ANDRESSA FARIA
EDILSON JUSTINIANO

BANCO DE DADOS ORIENTADO A GRAFOS APLICADO À BUSCA POR MÃO DE OBRA

UNIVERSIDADE DO VALE DO SAPUCAÍ
POUSO ALEGRE – MG
2015

ANDRESSA FARIA
EDILSON JUSTINIANO

BANCO DE DADOS ORIENTADO A GRAFOS APLICADO À BUSCA POR MÃO DE OBRA

Trabalho de Conclusão de Curso apresentado como requisto parcial à obtenção do título de Bacharel em Sistemas de Informação na Universidade do Vale do Sapucaí – UNIVAS.

Orientador: Prof. MSc. Márcio Emílio Cruz Vono de Azevedo

UNIVERSIDADE DO VALE DO SAPUCAÍ
POUSO ALEGRE – MG
2015

Faria, Andressa; Justiniano, Edilson

BANCO DE DADOS ORIENTADO A GRAFOS APLICADO À BUSCA POR MÃO DE OBRA / Andressa Faria, Edilson Justiniano – Pouso Alegre – MG: Univás, 2015.
52 f. : il.

Trabalho de Conclusão de Curso (graduação) – Universidade do Vale do Sapucaí, Univás, Sistemas de Informação.
Orientador: Prof. MSc. Márcio Emílio Cruz Vono de Azevedo

1. mão de obra. 2. grafo. 3. banco de dados.

ANDRESSA FARIA
EDILSON JUSTINIANO

BANCO DE DADOS ORIENTADO A GRAFOS APLICADO À BUSCA POR MÃO DE OBRA

Trabalho de conclusão de curso defendido e aprovado em 01/01/2016 pela banca examinadora constituída pelos professores:

Prof. MSc. Márcio Emílio Cruz Vono de Azevedo
Orientador

Prof[a]. MSc. Nome da professora
Avaliadora

Prof. MSc. Nome do professor
Avaliador

De Andressa Faria.

Dedico este trabalho . . .

De Edilson Justiniano.

Dedico este trabalho . . .

AGRADECIMENTOS

De Andressa Faria

Agradeço . . .

De Edilson Justiniano

Agradeço . . .

“Complicar é simples,
simplificar que é complicado.
(Paulo Sérgio dos Santos)

FARIA, Andressa; JUSTINIANO, Edilson. **BANCO DE DADOS ORIENTADO A GRAFOS APLICADO À BUSCA POR MÃO DE OBRA**. 2015. Monografia – Curso de SISTEMAS DE INFORMAÇÃO, Universidade do Vale do Sapucaí, Pouso Alegre – MG, 2015.

RESUMO

Este trabalho apresenta ...

Palavras-chave: mão de obra. grafo. banco de dados.

FARIA, Andressa; JUSTINIANO, Edilson. **BANCO DE DADOS ORIENTADO A GRAFOS APLICADO À BUSCA POR MÃO DE OBRA**. 2015. Monografia – Curso de SISTEMAS DE INFORMAÇÃO, Universidade do Vale do Sapucaí, Pouso Alegre – MG, 2015.

ABSTRACT

This work presents . . .

Key words: manpower. graph. database.

LISTA DE FIGURAS

LISTA DE SIGLAS E ABREVIATURAS

ACID	Atomicidade, Consistência, Isolamento e Durabilidade
API	*Application Programming Interface*
HTTP	*Hypertext Transfer Protocol*
JCP	*Java Community Process*
JSF	*JavaServer Faces*
JSP	*JavaServer Pages*
JVM	*Java Virtual Machine*
MVC	*Model – View – Controller*
NoSQL	*Not Only SQL*
SQL	*Structured Query Language*
UML	*Unified Modeling Language*

SUMÁRIO

INTRODUÇÃO

Com a constante evolução tecnológica é possível notar que, a cada dia, mais pessoas estão sendo inseridas em um mundo globalizado. Pertencer a este meio tem mudado completamente a maneira de se realizar tarefas, uma vez que, a revolução tecnológica busca facilitar o que até então era trabalhoso. Atualmente, nos deparamos com situações nas quais as pessoas desempenham mais de um papel, dividindo o seu tempo entre tarefas profissionais, pessoais e aquelas que chamamos de rotineiras as quais muitas vezes, não são realizadas por elas e sim por terceiros. Um profissional terceirizado é capaz de cuidar de todas as tarefas extras que não cabem na rotina, no entanto, encontrá-los tem se tornado cada vez mais difícil, uma vez que confiar a sua residência ou, ainda, a sua intimidade a uma pessoa não conhecida gera muita insegurança. Ainda, vale ressaltar que este tipo de trabalho está cada vez mais escasso e raro no cenário atual.

Com o advento da internet, e mais tarde sua popularização, uma série de aplicações para diferentes fins vem sendo desenvolvidas. Estes aplicativos tem como finalidade apresentar soluções para os mais diversos tipos de problemas. É possível notar que as redes sociais geram um grande impacto no modelo de vida que seguimos. Hoje, sabemos que tudo e todos estão conectados, direta ou indiretamente, e que as informações são trocadas a uma velocidade surpreendente. Precisamos obter informações de forma clara e rápida (BARBOSA, 2011).

Com intuito de obter acesso à informação de maneira instantânea, a migração dos antigos computadores pessoais (*desktops*) para dispositvos cada vez mais portáteis tornou-se comum. Podemos citar, como exemplo, a criação dos *tablets* que proporcionaram uma grande revolução tecnológica, pois, a partir desta invenção as pessoas passaram a ter liberdade de acesso, uma vez que as informações se encontram na palma de suas mãos. Seguindo esta tendência de mobilidade, os *smartphones* foram desenvolvidos.

O sucesso obtido por estes dispositivos móveis, somado a evolução das redes de dados 3G e 4G, permitiu a conexão entre as pessoas de qualquer lugar e a qualquer momento bastando alguns toques, tornando assim ainda mais fácil localizá-las e comunicar-se com elas.

A pesquisa realizada pelo Instituto Nielsen reafirma o contexto anteriormente mencionado. Ela comprova que o tempo gasto pelos usuários em seus *smartphones* é maior do que em seus computadores pessoais, sendo que boa parte deste tempo é utilizado para acesso às redes sociais. Isto ocorre pois os dispositivos móveis atuais são capazes de substituir os computadores pessoais em uma série de atividades, tornando-os um computador de mão disponível a qualquer

instante. A pesquisa, ainda demonstra o sucesso das redes sociais que, segundo a mesma, se deve a disseminação dos dispositivos móveis e ao modelo de vida que grande parte das pessoas segue atualmente.

Com a ampliação desta forma de comunicação, somada a evolução das tecnologias já discorridas, houve a necessidade de se realizar algumas melhorias voltadas aos bancos de dados, a fim de otimizar as tarefas realizadas por ele, levando assim, a criação de uma nova geração de bancos de dados, denominada NoSQL[1].

Dentre esta nova geração, o banco de dados orientado a grafos vem se destacando por apresentar grande potencial para manipulação de dados em diferentes áreas como: relacionamento interpessoal, biológicas, rotas geográficas, entre outras.

Penha e Carvalho (2013) utilizaram esta tecnologia para desenvolver uma aplicação, cujo principal objetivo é recomendar filmes aos seus usuários.

Silva e Pires (2013) realizaram uma comparação entre banco de dados orientado a grafos e os relacionais a fim de apresentar as vantagens que estes possuem sob os bancos de dados relacionais.

Junid et al. (2012) utilizaram a teoria dos grafos para tentar otimizar o processo de alinhamento da sequência de DNA a fim de determinar a região comum entre duas ou mais sequências deste.

Com o intuito de tentar solucionar o problema relacionado a busca por mãos de obra temporárias, este trabalho tem como proposta apresentar uma possível solução, através de uma aplicação *web*, seguindo o modelo de negócio das redes sociais, utilizando o conceito de orientação a grafos, a fim de gerar a familiarização desta ferramenta, uma vez que este tipo de serviço já está intrínseco na atualidade.

[1] NOSQL: *Not Only SQL* - Bancos de dados que utilizam não somente os recursos de Structure Query Language - SQL, a fim de obter melhor performance.

2 QUADRO TEÓRICO

Neste capítulo são discutidas as técnicas, metodologias e tecnologias que serão utilizadas no desenvolvimento deste trabalho.

2.1 Iconix

O ICONIX, segundo Rosenberg, Stephens e Collins-Cope (2005), foi criado em 1993 a partir de um resumo das melhores técnicas de desenvolvimento de *software* utilizando como ferramenta de apoio a *Unified Modeling Language* - UML[3]. Esta metodologia é mantida pela empresa ICONIX *Software Engineering* e seu principal idealizador é Doug Rosenberg.

Para Rosenberg e Scott (1999), o ICONIX possui como característica ser iterativo e incremental, somado ao fato de ser adequado ao padrão UML auxiliando, assim, o desenvolvimento e a documentação do sistema.

Atualmente, existem diversas metodologias de desenvolvimento de *software* disponíveis, contudo, o ICONIX, em especial, será utilizado para auxiliar no processo de desenvolvimento deste trabalho pois, segundo Silva e Videira (2001), esta metodologia nos permite gerar a documentação necessária para nortear o desenvolvimento de um projeto acadêmico.

De acordo com Rosenberg e Stephens (2007), os processos do ICONIX consistem em gerar alguns artefatos que correspondem aos modelos dinâmico e estático de um sistema e estes são elaborados e desenvolvidos de forma incremental e em paralelo, possibilitando ao analista dar maior ênfase no desenvolvimento do sistema do que na documentação do mesmo. A Figura 1, apresenta uma visão geral dos componentes do ICONIX.

[3] UML: *Unified Modeling Language* - Linguagem de modelagem para objetos do mundo real que habilita os desenvolvedores especificar, visualizar, construí-los a nível de software.

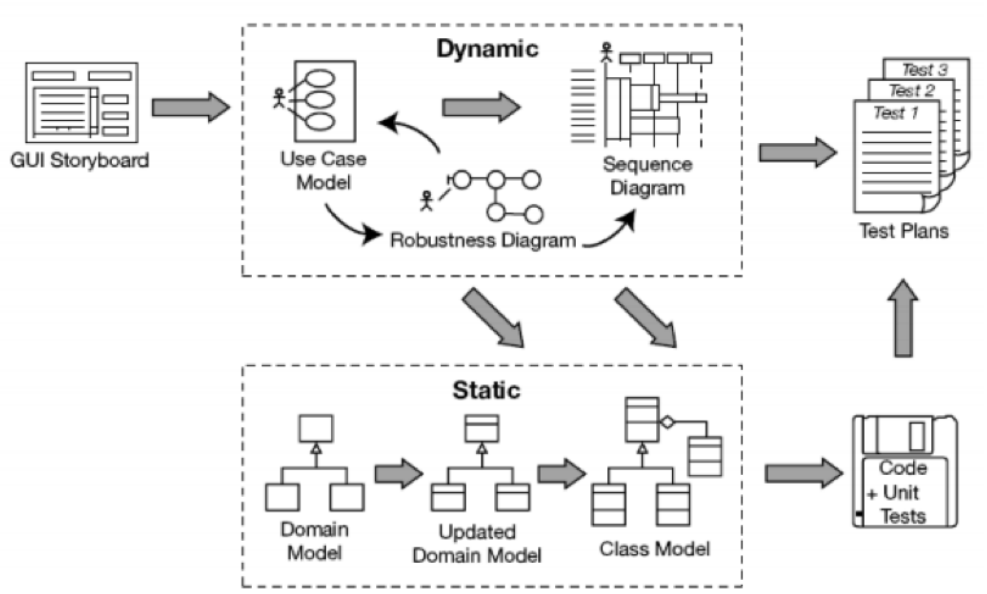

Figura 1 – Uma visão geral do ICONIX e seus componentes. **Fonte:** Rosenberg e Scott (1999)

Ao utilizar esta metodologia, o desenvolvimento do projeto passa a ser norteado por casos de uso(*use cases*) e suas principais fases são: análise de requisitos, análise e projeto preliminar, projeto e implementação. Abaixo será apresentado uma breve descrição de cada uma das fases do ICONIX, seguindo ideias de Azevedo (2010).

2.1.1 Análise de requisitos

Identificar os objetos do problema real e como eles serão abstraídos para um objeto de software através dos modelos de domínio; apresentar um protótipo das possíveis interfaces gráficas, e por fim descrever os casos de uso.

Se possível, elaborar também os diagramas de navegação para que o cliente possa entender melhor o funcionamento do sistema como um todo.

2.1.2 Análise e projeto preliminar

Detalhar todos os casos de uso identificados na fase de requisitos, por meio de diagramas de robustez, baseando-se no texto dos casos de uso (fluxo de eventos). O diagrama de robustez não faz parte dos diagramas padrões da UML. Porém, este é utilizado para descobrir as classes de análise e detalhar superficialmente o funcionamento dos casos de uso.

Em paralelo, deve-se atualizar o modelo de domínio adicionando os atributos às entidades identificados. A partir deste momento será possível gerar a base de dados do sistema.

2.1.3 Projeto detalhado

Elaborar o diagrama de sequência fundamentando-se nos diagramas de casos de uso identificados anteriormente, a fim de detalhar a implementação do caso de uso.

O diagrama de sequência deve conter as classes que serão implementadas e as mensagens enviadas entre os objetos, corresponderão aos métodos que realmente serão implementados futuramente.

2.1.4 Implementação

Desenvolver o código fonte e os testes do sistema. O ICONIX não define os passos a serem seguidos nesta fase, ficando a cargo da equipe de desenvolvimento.

Ao término de cada fase um artefato é gerado, sendo respectivamente: revisão dos requisitos, revisão do projeto preliminar, revisão detalhada e entrega.

O ICONIX é considerado um processo prático de desenvolvimento de software, pois a partir das iterações que ocorrem na análise de requisitos e na construção dos modelos de domínios (parte dinâmica), os diagramas de classes (parte estática) são incrementados e a partir destes, o sistema poderá ser codificado.

Por proporcionar esta praticidade, o ICONIX será empregado para o desenvolvimento deste projeto, pois por meio dele é possível obter produtividade no desenvolvimento do *software* ao mesmo tempo em que alguns artefatos são gerados, unindo o aspecto de abrangência e agilidade.

2.2 Teoria dos Grafos

A teoria dos grafos foi criada pelo matemático suiço *Leonhard* Euler no século XVIII com o propósito de solucionar um antigo problema, conhecido como as 7 pontes de *Königsberg* (HARJU, 2014).

Königsberg, atualmente conhecida como *Kaliningrad*, era uma antiga cidade medieval cortada pelo rio *Pregel* dividindo-a em 4 partes interligadas por 7 pontes. Ela era localizada na antiga Prússia, hoje, território Russo. O problema mencionado anteriormente consistia basicamente em atravessar toda a cidade, visitando todas as partes e utilizar todas as pontes desde que não repetisse uma das quatro partes ou uma das 7 pontes. A Figura 2 ilustra o problema mencionado.

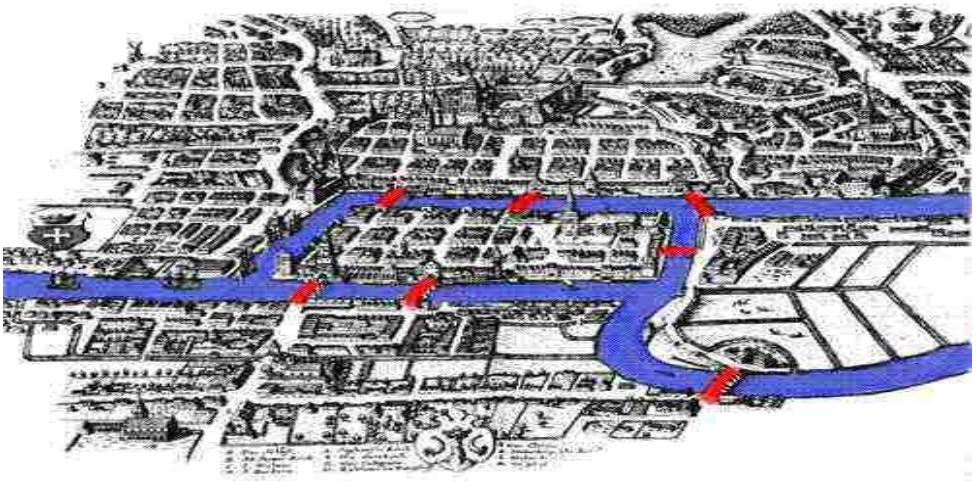

Figura 2 – O problema das 7 pontes de *Königsberg*. **Fonte:** Paoletti (2006)

De acordo com Bruggen (2014), para tentar solucionar o problema, Euler utilizou uma abordagem matemática ao contrário dos demais que tentaram utilizar a força bruta para solucionar tal problema, desenhando N números de diferentes possibilidades de rotas. Euler mudou o foco e passou a dar mais atenção ao número de pontes e não as partes da cidade. Através desta observação, foi possível perceber que realizar tal tarefa seria impossível, pois de acordo com sua teoria seria necessário possuir no mínimo mais uma ponte, uma vez que, o número de pontes era ímpar, não sendo possível realizar um caminho único e sem repetição. Desta forma, obteve-se a solução para este problema e criou-se o primeiro grafo no mundo.

Rocha (2013, p. 16) afirma que:

> um *grafo G = (V,E)* consiste em um conjunto finito *V* de vértices e um conjunto finito *E* de arestas onde cada elemento *E* possui um par de vértices que estão conectados entre si e pode ou não possuir um peso *P*.

Esta é a definição formal de um grafo. A partir desta definição, é possível identificar, no problema mencionado anteriormente, os vértices que neste caso são as pontes e as arestas que por sua vez são as partes da cidade.

Segundo Bondy e Murty (1976), muitas situações do mundo real podem ser descritas através de um conjunto de pontos conectados por linhas formando assim um grafo, como um

centro de comunicações e seus *links*, ou as pessoas e seus amigos, ou uma troca de emails entre pessoas, entre outras. Isto é possível pois, de acordo com Rocha (2013), existem muitos problemas atualmente que podem ser mapeados para uma estrutura genérica possibilitando assim utilizar a teoria de grafos para tentar solucioná-los, tais como: rotas geográficas, redes sociais, entre outros.

A Figura 3 demostra de maneira visual um grafo, conforme ideia de Bondy e Murty (1976), utilizando como exemplo o seguinte grafo G = {a, b, c, d, e, f, g, h} e suas respectivas arestas E_g = {(a, b), (a, h), (a, e), (b, f), (c, e), (c, d), (c, g), (d, e), (d, h), (d, g), (f, h)}, sendo que os vértices serão representados por círculos e as arestas que os interligam por linhas.

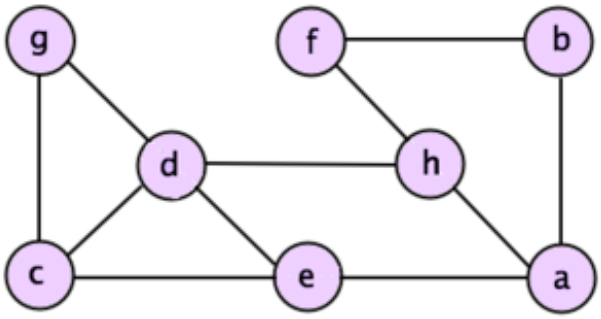

Figura 3 – Ilustração de uma representação gráfica de um simples grafo. **Fonte:** Rocha (2013)

Ruohonen (2013) afirma que os grafos podem ser gerados com a possibilidade de permitir *loops*[4] e arestas paralelas ou multiplas entre os vértices, obtendo um *multigraph*. A Figura 4 ilustra um simples *multigraph*.

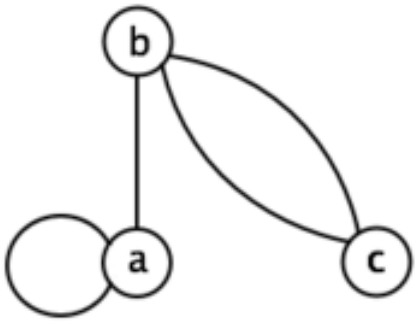

Figura 4 – Ilustração de uma representação gráfica de um *multigraph*. **Fonte:** Adaptado de Harju (2014)

Para Harju (2014), os grafos podem ser direcionados (*dígrafo*) ou não direcionados. Os direcionados são aqueles cujos vértices ligados a uma aresta são ordenados e permitem que uma

[4] *loops* - Uma aresta que interliga o mesmo vértice.

aresta que conecta os vértices *x* e *y* seja representada apenas de uma forma, sendo ela {x, y} ou {y, x}, ao contrário dos não direcionados que, para este mesmo caso, pode ser representado por ambas as formas Rocha (2013). A Figura 5 demonstra um grafo direcionado.

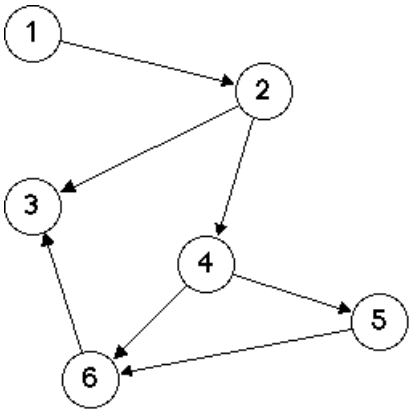

Figura 5 – Imagem de uma representação gráfica de um grafo direcionado. **Fonte:** Keller (2015)

Segundo Harju (2014), os tipos de grafos são:

- **grafo simples:** são aqueles grafos que não possuem *loops* ou arestas paralelas;
- **grafo completo:** são aqueles que, qualquer par de vértices são adjacentes;
- **subgrafos:** São pequenos grafos que em conjunto constituem um grafo maior.
- **grafos isomórficos:** dois grafos são isomórficos se, ambos possuírem a mesma estrutura de nós, e relacionamentos, exceto pelos identificadores de cada nó que podem ser diferentes, veja na Figura 6 um exemplo de dois grafos isomórficos (HARJU, 2014).

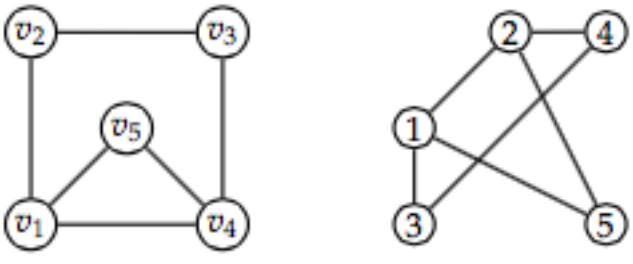

Figura 6 – Imagem de um grafo isomórfico. **Fonte:** Harju (2014)

- **caminho (travessia):** é uma sequência de vértices {v1, v2,...,vn} conectados por meio de arestas. Exemplo: {e1 = {v1, v2}, {v1, v3}..., {vn, vm}};
- **grafo conexo:** são aqueles que, para qualquer par vértices, há um caminho que os ligam;
- **grau do vértice:** é definido pela quantidade de arestas que se conectam ao vértice.

Para Rocha (2013) existem várias formas de se representar um grafo computacionalmente utilizando diferentes estruturas de dados. Entretanto, a mais utilizada e simples é a matriz adjacência.

Uma matriz adjacência consiste de uma matriz contendo o mesmo número de linhas e colunas (*n x n*). Veja na Figura 7 um exemplo de representação de um grafo utilizando esta estrutura.

$$A_G = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Figura 7 – Grafo representado por meio de uma matriz adjacência. **Fonte:** Rocha (2013)

Na Figura 7, as posições da matriz cujo o valor é igual a 1, definem que há uma aresta conectando os vértices, os tornando assim, adjacentes. Caso o valor seja 0, os vértices não estão conectados entre si no grafo e, portanto, não são vértices adjacentes.

Este conteúdo teórico foi escolhido para ser utilizado neste trabalho pois este, visa equacionar o problema relacionado à busca por mão de obra, através do modelo utilizado pelas redes sociais. Isto é possível pois, como mencionado anteriormente por meio desta teoria, é possível descrever várias situações do mundo real e como ela é muito bem aplicada à redes sociais, inclusive grandes empresas desta área já a utilizam. Portanto, esta teoria é utilizada para auxiliar no desenvolvimento deste projeto.

2.3 Tecnologias

Nesta seção serão abordadas as linguagens de programação e as tecnologias que serão utilizadas para o desenvolvimento deste trabalho.

2.3.1 Banco de dados

A expressão ”Banco de dados” teve origem a partir do termo inglês *Databanks*, que foi substituído, mais tarde, pela palavra *Databases* (Base de dados) por possuir um significado mais

apropriado (SETZER; SILVA, 2005).

De acordo com Date (2003), um banco de dados é uma coleção de dados persistentes, usada pelos sistemas de aplicação em uma determinada empresa. Sendo assim, um banco de dados é um local onde são armazenados os dados necessários para manter as atividades de determinadas organizações.

Um banco de dados possui, implicitamente, as seguintes propriedades: representa aspectos do mundo real; é uma coleção lógica de dados que possuem um sentido próprio e armazena dados para atender uma necessidade específica. O tamanho do banco de dados pode ser variável, desde que ele atenda às necessidades dos interessados em seu conteúdo (ELMASRI; NAVATHE, 2005).

A escolha do banco de dados que será utilizado em um projeto é uma decisão importante e que deve ser tomada na fase de planejamento, pois determina características da futura aplicação, como a integridade dos dados, o tratamento de acesso de usuários, a forma de realizar uma consulta e o desempenho. Portanto, essa decisão deve ser bem analisada, levando em consideração o tipo de software e no ambiente de produção que será utilizado.

Nas seções seguintes, são demonstrados os principais modelos de banco de dados, abordando suas características.

2.3.2 Banco de dados relacionais

O modelo de banco de dados relacional foi introduzido em 1970, por Edgar Frank Codd, em uma publicação com o título: "A relational model of data for large shared data banks", na revista *Association for Computing Machinery* (ACM). Essa publicação demonstrou como tabelas podem ser usadas para representar objetos do mundo real e como os dados podem ser armazenados para os objetos. Neste conceito, a integridade dos dados foi levada mais a sério do que em qualquer modelo de banco de dados antes visto. A partir desta publicação, surgiram muitos bancos de dados que passaram a utilizar este conceito e se tornaram muito utilizados no desenvolvimento de aplicações (MATTHEW; STONES, 2005).

Segundo Matthew e Stones (2005), conceito é baseado na teoria reacional da matemática e por isso há uma grande flexibilidade para o acesso e a manipulação de dados que são gravados no banco de dados. Utiliza-se técnicas simples, como normalização na modelagem do banco de dados, criando várias tabelas relacionadas, que servem como base para consultas usando uma

linguagem de consulta quase padronizada, a *Structured Query Language* – SQL[5].

A utilização de banco de dados relacionais geraram a necessidade de dividir os dados agregados utilizados na aplicação em várias relações conforme as regras da normalização. Para recuperar o mesmo dado agregado são necessárias consultas utilizando *joins*, uma operação que dependendo do tamanho das relações e da quantidade de dados pode não ser tão eficiente. Nos casos em que se precisa obter uma resposta rápida de um sistema isso pode ser uma desvantagem (SADALAGE; FOWLER, 2013).

Este foi um dos fatores determinantes que motivaram a criação de novas tecnologias, a fim de sanar o problema mencionado acima. A partir desta motivação foram desenvolvidos novos modelos de banco de dados, que serão apresentados a seguir.

2.3.3 Banco de dados NoSQL

A expressão NoSQL é um termo não definido claramente. Ela foi ouvida pela primeira vez em 1998 como um nome para o banco de dados relacional de Carlo Strozzi, que assim o nomeou por não fornecer uma SQL-API. O mesmo termo foi usado como nome do evento NoSQL Meetup em 2009, que teve como objetivo a discussão sobre sistemas de bancos de dados distribuídos.

Devido a explosão de conteúdos na *web* no início do século XXI, houve-se a necessidade de substituir os bancos de dados relacionais por bancos que oferececem maior capacidade de otimização e performance, a fim de suportar o grande volume de informações eminentes a esta mudança (BRUGGEN, 2014).

Rocha (2013, p. 27) afirma que NoSQL é "um acrônimo para Not only SQL, indicando que esses bancos não usam somente o recurso de Structured Query Language (SQL), mas outros recursos que auxiliam no armazenamento e na busca de dados em um banco não relacional".

Segundo Bruggen (2014), os banco de dados NoSQL podem ser categorizados de 4 maneiras diferentes, são elas: *Key-Value stores*[6], *Column-Family stores*[7], *Document stores*[8] e *Graph Databases*[9].

De acordo com Bruggen (2014), o banco de dados orientado a grafo (*graph database*) pertence a categoria NoSQL, contudo, ele possui particularidades que o torna muito diferente

[5] SQL: *Structured Query Language* - Linguagem para consultas e alterações em bancos de dados.
[6] *Key-Value stores* - armazenamento por um par de chave e valor.
[7] *Column-Family stores* - armazenamento por colunas e linhas.
[8] *Document stores* - armazenamento em arquivos.
[9] *Graph Databases* - banco de dados orientado a grafo.

dos demais tipos de bancos de dados NoSQL. A seguir, será descrito com maiores detalhes o banco de dados orientado a grafos Neo4j.

2.3.4 Neo4j

O Neo4j foi criado no início do século XXI por desenvolvedores que queriam resolver um problema em uma empresa de mídias. Porém, eles não obtiveram êxito ao tentar resolver tal problema utilizando as tecnologias tradicionais, portanto, decidiram arriscar e criar algo novo. A princípio, o Neo4j não era um sistema de gerenciamento de banco de dados orientado a grafos como é conhecido nos dias atuais. Ele era mais parecido com uma *graph library* (biblioteca de grafo) na qual as pessoas poderiam usar em seus projetos (BRUGGEN, 2014).

De acordo com Bruggen (2014), inicialmente ele foi desenvolvido para ser utilizado em conjunto com alguns bancos de dados relacionais como MySQL e outros, com a intenção de criar uma camada de abstração dos dados em grafos. Mas com o passar dos anos, os desenvolvedores decidiram tirar o Neo4j da estrutura dos bancos relacionais e criar sua própria estrutura de armazenamento em grafos.

O Neo4j, como vários outros, também é um projeto de sistema de gerenciamento de banco de dados NoSQL de código fonte aberto.

Segundo Robinson, Webber e Eifrem (2013), os bancos de dados orientados a grafos possuem como diferencial a sua performance, agilidade e flexibilidade. Entretanto, a performance é o que mais se destaca entre eles, pois, a maneira como eles armazenam e realizam buscas no banco de dados são diferentes dos bancos de dados convencionais. Primeiramente, este tipo de banco de dados não utiliza tabela; ele armazena os dados em vértices e arestas. Isto permite realizar buscas extremamente velozes através de *traversals* (travessias), uma vez que estas implementam algoritmos para otimizar tais funcionalidades, evitando assim o uso de *joins*[10] complexos, tornando-o tão veloz.

Neo4j (2013, p. 2) afirma que:

> *A single server instance can handle a graph of billions of nodes and relationships. When data throughput is insufficient, the graph database can be distributed among multiple servers in a high availability configuration.*[11]

[10] *joins* - função utilizada para realizar a junção entre tabelas, facilitando a busca em bancos de dados relacionais.

[11] Um único servidor pode manipular um grafo de bilhões de nós e relacionamentos. Quando a taxa de transferência de dados é insuficiente, o banco de dados orientado a grafo pode ser distribuído entre vários servidores em uma configuração de alta disponibilidade.

Com estas informações, é possível mensurar o quanto o Neo4j pode ser rápido e robusto, sendo possível, até mesmo distribuí-lo a fim de obter uma melhor configuração, organização e facilidade de manutenção.

Segundo Neo4j (2013), o banco de dados Neo4j é composto por nós (vértices), relacionamentos (arestas) e propriedades. Os relacionamentos são responsáveis por organizar os nós e ambos podem possuir seus atributos. É possível realizar as buscas e/ou alterações no Neo4j de duas formas diferentes. Sendo a primeira através da API *Cypher Query Language*, que é uma *query language* para banco de dados orientado a grafos muito próxima da linguagem humana, cuja sua descrição completa será apresentada a seguir. A segunda é o *framework*[12] *Traversal* que utiliza a API *Cypher* internamente para navegar pelo grafo.

Rocha (2013), afirma que o Neo4j permite criar mais de um relacionamento entre o mesmo par de vértices, desde que, estes sejam de tipos distintos. Isto possibilita navegar pelos vértices do grafo de forma mais rápida devido a estes diferentes tipos de arestas. O que torna possível implementar o algoritmo de busca desejado. A Figura 8 exemplifica um simples grafo utilizando um banco de dados Neo4j.

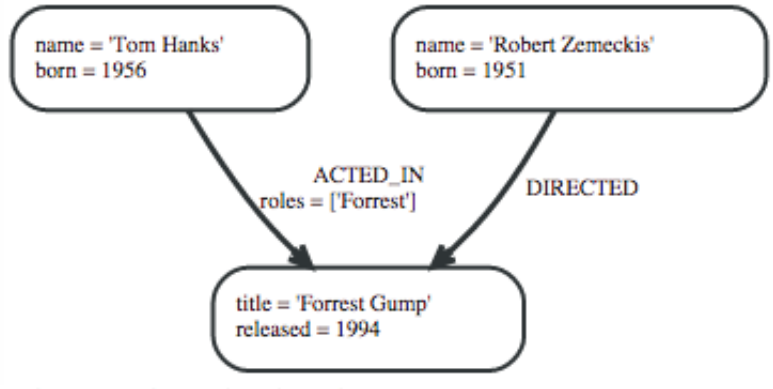

Figura 8 – Exemplo simples de um grafo no Neo4j. **Fonte:** Neo4j (2013)

Há duas formas de executar o Neo4j, segundo Robinson, Webber e Eifrem (2013). A primeira é conhecida como *Server* e a segunda *Embedded*. O modo *Server* é utilizado principalmente em *web-service* em conjunto com a API REST, este será aplicado neste trabalho. Já no modo *Embedded* o banco de dados é executado embarcado à aplicação Java.

Conforme Neo4j (2013), o Neo4j possui suporte as transações ACID (com atomicidade, consistência, isolamento e durabilidade).

O Neo4j é distribuído em duas versões sendo elas a *Entreprise* e a *Community*. A primeira possui um tempo de avaliação de 30 dias e após este tempo, é necessário comprar uma

[12] *Framework* - Abstração que une códigos comuns entre vários projetos de software, a fim de obter uma funcionalidade genérica.

licença para continuar a utilizá-lo. Como diferencial esta versão possui ferramentas para o gerenciamento do banco de dados, incluindo melhorias relacionadas à escalabilidade. A segunda versão é disponibilizada gratuitamente sem data limite de expiração, contudo, ela possui menos recursos como citado, mas é muito utilizada para fins didáticos e para pequenos projetos, aplicando-se perfeitamente a este projeto (NEO4J, 2013).

Por ser um banco de dados orientado a grafo bastante robusto, seguro e possuir uma documentação de fácil entendimento, além, é claro, de possuir um baixo custo de implantação devido a sua licença *open source*, este banco de dados foi escolhido para ser utilizado neste trabalho.

2.3.5 *Cypher Query Language*

O *Cypher Query Language* é uma linguagem para consultas em banco de dados orientado a grafo específica para o banco Neo4j. Ela foi criada devido à necessidade de manipular os dados e realizar buscas em grafos de uma forma mais simples, uma vez que, não é necessário escrever *traversals* (*travessias*) para navegar pelo grafo (NEO4J, 2013).

Robinson, Webber e Eifrem (2013), afirmam que, o *Cypher* foi desenvolvido para ser uma *query language* que utiliza uma linguagem formal, permitindo a um ser humano entendê-la. Desta forma, qualquer pessoa envolvida no projeto é capaz de compreender as consultas realizadas no banco de dados.

Segundo Neo4j (2013), o *Cypher* foi inspirado em uma série de abordagens e construído sob algumas práticas já estabelecidas, inclusive a SQL. Por este motivo, é possível notar que ele utiliza algumas palavras reservadas que são comuns na SQL como *WHERE* e *ORDER BY*.

De acordo com Neo4j (2013), o *Cypher* é composto por algumas cláusulas, dentre elas, se destacam:

- *START*: Define um ponto inicial para a busca, este ponto pode ser um relacionamento ou um nó.
- *MATCH*: Define o padrão de correspondência entre os nós. Para identificar um nó é necessário incluí-lo entre um par de parenteses, e os relacionamentos são identificados um hífen e um sinal de maior ou menor.
- *CREATE*: Utilizado para criar nós e relacionamentos no grafo.

- *WHERE*: Define um critério de busca.
- *RETURN*: Define quais nós, relacionamentos e propriedades de ambos devem ser retornados da *query* realizada.
- *SET*: Utilizado para editar as propriedades de um nó ou de um relacionamento.
- *UNION*: Possibilita juntar o resultado de duas ou mais consultas.
- *FOREACH*: Realiza uma ação de atualização para cada elemento na lista.

Outras *Query Languages* existem, inclusive com suporte ao Neo4j, porém devido as vantagens apresentadas acima, somada ao fato que ele possui uma curva de aprendizagem menor e é excelente para lhe oferecer uma base a respeito de grafos, este *framework* será utilizado para realizar as tarefas de manipulação dos dados no banco de dados.

2.3.6 Java

Segundo Schildt (2007), a primeira versão da linguagem Java foi criada por James Gosling, Patrick Naughton, Chris Warth, Ed Frank e Mike Sheridan na *Sun Microsystems* em 1991 e denominada "Oak"cujo seu principal foco era a interatividade com a TV. Mais tarde, em 1995, a *Sun Mycrosystems* renomeia esta linguagem e anuncia publicamente a tecnologia Java, focando nas aplicações *web*, que em pouco tempo e devido à grande ascensão da internet, cresceu e se mantém em constante evolução até os dias atuais.

De acordo com a Oracle (2010), a tecnologia Java não é apenas uma linguagem, mas também uma plataforma, que teve como modelo uma outra linguagem, o C++ que, por sua vez foi derivada da linguagem C. O C++ e o Java possuem em comum o conceito de orientação a objetos. O que permite a esta linguagem utilizar recursos como: generalização (herança), implementação, polimorfismo, entre outras. Tais funcionalidades permitem ao desenvolvedor escrever códigos reutilizáveis, a fim de facilitar o desenvolvimento do projeto.

Para Schildt (2007), o paradigma de orientação a objetos foi criado devido às limitações que o conceito estrutural apresentava quando era utilizado em projetos de grande porte, dificultando o desenvolvimento e manutenção dos mesmos. Este paradigma possibilita ao desenvolvedor aproximar o mundo real ao desenvolvimento de *software*, deixando os objetos do mundo real semelhantes a seus respectivos objetos da computação, possibilitando ao desenvolvedor modelar seus objetos de acordo com suas necessidades (FARIA, 2008).

Segundo Schildt (2007), o recurso denominado generalização (herança) permite ao desenvolvedor criar uma classificação hierárquica de classes. Além disso, é possível escrever uma classe genérica contendo comportamentos comum, e as demais classes, cujo tais comportamentos também serão aplicados a ela, somente precisa generalizar esta classe.

O polimorfismo se refere ao princípio da biologia em que um organismo pode ter diferentes formas ou estados. Este mesmo princípio, também pode ser aplicado à programação orientada a objeto. Desta forma, é possível definir comportamentos que serão compartilhadas entre as classes e suas respectivas subclasses, além de comportamentos próprios cujo apenas as sub classes possuem. Com isto, o comportamento pode ser diferente de acordo com a forma e/ou o estado do objeto (ORACLE, 2015a).

Retomando a ideia da Oracle (2010), uma das vantagens da tecnologia Java sob as demais é o fato de ela ser multiplataforma, possibilitando ao desenvolvedor escrever o código apenas uma vez e este, ser executado em qualquer plataforma inclusive em hardwares com menor desempenho. Isto é possível, pois, para executar um programa em Java é necessário possuir uma *Java Virtual Machine* - JVM[13] - instalada no computador. A JVM compreende e executa apenas *bytecodes*[14] e estes por sua vez são obtidos através do processo de compilação do código escrito em Java.

Todo programa que utiliza Java necessita passar por algumas etapas essenciais. Conforme ilustrado na Figura 9, o código é escrito em arquivo de texto com extensão `.java`, após isto ele será compilado e convertido para um arquivo com extensão `.class`, cujo o texto é transformado em *bytecodes*. Este arquivo com extensão `.class` é interpretado pela JVM que é responsável por executar todo o código do programa.

[13] JVM: *Java Virtual Machine* - Máquina virtual utilizada pela linguagem Java para execução e compilação de *softwares* desenvolvidos em Java.

[14] *Bytecode* é o código interpretado pela JVM. Ele é obtido por meio do processo de compilação de um programa java como mencionado anteriormente.

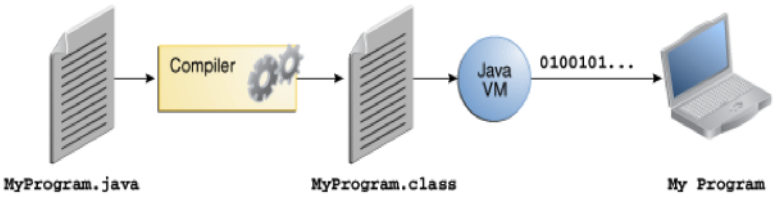

Figura 9 – Uma visão geral do processo de desenvolvimento de software. **Fonte:** Oracle (2010)

Por todas as vantagens descritas anteriormente, foi empregado o uso desta tecnologia neste trabalho.

2.3.7 Tomcat 7

O Tomcat é uma aplicação *container*, capaz de hospedar aplicações *web* baseadas em Java. A princípio ele foi criado para executar *servlets*[15] e *JavaServer Pages* - JSP[16]. Inicialmente ele era parte de um sub projeto chamado *Apache-Jakarta*, porém, devido ao seu sucesso ele passou a ser um projeto independente, e hoje, é responsabilidade de um grupo de voluntários da comunidade *open source* do Java (VUKOTIC; GOODWILL, 2011).

Segundo a Apache (2015), o Tomcat é um software que possui seu código fonte aberto e disponibilizado sob a *Apache License Version 2*. Isto o fez se tornar uma das aplicações *containers* mais utilizadas por desenvolvedores.

Containers são aplicações que são executadas em servidores e possuem a capacidade de hospedar aplicações desenvolvidas em Java *web*. O servidor ao receber uma requisição do cliente, entrega esta ao *container* no qual o é distribuído, o *container* por sua vez entrega ao *servlet* as requisições e respostas HTTP[17] e iniciam os métodos necessários do *servlet* de acordo com o tipo de requisição realizada pelo cliente (BASHMAN; SIERRA; BATES, 2010).

Brittain e Darwin (2007) afirmam que o Tomcat foi desenvolvido utilizando a linguagem de programação Java, sendo necessário possuir uma versão do Java SE[18] *Runtime Environment* - JRE - instalado e atualizado para executá-lo.

De acordo com Laurie e Laurie (2003), o Tomcat é responsável por realizar a comunicação entre a aplicação e o servidor Apache[19] por meio do uso de *sockets*.

[15] *Servlet* - Programa Java executado no servidor, semelhante a um *applet*.
[16] JSP: *JavaServer Pages* - Tecnologia utilizada para desenvolver páginas interativas utilizando Java *web*.
[17] HTTP: *Hypertext Transfer Protocol* - Protocolo de transferência de dados mais utilizado na rede mundial de computadores.
[18] Java SE - Corrigir.
[19] Apache - Corrigir.

Assim como outros *containers*, Bashman, Sierra e Bates (2010) afirmam que o Tomcat oferece gerenciamento de conexões *sockets*, suporta *multithreads*, ou seja, ele cria uma nova *thread* para cada requisição realizada pelo cliente e gerencia o acesso aos recursos do servidor, além de outras tarefas.

O Tomcat, em especial, foi escolhido para ser utilizado neste trabalho, pois o objetivo é desenvolver uma aplicação *web* e para hospedá-la em um servidor, uma aplicação *container* se faz necessária. Por este motivo, e somado a sua facilidade de configuração, além das vantagens acima descritas tal decisão foi tomada.

2.3.8 *Web Service* REST

A definição computacional de serviço é um programa que disponibiliza sua funcionalidade através de uma interface denominada contrato de serviço (ERL et al., 2012).

Web Service de acordo com Marzullo (2013), é uma materialização da ideia de um serviço que é disponibilizado na internet, e que, devido a isto, pode ser acessado em qualquer lugar do planeta e por diferentes tipos de dispositivos. Para ter acesso aos serviços que este *Web Service* disponibiliza o solicitante envia requisições de um tipo anteriormente definido e recebem respostas síncronas ou assíncronas.

Marzullo (2013), afirma que, a implementação de um *Web Service* é relativamente simples, uma vez que, há inúmeras ferramentas que facilitam a implementação do mesmo. Outro fator que permite a um *Web Service* ser mais dinâmico, é possuir uma estrutura interna fracamente acoplada, permitindo assim, mudanças em suas estruturas sem afetar a utilização pelo cliente.

Erl et al. (2012), afirma que, o REST[20] é uma das várias implementações utilizadas para criar serviços. Outras implementações também muito conhecidas são: a WSDL[21] e a SOAP[22].

O primeiro*Web Service* REST foi criado por Roy Fielding no ano 2000 na universidade da California. Ele foi criado para suceder o tecnologia SOAP, a fim dex facilitar o uso de tal tecnologia. (RODRIGUEZ, 2008).

Segundo Oracle (2015b), na arquitetura do REST os dados e as funcionalidades são considerados recursos e ambos são acessados por meio de URIs[23]. Ele funciona baseado no

[20] REST: *Representational State Transfer* - Tecnologia utilizada por *web services*

[21] WSDL: *Web Service Description Language* - Padrão de mercado utilizado para descrever *Web Services*.

[22] SOAP: *Simple Object Access Protocol* - Tecnologia utilizada por *web services* anteriores aos *Web services* REST.

protocolo HTTP, portanto, já possui os mecanismos de segurança, cabeçalhos e respostas embutidos.

Para Rodriguez (2008), o *Web service* REST segue quatro princípios básicos. São eles:

- Utilização dos métodos HTTP explicitamente;
- Orientação à conexão;
- Expõe a estrutura de diretório por meio das URIs;
- Trabalha com -*Extensible Markup Language* - XML[24], *Javascript Object Notation* - JSON[25] - ou ambos.

Esta tecnologia foi selecionada para ser utilizada neste trabalho, pois a forma de acesso ao banco de dados Neo4j utiliza uma API REST fornecida pelo próprio Neo4j.

2.3.9 HTML 5

Segundo W3C (2015a), *Hypertext Markup Language* - HTML[26] - é a linguagem usada para descrever o conteúdo das páginas *web*. Ela utiliza marcadores denominados *tags* para identificar aos navegadores de internet como eles devem interpretar tal documento.

Silva (2011a), afirma que, o HTML foi criado única e exclusivamente para ser uma linguagem de marcação e estruturação de documentos (páginas) *web*. Portanto, não cabe a ele definir os aspectos dos componentes como cores, espaços, fontes, etc.

A W3C (2015a), afirma que, a primeira versão do HTML foi criada no ano de 1991 pelo inventor da *web*, Tim Berners-Lee. A partir desta versão, o HTML foi e continua em constante atualização e hoje se encontra na sua oitava versão. As oito versões são: HTML, HTML +, HTML 2.0, 3.0, 3.2, 4.0, 4.01 e a versão atual é a 5.

Ao longo dos anos e da evolução propriamente dita do HTML, novas *tags* foram criadas, padrões adotados, e claro, novas versões foram criadas, até que, em maio de 2007 o *World Wide*

[23] URI: *Uniform Resource Identifier - Link* completo para acessar um determinado recurso.
[24] XML: *Extensible Markup Language* - Tecnologia utilizada para transferência de dados, configuração, entre outras atividades.
[25] JSON: *Javascript Object Notation* - Notação de objetos via Javascript, porém seu uso não é restrito apenas ao Javascript.
[26] HTML: *Hypertext Markup Language* - Linguagem usada para descrever o conteúdo das páginas *web*.

Web Consortium - W3C[27] - confirma a decisão de voltar a trabalhar na atual versão do HTML, também conhecida por HTML 5 (W3C, 2015a).

Silva (2011b), afirma que, em novembro daquele mesmo ano, o W3C publicou uma nota contendo uma série de diretrizes que descreve os princípios a serem seguidos ao desenvolver utilizando o HTML 5 em algumas áreas. Tais princípios, permitiram a esta versão, maior segurança, maior compatibilidade entre navegadores e interoperabilidade entre diversos dispositivos.

Por estes motivos, somado ao fato de que, ao utilizar esta tecnologia é possível obter uma maior flexibilidade no desenvolvimento das páginas *web*, Esta tecnologia em conjunto com outras como: CSS 3, Javascript e Angular JS, cujo suas descrições serão apresentadas nas próximas seções deste trabalho foi selecionada para ser utilizada neste projeto.

2.3.10 CSS 3

Silva (2011a), afirma que a principal função do *Cascading Style Sheet* - CSS[28] - é definir como os componentes anteriormente estruturados nos documentos *web*, por meio do HTML devem ser apresentados ao usuário.

Para W3C (2015b), o CSS é "*a simple mechanism for adding style (e.g., fonts, colors, spacing) to Web documents*[29]".

Tim Berners-Lee, inicialmente escrevia as estilizações de seus documentos *web*, mesmo que de forma simples e limitada, nos próprios documentos HTML. Isto se deve ao fato de que, ele acreditava que tal função deveria ser realizada pelos navegadores. Entretanto, em 1994 a primeira proposta de criação do CSS surgiu e em 1996, a primeira versão do CSS denominada CSS 1 foi lançada como recomendação do W3C (SILVA, 2011a).

De acordo com Silva (2011a), Atualmente o CSS possui quatro versões: A CSS 1, a 2, a 2.1 e atualmente a 3, que foi utilizada para o desenvolvimento deste trabalho.

Há três formas de incorporar o CSS em seu documento web segundo (SILVA, 2011a), são elas:

- ***Inline*:** É possível aplicar o estilo diretamente ao componente desejado, por meio do uso da propriedade `style` do componente HTML. Como é apresentado na Figura 10;

[27] W3C: *World Wide Web Consortium* - Consórcio internacional formado por empresas, instituições, pesquisadores, desenvolvedores e público em geral, com a finalidade de elevar a web ao seu potencial máximo.

[28] CSS: *Cascading Style Sheet* - Documentos que definem estilos aos componentes da página *web*.

[29] O CSS é um simples mecanismo para adicionar estilo, como: fontes, cores, espaços para os documentos *Web*.

```html
<!DOCTYPE html>
<html>
<head>
    <title>Título da página</title>
</head>
<body>
    <p style="color: red;">
        Exemplo de estilo CSS aplicado diretamente no componente
    </p>
</body>
</html>
```

Figura 10 – Exemplo de inclusão do estilo CSS inline. **Fonte:** Elaborado pelos autores.

- **Incorporado:** Outra forma, é escrever todo CSS referente ao documento *web* dentro da *tag* `style` do documento HTML. Para tanto, esta *tag* deve ser inserida entre o início e o fim da *tag* `head` do documento. Como é apresentado na Figura 11;

```html
<!DOCTYPE html>
<html>
<head>
    <title>Título da página</title>
    <style>
        p {
            color: red;
        }
    </style>
</head>
<body>
    <p>
        Exemplo de estilo CSS aplicado incorporado à página HTML por meio da tag STYLE
    </p>
</body>
</html>
```

Figura 11 – Exemplo de inclusão do estilo CSS incorporado à página HTML. **Fonte:** Elaborado pelos autores.

- **Externo:** A última forma, é criar um arquivo externo com extensão `.css` e definir todas as regras de estilização do documento *web* neste arquivo. Desta forma, para vincular tal arquivo à um documento HTML específico será necessário utilizar a *tag* `link` entre o início e o fim da *tag* `head` do documento. Como é apresentado na Figura 12;

```
<!DOCTYPE html>
<html>
<head>
    <title>Título da página</title>
    <link rel="stylesheet" href="estilo.css">
</head>
<body>
    <p>
        Exemplo de estilo CSS carregado a partir de um arquivo externo
    </p>
</body>
</html>
```

Figura 12 – Exemplo de inclusão do estilo CSS a partir de um arquivo externo. **Fonte:** Elaborado pelos autores.

O CSS 3 será utilizado neste projeto, pois, ele permite definir estilos aos componentes das páginas *web* e possui recursos que não são existem em suas versões anteriores, o que nos permite desenvolver páginas mais atrativas com menos recursos.

2.3.11 Javascript

Frank e Seibt (2002), afirmam que, o Javascript foi criado e lançado pela Netscape em 1995, em conjunto com o navegador de internet Netscape Navigator 2.0. A partir deste lançamento, as páginas *web* passaram a ganhar vida com a possibilidade de implementar um mínimo de dinamicidade. Isto se deve ao modo como a linguagem acessa e manipula os componentes do navegador. Contudo, ela pode ser utilizada em difentes dispositivos como *smartphones, smart tv*, entre outros, não limitando-se apenas à navegadores de internet.

O Javascript é uma linguagem de programação para *web*. A maioria dos sites usam esta linguagem, inclusive todos os navegadores mais modernos, vídeo games, *tablets*, *smart phones*, *smart tvs* possuem interpretadores de *Javascript*, o que a fez tornar, a linguagem de programação mais ambígua da história (FLANAGAN, 2011).

Segundo Frank e Seibt (2002), as semelhanças entre o Javascript e o Java se limitam apenas ao nome. A primeira linguagem não deriva da segunda, apesar de ambas compartilharem alguns conceitos e detalhes. O Javascript por ser uma linguagem interpretada, é mais flexível que o Java, que, por sua vez, é uma linguagem compilada.

De acordo com Flanagan (2011), o Javascript possui 6 versões, sendo elas: 1.0, 1.1, 1.2, 1.3, 1.4 e a atual versão 1.5.

Para Frank e Seibt (2002), há duas maneiras de incluir e executar o código escrito em Javascript nos documentos HTML. A primeira é incluir o código Javascript entre as *tags* `script` como mostra a Fígura 13.

```
<!DOCTYPE html>
<html>
<head>
    <title>Título da página</title>
</head>
<body>
    <script type="text/javascript">
        document.writeline('Exemplo de um código Javascript incorporado ' +
                           'ao documento HTML por meio da tag SCRIPT!');
    </script>
</body>
</html>
```

Figura 13 – Exemplo de inclusão do código em Javascript incorporado ao HTML. **Fonte:** Elaborado pelos autores.

A segunda forma é incluir um arquivo externo com extensão `.js` através da mesma *tag* `script`, veja um exemplo de inclusão de um arquivo contendo códigos em Javascript no documento HTML na Figura 14.

```
<!DOCTYPE html>
<html>
<head>
    <title>Título da página</title>
</head>
<body>
    <script type="text/javascript" src="Exemplo.js"></script>
</body>
</html>
```

Figura 14 – Exemplo de inclusão do código Javascript de um arquivo externo. **Fonte:** Elaborado pelos autores.

Para Flanagan (2011), as três tecnologias (HTML, CSS e Javascript) devem ser usadas em conjunto, uma vez que, cada uma delas possui seu papel específico, sendo eles: o HTML usado para especificar o conteúdo da página *web*, o CSS para especificar como os componentes serão apresentados e o Javascript para especificar o comportamento da página.

Pelos motivos acima mencionados, somado ao fato que o Javascript permite ao desenvolvedor criar páginas *web* mais dinâmicas e flexíveis, atendendo perfeitamente os requisitos deste projeto. Esta tecnologia será utilizada para auxiliar a criação das páginas *web* deste projeto.

2.3.12 Angular JS

Segundo Green e Seshadri (2013) o framework Angular JS foi criado para facilitar o desenvolvimento de aplicativos *web*, pois através dele, é possível criar um aplicativo *web* com poucas linhas.

Para o criador do Angular JS, Miško Hevery, o que o motivou a criar o Angular JS, foi a necessidade de ter que reescrever determinados trechos de códigos em todos os outros projetos.

O que para ele, estava se tornando chato e custoso. Portanto, Miško decidiu criar algo para facilitar o desenvolvimento de aplicativos *web* de uma forma que ninguém havia pensado antes.

O Angular JS é um framework *Model-View-Controller* - MVC[30] - escrito em Javascript. Ele é executado pelos navegadores de internet e ajuda os desenvolvedores a escreverem modernos aplicativos *web* (KOZLOWSKI; DARWIN, 2013).

Devido a estas facilidades que o Angular JS nos traz, é que o mesmo foi escolhido para ser utilizado, a fim de, auxiliar no desenvolvimento, não apenas das páginas web e seus respectivos conteúdos, como também na lógica e comunicação com o *Web Service* REST.

[30] MVC: *Model-View-Controller - Design pattern*.

3 QUADRO METODOLÓGICO

Por meio do quadro metodológico, serão apresentados os passos que se fizeram necessários para a realização deste projeto. Nele estão descritos desde a escolha da pesquisa, na qual ele se enquadra, até os procedimentos utilizados para o seu desenvolvimento. Gil (1999, p. 32), cita que a metodologia é um conjunto de procedimentos intelectuais e técnicos que trabalham para a realização do objetivo proposto.

Posterior ao estudo do levantamento teórico e técnico, foram definidos os procedimentos para a construção do projeto, iniciando pela escolha do tipo de pesquisa, demonstrada na sessão a seguir.

3.1 Tipo de pesquisa

Para Pádua (2007, p. 31), pesquisa é:

> Toda atividade voltada para a solução de problemas; como atividade de busca, indagação, investigação, inquirição da realidade, e a atividade que visa nos permitir, no âmbito da ciência, elaborar um conhecimento, ou um conjunto de conhecimentos, que nos auxilie na compreensão desta realidade e nos oriente em nossas ações.

De forma objetiva, a pesquisa é o meio utilizado para buscar respostas aos mais diversos tipos de indagações, tendo por base procedimentos racionais e sistemáticos. A pesquisa é realizada quando se tem um problema e não se têm informações suficientes para solucioná-lo. Por meio desta sessão, tem-se como objetivo explicar o tipo de pesquisa que norteou o desenvolvimento deste projeto, justificando também como ele se enquadra no tipo escolhido.

Pesquisar é um trabalho que envolve um planejamento análogo ao de um cozinheiro. O cozinheiro, ao preparar um prato, precisa saber o que ele quer fazer, possuir os ingredientes, assegurar-se de que todos os utensílios necessários estão à mão e cumprir as etapas requeridas no processo. O sucesso do prato dependerá do envolvimento do cozinheiro com o ato de cozinhar e também das suas habilidades técnicas na cozinha. O mesmo acontece com uma pesquisa, para que ela seja satisfatória, o pesquisador precisa estar envolvido e também desenvolver habilidades técnicas, que o levem a escolher o melhor caminho em busca da obtenção dos resultados.

Segundo Fonseca (2002, p. 20), ter um método de pesquisa envolve o estudo dos fatores que compõem o contexto da pesquisa, tais como, a escolha do caminho e o planejamento do percurso. Essa escolha inicia-se com a definição do tipo de pesquisa utilizada. Para este trabalho, é utilizada a pesquisa aplicada, que é aquela cujo o pesquisador tem como objetivo aplicar os conhecimentos obtidos durante o período da pesquisa, em um projeto real, a fim de conhecer os seus resultados. Gil (2007, p. 25), afirma que este tipo de pesquisa dirige-se à solução de problemas específicos, de interesses locais.

Neste projeto foram estudados os conceitos de banco de dados orientado à grafos e a sua aplicabilidade, desenvolvendo, por meio dos conhecimentos obtidos pela pesquisa, uma solução prática, disponibilizada por meio de um sistema *web*, que auxilia na busca por mão de obra temporária, que não caracterize vínculo empregatício.

Seguindo o enquadramento desta pesquisa, ela deve ser aplicada à um contexto especifico, conforme será abordado a seguir.

3.2 Contexto de pesquisa

O trabalho informal é um elemento estrutural da economia no Brasil e nos países em desenvolvimento. Ele faz parte do cenário atual, crescente a cada dia, e contribui ativamente com a geração de renda. É considerado como um desdobramento do excesso de mão de obra, definido a partir de pessoas que criam sua própria forma de trabalho como estratégia de sobrevivência ou como forma alternativa de recolocação no mercado de trabalho. O fortalecimento deste tipo de trabalho ocorre a partir da construção de redes, formadas por parentes e amigos, criando laços de confiança que são fundamentais para o desempenho da atividade. No entanto, há uma grande dificuldade em se encontrar estes profissionais, uma vez que não há um lugar centralizado para divulgar o seu perfil profissional.

O desenvolvimento deste projeto se propôs atuar sob essa limitação. Uma pesquisa informal, realizada no sul de Minas Gerais, com pessoas de diferentes perfis sociais, constatou que uma aplicação, capaz de centralizar a busca por estes profissionais, seria muito bem aceita. A partir deste resultado, validou-se a ideia de construir um ambiente *web* onde o trabalhador informal tem o espaço para centralizar suas habilidades e manter um perfil visível aos possíveis contratantes. Qualquer prestador de serviço informal pode ter acesso a este ambiente, desde que possua um dispositivo eletrônico capaz de se conectar a internet.

O ambiente desenvolvido também visa facilitar ao contratante, a busca por estes profis-

sionais, uma vez que não é fácil localizá-los por meio dos mecanismos de busca tradicionais. Desta forma existe um benefício mutuo, onde contratados e contratantes se despõem da praticidade.

Enfim, o contexto a quem esse projeto se destina busca ser bem abrangente, com o intuito de contribuir de forma relevante, proporcionando uma boa experiência aos envolvidos.

3.3 Instrumentos

Os instrumentos de pesquisa são as ferramentas usadas para a coleta de dados. Como afirma Marconi e Lakatos (2009, p. 117), os instrumentos de pesquisa abrangem "desde os tópicos de entrevista, passando pelo questionamento e formulário, até os testes ou escala de medida de opiniões e atitudes".

Para a realização deste projeto serão utilizados questionários, reuniões e análise documental como instrumentos de pesquisa.

As próximas subseções descreverão como serão utilizados os instrumentos de pesquisa neste projeto.

3.4 Procedimentos

Para que esta pesquisa seja levada a cabo, torna-se necessário a implementação de algumas ações, as quais são descritas abaixo.

- escolha da dupla;
- Decisão do tema (dezembro 2014)
- Escolha do orientador
- Levantamento das tecnologias que seriam utilizadas
- Estudo e pequenos testes a fim de validar as tecnologias
- Definição da metodologia de desenvolvimento de software.
- Pesquisa de mercado para saber a aceitação da ideia
- Inicio do desenvolvimento do pré projeto

- Escrita da introdução, objetivos e justificativa
- levantamento de requisitos textuais do sistema (através de possíveis cenários e simulações);
- Escrita do quadro teórico para o pré projeto
- Quadro metodológico
- Criação do cronograma de atividades;
- Revisão do pré projeto
- Banca de qualificação
- Correção do projeto de acordo com as sugestões da banca
- atualização do cronograma
- Iniciou-se o processo de elaboração da engenharia de software.
- Modelagem da base de dados
- Inicio do desenvolvimento do software.
- Configuração do ambiente de desenvolvimento (Instalação das ferramentas)
- Tentativa de desenvolvimento utilizando a seguintes tecnologias: - Neo4j embedded, primefaces e JSF.(criar conta) Porem não estava fluindo como o esperado, uma vez que o Neoo4j utilizado da maneira embedded não nos permitia conectar ao sistema de gerenciamento da base de dados (web)e abrir uma nova instancia de conexão simultânea devido a limitações do próprio Neo4j, já que um processo java já ocupava tal conexão com o socket. Outro problema encontrado ao utilizar tais tecnologias foi que tanto a parte cliente (front end) quanto a parte do servidor (back end, modelo de negócios) se encontravam totalmente acoplados em uma aplicação java web, portanto, a cada alteração realizada nos controllers (principalmente, pois este era alterado constantemente, uma vez que ele era diretamente relacionado a página web,seguindo o padrão de desenvolvimento estabelecido ao utilizar as tecnologias descritas acima), sendo necessário recompilar, construir e publicar a aplicação no servidor web (tomcat). Isto estava impedindo que o desenvolvimento do software fluísse naturalmente, como era esperado. Por estes motivos decidiu-se mudar algumas das tecnologias utilizadas no front end e em como o banco de dados era

acessado até então. Passou a utilizar então: - Html5, CSS3, Java Script e o framework Angular JS para auxiliar no desenvolvimento do front end, ao invés de primefaces e JSF. - Para acesso ao banco de dados, lançou-se mão da forma embedded disponibilizada pelo banco de dados e passou-se a utilizar a API REST também disponibilizada pelo mesmo. Tais decisões nos permitiram desacoplar o sistema e manter o front end e o back end independentes.

- Estudo a respeito de web service REST;
- Instalação e configuração do Angular JS.
- Testes utilizando o web Service REST a fim de validar a conexão com o banco de dados via API REST
- Testes utilizando Angular JS para enviar requisições ao servidor e receber as respostas.
- Configurar o acesso da aplicação cliente ao web service localizados em domínios distintos, utilizando o CORS.
- Implementação do caso de uso 'criar conta' utilizando as novas tecnologias.
- Desenvolvido o sistema de login com armazenamento da sessão no cliente de forma errada; (Colocando o usuário e senha decodificados na própria sessão do navegador).
- Desenvolvimento do Logoff
- Refatoração do sistema de login e armazenamento e validação da sessão. A partir deste ponto, foi utilizado o sistema de login via token. Gerando assim, sempre um novo token a fim de manter a segurança dos dados.
- Criar a página inicial do sistema (Página Home)
- Implementação da busca por parceiros (buscando todos os contratantes registrados no banco, ou seja, sem critérios para a busca, a fim de criar a rede de parcerias primeiramente);
- Implementação da funcionalidade 'Adicionar Parceria'.
- Implementação da fila de requisição em espera para análise do contratante no menu principal.
- Implementação da funcionalidade 'Aceitar parceria'.

- Implementação do serviço de localização de possíveis parceiros com base em sua rede de relacionamento (parcerias) na página inicial do sistema.
- Implementação da busca por todos os parceiros do contratante autenticado no sistema para ser apresentado na página de rede de parceria
- Implementação da busca por novos parceiros para o contratante autenticado no sistema na página de rede de parceria.
- Implementação da funcionalidade 'Listar todos os serviços' cujo o provedor de serviço possui atrelado a ele.
- Implementação da funcionalidade 'Adicionar serviços' para os provedores de serviços.
- Implementação de remoção de serviços.
- Implementação da funcionalidade Listar Mão de obra. Inicialmente, como feito na lista de parceiros, também feito na lista de provedores de serviços. (Listei todos os provedores de serviços que executavam o serviço procurado pela pessoa naquele momento), a fim de criar a rede de avaliações de serviços.
- Implementar a funcionalidade de Avaliar o serviço.
- Implementar a funcionalidade de Localização de mão de obra, agora baseando-se nas avaliações anteriormente realizadas pelos meus parceiros.
- Abranger a busca por provedores de serviço, que ainda não foram avaliadas, para localizar mão de obras dentro da mesma cidade do contratante.
- Implementar a busca pelas últimas avaliações realizadas pelo contratante autenticado no sistema.
- Criar a página de perfil do provedor de serviço e apresentar um relatório contendo a média das avaliações em diferentes situações (na rede de parceria, na empresa e na cidade)
- Criar a página de perfil dos contratantes apresentando a quantidade de parceiros em comum e o nome de alguns deles
- Desenvolver o design do sistema
- Refatorar a página Incial (Página Home) de ambos os tipos de acesso (Provedor de serviços e Contratantes ou ambos)

- Criar uma página com dicas randômicas para os prestadores de serviços
- Criar o feed de notícias na página inicial do sistema (parecido com o do facebook)
- Criar os gráficos (à desenvolver)

Realizando todos os passos descritos acima, será possível obter como resultado final a realização deste projeto.

4 RESULTADOS

Aqui deve aparecer a descrição dos resultados obtidos.

5 CONCLUSÃO

A conclusão deste trabalho é ...

Assim conclui-se que ...

REFERÊNCIAS

APACHE. ***Apache Tomcat***. 2015. Disponível em: <http://tomcat.apache.org/index.html>. Acesso em 15 de janeiro de 2015.

AZEVEDO, M. E. C. V. de. *Quantas fases tem o ICONIX?* 2010. <http://oengenheirodesoftware.blogspot.com.br/2010/11/quantas-fases-tem-o-iconix.html>. Acesso em: 22/07/2015.

BARBOSA, K. ***Por que as pessoas usam as redes sociais?*** 2011. Disponível em: <http://super.abril.com.br/blogs/tendencias/por-que-as-pessoas-usam-as-redes-sociais/>. Acesso em 28 de dezembro de 2014.

BASHMAN, B.; SIERRA, K.; BATES, B. ***Use a Cabeça! Servlets e JSP***. [S.l.]: Alta Books, 2010.

BONDY, J. A.; MURTY, U. S. R. **Graph Theory with applications**. 1976.

BRITTAIN, J.; DARWIN, I. F. ***Tomcat The Definitive Guide***. 2. ed. [S.l.: s.n.], 2007.

BRUGGEN, R. V. ***Learning Neo4j***. [S.l.]: Packt, 2014.

COOPER, D. R.; SCHINDLER, P. S. ***Métodos de Pesquisa em Administração***. 7. ed. [S.l.]: Bookman, 2003.

DATE, C. J. ***Introdução a sistemas de bancos de dados***. 8. ed. [S.l.]: Elsevier, 2003.

ELMASRI, R.; NAVATHE, S. B. ***Sistemas de Banco de Dados***. 1. ed. [S.l.]: Pearson, 2005.

ERL, T. et al. ***SOA with REST: principles, patterns & constraints for building enterprise solutions with REST***. [S.l.]: Novatec, 2012.

FARIA, M. A. de. *ASPECTJ: Programação orientada a aspecto em Java*. [S.l.: s.n.], 2008.

FLANAGAN, D. ***Javascript: The Definitive Guide***. 6. ed. [S.l.]: O' Reilly, 2011.

FONSECA, J. J. S. ***Metodologia da pesquisa Científica***. [S.l.]: UEC, 2002.

FRANK, D. R.; SEIBT, L. Javascript. 2002.

GIL, A. C. ***Métodos e técnicas de pesquisa social***. 5. ed. [S.l.]: Atlas, 1999.

GIL, A. C. ***Como elaborar projetos de pesquisa***. 4. ed. [S.l.]: Atlas, 2007.

GREEN, B.; SESHADRI, S. ***Javascript: The Definitive Guide***. [S.l.]: O' Reilly, 2013.

HARJU, T. Graph theory. 2014.

JUNID, S. A. M. A. et al. Potential of graph theory algorithm approach for dna sequence alignment and comparison. 2012.

KELLER, R. M. *Acylic Graph*. 2015. <http://www.cs.hmc.edu/~keller/courses/cs60/s98/examples/acyclic/>. Acesso em: 20 de Abril de 2015.

KOZLOWSKI, P.; DARWIN, P. B. ***Mastering Web Application Development with AngularJS***. [S.l.]: Packt Publishing, 2013.

LAURIE, B.; LAURIE, P. ***Apache: The Definitive Guide***. [S.l.]: RepKover, 2003.

MARCONI, M. A.; LAKATOS, E. M. ***Metodologia do trabalho científico***. 7. ed. [S.l.]: Atlas, 2009.

MARZULLO, F. P. ***SOA na prática: inovando seu negócio por meio de soluções orientadas a serviços***. [S.l.]: ServiceTech Press, 2013.

MATTHEW, N.; STONES, R. Beginnig databases with postgresql - from novice to professional. 2005.

NEO4J. ***The Neo4j Manual***. [S.l.: s.n.], 2013.

ORACLE. ***About the Java Technology***. 2010. Disponível em: <http://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html>. Acesso em 28 dez. 2014.

ORACLE. *Polymorphism*. 2015. <https://docs.oracle.com/javase/tutorial/java/IandI/polymorphism.html>. Acesso em: 22/07/2015.

ORACLE. *What are RESTful Web Services?* 2015. <https://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html>. Acesso em 23/07/2015.

PAOLETTI, T. *Leonard Euler's Solution to the Konigsberg Bridge Problem*. 2006. <http://www.maa.org/publications/periodicals/convergence/leonard-eulers-solution-to-the-konigsberg-bridge-problem>. Acesso em 26 de fevereiro de 2015.

PáDUA, E. M. M. de. ***Metodologia da Pesquisa: Abordagem Teorico-Pratica***. 16. ed. [S.l.]: Papirus, 2007.

PENHA, A. L. da S.; CARVALHO, W. *Sistema de recomendação de filmes utilizando graph database*. [S.l.: s.n.], 2013.

ROBINSON, I.; WEBBER, J.; EIFREM, E. ***Graph Databases***. [S.l.]: O'Reilly, 2013.

ROCHA, R. R. Algoritmos de particionamento e banco de dados orientado a grafos. 2013.

RODRIGUEZ, A. *RESTful Web services: The basics*. 2008. <http://www.ibm.com/developerworks/library/ws-restful/>. Acesso em 23/07/2015.

ROSENBERG, D.; SCOTT, K. ***Use Case Driven Object Modeling With UML A Pratical Approach***. 1. ed. [S.l.: s.n.], 1999.

ROSENBERG, D.; STEPHENS, M. ***Use Case Driven Object Modeling With UML Theory and Practice***. 1. ed. [S.l.: s.n.], 2007.

ROSENBERG, D.; STEPHENS, M.; COLLINS-COPE, M. **Agile Development with ICONIX Process: People, Process, and Pragmatism**. 2005.

RUOHONEN, K. **Graph Theory**. 2013.

SADALAGE, P. J.; FOWLER, M. ***NoSQL Distilled - A Brief Guide to the Emerging World of Polyglot Persistence***. [S.l.: s.n.], 2013.

SCHILDT, H. ***Java: The Complete Reference***. 7. ed. [S.l.: s.n.], 2007.

SETZER, V. W.; SILVA, F. S. C. da. ***Banco de dados: Aprenda o que são, melhore seu conhecimento, construa os seus***. [S.l.]: Edgard Blucher, 2005.

SILVA, A.; VIDEIRA, C. ***UML, Metodologias e Ferramentas Case***. 1. ed. [S.l.: s.n.], 2001.

SILVA, A. C. da; PIRES, J. P. *Banco de dados relacional versus banco de dados orientado a grafos aplicados a redes sociais*. [S.l.: s.n.], 2013.

SILVA, M. S. ***CSS 3: desenvolva aplicações web profissionais com o uso dos poderosos recursos de estilização das CSS 3***. [S.l.]: Novatec, 2011.

SILVA, M. S. ***HTML 5***. [S.l.: s.n.], 2011.

VUKOTIC, A.; GOODWILL, J. *Apache Tomcat 7*. [S.l.]: Apress, 2011.

W3C. *The Basics of HTML*. 2015. <https://docs.webplatform.org/wiki/guides/the_basics_of_html>. Acesso em 24/07/2015.

W3C. *What is CSS?* 2015. <http://www.w3.org/Style/CSS/>. Acesso em 24/07/2015.

Apêndices

TÍTULO DO APÊNDICE I

Aqui deve conter o texto do Apêndice 5. Na Figura 15 é ilustrada a primeira tela deste processo.

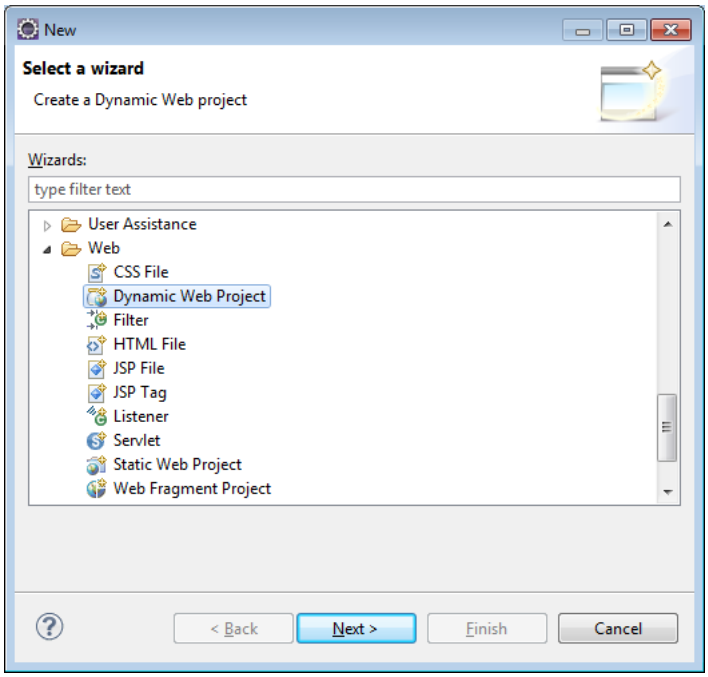

Figura 15 – Outra imagem. **Fonte:** Elaborado pelos autores

Após a seleção do tipo de projeto . . .

TÍTULO DO APÊNDICE 2

Primeira seção do apêndice 2

Neste apêndice é mostrado . . . de acordo com a Figura 16 é ilustrada a primeira tela deste processo.

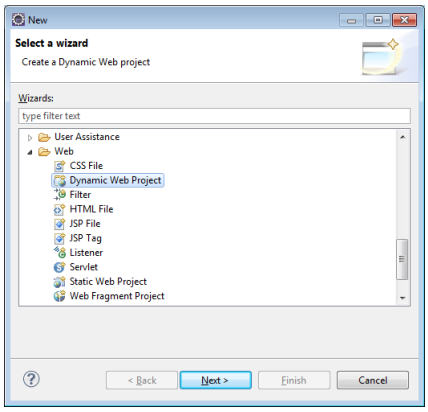

Figura 16 – Outra imagem ainda. **Fonte:** Elaborado pelos autores

Segunda seção do apêndice 2

Continuando . . . na figura Figura 17 é mostrado um exemplo de XML.

```
<project>
...
 <dependencies>
  ...
  <dependency>
   <groupId>org.neo4j</groupId>
   <artifactId>neo4j</artifactId>
   <version>1.9.4</version>
  </dependency>
  ...
 </dependencies>
 ...
</project>
```

Figura 17 – Exemplo de código XML. **Fonte:** Elaborado pelos autores

ANEXO I