Universidade Tecnológica Federal do Paraná

SISTEMAS DISTRIBUÍDOS

PROJETO MULTIDISCIPLINAR

Guido Margonar Moreira **2150948**

Orientador:

Prof. Lucio Agostinho Rocha

Data de submissão : 26/06/2023

Conteúdo

1 Introdução

Os sistemas distribuídos são uma área fundamental da computação que lida com o design, implementação e gerenciamento de sistemas compostos por vários componentes de hardware e software interconectados. Nesses sistemas, os recursos e tarefas são distribuídos entre diferentes computadores interligados em uma rede, trabalhando em conjunto como uma unidade, ao longo da disciplina de sistemas distribuídos foram estudados diversos métodos, arquiteturas e ferramentas para implementação dessa poderosa ferramenta.

Com o objetivo de demonstrar os conhecimentos adquiridos ao longo da disciplina foi proposta a elaboração de um projeto que faça uso de um middleware, comunicação entre objetos distribuídos, controle de concorrência e Web services, com isso em mente neste projeto foi implementado um aplicativo de leilão de sensores, usando o javaRMI para criar Peers atuando de middleware por meio dessa comunicação entre os objetos Cliente e Gerenciador que acessam um arquivo compartilhado só se a tranca não estiver ativa, e devovlem os valores requisitados pelo cliente, além disso foi implementada uma função de web service usando jakarta para permitir ao usuário converter moedas rapidamente, mais a frente também foi criada uma interface visual para o cliente utilizando o javaFx.

2 Metodologia

Este projeto foi desenolvido inteiramente no java, uma linguagem de programação de alto nível orientada a objetos, amplamente utilizada nas mais diversas áreas, conhecido por por sua portabilidade, segurança e robustez, sendo executado em uma máquina virtual chamada Java Virtual Machine (JVM), o que permite que o código Java seja executado em diferentes plataformas sem a necessidade de recompilação e possui uma vasta biblioteca padrão, o desenvolvimento dos softwares utilizou a a versão 11.0.19 do Java Jdk e para a criação do código a IDE foi o Apache Netbeans IDE 18, devido as facilitações oferecidas na preparação do ambiente para trabalhar com as demais bibliotecas.

2.1 Bibliotecas

Além de algumas bibiotecas padrões do java utilizadas para manipulação de arquivos, variáveis e outros dados, foram utilizadas bibliotecas especificas para criação do sistema distrivuido, elas foram: **jakarta.jws**, que permitiu a criação do web service no servidor Payara roteado localmente, **jakarta.xml**, usada pra acessar o serviço Web mencionado anteriormente, **java.rmi**, para criação dos Peers que permitem a execução dos métodos remotos para acesso de leitura e manipulação ao arquivo, por fim a biblioteca **java.io.File** foi utilizada para fazer a sincronização local por meio da criação de um arquivo para travar acesso as operações.

Na parte do Web Service foi utilizado o SOAP (Simple Object Access Protocol) um protocolo de comunicação utilizado para troca de mensagens entre sistemas distribuídos. No contexto do Jakarta EE, o Jakarta SOAP é uma especificação que define como implementar serviços web usando o protocolo SOAP permite que os aplicativos se comuniquem de forma interoperável, independentemente da plataforma ou linguagem de programação utilizada contanto que interpretem as mensagens XML para enviar solicitações e receber respostas, foi isso que foi utilizado para o programa requisitar a conversão de moeda com os parâmetros de valor total e taxa de conversão e receber o valor de volta no cliente.

Como mencionado anteriormente também foi implementado o RMI (Remote Method Invocation) tecnologia de comunicação e invocação de métodos em Java, que permite a interação entre objetos distribuídos em diferentes máquinas. Para criação dos Peers que executam essa funcionalidade foi necessária a funcionalidade do RMI Registry, um serviço de diretório que permite que os clientes localizem e obtenham referências para objetos remoto.

Por fim a biblioteca javafx foi utilizada para criar uma interface visual, criada por meio do programa Scene Builder, o que permite uma melhor visualização da plataforma como algo mais proximo de um produto real.

3 Código Fonte

Para a execução deste sistema foram criados um total de 3 projetos do netbeans que somam 12 códigos .java, o projeto mais simples é o SoapService visto que só foi necessário adaptar os conceitos vistos na aula 29, definindo as rotas para os serviços no servidor payara e trabalhando com as varíaveis passadas no Campo1 e Campo2

3.1 Soap Service

```java
package br.ws;


import jakarta.jws.WebService;
import jakarta.jws.WebMethod;
import jakarta.jws.WebParam;


@WebService(serviceName = "Consulta")
public class SoapService {


    @WebMethod(operationName = "read")
    public String hello(@WebParam(name = "name") String txt) {
        return "Hello      " + txt + " !";
    }

    @WebMethod(operationName = "convert")
    public float somar(@WebParam(name = "campo1") String CAMPO1,
    @WebParam(name = "campo2") String CAMPO2) {
            return Float.parseFloat(CAMPO1) * Float.parseFloat(
    CAMPO2);


    }
}
```

O código (também pode ser acessado no github: https://github.com/guidoMoreira/TrabalhoMu que faz essa requisição foi o SoapClient.java chamado na classe Principal.java do projeto SoapClient, cujas funções estão explicadas abaixo:

- A classe `SoapClient` possui duas variáveis estáticas `CAMPO1` e `CAMPO2` que são usadas para armazenar os valores dos campos que serão enviados na chamada ao serviço web.
- O construtor da classe `SoapClient` recebe dois parâmetros (`CAMPO1` e `CAMPO2`) e os atribui às variáveis estáticas correspondentes.
- O método `callSoapWebService` é responsável por estabelecer uma conexão SOAP, enviar a solicitação SOAP ao servidor e imprimir a resposta recebida.
- No método `callSoapWebService`, uma instância de `SOAPConnectionFactory` é criada utilizando `HttpSOAPConnectionFactory.newInstance()` e, em seguida, uma conexão SOAP é estabelecida usando essa fábrica.
- O método `createSOAPRequest` é chamado para criar uma mensagem SOAP de solicitação, passando o valor de `soapAction` como argumento.
- O método `createSOAPRequest` cria uma mensagem SOAP, adiciona um envelope SOAP e preenche o corpo SOAP com os campos `CAMPO1` e `CAMPO2` (que representam o valor que se tem a taxa de conversão para real respectivamente).
- O método `createSoapEnvelope` preenche o envelope SOAP com os namespaces adequados, cria um corpo SOAP e adiciona elementos SOAP correspondentes aos campos `CAMPO1` e `CAMPO2`.

- A mensagem SOAP de solicitação é exibida no console.
- A mensagem SOAP de solicitação é enviada para o endpoint SOAP especificado e a resposta é armazenada em `soapResponse`.
- A resposta SOAP é impressa no console.
- A conexão SOAP é fechada.
- Em caso de exceção, a mensagem de erro é impressa no console.

```
1  package br;
2
3  import com.sun.xml.messaging.saaj.client.p2p.
       HttpSOAPConnectionFactory;
4  import jakarta.xml.soap.MessageFactory;
5  import jakarta.xml.soap.MimeHeaders;
6  import jakarta.xml.soap.SOAPBody;
7  import jakarta.xml.soap.SOAPConnection;
8  import jakarta.xml.soap.SOAPConnectionFactory;
9  import jakarta.xml.soap.SOAPElement;
10 import jakarta.xml.soap.SOAPEnvelope;
11 import jakarta.xml.soap.SOAPException;
12 import jakarta.xml.soap.SOAPMessage;
13 import jakarta.xml.soap.SOAPPart;
14 //import com.sun.xml.messaging.saaj.client.p2p.
       HttpSOAPConnectionFactory;
15
16 public class SoapClient {
17
18
19     public static String CAMPO1;
20     public static String CAMPO2;
21
22     public SoapClient(String CAMPO1, String CAMPO2) {
23         this.CAMPO1 = CAMPO1;
24         this.CAMPO2 = CAMPO2;
25     }
26
27     public void callSoapWebService(String soapEndpointUrl,
28             String soapAction) {
29         try {
30
31 // Criar conexao SOAP
32             SOAPConnectionFactory soapConnectionFactory
33                     = (SOAPConnectionFactory)
       HttpSOAPConnectionFactory.newInstance();
34             SOAPConnection soapConnection
35                     = soapConnectionFactory.createConnection();
36 // Enviar SOAP Message para o server
37             SOAPMessage soapResponse
38                     = soapConnection.call(createSOAPRequest(
       soapAction),
39                             soapEndpointUrl);
40 // Imprimir resposta
41             System.out.println("Mensagem SOAP de resposta:");
42             soapResponse.writeTo(System.out);
43             System.out.println();
44             soapConnection.close();
45         } catch (Exception e) {
```

```
46              System.out.println("ERRO:");
47              System.out.println(e.getMessage());
48          }
49
50      }
51
52      private static SOAPMessage createSOAPRequest(String
    soapAction) throws Exception {
53
54  //criar mensagem SOAP
55          MessageFactory messageFactory
56                  = MessageFactory.newInstance();
57          SOAPMessage soapMessage
58                  = messageFactory.createMessage();
59  //criar envelope SOAP
60          createSoapEnvelope(soapMessage);
61  //MimeHeaders headers = soapMessage.getMimeHeaders();
62  //headers.addHeader("SOAPAction", soapAction);
63          soapMessage.saveChanges();
64  //Exibir mensagem
65          System.out.println("Request SOAP Message:");
66          soapMessage.writeTo(System.out);
67          System.out.println("\n");
68          return soapMessage;
69      }
70
71      private static void createSoapEnvelope(SOAPMessage
    soapMessage) throws SOAPException {
72          SOAPPart soapPart = soapMessage.getSOAPPart();
73  //verificar no wsdl o namespace utilizado
74          String myNamespace = "ns2";
75          String myNamespaceURI = "http://ws.br/";
76  // Preencher SOAP Envelope
77          SOAPEnvelope envelope = soapPart.getEnvelope();
78          envelope.addNamespaceDeclaration(myNamespace,
79                  myNamespaceURI);
80  // Preencher SOAP Body
81          SOAPBody soapBody = envelope.getBody();
82          SOAPElement soapBodyElem
83                  = soapBody.addChildElement("convert", myNamespace
    );
84          SOAPElement soapBodyElem1
85                  = soapBodyElem.addChildElement("campo1"); //o
    child name foi criado sem namespace
86                  soapBodyElem1.addTextNode(CAMPO1);
87          SOAPElement soapBodyElem2
88                  = soapBodyElem.addChildElement("campo2"); //o
    child name foi criado sem namespace
89                  soapBodyElem2.addTextNode(CAMPO2);
90      }
91  }
```

3.2 Soap Client

Ainda no SoapClient o arquivo Principal.Java utiliza a classe SoapClient e implementa em conjunto a requisição RMI usando das classes Mensagem, Java e Peer, ele é também responsável por criar a interface que o cliente usa para interagir com os servidores, permitindo a leitura de todos os lances, tentar dar novos lances, colocar sensores para o leilão e a conversão de moeda mostrada

anteriormente no web service isso tudo funciona da seguinte maneira:

- A classe `Principal` contém métodos para ler, escrever e sobrescrever mensagens usando um objeto RMI chamado `stub` que implementa a interface `IMensagem`.
- No método `main`, é obtido um registro RMI utilizando o método `LocateRegistry.getRegist` passando o endereço IP e o número da porta.
- É escolhido um peer aleatório da lista de peers disponíveis e é realizado um loop para tentar conectar ao peer até que a conexão seja bem-sucedida.
- Após a conexão ser estabelecida, o cliente exibe uma mensagem indicando o peer ao qual está conectado.
- Em seguida, é exibido um menu com opções para realizar diferentes ações, como visualizar sensores disponíveis, colocar um sensor à venda e dar lance em um sensor existente.
- Há também uma opção adicional para acessar um Web Service auxiliar que converte moedas de acordo com a cotação do real.
- O programa lê a opção selecionada pelo usuário e chama o método correspondente (`read`, `write`, `overwrite` ou a chamada ao Web Service) passando o `stub` e outros parâmetros necessários.
- O loop continua até que o usuário selecione a opção "x"para sair do programa.

```
1  /*
2   * To change this license header, choose License Headers in
3  Project Properties.
4   * To change this template file, choose Tools | Templates
5   * and open the template in the editor.
6   */
7  package br;
8  /**
9   *
10  * @author lucio
11  */
12
13 import java.rmi.registry.LocateRegistry;
14 import java.rmi.registry.Registry;
15 import java.security.SecureRandom;
16 import java.util.ArrayList;
17 import java.util.List;
18 import java.util.Scanner;
19 import java.rmi.RemoteException;
20 import java.util.Arrays;
21
22
23 public class Principal {
24
25
26   public static void read(IMensagem stub,String opcao){
27     Mensagem mensagem = new Mensagem("", opcao);
28     try{
```

```
29      Mensagem resposta = stub.enviar(mensagem); //dentro da
        mensagem tem o campo 'read'
30      System.out.println(resposta.getMensagem());
31      } catch (RemoteException e) {
32          System.out.println("Erro ao pedir metodo ao servidor");
33        }
34    }
35    public static void write(IMensagem stub,String opcao,Scanner
       leitura){
36      System.out.print("Digite o  ndice : ");
37      String fortune = leitura.next();
38    fortune += leitura.nextLine();
39
40      Mensagem mensagem = new Mensagem(fortune, opcao);
41      try{
42      Mensagem resposta = stub.enviar(mensagem); //dentro da
        mensagem tem o campo 'write'
43      System.out.println(resposta.getMensagem());
44      } catch (RemoteException e) {
45          System.out.println("Erro ao pedir metodo ao servidor");
46        }
47    }
48  public static void overwrite(IMensagem stub,String opcao,Scanner
       leitura){
49      System.out.print("Digite o  ndice : ");
50      String fortune = leitura.next();
51    fortune += leitura.nextLine();
52      fortune +=";";
53      System.out.print("Digite o novo valor: ");
54    fortune += leitura.nextLine();
55     fortune +=";";
56      System.out.print("Digite seu CEP: ");
57    fortune += leitura.nextLine();
58
59      Mensagem mensagem = new Mensagem(fortune, opcao);
60      try{
61      Mensagem resposta = stub.enviar(mensagem); //dentro da
        mensagem tem o campo 'write'
62      System.out.println(resposta.getMensagem());
63      } catch (RemoteException e) {
64          System.out.println("Erro ao pedir metodo ao servidor");
65        }
66
67    }
68
69   public static void main(String[] args) {
70
71
72
73          try {
74
75              Registry registro = LocateRegistry.getRegistry("
        127.0.0.1", 1099);
76
77
78            //Escolhe um peer aleatorio da lista de peers para
        conectar
79              SecureRandom sr = new SecureRandom();
80
81              IMensagem stub = null;
```

```
82            Peer peer = null;
83            List<Peer> listaPeers = Arrays.asList(Peer.values());
84        boolean conectou=false;
85        while(!conectou){
86          peer = listaPeers.get(sr.nextInt(listaPeers.size()));
87          try{
88            stub = (IMensagem) registro.lookup(peer.getNome());
89            conectou=true;
90          } catch(java.rmi.ConnectException e){
91            System.out.println(peer.getNome() + " indisponivel.
    ConnectException. Tentanto o proximo...");
92          } catch(java.rmi.NotBoundException e){
93            System.out.println(peer.getNome() + " indisponivel.
    NotBoundException. Tentanto o proximo...");
94          }
95        }
96            System.out.println("Conectado no peer: " + peer.
    getNome());
97
98
99            String opcao="";
100           Scanner leitura = new Scanner(System.in);
101           do {
102             System.out.println("1) Ver Sensores Dispon veis");
103             System.out.println("2) Colocar Sensor a venda");
104    System.out.println("3) Dar lance em um sensor j  existente")
    ;
105               System.out.println("Web Services auxiliares");
106               System.out.println("4) Converta as moedas de
    acordo com a cota  o do real");
107           System.out.println("x) Exit");
108           System.out.print(">> ");
109           opcao = leitura.next();
110           switch(opcao){
111           case "1": {
112             read(stub,opcao);
113             break;
114           }
115           case "2": {
116             //Monta a mensagem
117             write(stub,opcao,leitura);
118             break;
119           }
120    case "3": {
121      overwrite(stub,opcao,leitura);
122      break;
123    }
124               case "4": {
125                       System.out.println("Enviando requesi  o
    para o servidor");
126                       String soapEndpointUrl = "http://
    localhost:8080/SoapServer/Consulta?wsdl";
127                       String soapAction = "http://localhost
    :8080/SoapServer/Consulta";
128 //Soma 1+2
129 //float c2=2;
130
131 System.out.println("Digite o valor na sua moeda: ");
132
133
```

```
134  Scanner le2 = new Scanner(System.in);
135
136  String CAMPO1 = leitura.next();
137
138  System.out.println("Digite o valor da sua moeda em Reais: ");
139  String CAMPO2 = leitura.next();;
140
141  SoapClient sc = new SoapClient(CAMPO1,CAMPO2);
142  sc.callSoapWebService(soapEndpointUrl, soapAction);
143  //System.out.println("Fim!");
144                          break;
145                  }
146                }
147              } while(!opcao.equals("x"));
148
149          } catch(Exception e) {
150              e.printStackTrace();
151          }
152
153      }
154  }
```

3.3 Gerenciador

O ultimo projeto que recebeu o nome de Gerenciador também utiliza a classe IMensagem, Mensagem e Peer, que são usados para implementar o ServidorImpl, porém antes de explica-lo vale passar na classe Principal.java que apesar do nome é a classe responsável pelas manipulações do arquivo registro.txt que está junto deles no package br(vale destacar que é importante que tanto o package de Soap-Client quanto do Gerenciador tenham o mesmo nome para o java RMI funcionar corretamente).

- O código começa com algumas importações de classes e pacotes necessários.
- A classe `Principal` possui uma constante chamada `path` que representa o caminho do arquivo de registro contendo os sensores com seu id;preço;CEPComprador;Nome do sensor.
- Há uma variável `NUM_FORTUNES` que conta o número de sensores registrados no leilão e é inicializada como 0.
- A classe interna `FileReader` é definida dentro da classe `Principal`.
- A classe `FileReader` possui métodos para contar os itens no arquivo de registro, fazer o parsing (análise) do arquivo e ler, escrever e sobrescrever cada dado do leilão nele.
- Dentro do método `read` da classe `FileReader`, todos os leilões são lidos do arquivo, armazenados em um `HashMap` e retornados para o Servidor que os requisitou.
- O método `write` da classe `FileReader` permite escrever um novo registro no arquivo.
- O método `overwrite` da classe `FileReader` permite substituir um lance existente no arquivo de registro por um novo de maior valor.

- A classe `Principal` possui métodos públicos chamados `write`, `overwrite` e `read` que interagem com o objeto `FileReader` para realizar operações de leitura, escrita e sobrescrita no arquivo de registro.
- Os métodos públicos da classe `Principal` fazem uso do objeto `FileReader` para executar as operações necessárias.
- O código também trata exceções de arquivo não encontrado e exibe mensagens de erro correspondentes.

```
1  /**
2    * Laboratorio 4
3    * Autor: Lucio Agostinho Rocha
4    * Ultima atualizacao: 04/04/2023
5    */
6  package br;
7
8  import java.io.*;
9  import java.nio.file.Path;
10 import java.nio.file.Paths;
11 import java.security.SecureRandom;
12 import java.util.ArrayList;
13 import java.util.HashMap;
14 import java.util.HashSet;
15 import java.util.Map;
16 import java.util.Scanner;
17 import java.util.Set;
18
19 import java.io.RandomAccessFile;
20 import java.nio.channels.FileChannel;
21 import java.nio.channels.FileLock;
22
23
24 public class Principal {
25
26   public final static Path path = Paths
27       .get("/home/usuario/V deos/Trabalho Multidisciplinar
     Sistemas Distribuidos/ProjetoFinal/Gerenciador/Gerenciador/src
     /main/java/br/registro.txt");
28   private int NUM_FORTUNES = 0;
29
30   private FileReader fr;
31
32   public class FileReader {
33
34     public int countFortunes() throws FileNotFoundException {
35
36       int lineCount = 0;
37
38       InputStream is = new BufferedInputStream(new
     FileInputStream(
39           path.toString()));
40       try (BufferedReader br = new BufferedReader(new
     InputStreamReader(
41           is))) {
42
43         String line = "";
44         while (!(line == null)) {
45
```

```
46          if (line.equals("%"))
47            lineCount++;
48
49          line = br.readLine();
50
51        }// fim while
52
53        //System.out.println(lineCount);
54      } catch (IOException e) {
55        System.out.println("SHOW: Excecao na leitura do arquivo."
    );
56      }
57      return lineCount;
58    }
59
60    public void parser(HashMap<Integer, String> hm)
61        throws FileNotFoundException {
62
63      InputStream is = new BufferedInputStream(new
    FileInputStream(
64          path.toString()));
65      try (BufferedReader br = new BufferedReader(new
    InputStreamReader(
66          is))) {
67
68        int lineCount = 0;
69
70        String line = "";
71        while (!(line == null)) {
72
73          if (line.equals("%"))
74            lineCount++;
75
76          line = br.readLine();
77          StringBuffer fortune = new StringBuffer();
78          while (!(line == null) && !line.equals("%")) {
79            fortune.append(line + "\n");
80            line = br.readLine();
81            // System.out.print(lineCount + ".");
82          }
83
84          hm.put(lineCount, fortune.toString());
85          //System.out.println(fortune.toString());
86
87          //System.out.println(lineCount);
88        }// fim while
89
90      } catch (IOException e) {
91        System.out.println("SHOW: Excecao na leitura do arquivo."
    );
92      }
93    }
94
95    public String read(HashMap<Integer, String> hm)
96        throws FileNotFoundException {
97
98      String result=";";
99
100
101     InputStream is = new BufferedInputStream(new
```

```
   FileInputStream(
102          path.toString()));
103      try (BufferedReader br = new BufferedReader(new
   InputStreamReader(
104          is))) {
105
106        for(int i = 0; i < NUM_FORTUNES; i++){
107        int lineSelected = i;
108
109
110        result += hm.get(lineSelected);
111        result +=";";
112        }
113      } catch (IOException e) {
114        System.out.println("SHOW: Excecao na leitura do arquivo."
   );
115      }
116      return result;
117    }
118
119    public String write(HashMap<Integer, String> hm, String
   fortune)
120        throws FileNotFoundException {
121      String[] partes = fortune.split(";");
122
123      //int id = Integer.parseInt(partes[0]);
124      float valor = Float.parseFloat(partes[0].replace(",", "."))
   ;
125      String CEP = partes[1];
126      String nome = partes[2];
127
128      OutputStream os = new BufferedOutputStream(new
   FileOutputStream(
129          path.toString(),true)); //true=append
130      try (BufferedWriter bw = new BufferedWriter(
131                    new OutputStreamWriter(os))) {
132        /*String old = hm.get(id);
133        String[] partesold = fortune.split(";");
134        int ido = Integer.parseInt(partes[0]);
135        float valoro = Float.parseFloat(partes[1].replace(",",
   "."));
136
137        if(valor > valoro){*/
138        Scanner input = new Scanner(System.in);
139        //System.out.print("Add fortune: ");
140        //String fortune = input.next();
141
142        String fort = Integer.toString(NUM_FORTUNES)+ ";"+fortune
   ;
143        NUM_FORTUNES++;
144        hm.put(NUM_FORTUNES, fort);
145
146        //System.out.println(hm.get(NUM_FORTUNES));
147
148        //Append file
149        bw.append(fort+"\n%\n");
150        return fort;
151        /*}else{
152          return "Valor muito baixo";
153        }*/
```

```
154
155       } catch (IOException e) {
156         System.out.println("SHOW: Excecao na leitura do arquivo."
    );
157         return "Erro na leitura do arquivo";
158       }
159
160     }
161     public String overwrite(HashMap<Integer, String> hm, String
    fortune)
162         throws FileNotFoundException {
163       String[] partes = fortune.split(";");
164
165       int id = Integer.parseInt(partes[0]);
166       float valor = Float.parseFloat(partes[1].replace(",", "."))
    ;
167       String CEP = partes[2];
168
169
170       OutputStream os = new BufferedOutputStream(new
    FileOutputStream(
171           path.toString(),true)); //true=append
172       if(id < NUM_FORTUNES){
173       try (BufferedWriter bw = new BufferedWriter(new
    OutputStreamWriter(os))) {
174         String old = hm.get(id);
175         String[] partesold = old.split(";");
176         int ido = Integer.parseInt(partesold[0]);
177         float valoro = Float.parseFloat(partesold[1].replace(",",
     "."));
178         String nom = partesold[3];
179         System.out.print(valoro);
180         if(valor > valoro){
181         Scanner input = new Scanner(System.in);
182         //System.out.print("Add fortune: ");
183         //String fortune = input.next();
184
185         NUM_FORTUNES++;
186
187         hm.replace(id, fortune+";"+nom);
188
189         //System.out.println(hm.get(NUM_FORTUNES));
190
191
192         //Append file
193                                 PrintWriter writer = new
    PrintWriter(path.toString());
194                                 writer.print("");
195                                 writer.close();
196                                 for(int i = 0; i < NUM_FORTUNES;i
    ++){
197                                     if(i != id)
198                                         bw.append(hm.get(i)+"\n%\
    n");
199                                     else
200                                         bw.append(hm.get(id)+"\n
    %\n");
201                                 }
202           return fortune+";"+nom;
203         }else{
```

```
204           return "Valor muito baixo";
205         }
206
207       } catch (IOException e) {
208         System.out.println("SHOW: Excecao na leitura do arquivo."
    );
209         return "Erro na leitura do arquivo";
210       }
211       }else{
212         return "Id muito alto";
213 }
214     }
215   }
216
217   public String write(String fortune){
218     fr = new FileReader();
219     try {
220       NUM_FORTUNES = fr.countFortunes();
221       HashMap hm = new HashMap<Integer, String>();
222       fr.parser(hm);
223       fr.read(hm);
224       return fr.write(hm, fortune);
225     } catch (FileNotFoundException e) {
226       e.printStackTrace();
227       return "Arquivo n o encontrado";
228     }
229   }
230   public String overwrite(String nV){
231     fr = new FileReader();
232     try {
233       NUM_FORTUNES = fr.countFortunes();
234       HashMap hm = new HashMap<Integer, String>();
235       fr.parser(hm);
236       fr.read(hm);
237       return fr.overwrite(hm, nV);
238     } catch (FileNotFoundException e) {
239       e.printStackTrace();
240       return "Arquivo n o encontrado";
241     }
242
243   }
244   public String read(){
245     String result="-1";
246
247     fr = new FileReader();
248     try {
249       NUM_FORTUNES = fr.countFortunes();
250       HashMap hm = new HashMap<Integer, String>();
251       fr.parser(hm);
252       result = fr.read(hm);
253     } catch (FileNotFoundException e) {
254       e.printStackTrace();
255     }
256     return result;
257   }
258
259
260 }
```

Terminando a questão de sistemas distribuidos na classe ServidorImpl.java é criado um peer de acordo com os disponíveis no registro, esse Peer então fica

aguadando por uma conexão do Cliente RMI que meste caso é o Principal do ClienteSoap, ao fazer a conexão ele recebe a String no formato Json contem o método que vária de read(para ler todos o sensores sendo leiloados), write(Para registrar um novo sensor no leilão) e writeA(para tentar dar um lance em um sensor especifico), e para impedir que dois Peers acessem o arquivo ao mesmo tempo foi craido um arquivo chamado lockfile que é criado durante o uso do arquivo registro.txt e é deletado ao fim do uso permitindo a sincronização dos diferentes processos, o código parte por parte funciona assim:

- O código começa importando as classes e pacotes necessários.
- A classe `ServidorImpl` implementa a interface `IMensagem` e contém a lógica para processar as mensagens recebidas.
- A classe possui uma lista de objetos `Peer` chamada `alocados` e um objeto `File` chamado `lockFile`.
- O construtor da classe inicializa a lista `alocados` e cria o objeto `lockFile`.
- O método `enviar` é uma implementação do método definido na interface `IMensagem` e recebe uma mensagem como parâmetro. Ele processa a mensagem, realiza o parsing do JSON e retorna uma resposta.
- O método `parserJSON` recebe uma string JSON como entrada e realiza o parsing do JSON, armazenando os valores em um mapa.
- O método `parserJSON` verifica se a string JSON começa e termina com chaves, remove as chaves, divide a string em pares chave-valor e extrai as chaves e valores.
- O método verifica se o valor é uma string e, em seguida, adiciona a chave e o valor ao mapa.
- O método também possui lógica para criar um arquivo de tranca para que só um modifique/leia o arquivo de uma vez, chamar métodos da classe `Principal` com base na requisição que ele leu no JSON e após receber as respostas ele excluir o arquivo de tranca para permitir aos outros Peer acessa-lo.
- O método `iniciar` configura o servidor RMI, cria um registro de serviço, selecionando aleatoriamente um objeto `Peer` até encontrar um válido ou verificar que o registro está cheio.
- O método `main` cria uma instância do servidor e o inicia.

```
1  /**
2    * Laboratorio 4
3    * Autor: Guido Margonar Moreira
4    * Ultima atualizacao: 06/05/2023
5    */
6  package br;
7
8  import java.io.File;
9  import java.io.IOException;
10 import java.nio.file.Path;
```

```
11 import java.nio.file.Paths;
12 import java.rmi.RemoteException;
13 import java.rmi.registry.LocateRegistry;
14 import java.rmi.registry.Registry;
15 import java.rmi.server.UnicastRemoteObject;
16 import java.security.SecureRandom;
17 import java.util.ArrayList;
18 import java.util.List;
19 import java.util.regex.Matcher;
20 import java.util.regex.Pattern;
21 import java.util.Arrays;
22 import java.util.Map;
23 import java.util.HashMap;
24
25 public class ServidorImpl implements IMensagem{
26     private File lockFile;
27   ArrayList<Peer> alocados;
28     public final static Path lockFilePath = Paths.get("/home/
    usuario/V deos/Trabalho Multidisciplinar Sistemas
    Distribuidos/ProjetoFinal/Gerenciador/Gerenciador/src/main/
    java/br/lock.txt");
29
30     public ServidorImpl() {
31           alocados = new ArrayList<>();
32           lockFile = new File(lockFilePath.toString());
33     }
34
35     //Cliente: invoca o metodo remoto 'enviar'
36     //Servidor: invoca o metodo local 'enviar'
37     @Override
38     public Mensagem enviar(Mensagem mensagem) throws
    RemoteException {
39         Mensagem resposta;
40         try {
41           System.out.println("Mensagem recebida: " + mensagem.
    getMensagem());
42       resposta = new Mensagem(parserJSON(mensagem.getMensagem()))
    ;
43     } catch (Exception e) {
44       e.printStackTrace();
45       resposta = new Mensagem("{\n" + "\"result\": false\n" + "}"
    );
46     }
47         return resposta;
48     }
49
50     public String parserJSON(String json) throws IOException {
51   String jsonString = json.replaceAll("\n", "");//json.replaceAll
    ("\\s+", "");
52
53         // Criar um mapa para armazenar os valores
54         Map < String, Object > values = new HashMap < String,
    Object > ();
55
56         // Verificar se a string come a e termina com chaves
57
58         if (jsonString.startsWith("{") && jsonString.endsWith("}"
    )) {
59           // Remover as chaves da string
60           jsonString = jsonString.substring(1, jsonString.length
```

```
   () - 1);

          // Dividir a string em pares chave-valor
          String[] keyValuePairs = jsonString.split(",");

          // Para cada par chave-valor, extrair a chave e o valor
          for (String pair: keyValuePairs) {
            String[] keyValue = pair.split(":");
            String key = keyValue[0].replaceAll("\"", "");
            String value = keyValue[1];

            // Verificar se o valor    uma string
            if (value.startsWith("\"") && value.endsWith("\"")) {
              // Adicionar a string sem aspas ao mapa
                values.put(key, value.substring(1, value.length()
    - 1));
            }else if(value.startsWith("[") && value.endsWith("]")
   ){
    values.put(key, value.substring(2, value.length() - 2));
  }

          }
        }
        String method = (String) values.get("method");
  String args = (String) values.get("args");

        //Enquanto existir arquivo de tranca ele fica esperando
        while(lockFile.exists()){
            System.out.print("Esperando");
        }

        //Arquivo de Tranca Criado
       lockFile.createNewFile();
      String resultado = "";
      Principal fr = new Principal();
    if(method.equals("read")){
        String f = fr.read();
        resultado = "{\n" + "\"result\": "+ f +"\n" + "}";
System.out.println("Fortuna enviada: "+f);
      }
    else if (method.equals("write")){
    //Nova fortuna

      String res = fr.write(args);
      resultado = "{\n" + "\"result\": "+ res +"\n" + "}";
  System.out.println("Adicionada fortuna: "+res);
    }else if(method.equals("writeA")){
    //Nova fortuna

      String res = fr.overwrite(args);
      resultado = "{\n" + "\"result\": "+ res +"\n" + "}";
  System.out.println("Adicionada fortuna: "+res);
    }

    //Tranca Liberada
    lockFile.delete();
    return resultado;
  }

    public void iniciar(){
```

```
118
119     try {
120         //TODO: Adquire aleatoriamente um 'nome' do arquivo Peer.
    java
121         List<Peer> listaPeers = Arrays.asList(Peer.values());
122
123         Registry servidorRegistro;
124         try {
125           servidorRegistro = LocateRegistry.createRegistry(1099);
126         } catch (java.rmi.server.ExportException e){ //Registro
    jah iniciado
127           System.out.print("Registro ja iniciado. Usar o ativo.\n
    ");
128         }
129         servidorRegistro = LocateRegistry.getRegistry(); //
    Registro eh unico para todos os peers
130         String [] listaAlocados = servidorRegistro.list();
131         for(int i=0; i<listaAlocados.length;i++)
132           System.out.println(listaAlocados[i]+" ativo.");
133
134         SecureRandom sr = new SecureRandom();
135         Peer peer = listaPeers.get(sr.nextInt(listaPeers.size()))
    ;
136
137         int tentativas=0;
138         boolean repetido = true;
139         boolean cheio = false;
140         while(repetido && !cheio){
141           repetido=false;
142           peer = listaPeers.get(sr.nextInt(listaPeers.size()));
143           for(int i=0; i<listaAlocados.length && !repetido; i++){
144
145             if(listaAlocados[i].equals(peer.getNome())){
146               System.out.println(peer.getNome() + " ativo.
    Tentando proximo...");
147               repetido=true;
148               tentativas=i+1;
149             }
150
151           }
152           //System.out.println(tentativas+" "+listaAlocados.
    length);
153
154           //Verifica se o registro estah cheio (todos alocados)
155           if(listaAlocados.length>0 && //Para o caso inicial em
    que nao ha servidor alocado,
156                                        //caso contrario, o teste
    abaixo sempre serah true
157             tentativas==listaPeers.size()){
158             cheio=true;
159           }
160         }
161
162         if(cheio){
163           System.out.println("Sistema cheio. Tente mais tarde.");
164           System.exit(1);
165         }
166
167           IMensagem skeleton  = (IMensagem) UnicastRemoteObject
    .exportObject(this, 0); //0: sistema operacional indica a
```

```
    porta (porta anonima)
168             servidorRegistro.rebind(peer.getNome(), skeleton);
169             System.out.print(peer.getNome() +" Servidor RMI:
    Aguardando conexoes...");
170 
171         } catch(Exception e) {
172             e.printStackTrace();
173         }
174 
175     }
176 
177     public static void main(String[] args) {
178         ServidorImpl servidor = new ServidorImpl();
179         servidor.iniciar();
180     }
181 }
```

Por fim foi criada a interface visual, carregando um arquivo fxml criado no programa Scene Builder, os botões, áreas de texto e labels foram associados as funções do programa a principal diferença entre essa interface e o console é que nessa versão a conexão com os peers só é feita quando o programa apertar um botão, logo não é necessário um peer ativo para inicializar o aplicativo (não há muito a explicar no código abaixo no geral as mudanças saõ referentes ao Javafx sendo preciso referenciar os atributos da janela pelo id e as funcionalidades da classe Principal foram adaptadas para esse código).

```
1  /*
2   * To change this license header, choose License Headers in
3  Project Properties.
4   * To change this template file, choose Tools | Templates
5   * and open the template in the editor.
6   */
7  package br;
8  /**
9   *
10  * @author Guido
11  */
12 
13 import java.rmi.registry.LocateRegistry;
14 import java.rmi.registry.Registry;
15 import java.security.SecureRandom;
16 import java.util.ArrayList;
17 import java.util.List;
18 import java.util.Scanner;
19 import java.rmi.RemoteException;
20 import java.util.Arrays;
21 
22 
23 public class Principal {
24 
25 
26   public static void read(IMensagem stub,String opcao){
27     Mensagem mensagem = new Mensagem("", opcao);
28     try{
29     Mensagem resposta = stub.enviar(mensagem); //dentro da
    mensagem tem o campo 'read'
30     System.out.println(resposta.getMensagem());
31     } catch (RemoteException e) {
32         System.out.println("Erro ao pedir metodo ao servidor");
33       }
```

```
34    }
35    public static void write(IMensagem stub,String opcao,Scanner
       leitura){
36      System.out.print("Digite o  ndice : ");
37      String fortune = leitura.next();
38    fortune += leitura.nextLine();
39
40      Mensagem mensagem = new Mensagem(fortune, opcao);
41      try{
42      Mensagem resposta = stub.enviar(mensagem); //dentro da
       mensagem tem o campo 'write'
43      System.out.println(resposta.getMensagem());
44      } catch (RemoteException e) {
45          System.out.println("Erro ao pedir metodo ao servidor");
46        }
47    }
48  public static void overwrite(IMensagem stub,String opcao,Scanner
       leitura){
49      System.out.print("Digite o  ndice : ");
50      String fortune = leitura.next();
51    fortune += leitura.nextLine();
52      fortune +=";";
53      System.out.print("Digite o novo valor: ");
54    fortune += leitura.nextLine();
55     fortune +=";";
56      System.out.print("Digite seu CEP: ");
57    fortune += leitura.nextLine();
58
59      Mensagem mensagem = new Mensagem(fortune, opcao);
60      try{
61      Mensagem resposta = stub.enviar(mensagem); //dentro da
       mensagem tem o campo 'write'
62      System.out.println(resposta.getMensagem());
63      } catch (RemoteException e) {
64          System.out.println("Erro ao pedir metodo ao servidor");
65        }
66
67    }
68
69   public static void main(String[] args) {
70
71
72
73          try {
74
75              Registry registro = LocateRegistry.getRegistry("
       127.0.0.1", 1099);
76
77
78            //Escolhe um peer aleatorio da lista de peers para
       conectar
79              SecureRandom sr = new SecureRandom();
80
81              IMensagem stub = null;
82              Peer peer = null;
83              List<Peer> listaPeers = Arrays.asList(Peer.values());
84          boolean conectou=false;
85          while(!conectou){
86            peer = listaPeers.get(sr.nextInt(listaPeers.size()));
87            try{
```

```
88            stub = (IMensagem) registro.lookup(peer.getNome());
89            conectou=true;
90          } catch(java.rmi.ConnectException e){
91            System.out.println(peer.getNome() + " indisponivel.
    ConnectException. Tentanto o proximo...");
92          } catch(java.rmi.NotBoundException e){
93            System.out.println(peer.getNome() + " indisponivel.
    NotBoundException. Tentanto o proximo...");
94          }
95        }
96            System.out.println("Conectado no peer: " + peer.
    getNome());
97
98
99            String opcao="";
100           Scanner leitura = new Scanner(System.in);
101           do {
102             System.out.println("1) Ver Sensores Dispon veis");
103             System.out.println("2) Colocar Sensor a venda");
104    System.out.println("3) Dar lance em um sensor j   existente")
    ;
105               System.out.println("Web Services auxiliares");
106               System.out.println("4) Converta as moedas de
    acordo com a cota   o do real");
107             System.out.println("x) Exit");
108             System.out.print(">> ");
109             opcao = leitura.next();
110             switch(opcao){
111             case "1": {
112               read(stub,opcao);
113               break;
114             }
115             case "2": {
116               //Monta a mensagem
117               write(stub,opcao,leitura);
118               break;
119             }
120    case "3": {
121      overwrite(stub,opcao,leitura);
122      break;
123    }
124               case "4": {
125                         System.out.println("Enviando requesi   o
    para o servidor");
126                         String soapEndpointUrl = "http://
    localhost:8080/SoapServer/Consulta?wsdl";
127                         String soapAction = "http://localhost
    :8080/SoapServer/Consulta";
128 //Soma 1+2
129 //float c2=2;
130
131 System.out.println("Digite o valor na sua moeda: ");
132
133
134 Scanner le2 = new Scanner(System.in);
135
136 String CAMPO1 = leitura.next();
137
138 System.out.println("Digite o valor da sua moeda em Reais: ");
139 String CAMPO2 = leitura.next();;
```

```
140
141 SoapClient sc = new SoapClient(CAMPO1,CAMPO2);
142 sc.callSoapWebService(soapEndpointUrl, soapAction);
143 //System.out.println("Fim!");
144                         break;
145                 }
146             }
147         } while(!opcao.equals("x"));
148
149         } catch(Exception e) {
150             e.printStackTrace();
151         }
152
153     }
154 }
```

4 Resultados

Após as correções de inúmeros bugs e ajustes nos designes para as limitações e potenciais de cada arquitetura o projeto funcionou com sucesso, sendo necessário primeiramente executar os servidores Gerenciador, Payara server(e instanciar o web service) para depois iniciar as conexões com os clientes, o menu exibido para o cliente pode ser visto na Figura 1.

```
--- exec:3.1.0:exec (default-cli) @ SoapClient ---
Conectado no peer: PEER3
1) Ver Sensores Disponíveis
2) Colocar Sensor a venda
3) Dar lance em um sensor já existente
Web Services auxiliares
4) Converta as moedas de acordo com a cotação do real
x) Exit
>>
```

Figura 1: Menu Autoria própria.

Ao selecionar a primeira opção todos os dados são retornados como mostra a Figura 2, vale destacar que os dados estão no modelo **id;PreçoLance;CEP;Nome do sensor** infelizmente não foi possível colcoalos em uma interface visual no Javafx(a janela está na pasta do ClientSoap mas não foi usada).

```
1) Ver Sensores Disponíveis
2) Colocar Sensor a venda
3) Dar lance em um sensor já existente
Web Services auxiliares
4) Converta as moedas de acordo com a cotação do real
x) Exit
>> 1
{
"result": ;0;26,2;1234;Sensor de pressao
;1;27,2;1534;Sensor térmico
;2;50;12345;Sensor Oxigenio
;3;25;6;Sensor Barometrico
;4;2315;6465;LVDT
;5;151.5;22221;sensor de Radiação
;6;10.0;2;Multimetro
;
}
```

Figura 2: Opção 1 do menu Autoria própria.

Na Figura 3 é possível ver que o servidor conseguiu adicionar um novo sensor a venda recebendo o preço, CEP do vendedor e nome do produto.

```
1) Ver Sensores Disponíveis
2) Colocar Sensor a venda
3) Dar lance em um sensor já existente
Web Services auxiliares
4) Converta as moedas de acordo com a cotação do real
x) Exit
>> 2
Digite o índice: 200;456789;Sensor sEMG
{
"result": 7;200;456789;Sensor sEMG
}
```

Figura 3: Opção 2 do menu Autoria própria.

O sistema de lances também está funcionando permitindo dar um lance ao passar o índice o valor do lance e o CEP do comprador, nesse caso o lance estava acima do anterior por isso ele sobrescreveu o valor como mostra a 4

```
1) Ver Sensores Disponíveis
2) Colocar Sensor a venda
3) Dar lance em um sensor já existente
Web Services auxiliares
4) Converta as moedas de acordo com a cotação do real
x) Exit
>> 3
Digite o índice: 7
Digite o novo valor: 201
Digite seu CEP: 12345
{
"result": 7;201;12345;Sensor sEMG

}
```

Figura 4: Opção 3 do menu Autoria própria.

E o sistema de conversão do valor da moeda realiza o pedido para o payara server e recebe a resposta com o valor já calculado tudo de acordo com o esperado 6

```
1) Ver Sensores Disponíveis
2) Colocar Sensor a venda
3) Dar lance em um sensor já existente
Web Services auxiliares
4) Converta as moedas de acordo com a cotação do real
x) Exit
>> 4
Enviando requesição para o servidor
Digite o valor na sua moeda:
250
Digite o valor da sua moeda em Reais:
0.4
Request SOAP Message:
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns2="http://ws.br/"><SOAP-ENV:Header/><SOAP-ENV:Body><ns2:convert><campo1>250</campo1><campo2>0.

Mensagem SOAP de resposta:
<?xml version='1.0' encoding='UTF-8'?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"><S:Body><ns2:convertResponse xmlns:ns2="http://ws.br/"><return>100.0</return><
```

Figura 5: Opção 4 do menu Autoria própria.

Por fim a interface melhorada que implementa todos os features citados acima:

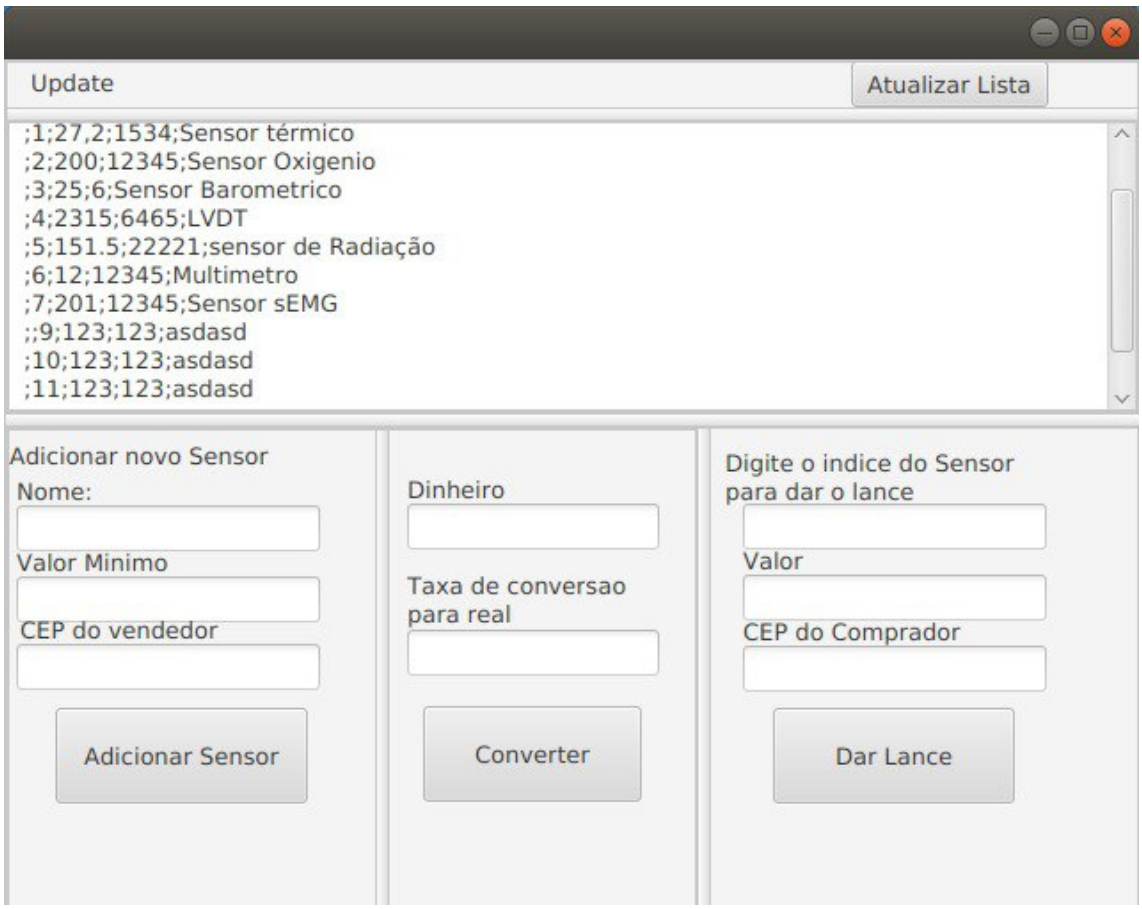

Figura 6: Interface visual no Javafx.

5 Conclusão

Ao levar em conta todos os dados apresentados anteriormente é possível concluir que os conceitos apresentados na disciplina foram bem ensinados e aplicados, embora o sistema não seja a prova de falhas e ainda haja muito espaço para melhoria na interpretação dos dados pela interface visual para melhor visualização das informações mas principalmente quanto a concorrência que ainda não tem suporte para peers em diferentes máquinas (seria necessário sincronizar um compartilhamento o valor do arquivo Registro e do arquivo de lock), escalabilidade para permitir mais peers, usuários e dados mais distribuídos de forma dinâmica, por fim também seria possível melhorar a tolerância a falhas visto que nem todos os possíveis erros pegos pelo try catch são tratados da melhor maneira, portanto apesar de alcançados todos os objetivo propostos o tempo acabou impedindo uma melhor otimização.