível dinamizar os formulários (Figura 4.9) e transformar os dados para que se possa fazer a comunicação com o Neo4j. Através do jQuery, criou-se as funções capazes de fazer a conexão com o REST API do banco e, com o auxílio do Popoto.js, a visualização das entidades e relacionamentos do Neo4j se tornou realizável (Figura 4.11). Ela usa os conceitos de ontologia da seção 2.5 para criar o processo de visualização da informação. O Popoto.js, então, usa os metadados para classificar os dados, ou seja, ela busca as entidades e relacionamentos, monta a visualização e as classifica de acordo com seu rótulo (*label*) e propriedades.

Depois de usar a aplicação final, ela conseguiu inserir dados (no teste, utilizou-se dados abertos do Portal da Transparência) através de um arquivo CSV e visualizar os dados

em forma de grafos de uma instância do banco de dados, após um processo de autenticação no serviço do banco ativo escolhido. A primeira funcionalidade utilizou um arquivo CSV como formato que estruturou os dados de entrada. Como era necessário criar uma forma eficiente de se adicionar os dados na aplicação, optou-se por este tipo de administração de dados por ser simples e popular em sistemas de dados abertos, como é o caso do Portal da Transparência. Isso tornou esse procedimento muito eficaz e capaz de gerir os dados para que eles fossem trabalhados conforme as regras de negócio do software. Com isso, o resultado final foi positivo e, no teste, conseguiu-se inserir com sucesso um arquivo com as licitações de propaganda do Ministério do Turismo.

Já a segunda funcionalidade (descrita no parágrafo anterior) é um item que traz grande valor ao software, pois ele está intimamente ligado com a maneira que os dados são vistos pelo usuário. Desse modo, essa implementação foi fundamental, porque ela vai fazer a investigação das informações adicionadas pela primeira funcionalidade destacada. A resposta final foi muito satisfatória, por proporcionar uma solução computacional que pudesse caracterizar os dados em forma de grafo. Esse efeito foi essencial para a criação das relações entre as entidades. Na execução dos testes, quando se usou o exemplo dos presidentes e seus ministros, viu-se claramente o quanto alguns ministros permanecem no cargo, mesmo após a saída de um presidente. Esse foi um exemplo de como essa aplicação montou um modelo visual interessante.

O processo de autenticação com o banco de dados fez com que se pudesse dar a Ana Lisa a autonomia de escolha de qual banco usar, informando parâmetros básicos como local, protocolo, porta, usuário e senha. Além de trazer mais segurança das transações que estão sendo feitas na aplicação, a fim de que se atingisse os princípios da Segurança da Informação (Integridade, Disponibilidade, Confiabilidade e Autenticidade) com eficácia e eficiência. Desse modo, nos testes, o software foi íntegro por não permitir que a aplicação fosse violada através de um acesso de uma pessoa não autorizada; foi disponível, pois ela estava disponibilizada quando solicitado por um usuário quando os dados de autenticação estavam corretos; foi confiável, pelo fato de haver um controle de acesso do usuário e do banco que se deseja usar e; por fim, foi autêntico por saber de quem foi a autoria das inserções no banco que são previamente controladas. Isso é extremamente relevante, porque pensar em Segurança da Informação na construção de softwares é primordial para aumentar a confiança do produto e evitar incidentes que tragam impactos negativos e prejuízos. A autenticação trouxe mais confiança a Ana Lisa através de um simples formulário de controle de usuário e de uma regra de negócio clara da aplicação que priorizasse esses conceitos de segurança. Isso ficou visível no resultado final na fase de testes.

5 – CONCLUSÕES E TRABALHOS FUTUROS

A visualização da informação é muito importante na vida pessoal e profissional do ser humano, mesmo porque ela é indispensável para trazer valor aos dados e informações disponíveis. Nesse campo, criar aplicações capazes de realizar essa ação através de uma aplicação Web é uma alternativa para criar uma interface que possibilite inserir dados e visualizá-los para uma melhor administração dos dados, que ocasiona em uma análise investigatória das relações de modo mais efetivo.

Esse trabalho realizou uma aplicação Web cliente que fosse possível visualizar grafos armazenados em um banco de dados orientados a grafos, a fim de que se pudesse construir relações com os nós criados. Além disso, o software é capaz de inserir novos dados, como relacionamentos e entidades, a partir de um arquivo CSV estruturado para que se possa ser visualizado posteriormente. Dessa forma, pensou-se em fazer um serviço, onde se visualizasse e inserisse grafos para fazer mapeamento de relações.

É nesse contexto que nasceu a Ana Lisa, a analista de dados capaz de fazer relações com as entidades armazenadas. Apesar de a aplicação ter esse escopo bem definido, ela possui certas limitações que podem ser trabalhadas em trabalhos futuros:

- A aplicação foi construída para inserir dados em um serviço do Neo4j já instalado. Ela não faz nenhum processo de orquestração dos serviços, nem criação de novos. O usuário apenas informa onde está hospedado o banco de dados e sua porta. Para um futuro melhoramento, poder-se-á instalar, por exemplo, um Docker [16] para gerenciar a criação e administração desses serviços dentro de container que facilite a replicação e a gerência dos bancos. Nesse caso, usar-se-á o Docker Swarm [17] para gerir todos os motores Docker e fazer a interface com o Neo4j [18];
- Um recurso que seria muito interessante de implantar, já que o projeto original tem isso como escopo, seria uma melhor administração dos dados no banco. No estado atual, só é permitido inserir novos registros; entretanto, seria de grande valia poder excluir lançamentos equivocados ou ultrapassados para facilitar na hora da escolha das entidades que serão visualizadas;
- O processo de análise de dados da aplicação Ana Lisa na versão atual só verifica relacionamentos e suas entidades. O Neo4j dispõe de diversos recursos interessantes para análise de dados, como a execução de algoritmos e filtros. Nas próximas implementações, poder-se-á fazer o uso de algoritmos

como o *ShortestPaths* (que usa o algoritmo de Dijkstra) para ver a proximidades entre as entidades e buscar novas relações e informações sobre eles, e o *PageRank* para mostrar relevância entre as entidades e relações. Os filtros que podem ser utilizados futuramente seriam: o *ORDERBY* (ordenar a saída), o WHERE (filtrar por um parâmetro) e o LIMIT (limitar a saída);

- O uso do Popoto.js foi essencial para construir a visualização da Ana Lisa; no entanto, ela limita um pouco a atuação da análise dos dados, por suas estruturas já definidas. Nesse projeto em questão, já foi feita uma adaptação dessa biblioteca para atuar conforme o projeto, principalmente mudando um pouco a interface de visualização, folhas de estilos e *scripts* e traduzindo certos conteúdos. O ideal, em próximas versões, é usar outras bibliotecas que tragam mais liberdade na visualização, que possibilitam o emprego de algoritmos específicos (como o *PageRank* e o *ShortestPaths*) e o uso de filtros no processo de visualização mencionado no item acima. Ainda assim, ele foi fundamental para dar agilidade ao projeto e terminá-lo dentro do prazo. Ficam como sugestões as bibliotecas de visualização: D3.js, vis.js [20] e Cytoscape.js [21].

Mesmo com as limitações vistas, esse software foi capaz de trazer uma primeira visão, na forma de produto, de como uma solução de análise de dados interativa deve ser, ainda que envolto em uma concepção minimalista. Com isso, essa primeira versão da Ana Lisa será um excelente ponto de partida para a construção de uma aplicação mais abrangente e funcional ao usuário final. Com isso, a aplicação serviu como um mote para explicitar novos requisitos para as versões futuras. Desse modo, vê-se, depois de executado esse projeto, que a aplicação pode tomar outras proporções e, como sugestão, pode abarcar as seguintes funcionalidades:

- Dentre as várias aplicações, as de geoprocessamentos estão entre as mais vitais para gerar informação e conhecimento dentro de certos contextos. Por exemplo, em um software onde se deseja explorar pontos da cidade, nada é mais natural do que trabalhar com mapas. O banco de dados orientado a grafo é uma excelente escolha para lidar com essa questão, já que ele formaliza relacionamento entre as entidades. Nessa situação, ele pode criar relações entre os pontos da cidade (que seriam as entidades), as quais poderiam criar relações entre si e se organizar em categorias em um modelo ontológico. Por exemplo, criar uma arquitetura onde os locais e as categorias são as entidades e, a relação poder ser *servem comida*. Nessa situação, poder-se-ia usar

Brasileira como categoria das entidades restaurantes que se associam através da relação *serve comida*. Com isso, os restaurantes que tem esse relacionamento são destacados no mapa. Além desse exemplo, diversos outros podem ser incluídos à Ana Lisa em versões futuras não só com grafos tradicionais; como também, com mapas;

- No atual momento, a aplicação funciona em um contexto monolítico, onde as *queries* são estáticas e baseadas nos processos da interface. Para estruturas mais complexas, o interessante seria projetar uma arquitetura baseada em microserviços para abranger mais situações. O Neo4j já possui isso dentro de sua API, o que facilita muito no processo de desenvolvimento dentro da Ana Lisa em próximas versões. Com isso, a aplicação poderá gerir mais recursos que o banco oferece e melhorará a performance consideravelmente.

Ainda assim, a atual versão da Ana Lisa, já realiza, de forma bastante eficaz e simples, certas tarefas que resolvem a questão da visualização de entidades e relações e, de investigação e estruturação dos dados. Desse modo, pode-se destacar imediatamente esses pontos:

- Ao final do projeto, percebe-se que o objetivo de visualizar grafos em uma página Web foi cumprido com sucesso. Ao mesmo passo que se criou toda uma estrutura para essa realização, ou seja, elaborou-se um conceito, a Ana Lisa, como uma analista de dados capaz de analisar os dados armazenados e expor suas relações e transpô-los em uma tabela, já com o resultado das relações inseridas. Isso afirma sua capacidade investigativa dos dados inseridos;
- Um outro objetivo, que foi executado, foi a inserção de novos dados, a partir de um arquivo CSV. Dessa forma, a aplicação alcançou outro nível de maturação por possibilitar a leitura de um padrão de texto tão difundido que pudesse ser manipulado e caracterizado em termo de nós e relacionamentos. Mesmo porque, tem-se que se preocupar em como se deve formatar os dados, que, muitas vezes, não estão estruturados. Dessa maneira, com o auxílio do Neo4j, a inserção dos dados pela Ana Lisa a esse banco foi completada com sucesso para sua visualização posterior através do mesmo software;
- Uma questão que se pode destacar é que a solução buscou se basear em uma interface simples e de fácil compreensão, o que facilita muito na sua popularização e no processo de venda do software. Apesar de ainda não ser um produto final, ele já tem esse conceito desde sua concepção com a ideia de

uma aplicação que não, necessariamente, precise de um profissional da área para sua operação. Nesse contexto, os conceitos da Usabilidade foram fundamentais para assimilar essa questão dentro da construção da Ana Lisa. Dessa maneira, pensou-se muito na utilidade da aplicação, se ela atendia os requisitos, se ela era fácil de uso e se ela era eficiente. Pelos testes e pelas demonstrações que foram realizadas com o orientador, colegas do departamentos e amigos, viu-se uma grande aceitação da aplicação quanto as questões de usabilidade e experiência deles com a aplicação. Desse modo, esse objetivo foi cumprido e será ajustado à medida que são desenvolvidas novas versões da Ana Lisa.

Voltando-se para as questões da seção 3, sobre a comparação entre dois bancos de dados orientado a grafos, percebe-se que esse tipo de banco de dados se tornou uma excelente alternativa ao modelo relacional em aplicações de rede social. Isso se deve por ele apresentar uma melhor forma de visualização e suprir a necessidade do alto grau de disponibilidade, escalabilidade e alto volume de dados por parte das aplicações contemporâneas. Viu-se também no experimento que o Neo4j e o OrientDB são capazes de armazenar os dados de uma arquitetura complexa com diversas relações. Ao testar a performance de ambas, nota-se uma ligeira vantagem do Neo4j quanto a performance na grande maioria dos aspectos estudados.

De modo geral, o software desenvolvido estava de acordo com as expectativas por trazer à luz novas formas de banco de dados NoSQL e por produzir uma aplicação capaz de visualizar e inserir dados, muitas vezes não estruturados. Estes, por sua vez, vão se moldar em forma de grafo, usando o SGBD, para investigar informações úteis para o contexto desses dados. Com isso, concluiu-se que todos os objetivos estabelecidos e o escopo definido foram executados conforme o programado.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] RAQUELINE R. M. PENTEADO, REBECA SCHROEDER, DIEGO HOSS, JAQUELINE NANDE, RICARDO M. MAEDA, WALMIR O. COUTO, CARMEM S. HARAUM. Estudo sobre Bancos de Dados em Grafos Nativos. UFPR, 2014;

[2] BERNADETTE FARIAS LÓSCIO, HÉLIO RODRIGUES DE OLIVEIRA, JONAS CÉSAR DE SOUSA PONTES. NoSQL no desenvolvimento de aplicações Web colaborativas. SBSC 2011;

[3] N.M.A. MOHAMED ALI, DR. T. PADMA. Graph Database: A Contemporary Storage Mechanism for Connected Data. International Journal of Advanced Research in Computer and Communication Engineering, Vol. 5, Issue 3, Março de 2016;

[4] GUILHERME M. ALVAREZ, FLÁVIO CECI, ALEXANDRE L. GONÇALVES. Análise Comparativa dos Bancos Orientados a Grafos de Primeira e Segunda Geração – Uma Aplicação na Análise Social. III Encontro de Inovação em SI, Florianópolis, SC, 17 a 20 de Maio de 2016;

[5] IAN ROBINSON, JIM WEBBER & EMIL EIFREM. Graph Databases. O'Reilly Media, Inc. Second Edition: Junho 2015;

[6] SILVA, Gustavo Henrique Moreira Alvares da. Um modelo de visualização de dados utilizando banco de dados orientado a grafo suportado por big data. 2016. xxiii, 150 f., il. Dissertação (Mestrado em Engenharia Elétrica)— Universidade de Brasília, Brasília, 2016;

[7] Portal da Transferência. Disponível em: <http://www.portaltransparencia.gov.br/>. Acesso em 21 de abril de 2017;

[8] DE LIMA, JÚNIO CÉSAR; DE CARVALHO, CEDRIC L. Ontologias-owl (web ontology language). Universidade Federal de Goiás, 2005;

[9] ALVES, LUCAS JK; BECKER, KARIN. Implementação de um Repositório de Versões de Serviços Web usando OrientDB;

[10] FORTE, MARCOS; SOUZA, WANDERLEY LOPES DE; PRADO, ANTONIO FRANCISCO DO. Utilizando ontologias e serviços web na computação ubíqua. XX Simpósio Brasileiro de Engenharia de Software, 2006;

[11] DE OLIVEIRA, SAMUEL SILVA. Bancos de Dados Não-Relacionais: Um Novo Paradigma para Armazenamento de Dados em Sistemas de Ensino

Colaborativo. Revista da Escola de Administração Pública do Amapá, v. 2, n. 1, p. 184-194, 2014;

[12] DAI, JIANGPENG ET AL. Analysis of distribution network outrage region based on graph database. Electricity Distribution (CICED), 2016 China International Conference on. IEEE, 2016. p. 1-4;

[13] BARBOSA, LEONARDO MAIA; ATTUX, ROMIS. Uma Proposta para Identificar Informação em Artigos Científicos Utilizando Redes Complexas. VII EADCA–Encontro dos Alunos e Docentes do Departamento de Computação e Automação Industrial, 2014;

[14] MILLER, JUSTIN J. Graph database applications and concepts with Neo4j. Proceedings of the Southern Association for Information Systems Conference, Atlanta, GA, USA, 2013;

[15] HOLZSCHUHER, FLORIAN; PEINL, RENÉ. Performance of graph query languages: comparison of cypher, gremlin and native access in Neo4j. Proceedings of the Joint EDBT/ICDT 2013 Workshops. ACM, 2013. p. 195-204;

[16] Docker. Disponível em: <https://www.docker.com/>. Acesso em 26 de junho de 2017;

[17] AGGARWAL, DIPPY. Neo4j Container Orchestration with Kubernetes, Docker Swarm & Mesos. Disponível em: <https://neo4j.com/blog/neo4j-container-orchestration-kubernetes-docker-swarm-mesos/>. Acesso em 26 de junho de 2017;

[18] BOYD, RYAN. Graph Compute with Neo4j: Built-in Algorithms, Spark & Extensions. Disponível em: <https://neo4j.com/blog/graph-compute-neo4j-algorithms-spark-extensions/>. Acesso em 26 de junho de 2017;

[19] vis.js. Disponível em: <http://visjs.org/>. Acesso em 26 de junho de 2017;

[20] Cytoscape. Disponível em: <http://www.cytoscape.org/index.html>. Acesso em 26 de junho de 2017;

[21] CARVALHO, AFONSO A. PRAÇA. Arquitetura e Desenvolvimento de uma Aplicação Web Distribuída Baseada em Javascipt. Trabalho de Conclusão de Curso. UNIRIO, 2014;

[22] NETO, MACHADO; PETERSEN, OGGO. Análise de bibliotecas para geração de gráficos na web. (2013);

[23] Popoto.js. Disponível em: <http://www.popotojs.com/>. Acesso em 27 de junho de 2017;

[24] D3.js. Disponível em: <https://d3js.org/>. Acesso em 27 de junho de 2017;

[25] Papaparse. Disponível em: <http://papaparse.com/>. Acesso em 27 de junho de 2017;

[26] MediaWiki, VisualEditor. Disponível em: <https://www.mediawiki.org/wiki/VisualEditor>. Acesso em 27 de junho de 2017;

[27] LAUFER, CARLOS. Guia de Web Semântica. 2015. Disponível em: <http://ceweb.br/media/docs/publicacoes/13/Guia_Web_Semantica.pdf>. Acesso em 15 de junho de 2017;

[28] VAZ, FERNANDO ROSA; C. L. CARVALHO. Visualização de Informações. Universidade Federal de Goiás, 2004;

[29] BASSO, CARLA DE ALMEIDA MARTINS; GOMES, BRUNA CARVALHO KALLES. Desempenho de Banco de Dados Não Relacionais com Big Data. CONTECSI - International Conference on Information Systems and Technology Management, 2015;

[30] RUBINO, GABRIEL. Visualização de Redes Gênicas a partir da Integração de Dados Biológicos. Trabalho de Conclusão de Curso. Universidade Tecnológica Federal do Paraná, 2016;

[31] PENTEADO, RAQUELINE RM ET AL. Um estudo sobre bancos de dados em grafos nativos. X ERBD-Escola Regional de Banco de Dados. 2014;

[32] BOSTOCK, MICHAEL; OGIEVETSKY, VADIM; HEER, JEFFREY. D^3 data-driven documents. IEEE transactions on visualization and computer graphics, v. 17, n. 12, p. 2301-2309, 2011;

[33] BAO, FAN; CHEN, JIA. Visual framework for big data in d3. js. Electronics, Computer and Applications, 2014 IEEE Workshop on. IEEE, 2014. p. 47-50;

[34] PINTO, ADILSON LUIZ; DA SILVA, ARMANDO MALHEIRO; SENA, PRISCILA MACHADO BORGES. Ontologias baseadas na visualização da informação das redes sociais. Prisma.com, n. 13, 2017;

[35] DE MENDONÇA, JÔNATAS MEDEIROS; DA SILVA, RODRIGO MEDEIROS SOARES. Técnicas de usabilidade e testes automatizados em processos de desenvolvimento de software empírico. Trabalho de Conclusão de Curso. Universidade de Brasília, 2014;

[36] GONÇALVES, MILENI KAZEDANI. Usabilidade de Software: Estudo de Recomendações Básicas para Verificação do Nível de Conhecimento dos Alunos dos Cursos de Design Gráfico e Sistemas de Informação da Unesp/Bauru. Dissertação. Universidade Estadual Paulista Júlio de Mesquita Filho, 2008;

[37] DAMASIO, JULIANA; INACIO, TASMAY. Avaliação de usabilidade e de impacto cognitivo de uma aplicação baseada em áudio para a navegação de pessoas com deficiência visual. Revista da Graduação, v. 8, n. 1, 2015;

[38] RODRIGUES, Adriana Alves; DIAS, Guilherme Ataíde. Estudos sobre visualização de dados científicos no contexto da Data Science e do Big Data. Pesquisa Brasileira em Ciência da Informação e Biblioteconomia, v. 12, n. 1, 2017;

[39] DE SOUZA, Alexandre Morais et al. Critérios para Seleção de SGBD NoSQL: o Ponto de Vista de Especialistas com base na Literatura. Anais doX Simpósio Brasileiro de Sistemas de Informação (Londrina–PR, Brasil. 27 a 30/05/2014, 2014;

[40] BRETERNITZ, Vivaldo José; SILVA, Leandro Augusto. Big data: Um novo conceito gerando oportunidades e desafios. Revista Eletrônica de Tecnologia e Cultura, v. 2, n. 2, 2013.