# SISTEMAS OPERACIONAIS: MICRO BLOG RMI

VITOR PALMA ADERALDO
VITOR HENRIQUE ROSA BATISTA DE OLIVEIRA

# Índice Hierárquico

## Hierarquia de Classes

Esta lista de hierarquias está parcialmente ordenada (ordem alfabética):

- Banco
- BlogCliente
- BlogServidor
- OraoraSherlock
- palavras
- Runnable
  - eleitor
- Thread
  - Semaforo
- Remote
  - Blog
    - BlogImp
- Serializable
  - Usuario
- UnicastRemoteObject
  - BlogImp

# Índice dos Componentes

## Lista de Componentes

Aqui estão as classes, estruturas, uniões e interfaces e suas respectivas descrições:


**Banco**

**Blog**

**BlogCliente**

**BlogImp**

**BlogServidor**

**eleitor**

**oraoraSherlock**

**palavras**

**Semaforo**

**Usuario**

# Índice dos Arquivos

## Lista de Arquivos

Esta é a lista de todos os arquivos e suas respectivas descrições:


**/home/vitor/Desktop/src/Banco.java**

**/home/vitor/Desktop/src/Blog.java**

**/home/vitor/Desktop/src/BlogCliente.java**

**/home/vitor/Desktop/src/BlogImp.java**

**/home/vitor/Desktop/src/BlogServidor.java**

**/home/vitor/Desktop/src/eleitor.java**

**/home/vitor/Desktop/src/Eleitores.java**

**/home/vitor/Desktop/src/oraoraSherlock.java**

**/home/vitor/Desktop/src/palavras.java**

**/home/vitor/Desktop/src/Semaforo.java**

**/home/vitor/Desktop/src/Usuario.java**

# Classes

## Referência da Classe Banco

### Métodos Públicos

- void **postar** (String usuario, String topico, String texto)
- void **seguir** (String usuario, String topico)
- void **pararDeseguir** (String usuario, String topico)
- List< String > **pegarPosts** (String usuario, String dia, String mes, String ano)
- List< String > **pegarPostsTopico** (String usuario, String topico, String dia, String mes, String ano)

---

### Métodos

#### void Banco.pararDeseguir (String *usuario*, String *topico*)

```
62                                                                  {
63
64        try {
65            stmt = c.createStatement();
66            String sql = "delete from follow where usuario = '" + usuario + "' and
topico='"+topico+"'";
67            stmt.executeUpdate(sql);
68
69            stmt.close();
70            c.commit();
71
72        } catch (SQLException ex) {
73            Logger.getLogger(Banco.class.getName()).log(Level.SEVERE, null, ex);
74        }
75    }
```

#### List<String> Banco.pegarPosts (String *usuario*, String *dia*, String *mes*, String *ano*)

```
77
{
78
79        List<String> lista = new ArrayList<String>();
80
81        String data = "'" + ano+"-"+mes+"-"+dia+"'";
82        String user = "'"+usuario+"'";
83
84        String us = "";
85        String top = "";
86        String texto  = "";
87        String hr = "";
88        try {
89            stmt = c.createStatement();
90            String sql = "select usuario,topico,texto,datahora from posts3 where datahora
>= "+ data+" and topico in ( select topico from follow where usuario = "+user+")";
91
92            ResultSet rs = stmt.executeQuery(sql);
93
94             while ( rs.next() ){
95
96                  lista.add(rs.getString("usuario"));
97                  lista.add(rs.getString("topico"));
98                  lista.add(rs.getString("texto"));
99                  lista.add(rs.getTimestamp("datahora").toString());
100
101
102          }
```

```
103
104             stmt.close();
105             c.commit();
106             rs.close();
107
108         } catch (SQLException ex) {
109             Logger.getLogger(Banco.class.getName()).log(Level.SEVERE, null, ex);
110         }
111
112         return lista;
113     }
```

**List<String> Banco.pegarPostsTopico (String *usuario*, String *topico*, String *dia*, String *mes*, String *ano*)**

```
115
116 {
117
118         List<String> lista = new ArrayList<String>();
119
120         String data = "'" + ano+"-"+mes+"-"+dia+"'";
121         String user = "'"+usuario+"'";
122         String top2 = "'"+topico+"'";
123
124
125         String us = "";
126         String top = "";
127         String texto  = "";
128         String hr = "";
129         try {
130             stmt = c.createStatement();
131             String sql = "select usuario,topico,texto,datahora from posts3 where datahora
>= "+ data+" and topico in ( select topico from follow where usuario = "+user+")" +" and topico
= " + top2;
132
133             ResultSet rs = stmt.executeQuery(sql);
134
135              while ( rs.next() ){
136
137                   lista.add(rs.getString("usuario"));
138                   lista.add(rs.getString("topico"));
139                   lista.add(rs.getString("texto"));
140                   lista.add(rs.getTimestamp("datahora").toString());
141
142
143           }
144
145             stmt.close();
146             c.commit();
147             rs.close();
148
149         } catch (SQLException ex) {
150             Logger.getLogger(Banco.class.getName()).log(Level.SEVERE, null, ex);
151         }
152
153         return lista;
154     }
```

**void Banco.postar (String *usuario*, String *topico*, String *texto*)**

```
32                                                                          {
33
34         try {
35             stmt = c.createStatement();
36             String banco_inserir_post = "insert into posts3 (usuario,topico,texto) values
('"+usuario+"','"+topico+"','"+texto+"')";
37             stmt.executeUpdate(banco_inserir_post);
38
39             stmt.close();
40             c.commit();
41
```

```
42        } catch (SQLException ex) {
43            Logger.getLogger(Banco.class.getName()).log(Level.SEVERE, null, ex);
44        }
45    }
```

### void Banco.seguir (String *usuario*, String *topico*)

```
47                                                      {
48
49        try {
50            stmt = c.createStatement();
51            String sql = "insert into follow (usuario,topico) values
('"+usuario+"','"+topico+"')";
52            stmt.executeUpdate(sql);
53
54            stmt.close();
55            c.commit();
56
57        } catch (SQLException ex) {
58            Logger.getLogger(Banco.class.getName()).log(Level.SEVERE, null, ex);
59        }
60    }
```

---

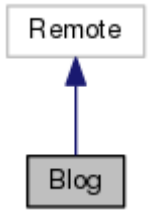
**A documentação para esta classe foi gerada a partir do seguinte arquivo:**

- /home/vitor/Desktop/src/**Banco.java**

## Referência da Interface Blog

Diagrama de Hierarquia para Blog:


Diagrama de colaboração para Blog:




### Métodos Públicos

- String **apresentar** () throws RemoteException
- String **postar** (String usuario, String topico, String texto) throws RemoteException
- String **seguir** (String usuario, String topico) throws RemoteException
- String **pararseguir** (String usuario, String topico) throws RemoteException
- List< String > **pegarPosts** (String usuario, String dia, String mes, String ano) throws RemoteException
- List< String > **pegarPostsTop** (String usuario, String topico, String dia, String mes, String ano) throws RemoteException
- int **definirId** () throws RemoteException
- int **definirLider** (int id, int forca) throws RemoteException
- int **abdicarLideranca** (int id) throws RemoteException
- void **setPermitir** (int a) throws RemoteException
- int **getPermitir** () throws RemoteException
- int **getLider** () throws RemoteException
- void **printar** () throws RemoteException
- int **eleger** () throws RemoteException
- void **setTime** (int id, int time) throws RemoteException
- void **setUpdate** (int i) throws RemoteException
- int **getUpdate** () throws RemoteException
- int **calcularMedia** () throws RemoteException
- void **zerarLista** () throws RemoteException
- int **pegarMedia** () throws RemoteException
- int **getEscrita** () throws RemoteException
- void **setEscrita** (int a) throws RemoteException
- int **getLeitura** () throws RemoteException
- void **setLeitura** (int a) throws RemoteException

- int **jesus** ()  throws RemoteException

---

## Métodos

### int Blog.abdicarLideranca (int *id*) throws RemoteException

Implementado por **BlogImp** (*pag.17*).

### String Blog.apresentar () throws RemoteException

Implementado por **BlogImp** (*pag.17*).

### int Blog.calcularMedia () throws RemoteException

Implementado por **BlogImp** (*pag.18*).

### int Blog.definirId () throws RemoteException

Implementado por **BlogImp** (*pag.18*).

### int Blog.definirLider (int *id*, int *forca*) throws RemoteException

Implementado por **BlogImp** (*pag.18*).

### int Blog.eleger () throws RemoteException

Implementado por **BlogImp** (*pag.18*).

### int Blog.getEscrita () throws RemoteException

Implementado por **BlogImp** (*pag.18*).

### int Blog.getLeitura () throws RemoteException

Implementado por **BlogImp** (*pag.19*).

### int Blog.getLider () throws RemoteException

Implementado por **BlogImp** (*pag.19*).

### int Blog.getPermitir () throws RemoteException

Implementado por **BlogImp** (*pag.19*).

**int Blog.getUpdate () throws RemoteException**

Implementado por **BlogImp** (*pag.19*).

**int Blog.jesus () throws RemoteException**

Implementado por **BlogImp** (*pag.19*).

**String Blog.pararseguir (String *usuario*, String *topico*) throws RemoteException**

Implementado por **BlogImp** (*pag.19*).

**int Blog.pegarMedia () throws RemoteException**

Implementado por **BlogImp** (*pag.19*).

**List<String> Blog.pegarPosts (String *usuario*, String *dia*, String *mes*, String *ano*) throws RemoteException**

Implementado por **BlogImp** (*pag.20*).

**List<String> Blog.pegarPostsTop (String *usuario*, String *topico*, String *dia*, String *mes*, String *ano*) throws RemoteException**

Implementado por **BlogImp** (*pag.20*).

**String Blog.postar (String *usuario*, String *topico*, String *texto*) throws RemoteException**

Implementado por **BlogImp** (*pag.20*).

**void Blog.printar () throws RemoteException**

Implementado por **BlogImp** (*pag.20*).

**String Blog.seguir (String *usuario*, String *topico*) throws RemoteException**

Implementado por **BlogImp** (*pag.20*).

**void Blog.setEscrita (int *a*) throws RemoteException**

Implementado por **BlogImp** (*pag.20*).

**void Blog.setLeitura (int *a*) throws RemoteException**

Implementado por **BlogImp** (*pag.21*).

### void Blog.setPermitir (int *a*) throws RemoteException

Implementado por **BlogImp** (*pag.21*).

### void Blog.setTime (int *id*, int *time*) throws RemoteException

Implementado por **BlogImp** (*pag.21*).

### void Blog.setUpdate (int *i*) throws RemoteException

Implementado por **BlogImp** (*pag.21*).

### void Blog.zerarLista () throws RemoteException

Implementado por **BlogImp** (*pag.21*).

---

**A documentação para esta interface foi gerada a partir do seguinte arquivo:**

- /home/vitor/Desktop/src/**Blog.java**

# Referência da Classe BlogCliente

## Métodos Públicos Estáticos

- static void **main** (String[] args)

---

## Métodos

### static void BlogCliente.main (String[] *args*) `[static]`

```
12                                          {
13
14        try{
15            Blog c = (Blog) Naming.lookup("//127.0.0.1:1099/CalculatorService");
16            palavras p = new palavras();
17
18
19
20            String entrada="";
21
22            Scanner post = new Scanner (System.in);
23
24            System.out.println(c.apresentar());
25
26            System.out.print("Usuario: ");
27            Scanner scanner = new Scanner(System. in);
28            String nome = scanner. nextLine();
29
30            eleitor e = new eleitor(c);
31
32            System.out.print("Horario: ");
33            scanner = new Scanner(System. in);
34            String horario = scanner. nextLine();
35
36            if(horario.equals("auto")){
37                //String hora =  new SimpleDateFormat("HH:mm:ss").format(new Date());
38                e.setAuto(1);
39                //System.out.println(hora);
40            }else{
41                e.setAuto(0);
42                e.setHora(horario);
43            }
44
45
46
47            e.definirId();
48            e.definirLider();
49
50
51            Thread threadEleicao = new Thread(e);
52            threadEleicao.start();
53
54
55            System.out.println("Logado com sucesso ! ");
56            System.out.println("");
57            Scanner scan = new Scanner (System.in);
58            System.out.print ("-> ");
59
60            entrada = scan.nextLine();
61            while (!entrada.equals("quit")){
62
63               if(p.achar(entrada)==1){
64
65                    if(c.getEscrita()==1){   //TEM GENTE ESCREVENDO
```

```
66                          System.out.println("Alguem esta realizando operacao de escrita,
aguarde...");
67                      }
68                      while(c.getEscrita()==1){
69
70                      }
71
72                      if(c.getLeitura()==1){
73                          System.out.println("Alguem esta realizando operacao de leitura,
aguarde...");
74                      }
75                      while(c.getLeitura()==1){
76
77                      }
78
79                      c.setEscrita(1);
80
81                      System.out.print("Topico: ");
82                      String topico = post. nextLine();
83
84                      System.out.print("Texto: ");
85                      String texto = post. nextLine();
86
87
88                      System.out.println(c.postar(nome, topico, texto));
89
90                    c.setEscrita(0);
91
92                }else if(p.achar(entrada)==2){
93
94                      System.out.print("Topico: ");
95                      String topico = post. nextLine();
96
97                    System.out.println(c.seguir(nome, topico));
98
99
100                }else if(p.achar(entrada)==3){
101
102                     System.out.println("Topico: ");
103                     String topico = post. nextLine();
104
105                     System.out.println(c.pararseguir(nome, topico));
106
107                }else if(p.achar(entrada)==4){
108
109                      if(c.getEscrita()==1){
110                         System.out.println("Alguem esta realizando operacao de escrita,
aguarde...");
111                     }
112
113                     while(c.getEscrita()==1){
114
115                     }
116
117                     c.setLeitura(1);
118
119                     System.out.print("Dia: ");
120                     String dia = post. nextLine();
121
122                     System.out.print("Mes: ");
123                     String mes = post. nextLine();
124
125                     System.out.print("Ano: ");
126                     String ano = post. nextLine();
127
128                     List<String> lista = c.pegarPosts(nome, dia, mes, ano);
129
130
131
132
System.out.println("-----------------------------------------");
```

```
133                     System.out.println("");
134                      for(int i = 0; i < lista.size();) {
135                          System.out.println("Usuario: "+lista.get(i)+" - Topico: "+
lista.get(i+1) + " - "+ p.getData(lista.get(i+3)));
136                          System.out.println("Texto: "+lista.get(i+2));
137 
System.out.println("-----------------------------------------");
138                          i=i+4;
139                      }
140 
141                      c.setLeitura(0);
142 
143                      System.out.println("");
144 
145 
146                 }else if(p.achar(entrada)==5){
147 
148                     if(c.getEscrita()==1){
149                         System.out.println("Alguem esta realizando operacao de escrita,
aguarde...");
150                     }
151 
152                     while(c.getEscrita()==1){
153 
154                     }
155 
156                     c.setLeitura(1);
157 
158                     System.out.print("Topico: ");
159                     String topi = post. nextLine();
160 
161                     System.out.print("Dia: ");
162                     String dia = post. nextLine();
163 
164                     System.out.print("Mes: ");
165                     String mes = post. nextLine();
166 
167                     System.out.print("Ano: ");
168                     String ano = post. nextLine();
169 
170                     List<String> lista = c.pegarPostsTop(nome,topi, dia, mes, ano);
171 
172                     System.out.println("\n\nPosts:\n");
173 
System.out.println("-----------------------------------------");
174                     System.out.println("");
175                      for(int i = 0; i < lista.size();) {
176                          System.out.println("Usuario: "+lista.get(i)+" - Topico: "+
lista.get(i+1) + " - "+ p.getData(lista.get(i+3)));
177                          System.out.println("Texto: "+lista.get(i+2));
178 
System.out.println("-----------------------------------------");
179                          i=i+4;
180                      }
181 
182                      c.setLeitura(0);
183 
184                      System.out.println("");
185 
186                 }else if(p.achar(entrada)==6){
187                    System.out.println(p.ajudar());
188                 }else if(p.achar(entrada)==7){
189                     e.getId();
190                     e.souLider();
191                     e.printa();
192 
193                 }else if(p.achar(entrada)==8){
194                     e.abdicar();
195 
196                 }else if(p.achar(entrada)==9){
197                     e.updateRelogio();
```

```
198
199                 }else System.out.println("Comando Invalido");
200
201
202                  System.out.print("-> ");
203                  entrada = scan.nextLine();
204           }
205
206
207
208
209
210         }catch (Exception e){
211             System.out.println(e);
212         }
213     }
```

---

**A documentação para esta classe foi gerada a partir do seguinte arquivo:**

- /home/vitor/Desktop/src/**BlogCliente.java**

## Referência da Classe BlogImp

Diagrama de Hierarquia para BlogImp:

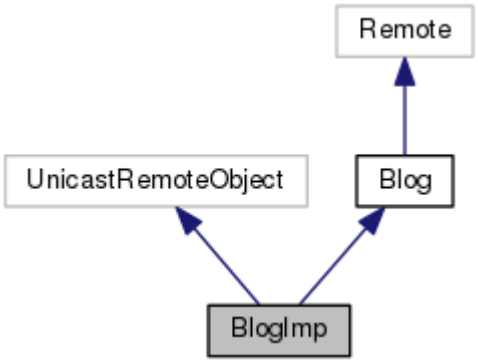
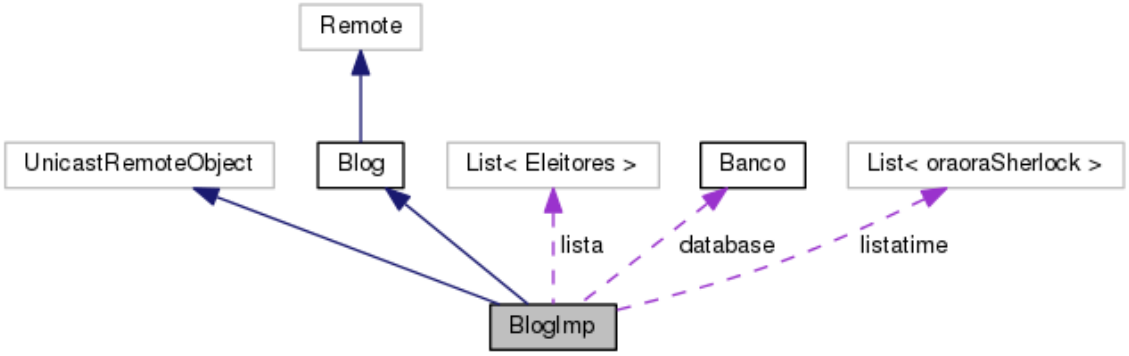



Diagrama de colaboração para BlogImp:




### Métodos Públicos

- int **pegarMedia** () throws RemoteException
- int **jesus** () throws RemoteException
- int **calcularMedia** () throws RemoteException
- int **getLider** () throws RemoteException
- int **getEscrita** () throws RemoteException
- void **setEscrita** (int a) throws RemoteException
- int **getLeitura** () throws RemoteException
- void **setLeitura** (int a) throws RemoteException
- void **setUpdate** (int i) throws RemoteException
- int **getUpdate** () throws RemoteException
- void **setPermitir** (int a) throws RemoteException
- int **getPermitir** () throws RemoteException
- void **zerarLista** () throws RemoteException
- void **setTime** (int id, int time) throws RemoteException
- int **eleger** () throws RemoteException
- String **apresentar** () throws RemoteException
- String **postar** (String usuario, String topico, String texto) throws RemoteException
- String **seguir** (String usuario, String topico) throws RemoteException
- String **pararseguir** (String usuario, String topico) throws RemoteException
- List< String > **pegarPosts** (String usuario, String dia, String mes, String ano) throws RemoteException

- List< String > **pegarPostsTop** (String usuario, String topico, String dia, String mes, String ano) throws RemoteException
- int **definirId** () throws RemoteException
- int **definirLider** (int id, int forca) throws RemoteException
- int **abdicarLideranca** (int id) throws RemoteException
- void **printar** () throws RemoteException

## Métodos Protegidos

- **BlogImp** () throws RemoteException

---

## Construtores & Destrutores

### BlogImp.BlogImp () throws RemoteException `[protected]`

```
27                                                  {
28         super();
29     }
```

---

## Métodos

### int BlogImp.abdicarLideranca (int *id*) throws RemoteException

Implementa **Blog** (*pag.9*).

```
172                                                                 {
173          int i = 0;
174          for(Eleitores a : lista){
175              if(a.getId()==id){
176                  a.setForca(-1);
177                  lista.set(i, a);
178                  forcaLider = -1;
179              }
180              i++;
181          }
182
183          return eleger();
184      }
```

### String BlogImp.apresentar () throws RemoteException

Implementa **Blog** (*pag.9*).

```
116                                                     {
117         String help1 = " post(@username,#topic,text)     |
follow(@username,#topic)\n";
118     String help4 = "
------------------------------------------------------------------\n";
119     String help5 = "\n                              Formato:                      \n";
120     String help2 = " unsubscribe(@username,#topic)   |
retrievetime(@username,date)\n";
121     String help3 = " retrievetopic(@username,#topic,date)\n";
122
123     String apresenta = "\n        Bem vindo ao Blog RPC, digite quit para sair,\n";
124     String apresenta2 = "       caso tenha duvidas digite help para saber mais.\n";
125
126     return (apresenta+apresenta2+help5+help4+help1+help2+help3+help4);
127
128     }
```

### int BlogImp.calcularMedia () throws RemoteException

Implementa **Blog** (*pag.9*).

```
39                                                          {
40          int total = 0;
41          int i = 0;
42          for(oraoraSherlock x : listatime){
43              total = total + x.getHora();
44              i++;
45          }
46          horao = (total/i) + 1;
47          return horao;
48
49      }
```

### int BlogImp.definirId () throws RemoteException

Implementa **Blog** (*pag.9*).

```
154                                                         {
155          id = id+1;
156          return id;
157      }
```

### int BlogImp.definirLider (int *id*, int *forca*) throws RemoteException

Implementa **Blog** (*pag.9*).

```
159                                                                                 {
160
161
162             Eleitores e = new Eleitores();
163             e.setForca(forca);
164             e.setId(id);
165
166             lista.add(e);
167
168
169          return eleger();
170      }
```

### int BlogImp.eleger () throws RemoteException

Implementa **Blog** (*pag.9*).

```
101                                             {
102         for(Eleitores x : lista){
103             if(x.getForca()>forcaLider){
104                 idLider = x.getId();
105                 forcaLider = x.getForca();
106             }
107         }
108
109         if(forcaLider==-1){
110             idLider = 0;
111             forcaLider = forca;
112         }
113         return idLider;
114     }
```

### int BlogImp.getEscrita () throws RemoteException

Implementa **Blog** (*pag.9*).

```
55                                                      {
56          return escrita;
57      }
```

### int BlogImp.getLeitura () throws RemoteException

Implementa **Blog** (*pag.9*).

```
63                                                        {
64          return leitura;
65      }
```

### int BlogImp.getLider () throws RemoteException

Implementa **Blog** (*pag.9*).

```
51                                                    {
52          return idLider;
53      }
```

### int BlogImp.getPermitir () throws RemoteException

Implementa **Blog** (*pag.9*).

```
84                                                      {
85          return permitir;
86      }
```

### int BlogImp.getUpdate () throws RemoteException

Implementa **Blog** (*pag.10*).

```
75                                                     {
76          return solicitaUpdate;
77
78      }
```

### int BlogImp.jesus () throws RemoteException

Implementa **Blog** (*pag.10*).

```
35                                                  {
36          return horao;
37      }
```

### String BlogImp.pararseguir (String *usuario*, String *topico*) throws RemoteException

Implementa **Blog** (*pag.10*).

```
140
{
141          database.pararDeseguir(usuario, topico);
142          return "Voce parou de seguir o topico "  + topico;
143       }
```

### int BlogImp.pegarMedia () throws RemoteException

Implementa **Blog** (*pag.10*).

```
31                                                      {
32          return horao;
```

```
33     }
```

### List<String> BlogImp.pegarPosts (String *usuario*, String *dia*, String *mes*, String *ano*) throws RemoteException

Implementa **Blog** (*pag.10*).

```
145
{
146         return database.pegarPosts(usuario, dia, mes, ano);
147     }
```

### List<String> BlogImp.pegarPostsTop (String *usuario*, String *topico*, String *dia*, String *mes*, String *ano*) throws RemoteException

Implementa **Blog** (*pag.10*).

```
150
{
151           return database.pegarPostsTopico(usuario, topico, dia, mes, ano);
152      }
```

### String BlogImp.postar (String *usuario*, String *topico*, String *texto*) throws RemoteException

Implementa **Blog** (*pag.10*).

```
130
{
131         database.postar(usuario, topico, texto);
132         return "Postado com sucesso !";
133      }
```

### void BlogImp.printar () throws RemoteException

Implementa **Blog** (*pag.10*).

```
186                                                 {
187      for(Eleitores a : lista){
188              System.out.println("id: "+a.getId()+" - forca: "+a.getForca());
189
190          }
191      }
```

### String BlogImp.seguir (String *usuario*, String *topico*) throws RemoteException

Implementa **Blog** (*pag.10*).

```
135                                                                                 {
136           database.seguir(usuario, topico);
137         return "Voce esta seguindo o topico " + topico;
138      }
```

### void BlogImp.setEscrita (int *a*) throws RemoteException

Implementa **Blog** (*pag.10*).

```
59                                                      {
60         escrita = a;
61     }
```

### void BlogImp.setLeitura (int *a*) throws RemoteException

Implementa **Blog** (*pag.10*).

```
67                                                          {
68          leitura = a;
69      }
```

### void BlogImp.setPermitir (int *a*) throws RemoteException

Implementa **Blog** (*pag.11*).

```
80                                                         {
81         permitir=a;
82      }
```

### void BlogImp.setTime (int *id*, int *time*) throws RemoteException

Implementa **Blog** (*pag.11*).

```
93                                                                  {
94
95           oraoraSherlock e = new oraoraSherlock();
96           e.setHora(time);
97           e.setId(id);
98           listatime.add(e);
99      }
```

### void BlogImp.setUpdate (int *i*) throws RemoteException

Implementa **Blog** (*pag.11*).

```
71                                                        {
72         solicitaUpdate = i;
73      }
```

### void BlogImp.zerarLista () throws RemoteException

Implementa **Blog** (*pag.11*).

```
88                                                  {
89          listatime = new ArrayList<oraoraSherlock>();
90      }
```

---

**A documentação para esta classe foi gerada a partir do seguinte arquivo:**

- /home/vitor/Desktop/src/**BlogImp.java**

# Referência da Classe BlogServidor

## Métodos Públicos Estáticos

- static void **main** (String[] args)

---

## Métodos

### static void BlogServidor.main (String[] *args*) `[static]`

```
13                                          {
14          new BlogServidor();
15      }
```

---

**A documentação para esta classe foi gerada a partir do seguinte arquivo:**

- /home/vitor/Desktop/src/**BlogServidor.java**

# Referência da Classe eleitor

Diagrama de Hierarquia para eleitor:

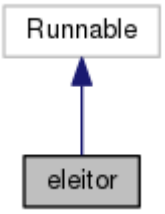
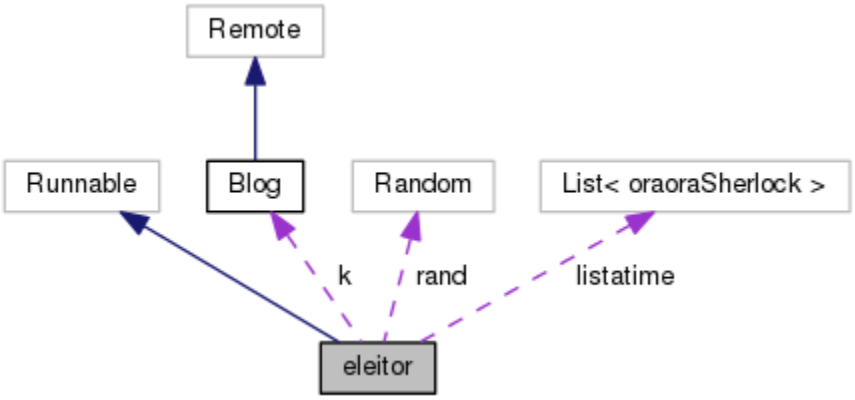



Diagrama de colaboração para eleitor:




## Métodos Públicos

- int **getForca** ()
- void **setAuto** (int a)
- void **setHora** (String b)
- void **souLider** ()
- void **getId** ()
- int **definirId** ()
- void **printa** ()
- int **definirLider** ()
- void **abdicar** ()
- void **run** ()
- int **qual_a_hora** ()
- void **updateRelogio** ()
- String **secs_to_String** (int x)

---

## Descrição Detalhada

**Autor:**

vitor

---

## Métodos

### void eleitor.abdicar ()

```
107                                 {
108             try {
109                 while(k.getPermitir()==0){
110 
111                 }
112 
113                  k.setPermitir(0);
114                  lider = k.abdicarLideranca(id);
115                  k.setPermitir(1);
116 
117             } catch (RemoteException ex) {
118                 Logger.getLogger(eleitor.class.getName()).log(Level.SEVERE, null,
ex);
119             }
120         }
```

### int eleitor.definirId ()

```
76                                  {
77             if(id==-1){
78                 try {
79                     id = k.definirId();
80                 } catch (RemoteException ex) {
81                     Logger.getLogger(eleitor.class.getName()).log(Level.SEVERE, null,
ex);
82                 }
83             }
84             return id;
85         }
```

### int eleitor.definirLider ()

```
95                                      {
96             try {
97                 int antigo = lider;
98                 lider = k.definirLider(id, forca);
99                 if(antigo!=lider) System.out.println("Um novo lider foi defindo !");
100 
101             } catch (RemoteException ex) {
102                 Logger.getLogger(eleitor.class.getName()).log(Level.SEVERE, null,
ex);
103             }
104             return lider;
105         }
```

### int eleitor.getForca ()

```
39                                  {
40             return forca;
41         }
```

### void eleitor.getId ()

```
57                          {
58             System.out.println("Voce é o processo de numero "+id + " com forca: "+forca);
59     }
```

### void eleitor.printa ()

```
87                                  {
88             try {
89                 k.printar();
90             } catch (RemoteException ex) {
91                 Logger.getLogger(eleitor.class.getName()).log(Level.SEVERE, null, ex);
92             }
93         }
```

**int eleitor.qual_a_hora ()**

```
151                                 {
152             int hr = 0;
153             String base="";
154 
155             if(auto==1) base =  new SimpleDateFormat("HH:mm:ss").format(new Date());
156             else base = hora;
157 
158             String[] aux = base.split(":");
159 
160             String h = aux[0];
161             String m = aux[1];
162             String s = aux[2];
163 
164             int hh = Integer.parseInt(h);
165             int mm = Integer.parseInt(m);
166             int ss = Integer.parseInt(s);
167 
168             hr = ss + (mm*60) + (hh*60*60);
169             //System.out.println("Horario: "+secs to String(hr));
170 
171             return hr;
172 
173         }
```

**void eleitor.run ()**

```
122                             {
123 
124             int antigo;
125 
126              try {
127                 while(true){
128                     while(k.getPermitir()==1){
129                         k.setPermitir(0);
130                         antigo = lider;
131                         lider = k.eleger();
132                         if(antigo!=lider) System.out.println("Um novo lider foi
defindo !");
133                         if(k.getUpdate()==1) k.setTime(id, qual_a_hora());
134                         if(k.getUpdate()==2){
135                             int sada= k.jesus();
136                             System.out.println("Novo horario:
"+secs_to_String(sada));
137                         }
138                         k.setPermitir(1);
139                         TimeUnit.SECONDS.sleep(1);
140                     }
141                 }
142             } catch (RemoteException ex) {
143                 Logger.getLogger(eleitor.class.getName()).log(Level.SEVERE, null,
ex);
144             } catch (InterruptedException ex) {
145                 Logger.getLogger(eleitor.class.getName()).log(Level.SEVERE, null,
ex);
146             }
147 
148 
149         }
```

**String eleitor.secs_to_String (int *x*)**

```
202                                             {
203             int total = x;
204             int h = (int) floor(x/3600);
205             total = total - (3600*h);
206             int m = (int) floor(total/60);
207             total = total - (60*m);
208 
209 
210             String ss = "";
```

```
211             if(h<=9) ss = "0"+Integer.toString(h) + ":";
212             else ss = Integer.toString(h)+":";
213             if(m<=9) ss = ss + "0"+ Integer.toString(m)+":";
214             else ss = ss + Integer.toString(m)+":";
215             if(total<=9) ss = ss + "0" + Integer.toString(total);
216             else ss = ss +Integer.toString(total);
217
218            return ss;
219         }
```

### void eleitor.setAuto (int *a*)

```
43                                      {
44             auto = a;
45         }
```

### void eleitor.setHora (String *b*)

```
47                                       {
48             hora = b;
49         }
```

### void eleitor.souLider ()

```
52                                  {
53             if(id==lider) System.out.println("Voce é o lider");
54             else System.out.println("O lider é o processo de numero "+lider);
55         }
```

### void eleitor.updateRelogio ()

```
179                                      {
180             try {
181                 if(id==lider){
182
183                     k.setUpdate(1);
184                     TimeUnit.SECONDS.sleep(1);
185                     k.setUpdate(0);
186                     int velho = qual_a_hora();
187                     hora = secs to String(k.calcularMedia());
188                     System.out.println("Seu horario: "+secs to String(velho)+"
Horario calculado: "+secs to String(k.calcularMedia()));
189                     k.zerarLista();
190                     k.setUpdate(2);
191                     TimeUnit.SECONDS.sleep(1);
192                     k.setUpdate(0);
193                 }else System.out.println("Voce n é o lider");
194             } catch (RemoteException ex) {
195                 Logger.getLogger(eleitor.class.getName()).log(Level.SEVERE, null,
ex);
196             } catch (InterruptedException ex) {
197                 Logger.getLogger(eleitor.class.getName()).log(Level.SEVERE, null,
ex);
198             }
199
200         }
```

---

**A documentação para esta classe foi gerada a partir do seguinte arquivo:**

- /home/vitor/Desktop/src/**eleitor.java**

# Referência da Classe oraoraSherlock

## Métodos Públicos

- int **getId** ()
- void **setId** (int id)
- int **getHora** ()
- void **setHora** (int hora)

---

## Métodos

### int oraoraSherlock.getHora ()

**Retorna:**

the hora

```
21                          {
22          return hora;
23      }
```

### int oraoraSherlock.getId ()

**Retorna:**

the id

```
7                           {
8           return id;
9       }
```

### void oraoraSherlock.setHora (int *hora*)

**Parâmetros:**

| | |
|---|---|
| *hora* | the hora to set |

```
28                                  {
29          this.hora = hora;
30      }
```

### void oraoraSherlock.setId (int *id*)

**Parâmetros:**

| | |
|---|---|
| *id* | the id to set |

```
14                                  {
15          this.id = id;
16      }
```

---

**A documentação para esta classe foi gerada a partir do seguinte arquivo:**

- /home/vitor/Desktop/src/**oraoraSherlock.java**

# Referência da Classe palavras

## Métodos Públicos

- String **getNome** (String a)
- String **getTopico** (String a)
- int **achar** (String texto)
- String **ajudar** ()
- String **getData** (String data)
- void **limpar** ()

---

## Descrição Detalhada

**Autor:**

vitor

---

## Métodos

### int palavras.achar (String *texto*)

```
49                                  {
50
51         if (texto.indexOf("post") >=0) return 1;
52     if ( texto.indexOf("follow") >=0 ) return 2;
53     if ( texto.indexOf("unsubscribe") >=0 ) return 3;
54     if ( texto.indexOf("retrievetime") >=0 ) return 4;
55     if ( texto.indexOf("retrievetopic") >=0 ) return 5;
56     if ( texto.indexOf("help")>=0) return 6;
57         if ( texto.indexOf("info")>=0) return 7;
58         if ( texto.indexOf("abdicar")>=0) return 8;
59         if ( texto.indexOf("setTime")>=0) return 9;
60
61     else return 0;
62     }
```

### String palavras.ajudar ()

```
64                            {
65
66         String help1 = " post(@username,#topic,text)     |
follow(@username,#topic)\n";
67     String help4 = "
-----------------------------------------------------------------\n";
68     String help5 = "\n                              Formato:                      \n";
69     String help2 = " unsubscribe(@username,#topic)   |
retrievetime(@username,date)\n";
70     String help3 = " retrievetopic(@username,#topic,date)\n";
71
72     String posta = "\n Post: Voce postar sobre um determinado topico e os\n outros usuarios
que seguem esse topico podem ver";
73     String segui = "\n Follow: Ao seguir um determinado topico voce podera\n ver todos
os posts relacionada a esse topico\n";
74     String unsegui = "\n Unsubscribe: Voce para de seguir determinado topico\n";
75     String retri = "\n Retrieve Time: Voce pode ver todos os posts feitos\n a partir da
data informada dos topicos que segue\n";
76     String retrit = "\n Retrieve Topic: Voce pode ver todos os posts feitos de\n determinado
topico a partir da data informada\n";
77
78
```

```
79
80      return
(help5+help4+help1+help2+help3+help4+posta+"\n"+segui+unsegui+retri+retrit);
81      }
```

### String palavras.getData (String *data*)

```
83                                      {
84
85          String ano = ""+data.charAt(0)+data.charAt(1)+data.charAt(2)+data.charAt(3);
86          String mes = ""+data.charAt(5)+data.charAt(6);
87          String dia = ""+data.charAt(8)+data.charAt(9);
88
89          String aux = dia+"/"+mes+"/"+ano;
90
91
92          return aux;
93      }
```

### String palavras.getNome (String *a*)

```
13                                    {
14
15          int indice = 5;
16          String aux="";
17
18          while(a.charAt(indice)!=','){
19              aux = aux+a.charAt(indice);
20              indice++;
21          }
22
23          return aux;
24      }
```

### String palavras.getTopico (String *a*)

```
26                                      {
27
28          int indice = 5;
29          String aux="";
30
31          while(a.charAt(indice)!=','){
32              indice++;
33          }
34
35          indice++;
36
37          while(a.charAt(indice)!=','){
38              aux = aux+a.charAt(indice);
39              indice++;
40          }
41
42          return aux;
43
44
45      }
```

### void palavras.limpar ()

```
95                          {
96          String ESC = "\033[";
97          System.out.print(ESC + "2J");
98      }
```

---

**A documentação para esta classe foi gerada a partir do seguinte arquivo:**

- /home/vitor/Desktop/src/**palavras.java**

# Referência da Classe Semaforo

Diagrama de Hierarquia para Semaforo:

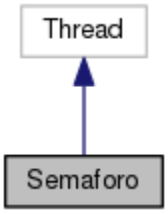
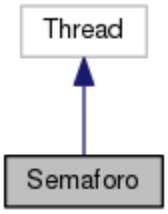



Diagrama de colaboração para Semaforo:




## Métodos Públicos

- **Semaforo** (int id, Semaphore semaphore)
- void **run** ()

---

## Construtores & Destrutores

### Semaforo.Semaforo (int *id*, Semaphore *semaphore*)

```
8                                                   {
9          this.idThread = id;
10          this.semaforo = semaphore;
11      }
```

---

## Métodos

### void Semaforo.run ()

```
33                          {
34          entrarRegiaoNaoCritica();
35          try{
36              semaforo.acquire();
37              entrarRegiaoCritica();
38          }catch (InterruptedException e) {
39          e.printStackTrace();
40          }finally {
41              semaforo.release();
42          }
43      }
```

---

**A documentação para esta classe foi gerada a partir do seguinte arquivo:**

- /home/vitor/Desktop/src/**Semaforo.java**

## Referência da Classe Usuario

Diagrama de Hierarquia para Usuario:

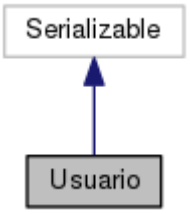
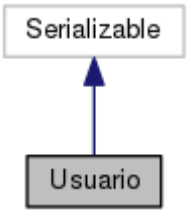



Diagrama de colaboração para Usuario:




### Métodos Públicos

- String **getNome** ()
- void **setNome** (String nome)
- String **getTopico** ()
- void **setTopico** (String topico)
- String **getTexto** ()
- void **setTexto** (String texto)
- String **getData** ()
- void **setData** (String data)

---

### Métodos

#### String Usuario.getData ()

**Retorna:**

the data

```
51                                  {
52          return data;
53      }
```

#### String Usuario.getNome ()

**Retorna:**

the nome

```
9                                   {
10          return nome;
11      }
```

### String Usuario.getTexto ()

**Retorna:**

the texto

```
37                                  {
38          return texto;
39      }
```

### String Usuario.getTopico ()

**Retorna:**

the topico

```
23                                   {
24          return topico;
25      }
```

### void Usuario.setData (String *data*)

**Parâmetros:**

| | |
|---|---|
| *data* | the data to set |

```
58                                        {
59          this.data = data;
60      }
```

### void Usuario.setNome (String *nome*)

**Parâmetros:**

| | |
|---|---|
| *nome* | the nome to set |

```
16                                        {
17          this.nome = nome;
18      }
```

### void Usuario.setTexto (String *texto*)

**Parâmetros:**

| | |
|---|---|
| *texto* | the texto to set |

```
44                                          {
45          this.texto = texto;
46      }
```

### void Usuario.setTopico (String *topico*)

**Parâmetros:**

| | |
|---|---|
| *topico* | the topico to set |

```
30                                            {
31          this.topico = topico;
32      }
```

---

**A documentação para esta classe foi gerada a partir do seguinte arquivo:**

- /home/vitor/Desktop/src/**Usuario.java**

# Arquivos

## Referência do Arquivo /home/vitor/Desktop/src/Banco.java

### Componentes

- class **Banco**

# Referência do Arquivo /home/vitor/Desktop/src/Blog.java

## Componentes

- interface **Blog**

# Referência do Arquivo /home/vitor/Desktop/src/BlogCliente.java

## Componentes

- class **BlogCliente**

# Referência do Arquivo /home/vitor/Desktop/src/BlogImp.java

## Componentes

- class **BlogImp**

# Referência do Arquivo /home/vitor/Desktop/src/BlogServidor.java

## Componentes

- class **BlogServidor**

# Referência do Arquivo /home/vitor/Desktop/src/eleitor.java

## Componentes

- class **eleitor**

# Referência do Arquivo /home/vitor/Desktop/src/Eleitores.java

## Componentes

- class **Eleitores**

# Referência do Arquivo /home/vitor/Desktop/src/oraoraSherlock.java

## Componentes

- class **oraoraSherlock**

# Referência do Arquivo /home/vitor/Desktop/src/palavras.java

## Componentes

- class **palavras**

# Referência do Arquivo /home/vitor/Desktop/src/Semaforo.java

## Componentes

- class **Semaforo**

# Referência do Arquivo /home/vitor/Desktop/src/Usuario.java

## Componentes

- class **Usuario**

# Índice

INDEX