PROPOSTA DE MODELAGEM DE UM SISTEMA DE PREVISÃO DE DEMANDA EM LINGUAGEM DE PROGRAMAÇÃO JAVA

Vanessa Elionara Souza Ferreira (UFERSA)
vanessaelionara@gmail.com
Monaliza Ferreira Rodrigues de Paula (UFERSA)
monaliza-ferreira@hotmail.com
Breno Barros Telles do Carmo (UFERSA)
brenotelles@hotmail.com
Antonia Claudenice Pinheiro de Almeida (UFERSA)
claudenice_sanus@hotmail.com

A fim de se prevenir das constantes variações do mercado, as previsões de demanda se apresentam de forma fundamental para facilitar na determinação dos recursos de uma organização, tanto financeiros quanto materiais e humanos, servindo de bbase para o desenvolvimento de uma cadeia de abastecimento consolidada, permitindo uma melhor integração de todo o processo produtivo. Com essa finalidade o estudo apresenta a implementação de um programa criado em linguagem de programação Java, o qual prevê a demanda por produtos, através de dados históricos para os quatro modelos quantitativos de previsão mais utilizados: média móvel simples, tendencial, sazonal e sazonal com tendência. Tal modelo proporciona uma aplicação prática em empresas de manufatura, pois gera as previsões, com seus valores e respectivos gráficos, a partir do lançamento dos dados no programa.

Palavras-chaves: Previsão, modelagem, java

1. Introdução

Toda empresa precisa se prevenir a fim de saber quanto e quando comprar material ou contratar funcionários, bem como quantificar sua capacidade produtiva, evitando com isso desperdícios de tempo, material, mão de obra e consequentemente, dinheiro.

Com essa finalidade, os modelos de previsão de demanda surgem no mercado como uma ferramenta para os gestores de produção. Concordando com isso, Stevenson (2001) entende que as previsões ajudam os gerentes a reduzir parte das incertezas permitindo-lhes desenvolver planos mais realistas.

De acordo com Slack (2009), existem diversos modelos de previsão, se subdividindo em qualitativos e quantitativos, porém, o referente trabalho se limitou aos modelos quantitativos mais conhecidos no mercado, o modelo de média móvel simples, tendencial simples, sazonal simples e sazonal com tendência.

O estudo objetiva a implementação de um programa criado em linguagem de programação orientada ao objeto, Java, o qual prevê demanda para os quatro tipos de previsão citados através de dados históricos.

2. Previsão de Demanda e seus Modelos

Previsões são estimativas de como se vai comportar o mercado demandante no futuro, são especulações sobre o potencial de compra do mercado (CORRÊA, 2009). Sabendo que demanda é a aspiração do cliente em consumir, segundo Dias (2004), existindo diferença entre fornecimento e demanda, considerando o tempo entre o início da produção e a disponibilização do produto para os consumidores, as empresas devem fazer uso da previsão de demanda a fim de antecipar o comportamento do mercado, o que possibilita aos consumidores a obtenção dos produtos no tempo desejável.

A previsão de demanda surge como uma tentativa de impedir erros relacionados à produção, acelerando o processo de decisão e deixando-o mais seguro. Moreira (2011) considera que a

previsão de demanda é um processo racional de busca de informações a cerca do valor das vendas futuras de um item ou de um conjunto de itens. E de acordo com Werner e Ribeiro (2003) prever a demanda torna-se essencial nas aplicações industriais pelo fato de revelar as tendências do mercado, colaborando assim, com o planejamento estratégico da empresa. O que é dito também por Stevenson (2001), o qual fala que as previsões reduzem parte das incertezas, o que permite ao gerente desenvolver planos mais próximos da realidade.

Segundo Slack (2009) existe dois métodos principais para a previsão, que são os qualitativos (baseiam-se em opiniões ou experiências) e quantitativos (baseiam-se os dados mais precisos e podem ser usadas para modelar dados). E existem alguns requisitos para a escolha do método de previsão a ser adotado. De acordo com Tubino (2009), um dos fatores que merecem destaque na escolha do método de previsão é os dados históricos. O mesmo autor define conceitos para os métodos de previsão mais conhecidos, que são o tendencial, sazonal simples e sazonal com tendência.

Para Tubino (2009), a média móvel simples usa dados de um número predeterminado de períodos, normalmente os mais recentes, para realizar a previsão e consiste na média aritmética dos n últimos períodos da demanda observada. Tendência refere-se ao movimento gradual de longo prazo da demanda e o cálculo da estimativa é realizado pela identificação da equação que descreva uma reta. A sazonalidade caracteriza-se pela ocorrência de variações, para cima ou para baixo, a intervalos regulares nas séries temporais da demanda e a forma mais simples de considerar a sazonalidade nas previsões da demanda consiste em empregar o último dado da demanda, no período sazonal em questão, e assumi-lo como previsão. Vale a ressalva de que forma mais usual de inclusão da sazonalidade nas previsões consiste em obter o índice de sazonalidade para os diversos períodos e aplicá-lo para o valor médio previsto para o período em questão, e o mesmo é obtido dividindo-se o valor da demanda no período pela média móvel centrada neste período. Tubino (2009) ainda corrobora que caso a demanda apresente sazonalidade e tendência, há necessidade de incorporar estas duas características no modelo de previsão.

3. Linguagem de Programação Java

Para Santos (2010), programação é um assunto abrangente, e linguagens de programação costumam ser razoavelmente complexas, e apesar disso, as ideias e conceitos usados nas linguagens de programação são simples. Nesse contexto, o Java se torna uma das principais linguagens de programação porque, de acordo com Deitel (2005), o Java é utilizado para desenvolver aplicativos de grande porte, aprimorar a funcionalidade dos servidores WEB (os computadores que fornecem o conteúdo que vemos em nossos navegadores da Web), fornecer aplicativos para dispositivos voltados para o consumo popular (celulares, pagers, etc.) e para outros propósitos diversos.

Segundo Bandeira (2010), Java é uma linguagem de programação que chama bastante atenção por ser portável para vários sistemas operacionais e hardwares.

Caelum (2010), tecnologia Java nasceu comum objetivo em mente, foi lançado com outro, mas, no final, decolou mesmo no desenvolvimento de aplicações do lado do servidor. Em 2011, a Oracle comprou a Sun, fortalecendo a marca. A Oracle sempre foi, junto com a IBM, uma das empresas que mais investiram e • zeram negócios através do uso da plataforma Java. No mesmo ano surge a versão Java 7 com algumas pequenas mudanças na linguagem.

Java é uma linguagem orientada a objetos e conforme Santos (2010), objeto é o elemento básico de sistema orientado a objetos, e é usado para representar uma entidade que aparece no domínio de um problema (BARNES; KOLLING, 2003, apud Santos, 2010).

Caelum (2010) e Deitel (2005) falam em suas obras que os objetos são criados dentro de uma classe, pois eles são instâncias das mesmas. Sendo assim, só podemos criar um objeto se antes criamos uma classe, pois toda estrutura de programas em Java se desenvolve dentro de uma classe. Ainda sob a ótica dos mesmos autores, podemos definir classe como um conjunto de dados (campos) e procedimentos (métodos) que executam esses dados.

Com uma linguagem orientada a objetos e madura como o Java, é extremamente mais fácil e rápido fazer alterações no sistema, desde que siga as boas práticas e recomendações sobre design orientado a objetos. (CAELUM, 2010).

4. Metodologia utilizada

Para a elaboração do sistema primeiramente foi realizada uma pesquisa bibliográfica que, segundo Gil (2002), é aquela realizada a partir de material já elaborado, principalmente a partir de livros e artigos. O material estudado abordava fundamentalmente temas acerca de métodos quantitativos de previsão de demanda, com a finalidade de se obter o conhecimento quanto à modelagem matemática de cada método de previsão para posteriormente transformá-los em linguagem de programação orientada a objetos.

O processo de desenvolvimento do programa se deu a partir da utilização de quatro modelos quantitativos de previsão de demanda baseados em séries temporais propostos por Tubino (2009), os quais foram:

a) Média móvel simples: A partir da demanda real de um espaço de tempo é estimada a demanda Média Móvel empregando-se diferentes números de períodos para o seu cálculo (Equação 1). Partindo-se do início da série é selecionada uma determinada quantidade de períodos e calculada a sua média, à medida que se adiciona um dado mais recente, descarta-se o mais antigo. Após a obtenção de todas essas médias é possível visualizar a previsão de demanda para os próximos períodos.

Equação 1 – Média Móvel Simples

$$Mm_n = \frac{\sum_{i=1}^{n} D_i}{n}$$

Fonte: Tubino (2009)

Legenda:
Mm_n = média móvel de *n* períodos
D_i = demanda ocorrida no período *i*;
n = número de períodos;
i = índice do período (*i* = 1, 2, 3...).

b) Sazonalidade simples: Após a análise do gráfico dos dados históricos, são identificados os ciclos de sazonalidade, a partir do intervalo médio de tempo entre dois vales ou dois picos de demanda. Em seguida, utilizando-se o método de ajustamento sazonal por Média Móvel Centrada (MMC), é calculada a média móvel para cada ciclo de

demanda, calculando a média do mesmo da mesma forma que é calculada a Média Móvel Simples (Equação 2). Caso a quantidade de período obtidos na média móvel centrada seja par é calculada o MMC de dois a dois elementos.

Equação 2 – Média Móvel Centrada

$$Mm_n = \frac{\sum_{i=1}^{n} D_i}{n}$$

Fonte: Tubino (2009)

Após a obtenção da MMC são calculados os índices de sazonalidade através da razão entre o dado histórico e cada MMC do seu período correspondente (Equação 3). Em seguida é calculada a média dos índices de sazonalidade pertencentes ao mesmo período de cada ciclo (Equação 4). Por fim, a previsão para o período subsequente é obtida através da aplicação dos índices de sazonalidade médios sobre a demanda média das Médias Móveis Centradas.

Equação 3 – Índice de Sazonalidade

$$IS_n = \frac{D_n}{MMC_n}$$

Fonte: Baseado nas informações do Tubino (2009)

Equação 4 – Índice de Sazonalidade Médio

$$ISM_n = \frac{IS_n + IS_{n+p} + IS_{n+2p} \ldots + D_{n+xp}}{x+1}$$

Fonte: Baseado nas informações do Tubino (2009)

Legenda:

IS_n = Índice de Sazonalidade

MMC_n = Média Móvel Centrada do período

D_n= Demanda do período

ISM_n = Índice de Sazonalidade Médio

n = Período do Ciclo

p = Tamanho do Ciclo

x+1 = Número de elementos somados

c) Tendência simples: A previsão tendencial da demanda é gerada por meio da aplicação da equação linear da reta (Equação 5), a qual é obtida através da série histórica de demanda.

Equação 5 – Obtenção da Equação Linear da Reta

$$b = \frac{n(\sum XY) - (\sum X).(\sum Y)}{n(\sum X^2) - (\sum X)^2}$$

$$a = \frac{(\sum Y) - b(\sum X)}{n}$$

$$Y = a + bX$$

Fonte: Tubino (2009)

Legenda:

Y = previsão de demanda par a o período *X*;

X = período da previsão;

a = ordenada de origem;

b = coeficiente angular.

d) Sazonalidade com tendência: Para a elaboração desse modelo primeiramente é realizado parte do processo de modelagem da Sazonalidade Simples, obtendo-se a Média Móvel Centrada, os Índices de Sazonalidade e os Índices de Sazonalidade Médios (Equação 2, 3 e 4). Em seguida é retirada a sazonalidade da série de dados históricos dividindo-se a demanda do período pelo seu Índice de Sazonalidade Médio correspondente. Posteriormente, a partir da demanda do período, livre dos ciclos de sazonalidade, é obtida a Equação Linear da Reta (Equação 5) para verificar a tendência dos dados. A partir da Equação da Tendência obtida é calculada a previsão tendencial dos dados históricos para o próximo período. Por fim, são aplicados os índices de sazonalidade sobre a previsão, multiplicando-se a demanda tendencial prevista de cada período por seu Índice de Sazonalidade Médio correspondente.

Após o entendimento da modelagem matemática dos métodos de previsão de demanda propostos por Tubino (2009), os modelos foram implementados na linguagem de programação JAVA, utilizando como ambiente de desenvolvimento a Plataforma Eclipse, que segundo Aniszczyk e Gallardo (2012) é um software livre desenvolvido pela IBM, que possui

um conjunto padrão de plug-ins, incluindo as Ferramentas de Desenvolvimento Java (JDT). Para a elaboração dos gráficos foi utilizada uma biblioteca gratuita chamada JFreeChart que, segundo Santos (2011), fornece um extenso conjunto de componentes que possibilitam a criação de diversos tipos de gráficos a partir dos dados fornecidos.

4. Aplicação da metodologia e resultados

Para a implementação do sistema foram criadas duas classes, a PrevisaoDeDemanda, na qual estão todos os métodos utilizados para a realização das atividades do programa, e a classe Execucao, que funciona como o ambiente de execução para os objetos da classe anterior.

4.1. PrevisaoDeDemanda

Para a implementação da classe PrevisaoDeDemanda foram importadas bibliotecas disponíveis no Eclipse, para leitura de dados e criação de Frames, e a biblioteca JFreeChart para criação dos gráficos. E foram criados os campos listados na Figura 1.

Figura 1 – PrevisaoDeDemanda

*PrevisaoDeDemanda.java | Execucao.java

```
import java.util.*;
import javax.swing.JFrame;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartFrame;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;



public class PrevisaoDeDemanda extends JFrame {


    // --------Campos---------------
    private double previsao [];
    private String meses[]= {"Janeiro", "Fevereiro","Março","Abril","Maio","Junho","Julho","Agosto", "Setembro", "Outubro", "Novembro", "Dezembro"};
    private Scanner ler = new Scanner(System.in);
    private XYSeries dados = new XYSeries("Dados históricos"); //Iniciado o objeto que adcionará valores ao gráfico
    private double [] [] valores;
    private double a, b, n, demandaMedia;
    private double somaX = 0;
    private double somaY=0;
    private double somaXY=0;
    private double somaX2=0;
    private double somaY2=0;
    private double r=0;
    private double [] indsazonalidade;
    private double [] indsazonalidademedio;
    private int numCiclos, periodo, numPrevisto, qntanos;
    // --------Métodos---------------

//Método construtor com dados pré-estabelecido
    public PrevisaoDeDemanda (int opcao){
        double [] vetorTemp1 = new double[0];
```

Fonte: Autoria Própria

Após a criação dos campos e importação das bibliotecas foram criados dois construtores, um sem parâmetro e um com parâmetro.

O construtor sem parâmetro partiu do pressuposto que o usuário, após informar a quantidade de anos dos seus dados históricos, entraria com os dados das demandas mensais de sua produção (Figura 2). Os dados seriam armazenados em uma matriz na sequência que fossem digitados e adicionados, através de um objeto pertencente a uma das classes do JFreeChart, a uma espécie de memória secundária por meio do método *add(x,y)* em forma de ordenada, onde sua posição X representa sua ordem no montante e sua posição Y representa a demanda digitada. Após o fim do laço, que constrói a matriz, é gerado um gráfico com todos os dados históricos cujo código pode ser visualizado na Figura 3.

Figura 2 - Construtor sem Parâmetro

```
    //-----------Recebimento de dados e geração de gráfico ----------------------
public PrevisaoDeDemanda (){
    /*Partindo-se da ideia de que o usuário digitará conjuntos de dados anuais, ou seja, digitara os dados mensais da serie historica
    System.out.println("Digite a quantidade de anos dos seus dados historicos");
    qntanos = ler.nextInt();

    valores = new double [qntanos][12]; //Matriz -> linhas representam os anos e colunas representam os meses
    int cont = 1;
    for(int i=0; i<qntanos; i++){ //Lançamento dos dados na matriz
        for(int j=0; j<12; j++){
            System.out.println("Digite a demanda do mes de "+ meses[j]+" do ano "+(i+1)+":");
            valores[i][j] = ler.nextDouble();
            dados.add((cont), valores[i][j]);
            cont++;
        }
    }
    this.gerarGrafico();
}
```

Fonte: Autoria Própria

Foi implementado um construtor com parâmetro para facilitar a visualização do funcionamento do programa. De acordo com o valor da opção digitada pelo usuário, durante a execução do programa, seria chamado um dos vetores pré-determinados e em seguida adicionados a matriz principal, sendo que as demais implementações desse construtor são semelhantes as do método construtor sem parâmetro.

Figura 4 - Construtor com Parâmetro

```
public PrevisaoDeDemanda (int opcao){
    double [] vetorTemp1 = new double[0];
    if(opcao==1){
    double [] teste1 = {1083, 1460, 2109, 2717, 2801, 2503, 2381, 3154, 3969, 4642, 4892, 4338, 3742, 4839, 5805,
            6747, 6880, 5683, 5487, 6194, 7642, 8821, 8469, 7139};
    vetorTemp1=teste1;
    }else{
        if(opcao==2){
            double [] teste2 = {3600, 3416, 2682, 2250, 2107, 2352, 2841, 3322, 3720, 3468, 3349, 2745, 2254, 2086,
                    2400, 2850, 3344, 3564, 3576, 3360, 2745, 2325, 1960, 2400};
            vetorTemp1=teste2;
        }else{
            if(opcao==3){
                double [] teste3 ={2373, 1664, 2365, 1891, 1731, 2441, 2088, 2467, 2321, 2460, 1478, 2915, 2662, 1882,
                        1922, 1928, 1594, 2020, 2150, 2635, 2564, 2200, 2445, 3280, 2122, 1983, 2291, 2162, 1969, 1845,
                        2432, 2519, 2669, 2667, 1868, 4540};
                vetorTemp1=teste3;
            }else{
                if(opcao==4){
                    double [] teste4 ={501.5, 645.5, 689.5, 521, 545.5, 411, 824.5, 744, 661, 1121, 1350, 1341, 1096.5,
                            810, 1045, 923, 930.5, 1106, 1160, 1030.5, 927, 887, 923, 849, 1083, 976.5, 758, 1088.5, 909,
                            1113, 990, 1031.5, 1429, 1319, 1573, 1268};
                    vetorTemp1=teste4;
                }else{
                System.out.println("Digite um valor válido");
                }}}}
    qntanos = (vetorTemp1.length/12);
    n=qntanos;
    valores = new double [qntanos][12]; //Matriz -> linhas representam os anos e colunas representam os meses
    int cont = 1;
    int cont2 = 0;
    for(int i=0; i<qntanos; i++){ //Lançamento dos dados na matriz
        for(int j=0; j<12; j++){
            valores[i][j] = vetorTemp1[cont2];
            dados.add((cont), valores[i][j]);
            cont++;
            cont2++;
        }}          this.gerarGrafico();        }
```

Fonte: Autoria Própria

Após a entrada dos dados realizada pelo usuário ou escolha da opção de dados já pré-estabelecido, como já mencionado, será gerado um gráfico onde poderá ser visualizado o comportamento dos dados históricos. Partindo-se da ideia de que o usuário, baseando-se na analise do gráfico gerado, tem conhecimento para escolher qual o melhor método de previsão a ser a ser utilizado, foi elaborado o método *escolhaModelo()*. De acordo com a opção escolhida pelo o usuário esse método retornará diferentes questionamentos quanto a informações necessárias para a utilização dos métodos de previsão de demanda (Figura 5).

Figura 5 - Escolha do Modelo de Previsão

```
// -----------------Escolha do Modelo-------------------
public void escolhaModelo(){
    System.out.println("Digite o valor do modelo de previsão deseja aplicar a seus dados:");
    System.out.println("1 - Média Móvel Simples \n 2 - Sazonalidade Simples \n 3 - Tendência Simples \n 4-Tendência com sazonalidade");
    int op = ler.nextInt();
    switch(op){
    case 1: // MMS
        System.out.println("Quantos períodos devem ser considerados para realizar a previsão?");
        int c = ler.nextInt();
        this.gerarMediaMovelSimples(c);
        this.mostrarPrevisaoMediaMovelSimples();
        break;
    case 2: //Sazonalidade
        System.out.println("O ciclo de demanda do gráfico gerado varia, em média, de quanto em quantos meses?");
        int d = ler.nextInt();
        this.gerarSazonalidadeSimples(d);
        this.mostrarSazonalidadeSimples();
        break;
    case 3: // Tendência
        this.gerarPrevisaoTend();
        this.mostrarPrevisaoTendencia();
        break;
    case 4: // Sazonalidade com Tendência
        System.out.println("O ciclo de demanda do gráfico gerado varia, em média, de quanto em quantos meses?");
        int f = ler.nextInt();
        this.gerarSazonalidadeComTendencia(f);
        this.mostrarSazonalidadeComTendencia();
        break;
    default:
        System.out.println("A opção digitada não é válida");
        break;

    }
}
```

Fonte: Autoria Própria

Após a escolha do modelo são chamados seus métodos correspondentes. Cada modelo é composto por 2 métodos, um para gerar a previsão e um que mostra na tela a previsão gerada. Foram implementados os seguintes métodos de previsão: Média Móvel Simples (Figura 6 e 7), Tendência Simples (Figura 8 e 9), Sazonalidade Simples e Sazonalidade com Tendência. A previsão de demanda proposta foi limitada a um único período subsequente.

Figura 6 - Gerar Média Móvel Simples

```
// ---------------------- MMS ------------------------
public void gerarMediaMovelSimples(int n){

    double soma;

    if(valores.length>0){ //Se a pessoa tiver entrado com algum valor
        double [] vetorTemporario = new double [qntanos*12];
        int cont = 0;
        for (int a=0; a<qntanos; a++){ //Colocar todos os dados em um vetor inteiro
            for (int b=0; b<12; b++){
                vetorTemporario[cont] = valores[a][b];
                cont++;
            }
        }
        periodo = (qntanos*12)-n+1;
        //Ex.: Uma lista com 5 dados em intervalo de 3 itens terá uma previsão para os proximos 3 períodos
        previsao = new double[periodo]; //Vetor da previsão
        if(periodo>0){ //se der menor ou igual a zero não tem como prever
            System.out.println("A previsão para os próximos "+periodo+" períodos é:");
            int p =n;
            int ac = 0; //acrescimo
            for (int i=0; i<previsao.length; i++){
                soma = 0;
                for (int j=ac; j<p; j++){
                    soma += vetorTemporario[j]; //soma dos valores da lista até o periodo p
                }
                previsao[i]=(soma/n);
                ac++;//+1, retirando o dado mais antigo
                p++;//até mais um espaço
                System.out.println(previsao[i]);
            }
        }else{
            System.out.println("Não foram adcionados valores suficiente para gerar a média móvel simples com esse intervalo");
        }   }   }
```

Fonte: Autoria Própria

Figura 7 - Mostrar Média Móvel Simples

```
public void mostrarPrevisaoMediaMovelSimples(){
    System.out.println("A previsão para os próximos "+periodo+" períodos:");
    int cont = (qntanos*12)+1;
    for (int i=0; i<previsao.length; i++){
        System.out.println(previsao[i]);
        dados.add((cont), previsao[i]);
        cont++;
    }
    this.gerarGrafico();
}
```

Fonte: Autoria Própria

Figura 8 - Gerar Tendência Simples

```
//------------------- Tendência Simples ----------------------
public void gerarPrevisaoTend(){
    //O código está preparado para realizar a previsão de mais de um ano, mas como declarado acima só consideraremos a previsão do ano seguinte
    int m=1;
    this.numPrevisto = m;
    int cont = 1;
    for (int i =0; i<qntanos;i++){
        for (int j =0; j<12;j++){
        somaX = somaX+cont; // cont é o X
        somaY = somaY + valores[i][j]; //valores[i][j] é o Y
        somaXY = somaXY + (cont*valores[i][j]);
        somaX2 = somaX2+ (cont*cont);
        somaY2 = somaY2+ (valores[i][j]*valores[i][j]);
        cont++; // Esse contador começa de 1 e vai até qntanos*120, é uma variável auxiliar para realizar a somatória do número de períodos.
        }
    }
    n = qntanos*12;
    b = ((n*somaXY)-(somaX*somaY))/((n*somaX2)-(somaX*somaX));
    a = (somaY - b*somaX)/n;

    previsao = new double[m*12];
    System.out.println("A previsão para os próximos "+m+" anos é:"); //Gerando a previsão tendencial
    if(b!=0){
        cont = (qntanos*12)+1; ///COMEÇA A CONTAR DO PRIMEIRO PERÍODO APÓS O ÚLTIMO ANO(EX.: Caso se tenha 24 períodos de dados históricos,
        //o contador da previsão começa em 25...
        int cont2 = 0;
        for (int z =0; z<m;z++){
            for (int x =0; x<12;x++){
            previsao[cont2]=(a+b*cont);
            cont++;
            cont2++;
            }
        }
    }else{
    System.out.println("Não foram adcionados valores para geração da equação da reta");
    }
}
```

Fonte: Autoria Própria

Figura 9 - Mostrar Previsão Tendencial

```
public void mostrarPrevisaoTendencia(){
    int cont =0;
    int cont2 =(qntanos*12)+1;
    for (int i =0; i<this.numPrevisto;i++){
        for (int j =0; j<12;j++){
            System.out.println("Ano "+(i+1)+" - "+meses[j]+" - "+previsao[cont]); //Código preparado para a previsão de mais de um ano
            dados.add((cont2), previsao[cont]); //Gerar gráfico com a previsão adcionada
            cont++;
            cont2++;
        }
        System.out.println("---------------");
    }
    this.gerarGrafico();
}
```

Fonte: Autoria Própria

4.2. Execucao

Na classe execução foi criado um objeto do tipo PrevisaoDeDemanda utilizando-se o construtor com parâmetro. De acordo com a opção escolhida pelo usuário é chamado o vetor de dados com os dados pré-estabelecidos, a opção é lida através do objeto *ler* da classe Scanner (Figura 10).

Figura 10 - Classe Execucao

PrevisaoDeDemanda.java | Execucao.java

```
import java.util.Scanner;
public class Execucao {


    public static void main(String[] args) {

        Scanner ler = new Scanner(System.in);

        System.out.println("Digite qual conjunto de dados deseja acessar:");
        System.out.println("1 - Dados sazonais/tendenciais com 2 anos de coleta");
        System.out.println("2 - Dados sazonais com 2 anos de coleta");
        System.out.println("3 - Dados aleatórios com 3 anos de coleta");
        System.out.println("4 - Dados tendenciais com 3 anos de coleta");
        try{
        int n = (int) ler.nextInt();
        PrevisaoDeDemanda d = new PrevisaoDeDemanda(n);
        d.escolhaModelo();
        }catch(Exception e){
            System.out.println("Ocorreu um erro durante a execução do programa");
        }

    }

}
```

Fonte: Autoria Própria

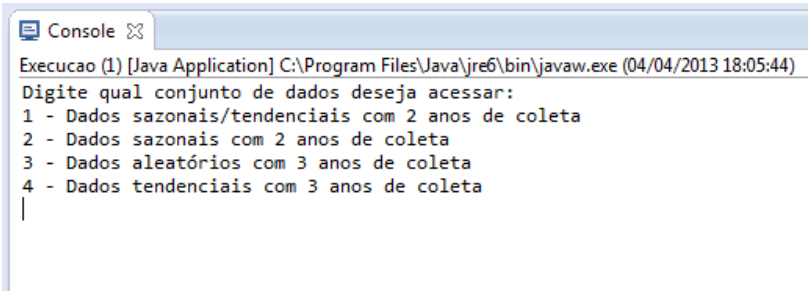
Foi utilizado o construtor com parâmetro para a melhor visualização do sistema. Utilizando o objeto da classe PrevisaoDeDemanda é chamado o método *escolhaModelo()*. Foram utilizados os comandos *try-catch*, de manipulação de erros, para garantir a execução do programa, caso o usuário entre com valores diferentes do esperado no programa.

4.3. Visualização da Execução do Programa

Para ilustrar a forma de execução do programa será mostrado abaixo o passo a passo do mesmo utilizando do Método de Previsão de Sazonalidade com Tendência.

a) Primeiramente aparece na tela as opções de demanda pré-estabelecidas a serem escolhidas pelo usuário.

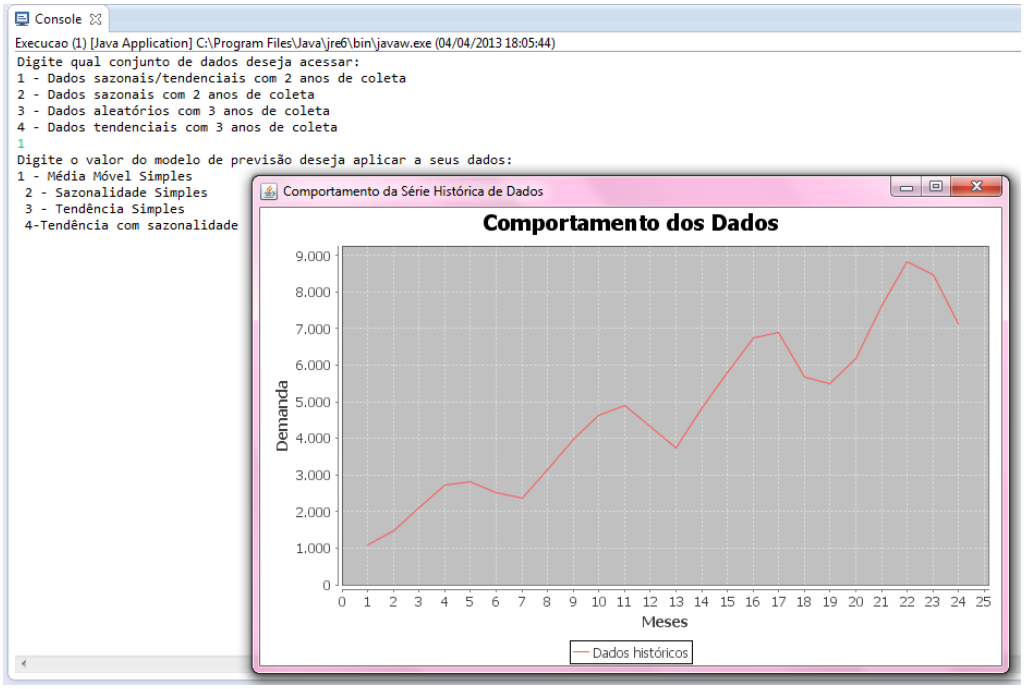
Figura 11 - Primeira Etapa



Fonte: Autoria Própria

b) Após o usuário escolher qual série histórica deseja visualizar o comportamento é gerado um gráfico da mesma.

Figura 12 - Segunda Etapa

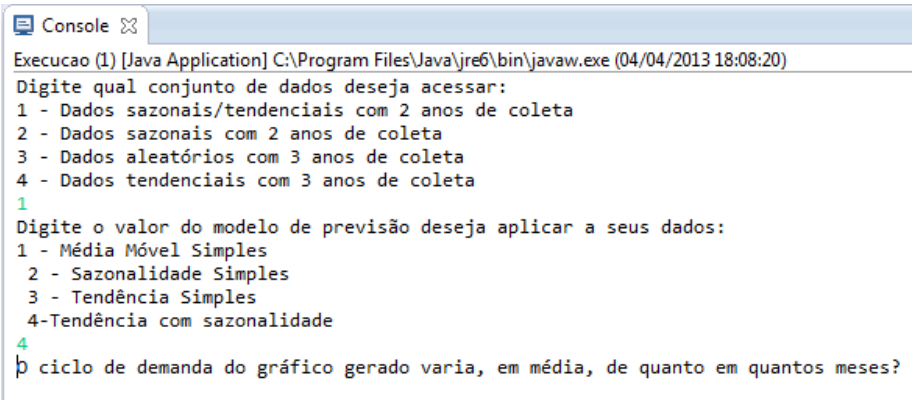


Fonte: Autoria Própria

c) Após a análise do gráfico gerado o usuário deverá escolher qual a forma de previsão de demanda que mais se adéqua aos seus dados. De acordo com o modelo escolhido aparece na tela um questionamento quanto a informação necessária para execução da

previsão. No caso, como a previsão escolhida foi a de Sazonalidade com Tendência, é solicitado ao usuário a informação quanto ao intervalo de meses existentes entre um período e outro.

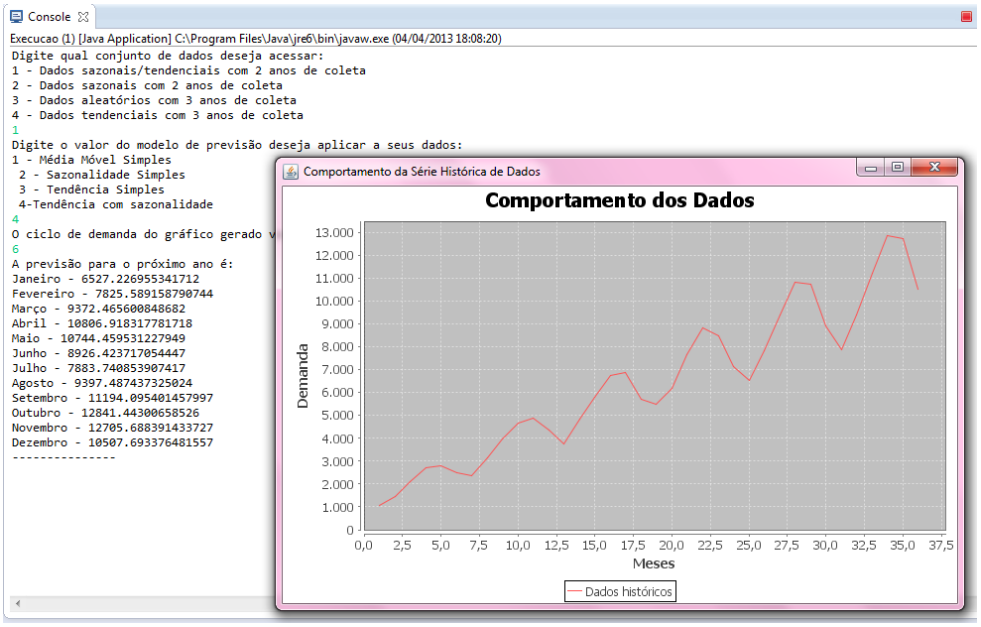
Figura 13 - Terceira Etapa



Fonte: Autoria Própria

d) Por fim é gerada a previsão para o próximo período, no caso o próximo ano, e um novo gráfico com a previsão adicionada à série histórica.

Figura 14 - Quarta Etapa



Fonte: Autoria Própria

5. Conclusão

Diante do exposto, percebeu-se que o objetivo do trabalho foi alcançado, uma vez que foi utilizada uma linguagem de programação orientada ao objeto, o Java, para a implementação de um modelo de previsão de demanda.

Verificou-se que o programa obteve êxito, o que caracteriza seu bom desempenho, tendo em vista que, a partir do fornecimento de dados históricos, as previsões são calculadas e os referentes gráficos são gerados. Os resultados são mostrados no console e os gráficos são mostrados em uma janela na tela para o usuário.

Dessa forma, de acordo com os dados do usuário, o mesmo analisa os gráficos gerados e, a partir de tal análise, escolhe qual modelo se adéqua melhor, por conseguinte, o programa gera a previsão da demanda. Isso mostra que o programa se apresenta com alto benefício, conveniência e eficiência para os que o utilizam, podendo ser utilizado em empresas de pequeno, médio e grande porte.

REFERÊNCIAS

ANISZCZYK, C.; GALLARDO, D. (2012). **Introdução à Plataforma Eclipse.** Disponível em: <http://www.ibm.com/developerworks/br/library/os-eclipse-platform/> Acesso em: 29 mar. 2013.

BANDEIRA, Thiago F. L. **Um Guia para o Desenvolvimento de Aplicações Web Usando Java.** Trabalho de Conclusão de Curso em Bacharelado em Ciência da Computação - Universidade Federal Rural do Semi-Árido, Mossoró, 2010.

CAELUM, 2010. **Java e Orientação a Objetos.** Apostila.

CORREA, H. L.; CORRÊA, C. A. **Administração de Produção e de Operações.** 1 ed. São Paulo: Atlas, 2009.

DEITEL, H. M.; DEITEL, P. J. **Java. Como Programar**. 2005.

DIAS, A. S. **Uso de conhecimentos teóricos e de especialista para Previsão de Demanda.** São Carlos, 2004. 181p. Dissertação (Engenharia de Produção) – Universidade Federal de São Carlos.

GARDIN, W. Disponível em: <http://wolmirgarbin.wordpress.com/2011/05/07/jfreechart-graficos-em-desktop/> Acesso em 05 ma. 2013.

GIL, Antônio Carlos. **Como elaborar projetos de pesquisas.** São Paulo: Atlas, 2002. 176p.

JFREE CHART. Disponível em: <http://www.jfree.org/jfreechart/> Acesso em: 05 mar. 2013.

LUSTOSA, L. et al. **Planejamento e Controle da Produção.** São Paulo: Elsevier, 2008.

MOREIRA, Daniel A. **Administração da Produção e Operações.** São Paulo: Cengage Learning, 2011.

SANTOS, Sátiro, R. de. **Desenvolvimento de um Sistema Computacional com Conteúdo Educacional para a Disciplina Programação Orientada a Objetos.** Trabalho de Conclusão de Curso em Bacharelado em Ciência da Computação - Universidade Federal Rural do Semi-Árido, Mossoró, 2010.

SANTOS, S. T. C. (2012). **Criando Gráficos com o JFreeChart.** Disponível em: <http://stthiaggo.blogspot.com.br/2011/12/criando-graficos-com-o-jfreechart.html/> Acesso em: 29 mar. 2013.

SLACK, Nigel et al. **Administração da produção.** São Paulo: Atlas, 2009.

STEVENSON, W. J. **Administração das Operações de Produção.** LTC. 6ª edição. Rio de Janeiro, 2001.

TUBINO, Dalvio F. **Planejamento e Controle da Produção – Teoria e Prática.** São Paulo: Atlas, 2009.

WERNER, L.; RIBEIRO, J. L. D. **Previsão de demanda: uma aplicação dos modelos Box-Jenkins na área de assistência técnica de computadores pessoais.** Revista Gestão & Produção, v. 10, n. 1, p. 47-67, 2003.