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

Projeto de pesquisa apresentado à disciplina de TCC 1 do curso de Sistemas de Informação como requisito para desenvolvimento do Trabalho de Conclusão de Curso sob a orientação do prof. Márcio Emílio Cruz Vono de Azevedo.

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

UNIVERSIDADE DO VALE DO SAPUCAÍ
POUSO ALEGRE – MG
2015

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 OBJETIVOS

Para que se possa levar esta proposta de pesquisa a cabo são colocados os seguintes objetivos:

2.1 Objetivo geral

Desenvolver uma aplicação *web* utilizando uma base de dados orientada a grafos, capaz de realizar buscas por profissionais que desempenham trabalhos temporários sem vínculo empregatício.

2.2 Objetivos específicos

- Demostrar o uso do banco de dados Neo4j, por meio do da *Application Program Interface* API Cypher;
- Representar o problema utilizando grafos;
- Projetar e implementar uma aplicação *web* para cadastro e busca de mão de obra.

[2] API: *Application Program Interface* - Conjunto de funções, métodos, ferramentas e protocolos utilizados para o desenvolvimento de um *software*.

3 JUSTIFICATIVA

Por meio de uma pesquisa informal, realizada com o auxílio de formulários disponibilizados na internet, na região de Pouso Alegre, foi constatado que não há um sistema que possua como principal objetivo localizar determinados tipos de mão de obra nos quais não existam vínculos empregatícios.

Este projeto propõem atuar sobre esta limitação, desenvolvendo um software cujo principal objetivo é localizar e apresentar aos usuários, profissionais temporários que possuam credibilidade e boas referências.

A fim de tornar possível a realização do mesmo serão utilizadas tecnologias gratuitas e de boa aceitação pelo mercado. Desta forma, será possível reduzir o custo de desenvolvimento, possibilitando a sua distribuição aos usuários. Com esta distribuição, seus benefícios terão acesso a uma parcela maior de pessoas, aumentando consideravelmente a facilidade na busca por este tipo de profissional.

Este projeto também visa agregar valor acadêmico, proporcionando uma base de conhecimentos e explanação de tecnologias atuais e valorizadas, sendo que algumas não fazem parte do escopo do curso, como exemplo, o conceito de banco de dados orientado a grafos e o Neo4J.

Agregando todos estes benefícios à possibilidade de melhoria contínua e acréscimos de novas funcionalidades, este projeto servirá como base para novos trabalhos acadêmicos.

4 QUADRO TEÓRICO

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

4.1 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.

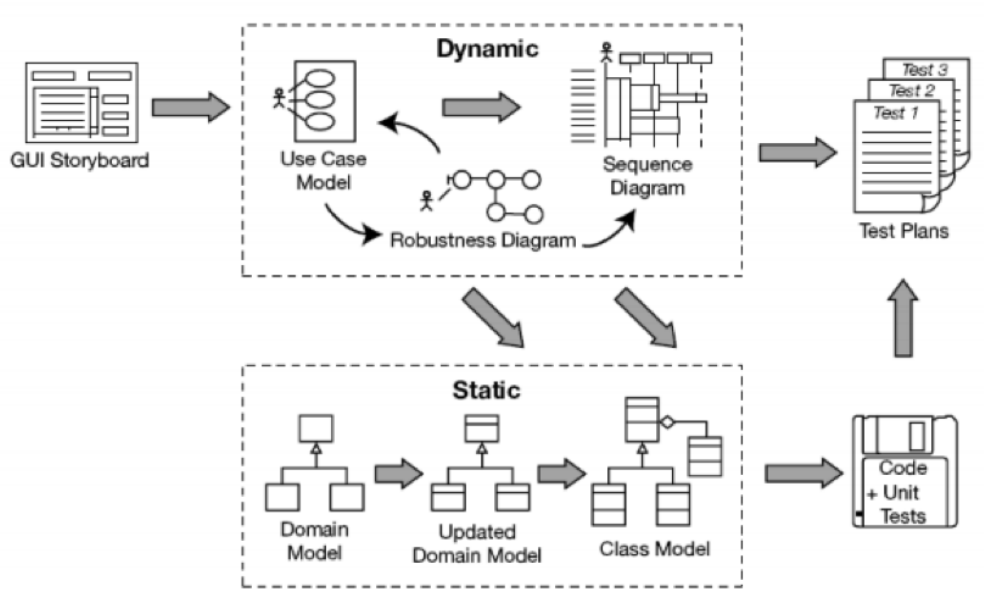

Figura 1 – Uma visão geral do ICONIX e seus componentes. **Fonte:** Rosenberg e Scott (1999)

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).

4.1.1 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.

4.1.2 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.

4.1.3 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.

4.1.4 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.

4.2 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.

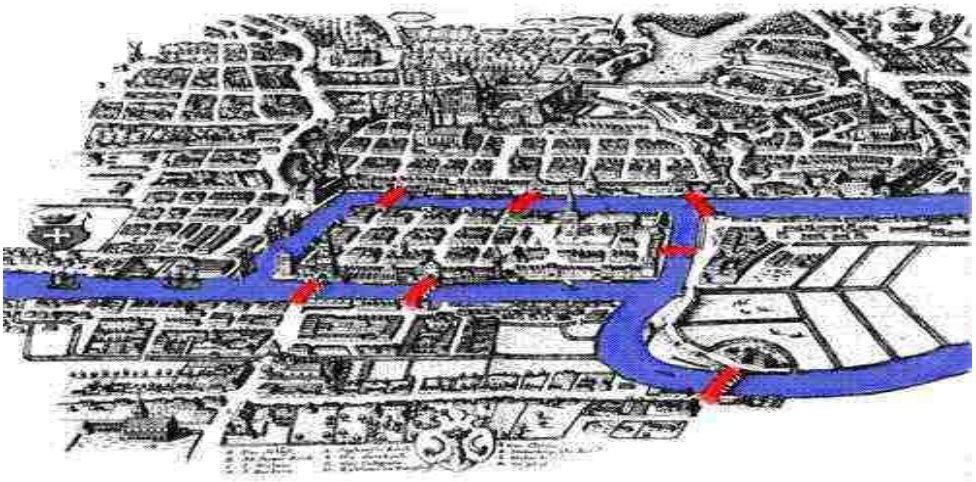

Figura 2 – O problema das 7 pontes de *Königsberg*. **Fonte:** Paoletti (2006)

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.

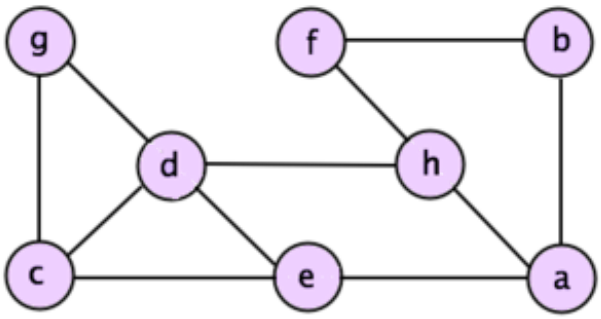

Figura 3 – Ilustração de uma representação gráfica de um simples grafo. **Fonte:** Rocha (2013)

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*.

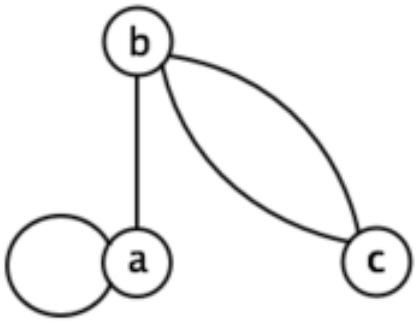

Figura 4 – Ilustração de uma representação gráfica de um *multigraph*. **Fonte:** Adaptado de Harju (2014)

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 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.

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

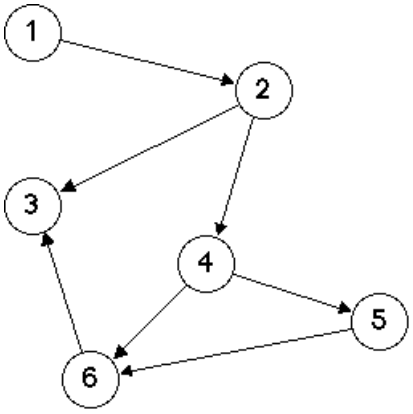

Figura 5 – Imagem de uma representação gráfica de um grafo direcionado. **Fonte:** Keller (2015)

Abaixo serão descritos alguns tipos de grafos existentes, de acordo com B. (1996).

- **grafo simples:** são aqueles grafos que não possuem *loops*, ou arestas paralelas.

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. Devido a estes motivos, esta teoria será utilizada para auxiliar no desenvolvimento deste projeto.

4.3 Tecnologias

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

4.3.1 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).

4.3.1.1 Tipos de bancos de dados

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, o desempenho. Portanto, essa decisão deve ser bem analisada, levando em consideração o tipo de software e no ambiente de produção que será utilizado.

A seguir, são demonstrados os principais modelos de banco de dados, abordando suas características.

4.3.2 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.

4.3.3 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].

[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.

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

4.3.4 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* 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.*[10]

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. 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*[11] *Traversal* que utiliza a API *Cypher* internamente para navegar pelo grafo.

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. Já no modo *Server* o banco de dados é executado embarcado à aplicação Java, que será utilizado por este trabalho.

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

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.

4.3.5 *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-

[10] 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.

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

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 nó inicial para a busca.
- *MATCH*: Define o padrão de correspondência entre os nós.
- *CREATE*: Cria nós e relacionanemtnos.
- *WHERE*: Define um critério de busca.
- *RETURN*: Define quais nós e/ou atributos devem ser retornados da *query* realizada.

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.

4.3.6 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 de programação 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.

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 sub classes, 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, 2015b).

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).

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[12] - instalada no computador. A JVM compreende e executa apenas *bytecodes*[13] 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 6, 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.

Por todas as vantagens descritas acima, será empregado o uso desta tecnologia neste projeto.

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

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

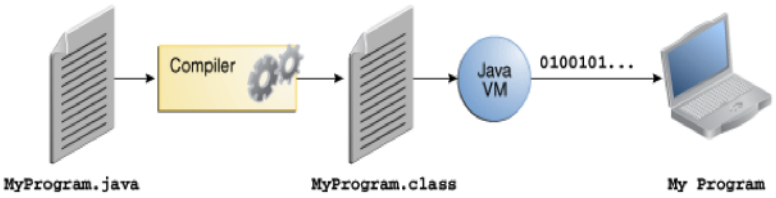

Figura 6 – Uma visão geral do processo de desenvolvimento de software. **Fonte:** Oracle (2010)

4.3.7 Rest

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.

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).

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[12] - instalada no computador. A JVM compreende e executa apenas *bytecodes*[13] e estes por sua vez são obtidos através do processo de compilação do código escrito em Java.

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

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

Todo programa que utiliza Java necessita passar por algumas etapas essenciais. Conforme ilustrado na figura 6, 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.

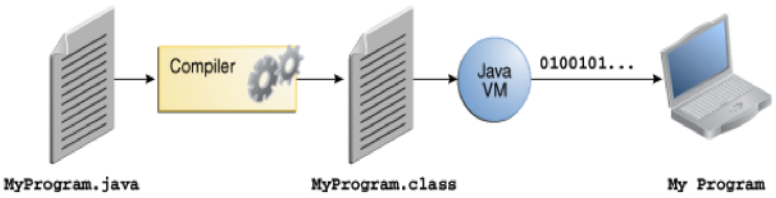

Figura 7 – Uma visão geral do processo de desenvolvimento de software. **Fonte:** Oracle (2010)

Por todas as vantagens descritas acima, será empregado o uso desta tecnologia neste projeto.

4.3.8 Tomcat 7

O Tomcat é uma aplicação *container*, capaz de hospedar aplicações *web* baseadas em Java. A princípio ele foi criado para executar *servlets*[14] e *JavaServer Pages* - JSP[15] -. 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[16] e iniciam os métodos necessários do *servlet* de acordo com o tipo de requisição realizada pelo cliente (BASHMAN; SIERRA; BATES, 2010).

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

Brittain e Darwin (2007) afirmam que o Tomcat foi desenvolvido utilizando a linguagem de programação Java, sendo necessário possuir uma versão do Java SE *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 por meio do uso de *sockets*.

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.

4.3.9 Javascript

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.

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).

Retomando a ideia da Oracle (2010), uma das vantagens da tecnologia Java sob as de-

mais 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[12] - instalada no computador. A JVM compreende e executa apenas *bytecodes*[13] 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 6, 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.

Figura 8 – Uma visão geral do processo de desenvolvimento de software. **Fonte:** Oracle (2010)

Por todas as vantagens descritas acima, será empregado o uso desta tecnologia neste projeto.

4.3.10 Angular Js

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

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

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

foi derivada da linguagem C. O C++ e o Java possuem em comum o conceito de orientação a objetos.

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).

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[12] - instalada no computador. A JVM compreende e executa apenas *bytecodes*[13] 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 6, 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.

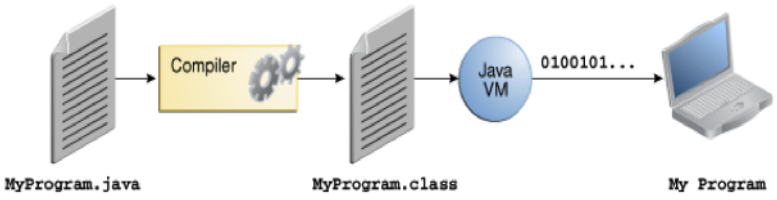

Figura 9 – Uma visão geral do processo de desenvolvimento de software. **Fonte:** Oracle (2010)

Por todas as vantagens descritas acima, será empregado o uso desta tecnologia neste projeto.

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

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

4.3.11 JavaServer Faces

Segundo Faria (2013), a tecnologia *JavaServer Faces* - JSF[17] - foi definida pelo JCP (*Java Community Process*), tornando-a um padrão de desenvolvimento, facilitando assim o trabalho dos desenvolvedores de software.

Bergsten (2004) afirma que o JSF é um *framework server-side* baseado em componentes *web*, cuja principal função é abstrair os detalhes de manipulação dos eventos e organização dos componentes na página *web*. Por meio dele é possível desenvolver páginas mais sofisticadas de forma simples, abstraindo inclusive, o tratamento de requisições e respostas. Isto permite ao desenvolvedor focar-se no *back-end* da aplicação, ou seja, na lógica, e não se preocupar com detalhes a respeito de requisições e respostas HTTP e como obter as informações recebidas e/ou enviadas através deste protocolo.

De acordo com Oracle (2015a), o JSF é de fácil aprendizado e utilização, pois possui sua arquitetura claramente definida, sendo dividida entre a lógica da aplicação e apresentação. Esta divisão é possível pois ele utiliza o padrão de projeto *Model-View-Controller* - MVC[18] -, tornando-o um importante *framework* para desenvolvimento de aplicações utilizando a plataforma Java *Web* e com alta demanda no mercado.

Segundo Gamma et al. (2009), o padrão de projeto MVC é dividido em três partes. O *Model* é a lógica da aplicação, a *View* é camada de apresentação e por último o *Controller* é responsável por definir a interface entre a lógica e a apresentação. Portanto, todo tipo de requisição ou resposta deve ser obrigatoriamente enviada ao *Controller*, que, por sua vez encaminhará para a camada de visão ou de lógica. A figura 7 demonstra um exemplo do modelo MVC utilizando o JSF.

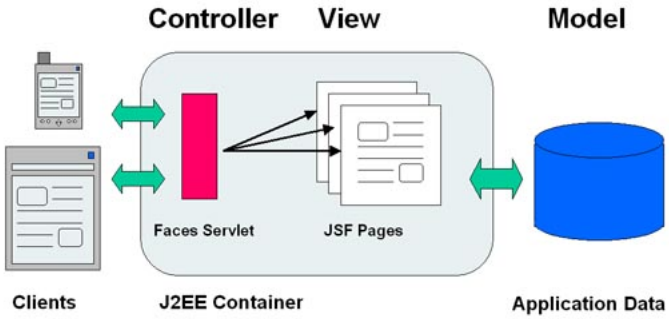

Figura 10 – Imagem de demonstração do Modelo MVC **Fonte:** http://www.javabeat.net/jsf-2/

[17] JSF: *JavaServer Faces* - Framework que permite desenvolver páginas *web* para aplicações desenvolvidas em Java.

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

Ao utilizar o JSF, toda e qualquer interação que o usuário realizar com a aplicação será executada por um *servlet* chamado *Faces Servlet*. Ela é a responsável por receber tais requisições da camada de visão e redirecioná-las à lógica da aplicação e, posteriormente, enviar a resposta ao usuário (FARIA, 2013).

Por possuir as vantagens descritas acima e possuir uma simples configuração, além de ser um *framework cross-browser*[19], este *framework* foi escolhido para auxiliar no desenvolvimento das páginas *web* deste projeto.

4.3.12 Primefaces

O Primefaces é uma biblioteca de componentes que implementa a especificação do JSF e deve ser utilizada em conjunto com o mesmo a partir da versão 2.0. Ele possui uma ampla gama de componentes disponíveis para auxiliar o desenvolvimento de interfaces *web* ricas, além de possuir o seu código fonte aberto (ROSS; BORSOI, 2012).

Segundo Juneau (2014), uma das grandes vantagens do Primefaces é a facilidade de integração entre ele e o JSF, bastando apenas incluir a biblioteca do Primefaces no projeto JSF. Salvo alguns componentes específicos, como o *file upload* que necessita de pequenas configurações adicionais. Estas mudanças, quando necessárias, devem ser realizadas no arquivo de configuração da aplicação, que por padrão, é chamado de *web.xml*, porém, o mesmo pode ser alterado pelo desenvolvedor.

Por todas as vantagens mencionadas acima, somado ao fato desta biblioteca possuir uma ótima documentação, em conjunto com uma grande comunidade de desenvolvedores que a utilizam e a alta demanda de desenvolvedores que a conhecem no mercado, ela foi escolhida em conjunto com o JSF para desenvolver as páginas *web* deste projeto.

[19] *Cross-Browser* - Compatibilidade com todos os tipos de dispositivos e navegadores

5 QUADRO METODOLÓGICO

Será apresentado neste capítulo a metodologia de pesquisa a ser utilizada para a realização deste projeto e os passos necessários até a sua conclusão.

5.1 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.

Conforme Cooper e Schindler (2003, p. 32), a pesquisa aplicada: "tem uma ênfase prática na solução de problemas, embora a solução de problemas nem sempre seja gerada por uma circunstância negativa".

Seguindo esta ideia, este tipo de pesquisa será utilizado no desenvolvimento deste projeto, pois, o objetivo é gerar uma solução para o problema de localização de mão de obra, por meio de um sistema *web*. Este projeto adequa-se perfeitamente ao tipo de pesquisa aplicada, uma vez que o mesmo busca analisar e gerar uma possível solução para o problema em destaque.

5.2 Contexto de pesquisa

Esta pesquisa terá como foco todas as pessoas da região do sul de Minas Gerais, que necessitam de determinados tipos de mão de obra (contratantes), bem como para aqueles que necessitam de um espaço para divulgar esta oferta (contratados ou prestadores de serviços). Tomando por base que encontrar este tipo de profissional tem se tornado uma tarefa cada vez mais complicada, que acaba gerando transtornos na vida da população, pois, em muitos casos, a falta de opções leva a contratações equivocadas e infelizes.

Como o sistema será desenvolvido em uma plataforma *web*, todas as pessoas neste contexto terão fácil acesso ao serviço, permitindo a disseminação desta ferramenta, sendo necessário apenas possuir uma comunicação com a internet.

5.3 Participantes

Participam deste projeto dois acadêmicos e um professor orientador. São eles:

- Andressa de Faria Giordano: acadêmica do 7º período do curso de bacharelado em Sistemas de Informação pela Universidade do Vale do Sapucaí - UNIVAS - com formação técnica e profissionalizante pelo Instituto Nacional de Pesquisa Tecnologica e Computacional - INPETTECC. Sua participação, neste projeto, será o levantamento dos requisitos e a elaboração de todos os diagramas descritos nas fases do ICONIX, a modelagem do banco de dados, a implantação da lógica de negócios, comunicação entre os módulos, acesso aos dados pelo sistema e a documentação do trabalho.

- Edilson Justiniano: acadêmico do 7º período do curso de bacharelado em Sistemas de Informação pela Universidade do Vale do Sapucaí - UNIVAS - com formação técnica e profissionalizante pelo Instituto Nacional de Pesquisa Tecnologica e Computacional - INPETTECC. Sua participação, neste projeto, será o levantamento dos requisitos e a elaboração de todos os diagramas descritos nas fases do ICONIX, a modelagem do banco de dados, a implantação da lógica de negócios, comunicação entre os módulos, acesso aos dados pelo sistema e a documentação do trabalho.

- Márcio Emílio Cruz Vono de Azevedo: professor orientador, engenheiro elétrico em modalidade eletrônica pelo Instituto Nacional de Telecomunicações - INATEL - e mestre em Ciência da Computação pela Universidade Federal de Itajubá - UNIFEI. Professor do INATEL e da Universidade do Vale do Sapucaí - UNIVAS - na área de engenharia de *software* e especialista em sistemas do Inatel.

A elaboração do texto escrito será realizado com base nas ações que serão desenvolvidas pelos participantes e serão feitas em conjunto.

5.4 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.

5.4.1 Questionários

Para Silva e Menezes (2005), questionário é:

> Uma série ordenada de perguntas que devem ser respondidas por escrito pelo informante. O questionário deve ser objetivo, limitado em extensão e estar acompanhado de instruções As instruções devem esclarecer o propósito de sua aplicação, ressaltar a importância da colaboração do informante e facilitar o preenchimento.

Serão disponibilizados questionários *on-line* a fim de avaliar o interesse da população por um sistema que ofereça o serviço de localização de mão de obra temporária. Para desenvolver tal *software*, uma bateria de testes será feita visando oferecer a melhor interação entre o usuário e o software, levando em conta sua aplicabilidade, desempenho e facilidade de utilização.

5.4.2 Análise documental

Para auxiliar no desenvolvimento deste projeto serão utilizados documentos como: manuais, tutoriais, livros, trabalhos e artigos acadêmicos, além de pesquisa web. Estes serão utilizados para o embasamento teórico que consiste no levantamento de requisitos e tecnologias a serem abordadas, e também para o desenvolvimento prático servindo como apoio e referencial para futuras consultas.

5.5 Procedimentos

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

- Realizar o levantamento de tecnologias a serem utilizadas;

- Fazer um levantamento de teorias em livros da área;
- Fazer um levantamento de requisitos;
- Definir todas as ações que o usuário poderá realizar, por meio de diagramas como o de caso de uso;
- Escrever os diagramas necessários para o início do desenvolvimento do projeto;
- Desenvolver a base de dados, de acordo com os requisitos levantados;
- Implementar o sistema utilizando as tecnologias descritas no quadro teórico;
- Realizar testes, a fim de, evitar que erros passem despercebidos;
- Disponibilizar o sistema na *web*.

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

5.6 Cronograma

Será exibido aqui uma tabela contendo o cronograma a ser seguido por este projeto, desde a sua definição até a sua conclusão.

Ação / Mês	Janeiro	Fevereiro	Março	Abril	Maio	Junho	Julho	Agosto	Setembro	Outubro	Novembro	Dezembro
Definição do Pré projeto	■											
Aprovação do Pré projeto		■										
Levantamento bibliográfico	■	■	■									
Primeira entrega do Pré projeto		■										
Orientação sobre Introdução		■										
Entrega da Introdução		■										
Orientações sobre Objetivos e Justificativas		■										
Entrega dos Objetivos e Justificativas		■										
Orientações sobre o Quadro Teórico		■	■									
Entrega do Quadro Teórico			■									
Orientações sobre o Quadro Metodológico			■	■								
Entrega do Quadro Metodológico				■								
Revisão de Referências				■	■							
Qualificação do projeto					■							
Levantamento de requisitos					■	■						
Desenvolvimento da pesquisa						■	■	■	■	■	■	

Tabela 1 – Cronograma do desenvolvimento do projeto. **Fonte:** Elaborado pelos autores

5.7 Orçamento

Abaixo serão apresentadas as despesas de forma geral previstas para a realização deste projeto.

Despesas	**Valor Previsto**
Impressão	R$ 70,00
Encadernação	R$ 35,00
Impressão em capa dura	R$ 80,00
Livros	R$ 1.700,00
Total	**R$ 1.885,00**

Tabela 2 – Orçamento previsto do projeto. **Fonte:** Elaborado pelos autores

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.

B., N. P. O. Grafos: teoria, modelos, algoritmo. Blucher, 1996.

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.

BERGSTEN, H. ***Javaserver Faces***. [S.l.: s.n.], 2004.

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.

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

FARIA, T. ***Java EE 7 com JSF, PrimeFaces e CDI***. [S.l.: s.n.], 2013.

GAMMA, E. et al. ***Design Patterns: Elements of Reusable Object-Oriented Software***. [S.l.: s.n.], 2009.

HARJU, T. Graph theory. 2014.

JUNEAU, J. *Primefaces in the Enterprise*. 2014. <http://www.oracle.com/technetwork/articles/java/java-primefaces-2191907.html>. Acesso em: 15 de janeiro de 2015.

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.

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.

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. *JavaServer Faces Technology Overview*. 2015. <http://www.oracle.com/technetwork/java/javaee/overview-140548.html>. Acesso em 15 de janeiro de 2015.

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

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.

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

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

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.

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.

ROSS, C. L.; BORSOI, B. T. **Uso de Primefaces no desenvolvimento de aplicações ricas para web**. 2012.

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, E. L. da; MENEZES, E. M. ***Metodologia da Pesquisa e Elaboração de Dissertação***. 4. ed. [S.l.]: UFSC, 2005.

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