JONATHAN RIBEIRO DOS SANTOS
SANDRO AUGUSTO DE OLIVEIRA

ALGORITMOS GENÉTICOS APLICADOS EM LINHA DE PRODUÇÃO

UNIVERSIDADE DO VALE DO SAPUCAÍ
POUSO ALEGRE – MG
2015

1 QUADRO TEÓRICO

Neste capítulo serão listados os conceitos e as tecnologias que serão utilizados no desenvolvimento da proposta de trabalho apontada na seção objetivos. Para tal, serão discutidos a definição, o histórico e as aplicabilidades de cada um deles, tomando por base autores fundantes e seus comentaristas. É importante ressaltar que o texto desta sessão, quando descreve a teoria da evolução das espécies, não tem como objetivo levantar questões sobre a origem dos seres vivos.

1.1 Algoritmos Genéticos

1.1.1 Fundamentos

Para Melanie (1999), desde o começo da era computacional, cientistas pioneiros, tais como Alan Turing, John von Neumann, Norbert Wiener e outros, tinham o objetivo de dotar os computadores de inteligência de maneira que eles pudessem tomar decisões, se adaptar a determinadas situações e até mesmo ter a capacidade de aprender. Com esta motivação, estes cientistas se interessaram por outras áreas, além da eletrônica, como a biologia e a psicologia e começaram então a realizar pesquisas para simular os sistemas naturais no mundo computacional a fim de alcançarem suas metas.

Vários conceitos computacionais baseados na natureza surgiram então ao longo do tempo, dentre eles a computação evolucionária inspirada na teoria da evolução natural a qual o exemplo mais proeminente são os AGs que foram introduzidos por Jhon Holland, seu aluno David Goldberg e outros estudantes da universidade de Michigan. Goldberg (1989) define os AGs como métodos de busca baseados na genética e no mecanismo de seleção natural que permitem a possibilidade de obter robustez e eficácia na tarefa de encontrar uma solução boa para um problema em um espaço de busca complexo em um tempo aceitável.

Segundo Linden (2012), a teoria da evolução foi proposta pelo naturalista inglês Charles Darwin por volta de 1850, quando este, em uma viagem de navio visitou vários lugares e, por ser uma pessoa com uma grande habilidade de observação, percebeu que indivíduos de uma mesma espécie vivendo em lugares diferentes possuíam também características distintas, ou

seja, cada indivíduo possuía atributos específicos que lhe permitia uma melhor adaptação em seu ecossistema.

O autor afirma então que, com base nesta observação, Darwin propôs que existe um processo de seleção natural, afirmando que, como os recursos na natureza, tais como água e comida, são limitados, os indivíduos competem entre si e aqueles que não possuem atributos necessários à adaptação ao seu ambiente tendem a ter uma probabilidade menor de reprodução e irão ser extintos ao longo do tempo, e por outro lado, aqueles com características que os permitem obter vantagens competitivas no meio onde vivem acabam tendo mais chances de sobreviver e gerar indivíduos ainda mais adaptados.

A teoria ressalta porém que, o processo não tem o objetivo de maximizar algumas características das espécies, pois os novos indivíduos possuem atributos que são resultados da mesclagem das características dos reprodutores, o que faz com que os filhos não sejam exatamente iguais aos pais, podendo assim ser superiores, uma vez que, estes herdem as qualidades de seus pais ou inferiores se os descendentes herdarem as partes ruins de seus reprodutores. Este processo de transferência de informação será explicado posteriormente (LINDEN, 2012).

Para entender a relação entre AGs e a evolução natural é necessário conhecer as principais terminologias biológicas, sendo importante ressaltar porém que, de acordo com Melanie (1999), apesar da analogia a certos termos da biologia, a forma com que os AGs são implementados é relativamente simples se comparado ao funcionamento biológico real.

Melanie (1999) afirma que todos os seres vivos são compostos de células e estas possuem um ou mais cromossomos que, basicamente, são manuais de instruções que definem as características do organismo. O cromossomo é formado por um conjunto de genes que, em grupo ou individualmente, são responsáveis por um determinado atributo do indivíduo como por exemplo, a cor do cabelo, a altura e etc. Cada gene possui uma localização dentro do cromossomo denominada locus e, por fim, o conjunto de todos os cromossomos dentro da célula é definido como genoma.

Considerando isto, Linden (2012, p.33) afirma que,

> Um conjunto específico de genes no genoma é chamado de genótipo. O genótipo é a base do fenótipo, que é a expressão das características físicas e mentais codificadas pelos genes e modificadas pelo ambiente, tais como cor dos olhos, inteligência etc. Daí, podemos concluir: nosso DNA codifica toda a informação necessária para nos descrever, mas esta informação está sob controle de uma grande rede de regulação gênica que, associada às condições ambientais, gera as proteínas na quantidade certa, que farão de nós tudo aquilo que efetivamente somos.

Uma vez descrita a complexidade dos organismos é necessário discorrer, de forma bá-

sica, sobre o processo de reprodução responsável pela transmissão da informação genética de geração para geração.

Linden (2012) afirma que, existem dois tipos de reprodução, a assexuada, em que não é necessário a presença de um parceiro e a sexuada que exige a presença de dois organismos. Os AGs simulam a reprodução sexuada em que cada um dos organismos envolvidos oferece um material genético denominado gametas. As gametas são formadas através de um processo denominado *crossing-over* ou *crossover* que tem início com a divisão de cada cromossomo em duas parte as quais irão se cruzar uma com a outra para formar dois novos cromossomos, que receberão um pedaço de cada uma das partes envolvidas no cruzamento.

Ainda segundo o autor, o resultado deste processo será então quatro cromossomos potencialmente diferentes que irão compor as gametas e farão parte do novo indivíduo. Neste processo pode ocorrer mutações que são resultados de alguns erros ou da influência de algum fator externo, como a radiação por exemplo. Estas mutações são pequenas mudanças nos genes dos indivíduos, podendo estas ser boas, ruins ou neutras.

E assim a informação genética é passada dos pais para os filhos, e como os componentes dos cromossomos definem as características do organismo, os filhos herdarão características dos pais porém serão ligeiramente diferente deles, como foi descrito anteriormente, o que permite que os novos indivíduos herdem características melhores ou piores que seus progenitores, porém, se os pais possuem características positivas, a probabilidade de gerarem filhos ainda melhores são maiores (LINDEN, 2012).

1.1.2 Características dos Algoritmos Genéticos

De acordo com Linden (2012), a analogia dos AGs com os processos biológicos se dá por meio da representação de cada termo descrito anteriormente em um modelo computacional voltado a encontrar soluções a um determinado problema em um processo aleatório. O fluxo de execução deste processo inicia-se com a criação aleatória de uma população inicial. Uma população contém um conjunto de indivíduos sendo que cada indivíduo representa uma possível solução para o problema.

O autor afirma ainda que, um indivíduo é formado por cromossomos que guardam as características da solução, ou seja, a forma que esta resolve o problema.

Segundo Melanie (1999), a execução de um algoritmo genético é basicamente realizada conforme os itens a seguir:

- Definição da população inicial;
- Avaliação e classificação dos indivíduos da população;
- Seleção de acordo com a qualidade;
- Cruzamento para geração de novos descendentes;
- Mutação aleatória dos novos indivíduos;
- Repetição do processo de seleção, cruzamento e mutação até se formar uma nova população;
- Avaliação e classificação dos indivíduos da nova população;
- A nova população substitui a anterior e o processo continua a partir do segundo ponto até que o número de populações criadas atinja um limite que é definido previamente ou até atingir outra condição definida pelo programador.

A figura Figura 1 ilustra os passos descritos acima.

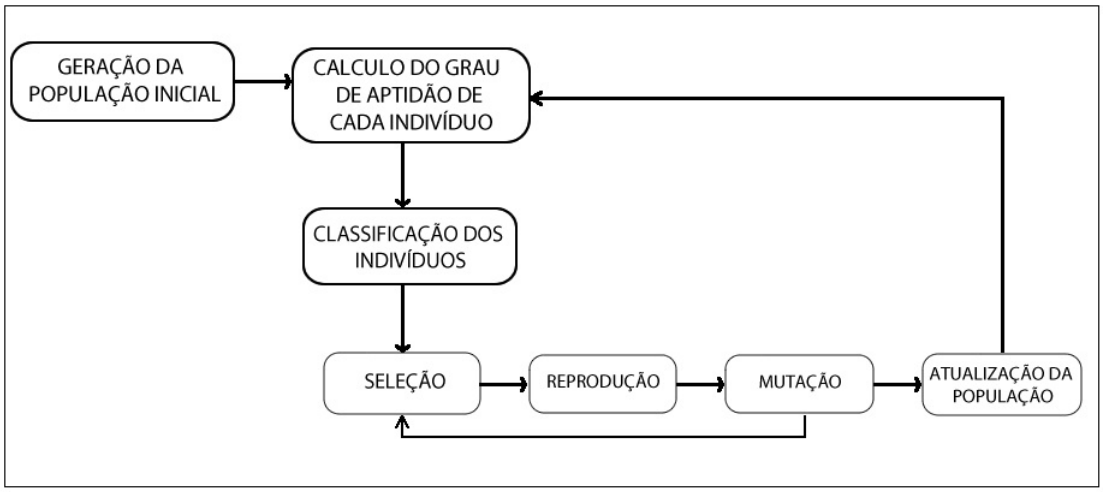

Figura 1 – Demonstração da execução de um AG **Fonte:** Desenvolvido pelos autores

1.1.2.1 População Inicial

De acordo com Melanie (1999), a população inicial é o conjunto dos primeiros indivíduos candidatos a resolução do problema. Estes indivíduos devem ser criados aleatoriamente,

seguindo a lógica definida pelo programador com base no contexto do problema a ser resolvido. Um exemplo seria a população inicial do algoritmo desenvolvido neste trabalho, em que a população inicial será formada por possíveis formas de se produzir um determinado lote de calças, ou seja, será formada uma população de possíveis formas de dividir o trabalho entre as costureiras.

1.1.2.2 Função de avaliação

Segundo Linden (2012) após a criação da população inicial esta é então avaliada através de uma função de avaliação que mede a qualidade de cada uma de suas soluções e é realizado então uma classificação que ordena as soluções das melhores para as piores, para então iniciar a formação de uma nova população. A nova população pode conter já inicialmente os dois melhores indivíduos existentes, este mecanismo é denominado elitismo e pode ser utilizado ou não.

Ainda segundo o autor a função de Avaliação ou função de aptidão penaliza as soluções inviáveis para a solução do problema, ou seja, ao verificar que uma certa solução não satisfaz o grau de aptidão necessário para o problema proposto esta solução é descartada e assim, só sobrevivem na execução da função as soluções com mais chance de resolver o problema e por isso torna-se o componente mais importante de qualquer algoritmo genético.

Devido a generalidade encontrada nos AGs a função de avaliação torna-se em muitos casos a única ligação verdadeira entre o programa e o problema real, por isso deve-se ter um certo cuidado ao desenvolvê-la, ela deve conter tudo o que se sabe sobre o problema a ser resolvido, quanto mais conhecimento a função possuir, maior será a chance da função entregar o melhor resultado (LINDEN, 2012).

1.1.2.3 Seleção

Segundo Linden (2012), o próximo passo para a formação da nova população é o cruzamento que inicia-se com a seleção de dois indivíduos que é realizada de acordo com a qualidade destes, porém, como é fundamental que esta escolha não despreze completamente os indivíduos com uma qualidade muito baixa, a seleção é feita de forma probabilística ou seja indivíduos com

boa qualidade possuem mais chances de serem selecionados e, por outro lado, indivíduos com menor nota de avaliação terão menos chance.

O autor ressalta que a razão de o mecanismo de seleção não escolher apenas as melhores soluções é devido ao fato de que aquelas com menor grau de avaliação também serem importantes para que se tenha uma maior diversidade de características na população envolvida para solução do problema, possibilitando assim que esta possa evoluir de forma satisfatória, pois se as novas populações forem constituídas somente das melhores soluções, elas serão compostas de indivíduos cada vez mais semelhantes impedindo assim que novas soluções ainda melhores sejam concebidas.

Existem vários tipos de seleção, no projeto será utilizado o método da roleta. Nesta forma de seleção, muito utilizada entre a maioria dos pesquisadores de AGs, metaforicamente, cada indivíduo da população ocupa uma porção da roleta, proporcional ao seu indicie de aptidão, ou seja, os indivíduos que possuírem maior aptidão ocuparão uma maior porção do que aqueles com menor aptidão. A roleta então, assim como mostra a figura 2, é girada e o indivíduo escolhido é aquele em que a seta parar sobre ele. (FILITTO, 2008).

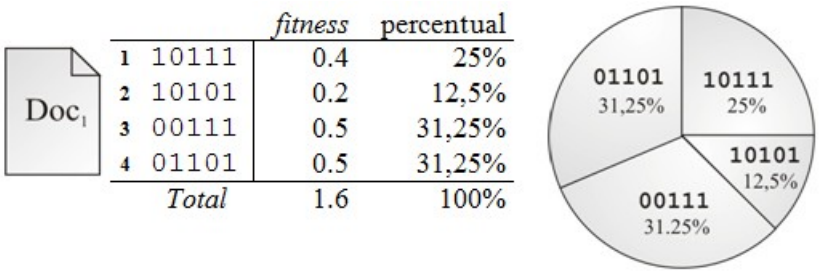

Figura 2 – Método de seleção Roleta **Fonte:** "http://www.dgz.org.br/fev09/Art04.htm"

1.1.2.4 Cruzamento

Segundo Linden (2012) Após a seleção dos pais irá ocorrer então o processo de cruzamento ou *crossover* e, como no cruzamento natural, neste processo dois novos indivíduos, ou seja, duas novas soluções são formadas a partir de características daquelas que se cruzaram, ou seja, serão geradas duas novas soluções que conterão alguns cromossomos de uma solução e alguns cromossomos de outra.

Para Lacerda e Carvalho (2015) o operador crossover é um mecanismo de busca dos AGs para explorar regiões desconhecidas do espaço de busca, ele é aplicado a cada par de cromossomos dos indivíduos retirados da população intermediária, gerando dois cromossomos

filhos até se formar um novo indivíduo. O crossover é aplicado com uma dada probabilidade a cada par de cromossomos selecionados. Essa probabilidade varia entre 60% e 90% e caso não ocorra o crossover os filhos serão iguais aos pais possibilitando que algumas soluções sejam preservadas.

Para FILITTO (2008), o cruzamento combina os cromossomos pais a fim de gerar cromossomos filhos e para isso existem vários tipos de cruzamento. Dentre eles o cruzamento em um ponto e cruzamento Uniforme.

O cruzamento em um ponto divide a lista de cromossomos selecionados em um ponto de sua cadeia, esse ponto é escolhido aleatoriamente. Após essa divisão é copiado uma parte dos cromossomos de cada pai para definir os cromossomos dos indivíduos filhos. Neste método é comum os pais gerarem dois novos filhos, mas a quantidade de filhos pode ser definida levando em consideração que o numero de alelos permita as diferentes combinações (FILITTO, 2008).

Já o cruzamento uniforme, segundo o autor, gera cada gene do descendente copiando o gene em questão de um dos pais, em que se usa uma mascara de cruzamento que é gerada aleatoriamente para fazer a escolha. Para criar cada cromossomo do novo indivíduo é feito uma iteração em todas as posições da máscara fazendo uma análise dos seus valores, quando o valor da posição for 1, o gene do primeiro pai referente a mesma posição da máscara é copiado, se o valor for 0 será copiado o gene do segundo pai, depois desse processo é gerado o novo descendente.

Segundo Lacerda e Carvalho (2015), o tipo de cruzamento uniforme se difere do cruzamento de um ponto uma vez que este sempre leva a metade dos bits de cada pai.

1.1.2.5 Mutação

Para Linden (2012), também pode ocorrer a mutação em que, da mesma forma que ocorre na natureza, aleatoriamente o valor dos cromossomos de um indivíduo pode ser alterado. A mutação ocorre de acordo com uma taxa definida. Basicamente é definido uma porcentagem baixa e então um número de 0 a 1 é sorteado e multiplicado por 100, se o resultado for menor que a porcentagem definida, irá ocorrer a mutação para aquele indivíduo.

Para Lacerda e Carvalho (2015), a mutação melhora a diversidade dos cromossomos na população, em contra partida depois de realizada a mutação se perde informações contidas no cromossomo assim para assegurar a diversidade deve-se usar uma taxa de mutação pequena.

Assim como no cruzamento há vários tipos de mutação, dentre eles a mutação de bit que será utilizada neste trabalho.

Segundo FILITTO (2008) este é o operador mais fácil de trabalhar podendo ser aplicado em qualquer forma de representação binária dos cromossomos. Este tipo de mutação gera uma probabilidade de mutação para cada bit do cromossomo, se caso a probabilidade sorteada estiver dentro da taxa de mutação definida o bit sofrerá mutação, recebendo um valor determinado de forma aleatória dentre os valores que podem ser assumidos pelo cromossomo.

No problema a ser solucionado na aplicação descrita neste trabalho há um cenário em que existem diversas soluções para se resolver o problema e deseja-se encontrar a melhor dentre elas. Conforme descrito anteriormente, os AGs é uma das melhores opções para se resolver este tipo de problema, por este motivo esta técnica foi escolhida.

1.2 Tecnologias

Abaixo serão listadas as tecnologias que serão utilizadas no desenvolvimento do projeto.

1.2.1 Linguagem de programação Java

Segundo Oracle (2015b), a Linguagem Java foi projetada para permitir o desenvolvimento de aplicações seguras, portáteis e de alto desempenho para a mais ampla gama de plataformas de computação.

De acordo com Schildt (2007), o Java foi criado em 1991 pela *Sun Microsystems* e foi baseado em uma linguagem já existente, o C++, que foi escolhida por ser orientada a objetos e por gerar códigos pequenos, o que era exatamente o que eles precisavam para implantar em pequenos aparelhos. Além dessas características, um requisito desejável era que a nova linguagem fosse independente de plataforma, para que fosse executado em qualquer arquitetura, tais como, TVs, telefones, entre outros, e então, para atender esta exigência, foi criado o conceito de máquina virtual, que ficou conhecido como *Java Virtual Machine* - JVM[3].

O ponto chave que permite o Java resolver o problema de portabilidade é o fato de o código ser compilado em *Bytecode* que é um conjunto genérico de instruções altamente otimizado

[3] O termo Java Virtual Machine será referenciado pela sigla JVM a partir deste ponto do trabalho.

que é executado pela JVM, e esta, por sua vez, traduz o mesmo para a arquitetura a qual ela está instalada o que possibilita a execução do programa em várias plataformas (SCHILDT, 2007).

Além da portabilidade, o Java também é *Multithread*, ou seja, permite a execução de múltiplas tarefas. A linguagem também conta com um *automatic garbage collector* que consiste em um mecanismo de gerenciamento de memória que a JVM acomoda. Além disso o Java suporta uma extensa biblioteca de rotinas que facilitam a interação com protocolos TCP/IP, como HTTP e FTP, além de possuir uma segurança em sua execução por controlar programas via rede com restrições (SCHILDT, 2007).

De acordo com Junior (2007),

> Atualmente a linguagem está organizada em três segmentos principais:
>
> - JavaMe (*Java Micro Edition*)- Destinado a pequenos dispositivos computacionais móveis, tais como celulares, PDAs e set-top boxes. É composto de máquina virtuais otimizadas para ambientes mais restritos, especificações de funcionalidades e uma API mais compacta;
>
> - JavaSE (*textJava Standard Edition*)- Integra os elementos padrão da plataforma e permite o desenvolvimento de aplicações de pequeno e médio porte. Inclui todas as APIs consideradas de base, além da máquina virtual padrão;
>
> - JavaEE (*Java Enterprise Edition*)- Voltada para o desenvolvimento de aplicações corporativas complexas. Adiciona APIs especificas aos elementos padrão da plataforma.

Java é uma linguagem Orientada a Objetos (OO). Para Santos (2003), este paradigma usa objetos, criados a partir de Modelos, também chamados de Classes, para representar e processar dados em aplicações computacionais. Basicamente uma Classe ou Modelo são as especificações de um Objeto, ou seja, que tipo de dados este deve ter e quais operações este terá e então após a criação da classe o programador pode criar um objeto desta. Uma analogia simples seria a planta de uma casa e a construção da mesma em si em que a planta representa a classe, ou seja, como a casa deve ser feita e a casa depois de construída representa o objeto.

Os dados e operações são guardados em objetos e estes são os elementos centrais do programa. Assim a aplicação como um todo é vista como uma coletânea de objetos que se relacionam uns com os outros. Cada objeto representa um conceito real de uma parte do problema. Isto permite que o desenvolvimento se torne menos complexo pois os conceitos são familiares as pessoas envolvidas no projeto pelo fato de que a aplicação não está organizada em processos estruturados mas sim em objetos que espelham o mundo real e interagem entre si (MANZONI, 2005).

Para Santos (2003) Os dados pertencentes aos modelos são representados por tipos nativos, característicos das linguagem de programação e podem também ser representados por outros modelos criados pelo programador.

O autor ainda exemplifica um modelo como a representação de um funcionário de uma empresa para fins de processamento de folha de pagamento. Neste caso o modelo representaria dentre outros dados, o nome, cargo, salário e horas extras trabalhadas dessa pessoa, e as operações seria aquelas relacionados com realização de cálculos de salário e impostos por exemplo.

Para o desenvolvimento do sistema de informação deste projeto o paradigma de orientação a objetos, que será implementado na linguagem Java, será imprescindível devido a alta complexidade do problema a ser resolvido. Pois cada objeto representará partes do algoritmo genético que será desenvolvido, tais como a representação do Indivíduo, Cromossomos e etc, o que facilitará muito o desenvolvimento.

A aplicação desenvolvida neste projeto irá fazer uso do Java EE e, como já foi dito em outras sessões anteriores, será implementado em plataforma WEB. Para isso faz se necessário o uso de um servidor de aplicações WEB. Dentre as opções disponíveis foi escolhido o *Tomcat* pelo fato de este ser o mais simples e ter os atributos mínimos necessários para a aplicação que será desenvolvida.

Segundo Vukotic e Goodwill (2011), o *Tomcat* é um servidor *open source* e *container* de aplicativos *web* baseados em Java. Ele foi criado para executar *servlets*, que também é uma denominação para Classe dentro da especificação do Java para WEB, e arquivos de marcação que definem os elementos visuais da tela que será explanado na sessão de interface gráfica. O *Tomcat* foi criado como um subprojeto da *Apache-Jakarta*, porém como ficou muito popular entre os desenvolvedores, a *Apache* o denominou como um projeto separado e vem sendo melhorado e apoiado por um grupo de voluntários da comunidade *Java Open Source*, que o faz uma excelente solução para desenvolvimento de uma aplicação *web* completa.

1.2.2 Interface Gráfica

Para a construção da interface gráfica (páginas web) da aplicação desenvolvida neste projeto, será utilizado uma framework nativo nas novas versões do Java denominado *Java Server Faces* ou JSF.

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[4] -, tornando-o um importante *framework* para desenvolvimento de aplicações utilizando a plataforma Java *Web*.

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 3 demonstra um exemplo do modelo MVC utilizando o JSF.

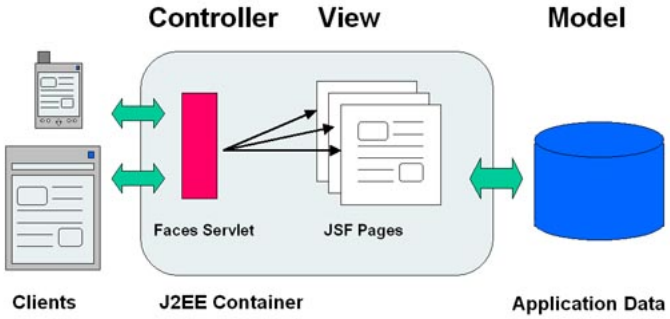

Figura 3 – Imagem de demonstração do Modelo MVC **Fonte:** http://www.javabeat.net/jsf-2/

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).

Segundo Lucknow e Melo (2010), a especificação do *JavaServer Faces* foi definida pela *Java Comunity Process* - JCP[5], que é uma entidade cujo objetivo é especificar a evolução do Java. E por este motivo grandes empresas como *Apache*, *IBM*, *Oracle* entre outras, participaram desta definição. As implementações mais conhecidas são da especificação do JSF são:

- *Sun Mojarra* (antes JSF R1) – implementação de referencia (http://javaserverfaces.java.net/)
- *MyFaces* da *Apache*(http://myfaces.apache.org/)

[4] MVC: *Model-View-Controller - Design pattern*.
[5] O termo *Java Comunity Process* será referenciado pela sigla JCP a partir deste ponto do trabalho.

Com essas implementações é possível utilizar todos os recursos do padrão JSF, como formulários, tabelas, *layout*, conversão e validação de eventos, além de toda a inteligência para a interpretação dos arquivos de configuração e interação com o contêiner Java. Como o JSF é um padrão de mercado, várias empresas investem no desenvolvimento de bibliotecas de componentes como, calendário, barras de progresso, menus, efeitos de transição entre outros.(LUCKNOW; MELO, 2010)

Algumas das principais bibliotecas de componentes são:

- *Trinidad*, da *Apache MyFaces* (http://myfaces.apache.org/trinidad/);
- *Tobago*, da *Apache MyFaces* (http://myfaces.apache.org/tobago/);
- *ICEFaces*, da *ICESoft* (http://www.icefaces.org/);
- *RichFaces*, da *JBoss* (http://www.jboss.org/richfaces/);
- *Tomahawk*, da *Apache MyFaces* (http://myfaces.apache.org/tomahawk/);
- *PrimeFaces* (http://www.primefaces.org/)

Neste projeto, será utilizado a biblioteca PrimeFaces, que 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).

para 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 possuir as vantagens descritas acima e possuir uma simples configuração, além de ser um *framework cross-browser*[6], o JSF foi escolhido para auxiliar no desenvolvimento das páginas *web* deste projeto. E por todas as vantagens do Primefaces 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.

1.2.3 Armazenamento de dados

A aplicação desenvolvida neste projeto irá demandar o armazenamento de informações das costureiras e da disponibilidade das mesmas. Além disso as saídas oferecidas pelo algoritmo genético deverá ser salva para uso posterior alimentando assim a base de dados do sistema, desta forma faz se necessário o uso de um banco de dados.

O gerenciador de banco de dados a ser utilizado nesta aplicação é o PostgreSQL, que foi escolhido por ser uma ferramenta robusta e open source. O PostgreSQL é um banco de dados relacional desenvolvido pela universidade da California por volta de 1970. Na época o projeto se chamava Ingres e só passou a se chamar Postgres por volta de 1986 quando Michael Stonebraker adicionou o conceito de orientação a objetos ao projeto e decidiu então definir um novo nome para a nova versão.(DOUGLAS; DOUGLAS, 2003),

Para Date (2004) , "um banco de dados é um sistema computadorizado cuja funcionalidade geral é armazenar informações e permitir que os usuários busquem e atualizem essas informações quando as solicitar". Já o conceito de banco de dados relacional é definido por Price (2008, p.30) como:

> uma coleção de informações relacionadas, organizadas em tabelas. Cada tabela armazena dados em linhas; os dados são organizados em colunas. As tabelas são armazenadas em esquemas de banco de dados, que são áreas onde os usuários podem armazenar suas próprias tabelas.

Para manipular e acessar as informações em um banco de dados relacional é usado uma linguagem denominada SQL (*Structured Query Language*), que foi projetada especificamente para este fim. A linguagem foi desenvolvida pela IBM por volta de 1970 que tomou como base o trabalho do Dr. E.F.Codd e possui uma sintaxe simples de fácil aprendizado e utilização. (PRICE, 2008)

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

2 QUADRO METODOLÓGICO

O quadro metodológico é a descrição dos passos realizados para a execução do projeto. Serão listados, nos tópicos a seguir, os itens essenciais no desenvolvimento do trabalho, sendo eles as técnicas, procedimentos, práticas e instrumentos utilizados, o contexto de aplicação, os participantes, o orçamento, o cronograma e o tipo de pesquisa utilizado.

2.1 Tipo de pesquisa

Para Gil (1999, p.42), a pesquisa tem um caráter pragmático, é um "processo formal e sistemático de desenvolvimento do método científico. O objetivo fundamental da pesquisa é descobrir respostas para problemas mediante o emprego de procedimentos científicos".

Este projeto terá como base a metodologia de pesquisa aplicada, pois será desenvolvida uma aplicação inteligente utilizando Algoritmos Genéticos para o auxílio na tomada de decisão sobre a produção de calças de uma confecção.

Gerhardt e Silveira (2009, p.35) afirmam que o método de pesquisa aplicada, "objetiva gerar conhecimentos para aplicação prática, dirigidos a solução de problemas específicos. Envolve verdades e interesses locais."

Segundo Zanella (2009), a pesquisa aplicada tem como motivação básica a solução de problemas concretos, práticos e operacionais e também pode ser chamada de pesquisa empírica pois o pesquisador precisa ir a campo, conversar com pessoas e presenciar relações sociais.

2.2 Contexto de pesquisa

Sabe-se que com a alta competitividade no mercado, empresas cada vez mais buscam diferenciais competitivos para seus produtos e, neste cenário, a ideia de redução de custos se torna essencial uma vez que tal redução pode ser refletida no preço dos produtos permitindo que estes se diferenciem. Dentre os fatores que viabilizam tais reduções está a otimização de processos que consistem em organizar os procedimentos relacionados à produção de forma que estes se tornem mais eficazes.

O software desenvolvido neste trabalho visa organizar uma linha de produção de forma que esta se torne o mais eficiente possível. Será utilizada como base uma determinada fábrica de confecção de calças situada na cidade de Cachoeira de Minas - MG, porém a base de conhecimento pode ser aplicada a outros tipos de negócios que seguem o mesmo padrão de desenvolvimento de produtos.

Como já explanado no quadro teórico, a estrutura do algoritmo genético é composta por populações que são formadas por indivíduos que por sua vez são formados por cromossomos. Cada indivíduo representa uma solução e cada cromossomo do indivíduo representa uma de suas características. Assim ocorre então um processo de cruzamento e mutação a fim de que possam ser gerados novos indivíduos que representem soluções ainda melhores que seus antecessores. As sessões abaixo descrevem como cada um destes elementos foram desenvolvidos no contexto do problema da fábrica de calças.

Como cada funcionário da fábrica citada trabalha em sua casa é preciso ter uma boa forma de distribuir a produção, para que o transporte da materia-prima seja eficaz contrbuindo para a empresa uma redução de custos e um tempo de produção melhor. Para isso, é nescessário que o software conheça os procedimentos de produção da fabrica, como seus funcionários trabalham e se estão sendo alocados de forma correta. A aplicação cruza todas essas informações gerando para o usuario uma relação de como ele deve distribuir a matéria-prima para produção das peças.

Como cada funcionário trabalha em sua casa é preciso saber qual a melhor rota para se entregar a materia-prima ou recolher o que foi produzido para que seja passado para a próxima etapa da produção. O software avalia a melhor rota levando em consideração as peças que precisam ser produzidas, as habilidades de cada funcionário e se os mesmos não estão alocados em outros processos de produção. Analisando esses fatos os software sabe a melhor forma para distribuição da produção e os funcionários tem suas tarefas distribuidas de forma correta para que não fiquem super alocados ou sem serviço.

2.2.1 Realização da Modelagem da base de dados

O software desenvolvido trabalha com a manipulação de dados cadastrados para gerar informações que serão utilizadas no processo de otimização de produção da fábrica sendo eles muito importantes para que se possa atingir o objetivo da aplicação.

O banco de dados armazena os dados dos costureiros e suas habilidades, esses dados são

importantes para que cada costureiro seja alocado de forma correta nas atividades necessárias para produção das calças. Desta forma, se sabe qual funcionário fabrica a parte da frente da calça, a parte de traz, o funcionário que coloca o zíper, que faz os bolsos, que junta as partes da peça, que faz a pintura e entre outras atividades necessárias.

A tabela processo tem como finalidade armazenar o modelo peça. Cada modelo pode possuir atividades diferentes para sua produção, com o cadastro do modelo da calça é possível separar as atividades de fabricação de cada modelo, ou seja, um modelo pode conter partes que outros modelos não possuem, cada modelo é representado por um processo.

No banco de dados também possui uma tabela chamada atividade onde são armazenados os processos e as habilidades dos costureiros necessários para produção de cada processo. Como há uma grande variedade de modelos e cortes das peças, os procedimentos para fabricação podem variar, alguns modelos podem ter procedimentos que em outros modelos não são necessários.

No processo de fabricação tem-se a necessidade de fabricar uma determinada parte da peça para que outra venha a ser fabricada até que se chegue no produto final que é a calça. A tabela atividade-ordem é responsável por organizar as atividades para que sejam executadas na ordem correta. Ela armazena as atividades

A figura Figura 4 ilustra o que foi descrito acima.

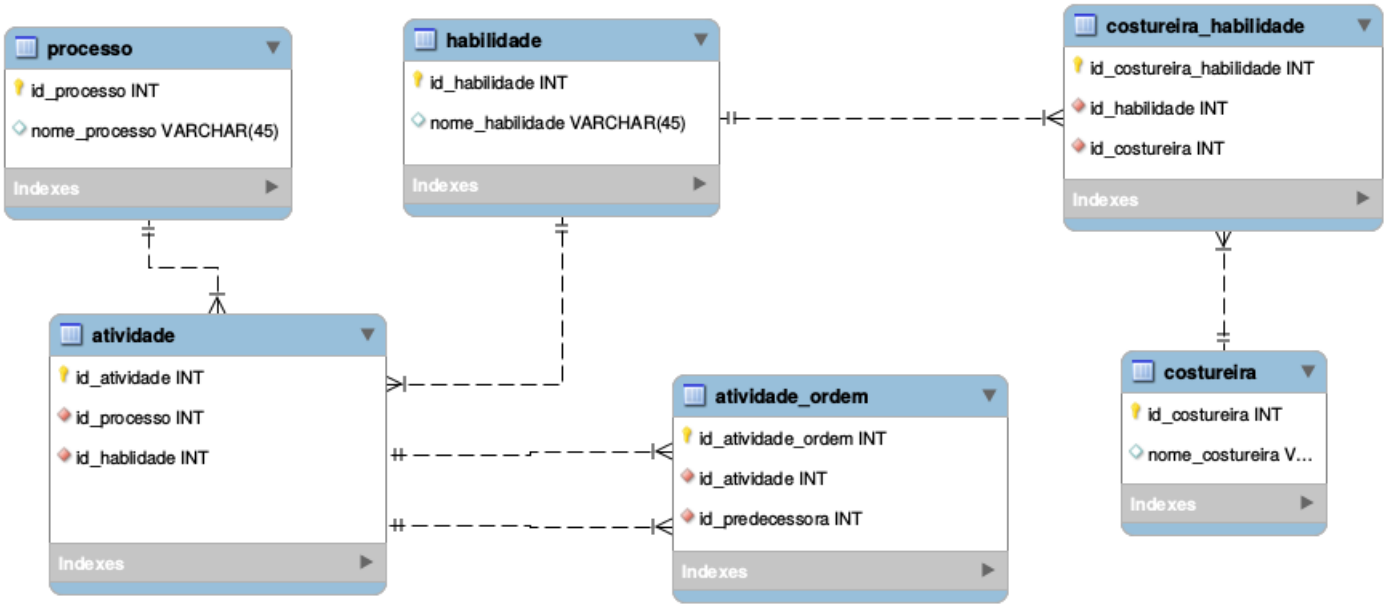

Figura 4 – Modelo do banco de dados **Fonte:** Desenvolvido pelos autores

2.2.2 Framework de desenvolvimento

Primeiramente é necessário ressaltar que, para o desenvolvimento da aplicação, foi utilizada uma base desenvolvida pelo professor Artur Barbosa durante as aulas de sistemas especialistas, do VII período do curso de sistemas de informação nesta universidade. Esta base também denominada *framework*, define regras a serem seguidas no desenvolvimento de cada elemento de um algorítimo genético. Este *framework* é definido dentro da seguinte estrutura:

- Classe `GAModel`:

  A classe `GAModel` é basicamente a classe mãe de todos os elementos de um algoritmo genético, ela representa o modelo que irá armazenar a população de indivíduos além de ser a classe que armazena os parâmetros que definem as configurações do algoritmo, tais como, tipo de cruzamento, tipo de mutação, tamanho da população etc.

  A classe contém os seguintes atributos:

  - *populationSize*: Este atributo define qual será o tamanho da população, ou seja, quantos indivíduos irão formar cada população;
  - *generationQuantity*: Como já explicado anteriormente, o processo de cruzamento e mutação se repete até que o número de indivíduos, definido no atributo anterior, seja atingido formando assim uma nova população e então, por sua vez, este processo de geração de novas populações se repete até que seja atingido um número de gerações definido pelo programador. Este atributo representa esta quantidade;
  - elitism: Atributo do tipo *boolean* que representa se o algoritmo vai ter a função de elitismo. Esta função, como já foi explicado anteriormente, quando está ativada (com valor *true*), no momento de começar a se criar uma nova população os dois melhores indivíduos da população que será substituída já começam a fazer parte da nova população, antes de começar o processo de cruzamento e mutação. Este mecanismo garante que a nova população terá pelo dois indivíduos iguais ao da antiga população, o que irá impedir que a nova população seja pior que a primeira;
  - foreignIndividualRate:
  - mutationRate: Como descrito no quadro teórico, a mutação é o fato de realizar pequenas alterações no indivíduo a fim de que este possa se tornar ainda melhor. Este parâmetro define uma porcentagem, geralmente baixa, que define quando o

indivíduo sofrerá mutação ou não. Esta questão ficará mais claras mais abaixo, quando será explicado o passo a passo da execução do algoritmo.

  - mutationQuantity: Caso a mutação for ocorrer para o indivíduo, a alteração aleatória será feita nos cromossomos. Este parâmetro define quantos cromossomos do indivíduo deve ser alterado pela mutação;
  - selectionType: Conforme descrito no quadro teórico, existem várias formas de seleção dos indivíduos para realizarem o cruzamento. Este parâmetro define qual será a forma escolhida pelo programador ao implementar o seu problema. No *framework* este parâmetro é do tipo `enum` e pode assumir 2 valores o *ROULETTE*, que representa o método roleta e o *CLASSIFICATION*, que representa o método de classificação;
  - crossType: Assim como a seleção, existe diversas formas de fazer o cruzamento dos indivíduos. Este atributo, também do tipo `enum` representa a forma de cruzamento e pode receber os valores *Binary*, *Permutation*, *Uniform* e *Aritmetic*;
  - mutationType: Segue as mesma forma que o selectionType e o crossType e pode assumir os valores *Permutation*, *Binary* e *Numerical*.

- Classe `Individual`:

  A classe abstrata `Individual`, encontra-se dentro do pacote edu.univas.edu.tcc.gacore e representa a estrutura básica de um indivíduo. A classe contém uma `lista` do tipo `Cromossomo`, que será descrito mais abaixo, que contém uma coleção de objetos que representam as características da solução.

  A Classe contém ainda um atributo chamado `valor` que irá armazenar a qualidade, ou seja, qual é o custo da solução representada pelo indivíduo, tal valor é recebido no retorno da operação calcularValor() descrita abaixo.

  Com relação as operações, além dos *getters and setters* e o construtor, que recebe a lista de cromossomos como parâmetro, a classe contém a operação abstrata `calcularValor()`, esta operação é quem realiza a função de avaliação, explicada anteriormente, que mede a qualidade do indivíduo. Desta forma, ao utilizar este *framework*, a classe que representa o indivíduo do problema deve herdar desta classe `Individual`. Fazendo isso tal classe passará a ter uma lista de cromossomos e o atributo que representa o seu valor e a classe obrigatoriamente terá que implementar a operação `calcularValor()`, já que esta

é abstrata na classe mãe, permitindo assim que o programador desenvolva a função de avaliação específica para o seu problema.

- Classe `Chromosome`:
- Classe `IndividualPair`:
- Classe `GAController`:

A Figura X representa o diagrama de classes que demonstra a estrutura em questão. (Desenhar diagrama de classe do framework no ASTAH)

Posterior a definição das regras de desenvolvimento, os próximos tópicos apresentam a implementação dos elementos, seguindo as definições do *framework*.

2.2.3 Definição do indivíduo e Cromossomos

O processo de definição indivíduo foi o primeiro passo do desenvolvimento da aplicação. Isso se deu devido ao fato de que o indivíduo é a parte crucial para que se possa definir a lógica a ser seguida para a definição da população inicial, o tipo de cruzamento a função de avaliação etc.

Em um primeiro momento, o modelamento do indivíduo e seus cromossomos foi feito da seguinte forma: Foi criado uma classe Java para representar o individuo denominada ProcessoIndividuo, esta classe é filha da classe Individual definida no framework.

todavia verificou-se que o algoritmo poderia apresentar problemas de performance. Um outro fator que influenciou para que se pensasse em uma nova abordagem foi o fato de somente parte da inteligência da distribuição das horas ficarem sob responsabilidade do algoritmo, assim, sob a orientação do professor Artur Barbosa (mudar nome), o indivíduo e seus cromossomos foram desenvolvidos como descrito abaixo. Esta nova forma também facilitou o processo de cruzamento e mutação que será explanado mais adiante.

2.3 Instrumentos

Segundo Faria (2015), instrumentos de pesquisa são a forma que os dados serão coletados para a realização do trabalho, podendo ser, dentre outras formas, por meio de reuniões,

questionários e entrevistas. Para este projeto utilizaremos os instrumentos descritos nas subsessões a seguir.

2.3.1 Entrevistas

Segundo Haguette (1997), entrevista é uma interação entre duas pessoas em que uma representa o entrevistador, que através de perguntas, obtêm informações por parte de outra pessoa que representa o entrevistado.

Será realizada uma entrevista com o dono da empresa de confecção com o objetivo de entender seu modelo de negócio para que então seja possível começar a fazer o levantamento dos requisitos do sistema. Para Pressman (2011, p.128), levantamento de requisitos de software consiste em

> perguntar ao cliente, aos usuários e aos demais interessados quais são os objetivos para o sistema ou produto, o que deve ser alcançado, como o sistema ou produto atenda às necessidades da empresa e, por fim, como o sistema ou produto deve ser utilizado no dia a dia.

2.3.2 Reuniões

De acordo com Carvalho (2012), reunião é o ajuntamento de pessoas para se tratar de um determinado assunto em que é necessário que se tenha conclusões sobre as questões que foram discutidas.

Durante o desenvolvimento do projeto poderão ser realizadas reuniões com o proprietário da fábrica de calças para saneamento de dúvidas, sugestões e outros assuntos que possam surgir.

2.4 Procedimentos

Esta sessão descreve os procedimentos a serem realizados na execução do projeto.

- Fazer uma prova de conceito com algoritmos genéticos para eliminar dúvidas cruciais sobre a viabilidade do projeto;

- Realizar a coleta dos requisitos com o gestor da empresa;

  Descrever sobre os requisitos

- Executar procedimentos referente à engenharia do software;

2.4.1 Realzação da Modelagem da base de dados

;

- Codificar o projeto; Problemas encontrados no desenvolvimento
- Realizar testes.

Realizando todos os passos descritos acima, teremos como resultado final o projeto finalizado.

REFERÊNCIAS

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

CARVALHO, L. *Aprenda algumas técnicas de reunião*. 2012. <http://www.administradores.com.br/artigos/negocios/aprenda-algumas-tecnicas-de-reuniao/62516/>. Acessado em 04 de abril.

DATE, C. J. *Introdução a sistemas de banco de dados*. 8º. ed. São Paulo: Campus, 2004.

DOUGLAS, K.; DOUGLAS, S. *PostgreSQL*: A comprehensive guide to building, programming, and administering postgresql databases. 1º. ed. EUA: Sams Publishing, 2003.

FARIA, J. de. *TCC I*. Pouso Alegre: Univás. Notas de Aula 26 de março: [s.n.], 2015.

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

FILITTO, D. Algoritimos genéticos: Uma visão explanatória. *Saber Acadêmico*, n. 06, p. 136–143, 2008.

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

GERHARDT, T. E.; SILVEIRA, D. T. *Métodos de Pesquisa.* 1º. ed. Porto Alegre: UFRGS, 2009.

GIL, A. C. *Métodos e técnicas de pesquisa social.* São Paulo: Atlas, 1999.

GOLDBERG, D. E. *Genetic Algorithms in search optimization and machine learning*. 1º. ed. New York: Addison-Wesley publishing company,inc., 1989.

HAGUETTE, T. M. F. *Metodologias qualitativas na Sociologia*. 5º. ed. Petrópolis: Vozes, 1997.

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.

JUNIOR, P. J. *P. de JAVA: Guia do Programador.* 1º. ed. [S.l.]: Novatec, 2007.

LACERDA, E. G. M. de; CARVALHO, A. C. P. L. F. de. *Introdução aos Algritmos Genéticos*. 2015. <http://www.leca.ufrn.br/~estefane/metaheuristicas/ag.pdf>. Acessado em 1 de Agosto.

LINDEN, R. *Algoritmos genéticos*. 1º. ed. Rio de Janeiro: Ciência Moderna, 2012.

LUCKNOW, D. H.; MELO, A. A. de. *Programação Java para Web*. 1º. ed. São Paulo: Novatec, 2010.

MANZONI, A. Desenvolvimento de um sistema computacional orientado a objetos para sistemas elÉtricos de potÊncia: AplicaÇÃo a simulaÇÃo rÁpida e anÁlise da estabilidade de tensÃo. In: . [S.l.: s.n.], 2005.

MELANIE, M. *An introduction to genetic algorithms*. 1º. ed. London: Massachusetts Institute of Technology, 1999.

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

ORACLE. *O que é Java?* 2015. <https://www.java.com/pt_BR/download/whatis_java.jsp>. Acessado em 12 de Fevereiro.

PRESSMAN, R. *Engenharia de Software*. McGraw Hill Brasil, 2011. ISBN 9788580550443. Disponível em: <http://books.google.com.br/books?id=y0rH9wuXe68C>.

PRICE, J. *Oracle Database 11g SQL*: Domine sql e pl/sql no banco de dados oracle. 1º. ed. Porto Alegre: bookman, 2008.

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

SANTOS, R. *Introdução à Programação Orientada a Objetos usando Java*. 3º. ed. Rio de Janeiro: Campus, 2003.

SCHILDT, H. *The complete reference Java*. 7º. ed. Nova York: MC Graw Hill Osborne, 2007.

VUKOTIC, A.; GOODWILL, J. *Apache Tomcat 7*. 1º. ed. Nova York: Apress, 2011.

ZANELLA, L. C. H. *Metodologia de Estudo e de Pesquisa em Administração.* 1º. ed. Florianópolis: CAPES, 2009.