

Solution générique de calcul GRID
exploitant des messageries instantanées
(Java / Python, XML, XMPP / IRC)

Joffrey Hérard

Responsable : Olivier Flauzac

2016-2017

Table des matières

1	Introduction	4
2	Les Acteurs	5
3	Les Échanges	6
4	Les Erreurs	7
4.1	Les Problèmes d'exécution	7
4.2	Les Problèmes réseaux	8
4.3	Gestions des erreurs	9
4.3.1	Gestions des erreurs sur l'exécution	9
4.3.2	Gestions des erreurs sur le réseaux	9
5	Modélisation	10
5.1	Connexions	10
5.1.1	Connexions du/des Provider(s)	10
5.1.2	Connexions du/des Worker(s)	10
5.2	Représentation des JOBS	11
5.3	Représentation des fichiers de lignes de commande	12
5.4	Les différentes fonctions principales	13
5.4.1	Contraintes	13
5.4.2	Split	13
5.4.3	Exec	16
5.4.4	Build	16
5.5	Description d'une exécution quelconque	17
6	Conclusion	18
7	Annexes	19
7.1	Organisation du Projet	19
7.1.1	Outils et langages	20

7.2	Code	21
7.2.1	XML	22
7.2.2	Perl	24
7.2.3	Java	25

Chapitre 1

Introduction

Sujet : Solution générique de calcul GRID exploitant des messageries instantanées (Java / Python, XML, XMPP / IRC) Durant ce TER, la mise en place d'un système de calcul repartie entre plusieurs machine avec l'évaluation de possibilité d'exécutions ou non par la machine cible, il fallait aussi évaluer quels échanges allaient être réalisés par les acteurs durant une exécution type et ceci en avec le protocole XMPP ou IRC .

Chapitre 2

Les Acteurs

Nous avons donc deux genres d'acteur pour chaque travail différent disponibles

- Fournisseur de travail/Provider, unique pour chaque travail.
- Des travailleurs/Workers, de 1 à n , n définit par le problèmes.

Chapitre 3

Les Échanges

Voici la liste des différents message qui transitent a travers une exécution type.

1. Nous avons en premier le message de type "ENVOI JOB" il contient :
 - l'identifiant du problème,
 - Le code des contraintes,
 - Le code a exécuter,
 - La ligne de commande pour l'exécuter.
2. Ensuite il y a le message ou le workers signale qu'il est prêt il contiens juste un message pour signale dans une chaîne de caractère " Je suis prêt".
3. Il y a enfin le message qui renvoi le résultat "REPONSE JOB" il contient :
 - L'identifiant pour savoir si le code a pu être exécuté.
 - L'identifiant du problème.
 - La valeur du retour de l'exécution.
 - Code de contraintes, si on a pas pu exécuter .
 - Code exécutable, si on a pas pu exécuter .
 - Ligne de commande associe, si on a pas pu exécuter .

Voici la liste des fichiers schéma XML associe ainsi que leurs locations au sein du projet :

- "ENVOI_JOB" = ../Schema_XML/ENVOI_RECEPTION.xsd.
- "READY"= ../Schema_XML/ENVOI_RECEPTION.xsd
- "REPONSE_JOB"= ../Schema_XML/ENVOI_RECEPTION.xsd.

Tout les codes du projet sont présenter en annexe.

Chapitre 4

Les Erreurs

4.1 Les Problèmes d'exécution

Les problèmes qui peuvent opérer à travers le système, sont d'abord pour la partie XMPP :

1. Mauvais nom de domaine
2. Problème de Chatroom déjà existante
3. Problème d'exécution : aucun worker peut exécuter le code, comment le détecter ?
- 4.

Les problèmes qui peuvent opérer à travers le système, sont d'abord pour la partie IRC :
Nous avons exactement les mêmes soucis

4.2 Les Problèmes réseaux

Nous avons plusieurs problèmes lie au réseaux quelque soit le protocole utilise :

1. Latence/Impossible a établir une connexion a la Chatroom
2. Latence/Impossible a envoyer un message d'un Provider vers un Worker
3. Latence/Impossible a envoyer un message d'un Worker vers un Provider
4. Un Worker est déconnecte en plein milieu de sa tache
5. Un Provider est déconnecte durant l'attente d'une réponse sur un JOB

4.3 Gestions des erreurs

4.3.1 Gestions des erreurs sur l'exécution

1. Mauvais nom de domaine → Redemander le nom de domaine jusqu'à validation .
2. Problème de Chatroom déjà existante → Message d'erreur un problème exactement identique est en cours d'exécution .
3. Problème d'exécution : aucun worker peut exécuter le code, comment le détecter ?
→ Mis en place d'un tableau de variable booléenne au départ initialisé à faux, si un worker renvoie avec une impossibilité d'exécution du code dictée par le code contraint, alors on met à vrai et on redistribue. Si aucun est capable on arrête l'exécution.

4.3.2 Gestions des erreurs sur le réseau

1. Latence/Impossible d'établir une connexion à la Chatroom → Message qui explicite le fait d'aller voir un Administrateur Réseaux
2. Latence/Impossible d'envoyer un message d'un Provider vers un Worker → Message qui explicite le fait d'aller voir un Administrateur Réseaux
3. Latence/Impossible d'envoyer un message d'un Worker vers un Provider → Message qui explicite le fait d'aller voir un Administrateur Réseaux
4. Un Worker est déconnecté en plein milieu de sa tâche → Détecteur de présence permis par le protocole XMPP sur une ChatRoom MultiUser
5. Un Provider est déconnecté durant l'attente d'une réponse sur un JOB → Évaluation de présence d'un Provider, si aucun alors arrêter le Job en cours, ou mis en place d'un Timeout.

Chapitre 5

Modélisation

5.1 Connexions

5.1.1 Connexions du/des Provider(s)

5.1.2 Connexions du/des Worker(s)

5.2 Représentation des JOBS

Nous allons dans cette partie du rapport montrer la représentation nécessaire et désirer pour représenter un travail. Donc un problèmes c'est quoi ?

- Identifiant d'un problème, un entier de 0 à n.
- Code de contraintes, ce code est forcément un code Perl avec un code de retour bien particulier 0 pour non exécutable et 3 pour exécutable et donc que l'on peut exécuter.
- Type du fichier par exemple ".c", ".cpp", ".cc", ".java", ".pl" etc..
- Code d'exécution, peut importer son langage. on peut l'exécuter si le code contraintes l'a validé
- La ligne de commande pour exécuter le code par exemple "perl monfichier.pl"

```
<xs:schema>
  <xs:element name="JOB">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="code_Perl" type="xs:string"/>
        <xs:element name="code_exec" type="xs:string"/>
        <xs:element name="cmd" type="xs:string"/>
        <xs:element name="rang" type="xs:Integer"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

5.3 Représentation des fichiers de lignes de commande

```
perl calcul.pl 2 3,  
perl calcul.pl 3 3,  
perl calcul.pl 4 3,  
perl calcul.pl 5 3,  
perl calcul.pl 26 3,  
perl calcul.pl 29 3#
```

5.4 Les différentes fonctions principales

5.4.1 Contraintes

```
#!/usr/bin/perl
use v5.14;

my $osname = $^O;

if( $osname eq 'MSWin32' ){
    print "We are on windows";
    # work around for historical reasons
    exit(1);
} else {
    print "Test si on est sur Mac\n";
    if ( $osname eq 'darwin' ) {
        print "We are on Mac OS X ...\n";
        exit(2);
    }
    else {
        print "Test si on est sur Linux\n";
        if ( $osname eq 'linux' ) {
            print "We are on Linux...\n";
            exit(3);
        }
    }
}
```

5.4.2 Split

```
1 public int split(ArrayList<identity> Liste_user,int Nombre_Participants,
2     String ProblemeCourant,XmppManager xmppManager,String choix)
3 {
4     for(int i=0;i<Nombre_Participants-1;i++)
5     {
6         String buddyJID = Liste_user.get(i).getId();
7         String buddyName = Liste_user.get(i).getName();
8
9         try {
10             xmppManager.createEntry(buddyJID, buddyName);
```

```

11
12      /* Recherche de la liste des exec*/
13
14      DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
15      DocumentBuilder builder = factory.newDocumentBuilder();
16
17      File fileXML = new File("DB_JOBS/"+choix);
18
19      Document xml = builder.parse(fileXML);
20
21      Element root = xml.getDocumentElement();
22
23      XPathFactory xpf = XPathFactory.newInstance();
24
25      XPath path = xpf.newXPath();
26
27
28
29      String expression = "/JOB/code_exec";
30
31      String strexec = (String)path.evaluate(expression, root);
32      System.out.print("DEBUT STR EXEC");
33      System.out.print(strexec);
34      System.out.print("FIN STR EXEC");
35
36      expression = "/JOB/code_Perl";
37
38      String strperl = (String)path.evaluate(expression, root);
39
40      expression = "/JOB/cmd";
41
42      String strcmd = (String)path.evaluate(expression, root);
43
44
45      String delims = "[,]";
46      String[] tokens =strcmd.split(delims);
47      System.out.print("affichage des tokens");
48      tokens[tokens.length-1]=tokens[tokens.length-1].substring(
49      0, tokens[tokens.length-1].length()-1
50      );
51      for(int j=0;j<tokens.length;j++)

```

```

52         System.out.println(j+" : "+tokens[j]);
53
54     final Document document= builder.newDocument();
55
56     final Element racine = document.createElement("JOB");
57     document.appendChild(racine);
58     final Element exec = document.createElement("exec");
59     exec.appendChild(document.createTextNode(strexec));
60
61     final Element contraintes = document.createElement("contraintes");
62     contraintes.appendChild(document.createTextNode(strperl));
63
64
65     final Element cmd = document.createElement("cmd");
66     cmd.appendChild(document.createTextNode(tokens[i]));
67     final Element id = document.createElement("id");
68     id.appendChild(document.createTextNode(""+i));
69     racine.appendChild(id);
70     racine.appendChild(contraintes);
71     racine.appendChild(exec);
72     racine.appendChild(cmd);
73
74     final TransformerFactory transformerFactory = TransformerFactory.newInstance();
75     final Transformer transformer = transformerFactory.newTransformer();
76     final DOMSource source = new DOMSource(document);
77     final StreamResult sortie = new StreamResult(new File("JOB_SEND/XML_send_"+i));
78     transformer.transform(source, sortie);
79
80     String Probleme_individuel=FileToString("JOB_SEND/XML_send_"+i);
81
82     xmppManager.sendMessage(Probleme_individuel, buddyName+"@"+xmppManager.NOM_HOTE);
83
84     }catch (Exception e) {
85         // TODO Auto-generated catch block
86         e.printStackTrace();
87     }
88     // Ici on fait apelle a la fonction split
89
90 }
91
92 return 0;

```

93 }

5.4.3 Exec

```
1  #!/usr/bin/perl
2  use v5.14;
3
4  exit $ARGV[0] +$ARGV[1] ;
```

5.4.4 Build

```
1  String from = message.getFrom();
2  String body = message.getBody();
3  System.out.println(String.format("Received message '%1$s' from %2$s", body, from));
4  // on regarde l'en tete du message apparence a un message xml reponse tel que lid est 1 si r
5  //si impossible a executer dans le cas echeant un troisieme champ correspond a lid du job non
6  //et 0 si envoie travail
7
8  // on procede donc au build un script ici un peu fictif mais pour prolonger pour voir si ca m
9  // ici on va juste sommer les results choses assez simple
10
11 String regex="[,]";
12 String[] en_tete = body.split(regex);
13
14 // indice 0 = en tete indice 1 = res
15 if(Integer.parseInt(en_tete[0])==1){
16 retour_Providing+=Integer.parseInt(en_tete[1]);
17 recu++;
18 if(recu==envoyer)
19 {
20     travail_terminer=true;
21 }
```

5.5 Description d'une exécution quelconque

Chapitre 6

Conclusion

Chapitre 7

Annexes

7.1 Organisation du Projet

- /
- /bin
 - Tout les fichiers .class
 - Images
 - Schema
- /DB_JOBS
 - Calculatoire.xml
 - Calculatoire2.xml
 - Langford.xml
 - etc..
- /Echantillon_Script_Cmd
 - Robin.dc
 - Toto.dc
 - etc..
- /Echantillon_Script_Exec
 - calcul.pl
- /Echantillon_Script_Perl
 - DitributionContraintes.pl
- /JDOM
- /JOB_REC
 -
 - /DATA_EXTRACT
 - fichier_extraits
 - xml_receive.xml
- /JOB_SEND

- XML_send_0.xml
- /openfire
- /Rapport
 - TER_Joffrey_Herard.pdf
- /Schema_XML
 - BDD_JOB.xsd
 - ENVOI_RECEPTION.xsd
 - JOB_REP.xsd
- /smack_3_1_0
- /src
 - Tout les fichiers .java
- README.md

7.1.1 Outils et langages

Le projet a été programme en Java version : "1.8.0_111" sous Eclipse

7.2 Code

7.2.1 XML

BDD_JOB.xsd

```
1 <xs:schema>
2   <xs:element name="JOB">
3     <xs:complexType>
4       <xs:sequence>
5         <xs:element name="code_Perl" type="xs:string"/>
6         <xs:element name="code_exec" type="xs:string"/>
7         <xs:element name="cmd" type="xs:string"/>
8         <xs:element name="rang" type="xs:Integer"/>
9       </xs:sequence>
10    </xs:complexType>
11  </xs:element>
12 </xs:schema>
```

ENVOI_RECEPTION.xsd

```
1 <xs:schema>
2   <xs:element name="JOB">
3     <xs:complexType>
4       <xs:sequence>
5         <xs:element name="id" type="xs:Integer"/>
6         <xs:element name="contraintes" type="xs:string"/>
7         <xs:element name="exec" type="xs:string"/>
8         <xs:element name="cmd" type="xs:string"/>
9       </xs:sequence>
10    </xs:complexType>
11  </xs:element>
12 </xs:schema>
```

JOB_REP.xsd

```
1 <xs:schema>
2   <xs:element name="JOB_REP">
3     <xs:complexType>
4       <xs:sequence>
5         <xs:element name="id_retour" type="xs:Integer"/>
6         <xs:element name="idduPRb" type="xs:Integer"/>
7         <xs:element name="reponse" type="xs:string"/>
8         <xs:element name="code_Perl" type="xs:string"/>
9         <xs:element name="exec" type="xs:string"/>
10        <xs:element name="cmd" type="xs:string"/>
11      </xs:sequence>
12    </xs:complexType>
13  </xs:element>
14 </xs:schema>
```

7.2.2 Perl

Les exemple de fichier de contrainte écrit en Perl DistributionContraintes.pl

```
1  #!/usr/bin/perl
2  use v5.14;
3
4  my $osname = $^O;
5
6
7  if( $osname eq 'MSWin32' ){
8      print "We are on windows";
9      # work around for historical reasons
10     exit(1);
11 } else {
12     print "Test si on est sur Mac\n";
13     if ($osname eq 'darwin') {
14         print "We are on Mac OS X ...\n";
15         exit(2);
16     }
17     else {
18         print "Test si on est sur Linux\n";
19         if ($osname eq 'linux') {
20             print "We are on Linux...\n";
21             exit(3);
22         }
23     }
24 }
```

7.2.3 Java

ButtonAjout.java :

```
1  import java.awt.Color;
2  import java.awt.FlowLayout;
3  import java.awt.GradientPaint;
4  import java.awt.Graphics;
5  import java.awt.Graphics2D;
6  import java.awt.Image;
7  import java.awt.event.MouseEvent;
8  import java.awt.event.MouseListener;
9
10 import java.io.File;
11 import java.io.IOException;
12
13 import javax.imageio.ImageIO;
14
15 import javax.swing.JButton;
16 import javax.swing.JFileChooser;
17 import javax.swing.JFrame;
18 import javax.swing.JLabel;
19 import javax.swing.JOptionPane;
20 import javax.swing.JTextField;
21 import javax.swing.filechooser.FileNameExtensionFilter;
22
23 import javax.xml.XMLConstants;
24 import javax.xml.transform.Source;
25 import javax.xml.transform.stream.StreamSource;
26 import javax.xml.validation.*;
27
28 import java.io.*;
29 import org.jdom2.*;
30 import org.jdom2.output.*;
31
32
33 @SuppressWarnings("unused")
34 public class ButtonAjout extends JButton implements MouseListener {
35
36     /**
37     *
```

```

38     */
39     private static final long serialVersionUID = 1L;
40     private String name;
41     private Image img;
42     private JFrame fenetreAjout;
43
44     private JLabel contraintes ;
45     private JLabel exec ;
46     private JLabel nom_p ;
47     private JLabel commande ;
48     private JLabel rang ;
49
50     private JTextField rang1 ;
51
52     private JTextField commande1 ;
53     private JTextField contraintes1 ;
54     private JTextField nom_p1 ;
55     private JTextField exec1 ;
56
57     private JButton bouton_ok ;
58
59
60
61     public ButtonAjout(String str){
62         super(str);
63         this.name = str;
64         this.addMouseListener(this);
65         this.fenetreAjout = new JFrame();
66         this.nom_p= new JLabel("Nom : ");
67         this.nom_p1= new JTextField("Nom du probleme");
68         this.contraintes= new JLabel("Chemin du fichier de contrainte");
69
70         this.exec= new JLabel("Chemin du fichier contenant l'Execution");
71         this.contraintes1 =new JTextField("Path_dufichier_perl_de_Contraintes");
72         this.exec1= new JTextField("path_du_fichier_du_code_de_exec");
73         this.bouton_ok=new JButton("OK");
74         this.commande= new JLabel("Ligne de commande d'execution");
75         this.commande1 =new JTextField("Ligne de commandes a executer");
76         this.rang= new JLabel("Rang du probleme");
77         this.rang1 =new JTextField("Rang");
78

```

```

79
80 }
81
82 public String FileToString(String PathFile)
83 {
84     String fic = "";
85     //lecture du fichier texte
86     try
87     {
88         InputStream ips=new FileInputStream(PathFile);
89         InputStreamReader ipsr=new InputStreamReader(ips);
90         BufferedReader br=new BufferedReader(ipsr);
91         String ligne;
92         while ((ligne=br.readLine())!=null)
93         {
94             System.out.println(ligne);
95             fic+=ligne+"\n";
96         }
97         br.close();
98         ipsr.close();
99         ips.close();
100     }
101     catch (Exception e)
102     {
103         System.out.println(e.toString());
104     }
105     return fic;
106 }
107 public int Ajouter_TYPE_DE_JOBS(String nom,String contraintesT,String exec1T,String commandeT)
108 {
109     /*Ici on va creer le fichier XML dans un dossier propre a la machine */
110     /*On va tester, l'existence du dossier */
111     int retour = -42;
112     String Fichier_Perl;
113     String Fichier_Exec;
114     String Fichier_Commande;
115
116     File d = new File ("DB_JOBS");
117     File f = new File ("DB_JOBS/"+nom+".xml");
118     if (d.exists() && d.isDirectory()){
119         /*Le fichier existe on va faire la creation du fichier XML associer*/

```

```

120     if (!f.exists())
121     {
122         try {
123
124
125             Element JOB = new Element("JOB");
126             Document doc = new Document(JOB);
127
128
129             Fichier_Perl=FileToString(contraintesT);
130
131             Fichier_Exec=FileToString(exec1T);
132             Fichier_Commande=FileToString(commandeT);
133
134             JOB.addContent(new Element("code_Perl").setText(Fichier_Perl));
135
136             JOB.addContent(new Element("code_exec").setText(Fichier_Exec));
137             JOB.addContent(new Element("cmd").setText(Fichier_Commande));
138             JOB.addContent(new Element("rang").setText(rang));
139
140
141
142             // new XMLOutputter().output(doc, System.out);
143             XMLOutputter xmlOutput = new XMLOutputter();
144
145             // display nice nice
146             xmlOutput.setFormat(Format.getPrettyFormat());
147             xmlOutput.output(doc, new FileWriter("DB_JOBS/"+nom+".xml"));
148
149             System.out.println("Fichier Enregistrer!");
150         } catch (IOException io) {
151             System.out.println(io.getMessage());
152         }
153         retour= 0;
154     }
155     else
156     {
157         retour= -1;
158     }
159
160 }else{

```

```

161     System.out.println("Mon répertoire n'existe pas");
162     d.mkdir();
163     try {
164
165         Element JOB = new Element("JOB");
166         Document doc = new Document(JOB);
167
168
169         Fichier_Perl=FileToString(contraintesT);
170
171         Fichier_Exec=FileToString(exec1T);
172
173         Fichier_Commande=FileToString(commandeT);
174
175         JOB.addContent(new Element("code_Perl").setText(Fichier_Perl));
176
177         JOB.addContent(new Element("code_exec").setText(Fichier_Exec));
178         JOB.addContent(new Element("cmd").setText(Fichier_Commande));
179         JOB.addContent(new Element("rang").setText(rang));
180
181
182
183
184         // new XMLOutputter().output(doc, System.out);
185         XMLOutputter xmlOutput = new XMLOutputter();
186
187         // display nice nice
188         xmlOutput.setFormat(Format.getPrettyFormat());
189         xmlOutput.output(doc, new FileWriter("DB_JOBS/"+nom+".xml"));
190
191         System.out.println("Fichier Enregistrer!");
192     } catch (IOException io) {
193         System.out.println(io.getMessage());
194     }
195     retour= 0;
196 }
197
198     return retour;
199 }
200
201 @Override

```

```

202 public void mouseClicked(MouseEvent arg0) {
203     // TODO Auto-generated method stub
204     JFrame fenetreAjout = new JFrame();
205     fenetreAjout.setTitle("Ajout d'une tache dans la base");
206     fenetreAjout.setSize(275, 275);
207     fenetreAjout.setLocationRelativeTo(null);
208     fenetreAjout.setLayout(new FlowLayout());
209     fenetreAjout.add(nom_p);
210     fenetreAjout.add(nom_p1);
211     fenetreAjout.add(contraintes);
212     fenetreAjout.add(contraintes1);
213
214     fenetreAjout.add(exec);
215     fenetreAjout.add(exec1);
216     fenetreAjout.add(commande);
217     fenetreAjout.add(commande1);
218     fenetreAjout.add(rang);
219     fenetreAjout.add(rang1);
220     fenetreAjout.add(bouton_ok);
221     fenetreAjout.setVisible(true);
222
223     commande1.addMouseListener(new MouseListener() {
224
225         public void mouseClicked(MouseEvent e) {
226             JFileChooser fc = new JFileChooser();
227
228             //Cr  er un filtre qui ne s  lectionnera que les fichiers .txt
229             FileNameExtensionFilter ff = new FileNameExtensionFilter("Fichiers liste de comm
230             fc.setFileFilter(ff);
231
232             int returnVal = fc.showOpenDialog(commande1);
233             String nomFic = "";
234             if (returnVal == JFileChooser.APPROVE_OPTION) {
235                 commande1.setText(fc.getSelectedFile().getAbsolutePath());
236             }
237         }
238
239         public void mousePressed(MouseEvent e) {
240
241         }
242

```

```

243     public void mouseReleased(MouseEvent e) {
244
245     }
246
247     public void mouseEntered(MouseEvent e) {
248
249     }
250
251     public void mouseExited(MouseEvent e) {
252
253     }
254
255 });
256 contraintes1.addMouseListener(new MouseListener() {
257
258     public void mouseClicked(MouseEvent e) {
259         JFileChooser fc = new JFileChooser();
260
261         //Cr  er un filtre qui ne s  lectionnera que les fichiers .txt
262         FileNameExtensionFilter ff = new FileNameExtensionFilter("Fichiers Perl", "pl");
263         fc.setFileFilter(ff);
264
265         int returnVal = fc.showOpenDialog(contraintes1);
266         String nomFic = "";
267         if (returnVal == JFileChooser.APPROVE_OPTION) {
268             contraintes1.setText(fc.getSelectedFile().getAbsolutePath());
269         }
270     }
271
272     public void mousePressed(MouseEvent e) {
273
274     }
275
276     public void mouseReleased(MouseEvent e) {
277
278     }
279
280     public void mouseEntered(MouseEvent e) {
281
282     }
283

```

```

284     public void mouseExited(MouseEvent e) {
285
286     }
287
288 });
289
290 exec1.addMouseListener(new MouseListener() {
291
292     public void mouseClicked(MouseEvent e) {
293         JFileChooser fc = new JFileChooser();
294
295         //Cr  er un filtre qui ne s  lectionnera que les fichiers .txt
296         FileNameExtensionFilter ff = new FileNameExtensionFilter("Fichiers executable",
297         fc.setFileFilter(ff);
298
299         int returnVal = fc.showOpenDialog(exec1);
300         String nomFic = "";
301         if (returnVal == JFileChooser.APPROVE_OPTION) {
302             exec1.setText(fc.getSelectedFile().getAbsolutePath());
303         }
304     }
305
306     public void mousePressed(MouseEvent e) {
307
308     }
309
310     public void mouseReleased(MouseEvent e) {
311
312     }
313
314     public void mouseEntered(MouseEvent e) {
315
316     }
317
318     public void mouseExited(MouseEvent e) {
319
320     }
321
322 });
323
324 bouton_ok.addMouseListener(new MouseListener() {

```



```

325
326     public void mouseClicked(MouseEvent e) {
327         int retour =Ajouter_TYPE_DE_JOBS(nom_p1.getText(),contraintes1.getText(),exec1.getT
328         switch(retour)
329         {
330             case -1:
331                 // le fichier existe deja
332
333                 JOptionPane.showMessageDialog(null, "Le fichier existe deja", "Erreur", JOption
334                 break;
335
336             case 0:
337                 // Good
338                 break;
339
340             default:
341                 // Une erreur inconnu
342                 break;
343         }
344     }
345
346     public void mousePressed(MouseEvent e) {
347
348     }
349
350     public void mouseReleased(MouseEvent e) {
351
352     }
353
354     public void mouseEntered(MouseEvent e) {
355
356     }
357
358     public void mouseExited(MouseEvent e) {
359
360     }
361
362     });
363
364 }
365

```

```

366
367     @Override
368     public void mouseEntered(MouseEvent arg0) {
369         // TODO Auto-generated method stub
370
371     }
372
373     @Override
374     public void mouseExited(MouseEvent arg0) {
375         // TODO Auto-generated method stub
376
377     }
378
379     @Override
380     public void mousePressed(MouseEvent arg0) {
381         // TODO Auto-generated method stub
382
383     }
384
385     @Override
386     public void mouseReleased(MouseEvent arg0) {
387         // TODO Auto-generated method stub
388
389     }
390
391     public String getName() {
392         return name;
393     }
394
395     public void setName(String name) {
396         this.name = name;
397     }
398
399     public Image getImg() {
400         return img;
401     }
402
403     public void setImg(Image img) {
404         this.img = img;
405     }
406

```

```

407     public JFrame getFenetreAjout() {
408         return fenetreAjout;
409     }
410
411     public void setFenetreAjout(JFrame fenetreAjout) {
412         this.fenetreAjout = fenetreAjout;
413     }
414
415     public JLabel getContraintes() {
416         return contraintes;
417     }
418
419     public void setContraintes(JLabel contraintes) {
420         this.contraintes = contraintes;
421     }
422
423
424     public JLabel getExec() {
425         return exec;
426     }
427
428     public void setExec(JLabel exec) {
429         this.exec = exec;
430     }
431
432
433     public JLabel getNom_p() {
434         return nom_p;
435     }
436
437     public void setNom_p(JLabel nom_p) {
438         this.nom_p = nom_p;
439     }
440
441     public JTextField getContraintes1() {
442         return contraintes1;
443     }
444
445     public void setContraintes1(JTextField contraintes1) {
446         this.contraintes1 = contraintes1;
447     }

```

```
448
449     public JTextField getNom_p1() {
450         return nom_p1;
451     }
452
453     public void setNom_p1(JTextField nom_p1) {
454         this.nom_p1 = nom_p1;
455     }
456
457
458     public JTextField getExec1() {
459         return exec1;
460     }
461
462     public void setExec1(JTextField exec1) {
463         this.exec1 = exec1;
464     }
465
466
467     public JButton getBouton_ok() {
468         return bouton_ok;
469     }
470
471     public void setBouton_ok(JButton bouton_ok) {
472         this.bouton_ok = bouton_ok;
473     }
474
475     public static long getSerialversionuid() {
476         return serialVersionUID;
477     }
478
479
480 }
```

ButtonContact.java :

```
1  import java.awt.Color;
2  import java.awt.FlowLayout;
3  import java.awt.GradientPaint;
4  import java.awt.Graphics;
5  import java.awt.Graphics2D;
6  import java.awt.Image;
7  import java.awt.event.MouseEvent;
8  import java.awt.event.MouseListener;
9  import java.io.File;
10 import java.io.IOException;
11 import javax.imageio.ImageIO;
12 import javax.swing.JButton;
13 import javax.swing.JFrame;
14 import javax.swing.JLabel;
15
16 @SuppressWarnings("unused")
17 public class ButtonContact extends JButton implements MouseListener {
18
19     /**
20     *
21     */
22     private static final long serialVersionUID = 1L;
23     private String name;
24     private Image img;
25     private JLabel label_creator ;
26     private JLabel label_director ;
27     private JLabel label_version ;
28     private JLabel label_date ;
29     private JLabel label_mail ;
30
31     public ButtonContact(String str){
32         super(str);
33         this.name = str;
34         this.addMouseListener(this);
35         label_creator = new JLabel("Createur : Joffrey Herard");
36         label_director = new JLabel("Responsable : Olivier Flauzac");
37         label_version = new JLabel("Version : 0.1.0");
38         label_date = new JLabel("2016");
39         label_mail = new JLabel("mail : joffrey.herard[at]etudiant.univ-reims.fr");
```

```

40     }
41
42     @Override
43     public void mouseClicked(MouseEvent arg0) {
44         // TODO Auto-generated method stub
45
46         JLabel label_creator = new JLabel("Createur : Joffrey Herard");
47         JLabel label_director = new JLabel("Reponsable : Olivier Flauzac");
48         JLabel label_version = new JLabel("Version : 0.1.0");
49         JLabel label_date = new JLabel("2016");
50         JLabel label_mail = new JLabel("mail : joffrey.herard[at]etudiant.univ-reims.fr");
51
52         JFrame fenetreContact = new JFrame();
53         fenetreContact.setTitle("Contact");
54         fenetreContact.setSize(400, 80);
55         fenetreContact.setLocationRelativeTo(null);
56         fenetreContact.setVisible(true);
57
58         fenetreContact.setLayout(new FlowLayout());
59         fenetreContact.getContentPane().add(label_creator);
60         fenetreContact.getContentPane().add(label_director);
61         fenetreContact.getContentPane().add(label_version);
62         fenetreContact.getContentPane().add(label_date);
63         fenetreContact.getContentPane().add(label_mail);
64
65     }
66
67
68     @Override
69     public void mouseEntered(MouseEvent arg0) {
70         // TODO Auto-generated method stub
71
72     }
73
74     @Override
75     public void mouseExited(MouseEvent arg0) {
76         // TODO Auto-generated method stub
77
78     }
79
80     @Override

```

```

81     public void mousePressed(MouseEvent arg0) {
82         // TODO Auto-generated method stub
83     }
84
85
86     @Override
87     public void mouseReleased(MouseEvent arg0) {
88         // TODO Auto-generated method stub
89     }
90
91
92     public String getName() {
93         return name;
94     }
95
96     public void setName(String name) {
97         this.name = name;
98     }
99
100    public Image getImg() {
101        return img;
102    }
103
104    public void setImg(Image img) {
105        this.img = img;
106    }
107
108    public JLabel getLabel_creator() {
109        return label_creator;
110    }
111
112    public void setLabel_creator(JLabel label_creator) {
113        this.label_creator = label_creator;
114    }
115
116    public JLabel getLabel_director() {
117        return label_director;
118    }
119
120    public void setLabel_director(JLabel label_director) {
121        this.label_director = label_director;

```

```
122     }
123
124     public JLabel getLabel_version() {
125         return label_version;
126     }
127
128     public void setLabel_version(JLabel label_version) {
129         this.label_version = label_version;
130     }
131
132     public JLabel getLabel_date() {
133         return label_date;
134     }
135
136     public void setLabel_date(JLabel label_date) {
137         this.label_date = label_date;
138     }
139
140     public JLabel getLabel_mail() {
141         return label_mail;
142     }
143
144     public void setLabel_mail(JLabel label_mail) {
145         this.label_mail = label_mail;
146     }
147
148     public static long getSerialversionuid() {
149         return serialVersionUID;
150     }
151
152 }
```

ButtonDel.java :

```
1  import java.awt.Color;
2  import java.awt.Dimension;
3  import java.awt.FlowLayout;
4  import java.awt.GradientPaint;
5  import java.awt.Graphics;
6  import java.awt.Graphics2D;
7  import java.awt.Image;
8  import java.awt.event.MouseEvent;
9  import java.awt.event.MouseListener;
10 import java.io.File;
11 import java.io.FileNameFilter;
12 import java.io.IOException;
13 import javax.imageio.ImageIO;
14 import javax.swing.JButton;
15 import javax.swing.JComboBox;
16 import javax.swing.JFileChooser;
17 import javax.swing.JFrame;
18 import javax.swing.filechooser.FileNameExtensionFilter;
19
20 @SuppressWarnings("unused")
21 public class ButtonDel extends JButton implements MouseListener {
22
23     /**
24      *
25      */
26     private static final long serialVersionUID = 1L;
27     private String name;
28     private Image img;
29     private JComboBox<String> comboPrb;
30     private static FileNameFilter xmlFileFilter = new FileNameFilter() {public boolean accept(
31     private File repertoire;
32     private File[] files;
33     private JButton bouton_ok;
34     private File fichier_Choisi;
35     private String choix;
36
37     public ButtonDel(String str){
38         super(str);
39         this.name = str;
```

```

40     this.addMouseListener(this);
41     comboPrb= new JComboBox<String>();
42     bouton_ok= new JButton("OK");
43 }
44
45 @Override
46 public void mouseClicked(MouseEvent arg0) {
47     // TODO Auto-generated method stub
48     JFrame fenetre = new JFrame();
49     fenetre.setTitle("Suppression d'une tache dans la base");
50     fenetre.setSize(250, 275);
51     fenetre.setLocationRelativeTo(null);
52     fenetre.setLayout(new FlowLayout());
53     repertoire = new File("DB_JOBS");
54     files =repertoire.listFiles(xmlFileFilter);
55
56     comboPrb.setPreferredSize(new Dimension(150, 40));
57     for(int i = 0;i<files.length;i++)
58     {
59         int taille_nom= files[i].getName().length();
60         comboPrb.addItem((files[i].getName()).substring(0, taille_nom-4));
61     }
62
63
64     fenetre.add(comboPrb);
65     fenetre.add(bouton_ok);
66
67     fenetre.setVisible(true);
68
69     bouton_ok.addMouseListener(new MouseListener() {
70
71         public void mouseClicked(MouseEvent e) {
72             choix =comboPrb.getSelectedItem().toString();
73
74             fichier_Choisi=new File("DB_JOBS/"+choix+".xml");
75             System.out.println("Fichier choisi : "+fichier_Choisi);
76
77             if(fichier_Choisi.delete())
78             {
79                 System.out.println("Fichier supprimer");
80             }

```

```

81         else
82         {
83             System.out.println("Fichier non supprimer");
84         }
85         fenetre.setVisible(false);
86         fenetre.setVisible(true);
87     }
88
89     public void mousePressed(MouseEvent e) {
90
91     }
92
93     public void mouseReleased(MouseEvent e) {
94
95     }
96
97     public void mouseEntered(MouseEvent e) {
98
99     }
100
101     public void mouseExited(MouseEvent e) {
102
103     }
104
105     });
106 }
107
108 @Override
109 public void mouseEntered(MouseEvent arg0) {
110     // TODO Auto-generated method stub
111
112 }
113
114 @Override
115 public void mouseExited(MouseEvent arg0) {
116     // TODO Auto-generated method stub
117
118 }
119
120 @Override
121 public void mousePressed(MouseEvent arg0) {

```

```

122     // TODO Auto-generated method stub
123
124 }
125
126 @Override
127 public void mouseReleased(MouseEvent arg0) {
128     // TODO Auto-generated method stub
129
130 }
131
132 public String getName() {
133     return name;
134 }
135
136 public void setName(String name) {
137     this.name = name;
138 }
139
140 public Image getImg() {
141     return img;
142 }
143
144 public void setImg(Image img) {
145     this.img = img;
146 }
147
148 public JComboBox<String> getComboPrb() {
149     return comboPrb;
150 }
151
152 public void setComboPrb(JComboBox<String> comboPrb) {
153     this.comboPrb = comboPrb;
154 }
155
156 public static FilenameFilter getXmlFileFilter() {
157     return xmlFileFilter;
158 }
159
160 public static void setXmlFileFilter(FilenameFilter xmlFileFilter) {
161     ButtonDel.xmlFileFilter = xmlFileFilter;
162 }

```

```

163
164     public File getRepertoire() {
165         return repertoire;
166     }
167
168     public void setRepertoire(File repertoire) {
169         this.repertoire = repertoire;
170     }
171
172     public File[] getFiles() {
173         return files;
174     }
175
176     public void setFiles(File[] files) {
177         this.files = files;
178     }
179
180     public JButton getBouton_ok() {
181         return bouton_ok;
182     }
183
184     public void setBouton_ok(JButton bouton_ok) {
185         this.bouton_ok = bouton_ok;
186     }
187
188     public File getFichier_Choisi() {
189         return fichier_Choisi;
190     }
191
192     public void setFichier_Choisi(File fichier_Choisi) {
193         this.fichier_Choisi = fichier_Choisi;
194     }
195
196     public String getChoix() {
197         return choix;
198     }
199
200     public void setChoix(String choix) {
201         this.choix = choix;
202     }
203

```

```
204     public static long getSerialversionuid() {  
205         return serialVersionUID;  
206     }  
207  
208 }
```

ButtonDoW.java :

```
1  import java.awt.Color;
2  import java.awt.Dimension;
3  import java.awt.FlowLayout;
4  import java.awt.GradientPaint;
5  import java.awt.Graphics;
6  import java.awt.Graphics2D;
7  import java.awt.Image;
8  import java.awt.event.MouseEvent;
9  import java.awt.event.MouseListener;
10 import java.io.File;
11 import java.io.FileNameFilter;
12 import java.io.IOException;
13 import javax.imageio.ImageIO;
14 import javax.swing.JButton;
15 import javax.swing.JComboBox;
16 import javax.swing.JFileChooser;
17 import javax.swing.JFrame;
18 import javax.swing.filechooser.FileNameExtensionFilter;
19
20 import org.jivesoftware.smack.*;
21 import org.jivesoftware.smack.Chat;
22 import org.jivesoftware.smack.ChatManager;
23 import org.jivesoftware.smack.ConnectionConfiguration;
24 import org.jivesoftware.smack.MessageListener;
25 import org.jivesoftware.smack.Roster;
26 import org.jivesoftware.smack.SmackConfiguration;
27 import org.jivesoftware.smack.XMPPConnection;
28 import org.jivesoftware.smack.XMPPException;
29 import org.jivesoftware.smack.ConnectionConfiguration.SecurityMode;
30 import org.jivesoftware.smack.packet.Message;
31 import org.jivesoftware.smack.packet.Presence;
32 import org.jivesoftware.smack.packet.Presence.Type;
33 import org.jivesoftware.smackx.muc.MultiUserChat;
34
35 @SuppressWarnings("unused")
36 public class ButtonDoW extends JButton implements MouseListener {
37
38     /**
39     *
```

```

40     */
41     private static final long serialVersionUID = 1L;
42     private String name;
43     private Image img;
44     private JComboBox<String> comboPrb;
45     private static FilenameFilter xmlFileFilter = new FilenameFilter() {public boolean accept(
46     private File repertoire;
47     private File[] files;
48     private JButton bouton_ok;
49     private File fichier_Choisi;
50     private String choix;
51
52     private String username ;
53     private String password ;
54
55     private XmppManager xmppManager;
56     private String ProblemeCourant;
57     private boolean isRunning ;
58
59     public ButtonDoW(String str){
60         super(str);
61         this.name = str;
62         this.addMouseListener(this);
63         this.xmppManager = new XmppManager(xmppManager.NOM_HOTE, 5222);
64     }
65
66     @Override
67     public void mouseClicked(MouseEvent arg0) {
68         try{
69             xmppManager.init();
70             xmppManager.performLogin(username, password);
71             xmppManager.setStatus(true, "YOLO");
72             xmppManager.setProvider(false);
73
74             // Create a MultiUserChat using an XMPPConnection for a room
75             MultiUserChat muc2 = new MultiUserChat(xmppManager.getConnection(), ProblemeCourant+"@"
76
77             // User2 joins the new room
78             // The room service will decide the amount of history to send
79             muc2.join(username);
80             xmppManager.createEntry("provider","BOT_Providing");

```



```

81         xmppManager.sendMessage("I am ready to work ", "provider@"+xmppManager.NOM_HOTE);
82         isRunning = true;
83
84         while (isRunning){
85             Thread.sleep(50);
86
87         }
88
89         xmppManager.destroy();
90
91     } catch (Exception exc) {
92         // TODO Auto-generated catch block
93         exc.printStackTrace();
94     }
95 }
96
97 @Override
98 public void mouseEntered(MouseEvent arg0) {
99     // TODO Auto-generated method stub
100
101 }
102
103 @Override
104 public void mouseExited(MouseEvent arg0) {
105     // TODO Auto-generated method stub
106
107 }
108
109 @Override
110 public void mousePressed(MouseEvent arg0) {
111     // TODO Auto-generated method stub
112
113 }
114
115 @Override
116 public void mouseReleased(MouseEvent arg0) {
117     // TODO Auto-generated method stub
118
119 }
120
121 public String getName() {

```

```

122     return name;
123 }
124
125 public void setName(String name) {
126     this.name = name;
127 }
128
129 public Image getImg() {
130     return img;
131 }
132
133 public void setImg(Image img) {
134     this.img = img;
135 }
136
137 public JComboBox<String> getComboPrb() {
138     return comboPrb;
139 }
140
141 public void setComboPrb(JComboBox<String> comboPrb) {
142     this.comboPrb = comboPrb;
143 }
144
145 public static FilenameFilter getXmlFileFilter() {
146     return xmlFileFilter;
147 }
148
149 public static void setXmlFileFilter(FilenameFilter xmlFileFilter) {
150     ButtonDoW.xmlFileFilter = xmlFileFilter;
151 }
152
153 public File getRepertoire() {
154     return repertoire;
155 }
156
157 public void setRepertoire(File repertoire) {
158     this.repertoire = repertoire;
159 }
160
161 public File[] getFiles() {
162     return files;

```

```

163     }
164
165     public void setFiles(File[] files) {
166         this.files = files;
167     }
168
169     public JButton getBouton_ok() {
170         return bouton_ok;
171     }
172
173     public void setBouton_ok(JButton bouton_ok) {
174         this.bouton_ok = bouton_ok;
175     }
176
177     public File getFichier_Choisi() {
178         return fichier_Choisi;
179     }
180
181     public void setFichier_Choisi(File fichier_Choisi) {
182         this.fichier_Choisi = fichier_Choisi;
183     }
184
185     public String getChoix() {
186         return choix;
187     }
188
189     public void setChoix(String choix) {
190         this.choix = choix;
191     }
192
193     public String getUsername() {
194         return username;
195     }
196
197     public void setUsername(String username) {
198         this.username = username;
199     }
200
201     public String getPassword() {
202         return password;
203     }

```

```
204
205     public void setPassword(String password) {
206         this.password = password;
207     }
208
209     public XmppManager getXmppManager() {
210         return xmppManager;
211     }
212
213     public void setXmppManager(XmppManager xmppManager) {
214         this.xmppManager = xmppManager;
215     }
216
217     public String getProblemeCourant() {
218         return ProblemeCourant;
219     }
220
221     public void setProblemeCourant(String problemeCourant) {
222         ProblemeCourant = problemeCourant;
223     }
224
225     public boolean isRunning() {
226         return isRunning;
227     }
228
229     public void setRunning(boolean isRunning) {
230         this.isRunning = isRunning;
231     }
232
233     public static long getSerialversionuid() {
234         return serialVersionUID;
235     }
236
237 }
```

ButtonLaunch.java :

```
1  import java.awt.BorderLayout;
2  import java.awt.Color;
3  import java.awt.Dimension;
4  import java.awt.FlowLayout;
5  import java.awt.GradientPaint;
6  import java.awt.Graphics;
7  import java.awt.Graphics2D;
8  import java.awt.Image;
9  import java.awt.event.MouseEvent;
10 import java.awt.event.MouseListener;
11 import java.io.BufferedReader;
12 import java.io.File;
13 import java.io.FileInputStream;
14 import java.io.FileWriter;
15 import java.io.FilterReader;
16 import java.io.IOException;
17 import java.io.InputStream;
18 import java.io.InputStreamReader;
19 import java.util.ArrayList;
20 import java.util.Collection;
21 import java.util.Iterator;
22 import java.util.List;
23
24 import javax.imageio.ImageIO;
25 import javax.swing.JButton;
26 import javax.swing.JComboBox;
27 import javax.swing.JFileChooser;
28 import javax.swing.JFrame;
29 import javax.swing.JLabel;
30 import javax.swing.filechooser.FileNameExtensionFilter;
31
32 import javax.xml.parsers.DocumentBuilder;
33 import javax.xml.parsers.DocumentBuilderFactory;
34 import javax.xml.parsers.ParserConfigurationException;
35 import javax.xml.transform.Transformer;
36 import javax.xml.transform.TransformerFactory;
37 import javax.xml.transform.dom.DOMSource;
38 import javax.xml.transform.stream.StreamResult;
39 import javax.xml.xpath.XPath;
```

```

40 import javax.xml.xpath.XPathConstants;
41 import javax.xml.xpath.XPathExpressionException;
42 import javax.xml.xpath.XPathFactory;
43
44 import org.w3c.dom.Document;
45 import org.w3c.dom.Element;
46 import org.w3c.dom.NodeList;
47 import org.xml.sax.SAXException;
48
49 import org.jivesoftware.smack.*;
50 import org.jivesoftware.smack.ConnectionConfiguration;
51 import org.jivesoftware.smack.MessageListener;
52 import org.jivesoftware.smack.SmackConfiguration;
53 import org.jivesoftware.smack.XMPPConnection;
54 import org.jivesoftware.smack.XMPPException;
55 import org.jivesoftware.smack.ConnectionConfiguration.SecurityMode;
56 import org.jivesoftware.smack.packet.Message;
57 import org.jivesoftware.smack.packet.Presence;
58 import org.jivesoftware.smack.packet.Presence.Type;
59 import org.jivesoftware.smackx.Form;
60 import org.jivesoftware.smackx.ServiceDiscoveryManager;
61 import org.jivesoftware.smackx.muc.Affiliate;
62 import org.jivesoftware.smackx.muc.MultiUserChat;
63 import org.jivesoftware.smackx.muc.Occupant;
64 import org.jivesoftware.smackx.muc.RoomInfo;
65 import org.jivesoftware.smackx.packet.DiscoverItems;
66
67
68 @SuppressWarnings({ "unused", "serial" })
69 public class ButtonLaunch extends JButton implements MouseListener {
70
71     private String name;
72     private Image img;
73     private JButton bouton_ok;
74     private JFrame fenetre;
75     private JComboBox<String> comboPrb;
76     private static FilenameFilter xmlFileFilter = new FilenameFilter() {public boolean accept
77     private File repertoire;
78     private File[] files;
79     private File fichier_Choisi;
80     private XmppManager xmppManager ;

```

```

81     private String ProblemeCourant;
82     private boolean isRunning ;
83     private ArrayList<Identity> Liste_user;
84     private JLabel res ;
85
86     public ButtonLaunch(String str){
87         super(str);
88         this.name = str;
89         this.addMouseListener(this);
90         fenetre = new JFrame();
91         comboPrb = new JComboBox<String>();
92
93         bouton_ok = new JButton("OK");
94         Liste_user = new ArrayList<Identity>();
95         res=new JLabel("Resultat");
96         xmppManager = new XmppManager(XmppManager.NOM_HOTE, 5222);
97     }
98     public String FileToString(String PathFile)
99     {
100         String fic = "";
101         //lecture du fichier texte
102         try
103         {
104             InputStream ips=new FileInputStream(PathFile);
105             InputStreamReader ipsr=new InputStreamReader(ips);
106             BufferedReader br=new BufferedReader(ipsr);
107             String ligne;
108             while ((ligne=br.readLine())!=null)
109             {
110                 System.out.println(ligne);
111                 fic+=ligne+"\n";
112             }
113             br.close();
114             ipsr.close();
115             ips.close();
116         }
117         catch (Exception e)
118         {
119             System.out.println(e.toString());
120         }
121         return fic;

```

```

122     }
123     @Override
124     public void mouseClicked(MouseEvent arg0) {
125         // TODO Auto-generated method stub
126
127         fenetre.setTitle("Lancement d'une tache");
128         fenetre.setSize(800, 400);
129         fenetre.setLocationRelativeTo(null);
130
131         fenetre.setBackground(Color.white);
132         fenetre.setLayout(new FlowLayout());
133         repertoire = new File("DB_JOBS");
134         files =repertoire.listFiles(xmlFileFilter);
135
136         comboPrb.setPreferredSize(new Dimension(150, 40));
137         for(int i = 0;i<files.length;i++)
138         {
139             int taille_nom= files[i].getName().length();
140             comboPrb.addItem((files[i].getName()).substring(0, taille_nom-4));
141         }
142
143
144         fenetre.add(comboPrb);
145
146         fenetre.add(bouton_ok);
147         fenetre.setVisible(true);
148
149         bouton_ok.addMouseListener(new MouseListener(){
150
151             public void mouseClicked(MouseEvent e) {
152                 /*Lancement du JOB*/
153                 // on recupere ce qui a ete choisi
154                 String choix =comboPrb.getSelectedItem().toString();
155
156                 String username = "provider";
157                 String password = "toto";
158                 choix= choix+".xml";
159                 ProblemeCourant= FileToString("DB_JOBS/"+choix);
160
161                 try {
162                     /*On initialise la connection */

```



```

163 xmppManager.init();
164 xmppManager.performLogin(username, password);
165 xmppManager.setStatus(true, "YOLO");
166 xmppManager.setProvider(true);
167
168 /*On crer la chatroom Multiuser */
169 //Get the MultiUserChatManager
170
171 //Create a MultiUserChat using an XMPPConnection for a room
172 String Name_room = "providing_room_"+comboPrb.getSelectedItemAt().toString()+"@com
173 MultiUserChat muc = new MultiUserChat(xmppManager.getConnection(), Name_room);
174
175 // Create the room identity
176 muc.create("BOT_Providing");
177
178 // Send an empty room configuration form which indicates that we want
179 // an instant room
180 muc.sendConfigurationForm(new Form(Form.TYPE_SUBMIT));
181 /*On essaye d'utiliser le XML selectioner*/
182
183 //On va essayer d'avoir la liste des utilisateurs selectionner et de leur envoyer
184
185
186 System.out.println("On enregistre le nombre de personne");
187 int Nombre_Participants=muc.getOccupantsCount();
188 System.out.println("Number of occupants et affichage de la liste:"+Nombre_Part
189
190 Iterator it = muc.getOccupants();
191
192 while(it.hasNext())
193 {
194     System.out.print("Membre : ");
195     System.out.println("Valeur "+it.next());
196 }
197
198 /*On attend un nombre d'utilisateur requis par le split ici on le simule seule
199
200 //On cherche a savoir combien d'utilisateurs par rapport au fichier xml enregi
201
202 System.out.println("On attend un nombre d'utilisateur precis");
203 int Nombre_requis=3;

```

```

204
205     DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
206
207
208     try {
209
210         DocumentBuilder builder = factory.newDocumentBuilder();
211
212         File fileXML = new File("DB_JOBS/"+choix);
213
214         Document xml = builder.parse(fileXML);
215
216         Element root = xml.getDocumentElement();
217
218         XPathFactory xpf = XPathFactory.newInstance();
219
220         XPath path = xpf.newXPath();
221
222
223
224         String expression = "/JOB/rang";
225
226         String str = (String)path.evaluate(expression, root);
227
228         Nombre_requis=Integer.parseInt(str);
229
230         System.out.println("-----");
231     } catch (ParserConfigurationException xe) {
232         xe.printStackTrace();
233     } catch (SAXException xe) {
234         xe.printStackTrace();
235     } catch (IOException xe) {
236         xe.printStackTrace();
237     } catch (XPathExpressionException xe) {
238         xe.printStackTrace();
239     }
240     System.out.println("NBR : "+Nombre_requis);
241
242     //modifier ici attention nombre requis = 2 ,valable uniquement pour les tests
243     while(Nombre_Participants<Nombre_requis){
244         Nombre_Participants=muc.getOccupantsCount();

```

```

245         System.out.println("Number of occupants et affichage de la liste:"+Nombre_Par
246
247         it = muc.getOccupants();
248
249         while(it.hasNext())
250         {
251             System.out.println("Affichage tableau : ");
252             System.out.println("Valeur "+it.next());
253         }
254
255     }
256
257     //Faire liste utilisateurs de la room
258
259
260     ArrayList<Occupant> Liste=(ArrayList<Occupant>)muc.getParticipants();
261
262
263     for(int i = 0;i<Liste.size();i++)
264     {
265         Liste_user.add(new Identity(Liste.get(i).getJid(),Liste.get(i).getRole(),Liste.get(i).getNick()));
266         System.out.println("Nickname "+i+Liste.get(i).getNick());
267     }
268
269
270     //maintenant que on a la liste des utilisateur connecte a la chatRoom on va le
271     //On a attendu que lon est assez de participant ou pas en fonction du split
272     //Voir pour filtrer son propre nom a savoir is on est tjr le premier ou pas
273     System.out.println("Debut split ");
274     //modifier ici attention on a pas le bon nombre de participant
275     split(Liste_user,Nombre_Participants,ProblemeCourant,xmppManager,choix);
276
277     System.out.println("Fin du split ");
278     muc.sendMessage("Lancement du probleme du"+comboPrb.getSelectedItem().toString());
279
280     /*Maintenant que l'on a envoyer plusieurs problemes on va essayer d'avoir leur reponse
281
282     isRunning = true;
283     fenetre.add(res);
284     while (xmppManager.isTravail_terminer()){
285         Thread.sleep(50);

```

```

286     }
287     //on a eu le resultat
288     res.setText("Resultat : "+xmppManager.getRetour_Providing());
289     fenetre.add(res);
290     Thread.sleep(5000);
291     xmppManager.destroy();
292
293
294     }
295     catch (XMPPException ex) {
296         // TODO Auto-generated catch block
297         ex.printStackTrace();
298     }
299     catch (Exception ex) {
300         // TODO Auto-generated catch block
301         ex.printStackTrace();
302     }
303
304 }
305
306 public void mousePressed(MouseEvent e) {
307
308 }
309
310 public void mouseReleased(MouseEvent e) {
311
312 }
313
314 public void mouseEntered(MouseEvent e) {
315
316 }
317
318 public void mouseExited(MouseEvent e) {
319
320 }
321 });
322 }
323
324 @Override
325 public void mouseEntered(MouseEvent arg0) {
326     // TODO Auto-generated method stub

```

```

327
328     }
329
330     @Override
331     public void mouseExited(MouseEvent arg0) {
332         // TODO Auto-generated method stub
333
334     }
335
336     @Override
337     public void mousePressed(MouseEvent arg0) {
338         // TODO Auto-generated method stub
339
340     }
341
342     @Override
343     public void mouseReleased(MouseEvent arg0) {
344         // TODO Auto-generated method stub
345
346     }
347     public int split(ArrayList<Identity> Liste_user,int Nombre_Participants,String ProblemeCour
348     {
349         //modifier ici attention
350         for(int i=0;i<Nombre_Participants-1;i++)
351         {
352             String buddyJID = Liste_user.get(i).getId();
353             String buddyName = Liste_user.get(i).getName();
354
355             try {
356                 xmppManager.createEntry(buddyJID, buddyName);
357
358                 //On va crer le message XML approprier puis l'envoyer
359
360
361                 //On construit le fichier XML avec le code a executer
362
363
364                 /* On va dans un premier temps rechercher l'ensemble des noms des patients de
365
366                 /* Recherche de la liste des exec*/
367

```

```

368
369 DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
370 DocumentBuilder builder = factory.newDocumentBuilder();
371
372 File fileXML = new File("DB_JOBS/"+choix);
373
374 Document xml = builder.parse(fileXML);
375
376 Element root = xml.getDocumentElement();
377
378 XPathFactory xpf = XPathFactory.newInstance();
379
380 XPath path = xpf.newXPath();
381
382
383
384 String expression = "/JOB/code_exec";
385
386 String strexec = (String)path.evaluate(expression, root);
387 System.out.print("DEBUT STR EXEC");
388 System.out.print(strexec);
389 System.out.print("FIN STR EXEC");
390
391 expression = "/JOB/code_Perl";
392
393 String strperl = (String)path.evaluate(expression, root);
394
395 /* On récupère tous les noeuds répondant au chemin //patient */
396
397
398 // On ajouter a sa la bonne ligne de commande a executer
399 expression = "/JOB/cmd";
400
401 String strcmd = (String)path.evaluate(expression, root);
402
403 /* On récupère tous les noeuds répondant au chemin //patient */
404
405 String delims = "[,]";
406 String[] tokens =strcmd.split(delims);
407 System.out.print("affichage des tokens");
408 tokens[tokens.length-1]=tokens[tokens.length-1].substring(0, tokens[tokens.length-1]

```

```

409     for(int j=0;j<tokens.length;j++)
410         System.out.println(j+" : "+tokens[j]);
411
412     //Faut parser la liste
413
414
415
416     final Document document= builder.newDocument();
417
418     final Element racine = document.createElement("JOB");
419     document.appendChild(racine);
420     final Element exec = document.createElement("exec");
421     exec.appendChild(document.createTextNode(strexec));
422
423     final Element contraintes = document.createElement("contraintes");
424     contraintes.appendChild(document.createTextNode(strperl));
425
426
427     final Element cmd = document.createElement("cmd");
428     cmd.appendChild(document.createTextNode(tokens[i]));
429     final Element id = document.createElement("id");
430     id.appendChild(document.createTextNode(""+i));
431     racine.appendChild(id);
432     racine.appendChild(contraintes);
433     racine.appendChild(exec);
434     racine.appendChild(cmd);
435
436     final TransformerFactory transformerFactory = TransformerFactory.newInstance();
437     final Transformer transformer = transformerFactory.newTransformer();
438     final DOMSource source = new DOMSource(document);
439     final StreamResult sortie = new StreamResult(new File("JOB_SEND/XML_send_"+i));
440     transformer.transform(source, sortie);
441
442     String Probleme_individuel=FileToString("JOB_SEND/XML_send_"+i);
443
444     xmppManager.sendMessage(Probleme_individuel, buddyName+"@"+xmppManager.NOM_HOTE);
445
446
447
448     } catch (Exception e) {
449         // TODO Auto-generated catch block

```

```

450         e.printStackTrace();
451     }
452     // Ici on fait appelle a la fonction split
453
454 }
455
456     return 0;
457 }
458 public String getName() {
459     return name;
460 }
461 public void setName(String name) {
462     this.name = name;
463 }
464 public Image getImg() {
465     return img;
466 }
467 public void setImg(Image img) {
468     this.img = img;
469 }
470 public JButton getBouton_ok() {
471     return bouton_ok;
472 }
473 public void setBouton_ok(JButton bouton_ok) {
474     this.bouton_ok = bouton_ok;
475 }
476 public JFrame getFenetre() {
477     return fenetre;
478 }
479 public void setFenetre(JFrame fenetre) {
480     this.fenetre = fenetre;
481 }
482 public JComboBox<String> getComboPrb() {
483     return comboPrb;
484 }
485 public void setComboPrb(JComboBox<String> comboPrb) {
486     this.comboPrb = comboPrb;
487 }
488
489 public static FilenameFilter getXmlFileFilter() {
490     return xmlFileFilter;

```



```

491     }
492     public static void setXmlFileFilter(FilenameFilter xmlFileFilter) {
493         ButtonLaunch.xmlFileFilter = xmlFileFilter;
494     }
495     public File getRepertoire() {
496         return repertoire;
497     }
498     public void setRepertoire(File repertoire) {
499         this.repertoire = repertoire;
500     }
501     public File[] getFiles() {
502         return files;
503     }
504     public void setFiles(File[] files) {
505         this.files = files;
506     }
507     public File getFichier_Choisi() {
508         return fichier_Choisi;
509     }
510     public void setFichier_Choisi(File fichier_Choisi) {
511         this.fichier_Choisi = fichier_Choisi;
512     }
513     public XmppManager getXmppManager() {
514         return xmppManager;
515     }
516     public void setXmppManager(XmppManager xmppManager) {
517         this.xmppManager = xmppManager;
518     }
519     public String getProblemeCourant() {
520         return ProblemeCourant;
521     }
522     public void setProblemeCourant(String problemeCourant) {
523         ProblemeCourant = problemeCourant;
524     }
525     public boolean isRunning() {
526         return isRunning;
527     }
528     public void setRunning(boolean isRunning) {
529         this.isRunning = isRunning;
530     }
531     public ArrayList<Identity> getListe_user() {

```

```
532     return Liste_user;
533 }
534 public void setListe_user(ArrayList<Identity> liste_user) {
535     Liste_user = liste_user;
536 }
537
538 }
```

ButtonWorker.java :

```
1  import java.awt.BorderLayout;
2  import java.awt.Color;
3  import java.awt.FlowLayout;
4  import java.awt.GradientPaint;
5  import java.awt.Graphics;
6  import java.awt.Graphics2D;
7  import java.awt.GridLayout;
8  import java.awt.Image;
9  import java.awt.event.MouseEvent;
10 import java.awt.event.MouseListener;
11 import java.io.File;
12 import java.io.IOException;
13 import java.util.ArrayList;
14 import java.util.Iterator;
15
16 import javax.imageio.ImageIO;
17 import javax.swing.JButton;
18 import javax.swing.JComboBox;
19 import javax.swing.JFrame;
20 import javax.swing.JLabel;
21 import javax.swing.JPanel;
22 import javax.swing.JTextField;
23
24 import org.jivesoftware.smack.Chat;
25 import org.jivesoftware.smack.ChatManager;
26 import org.jivesoftware.smack.MessageListener;
27 import org.jivesoftware.smack.XMPPException;
28 import org.jivesoftware.smack.packet.Message;
29 import org.jivesoftware.smackx.muc.HostedRoom;
30 import org.jivesoftware.smackx.muc.MultiUserChat;
31
32
33 @SuppressWarnings({ "serial", "unused" })
34 public class ButtonWorker extends JButton implements MouseListener {
35
36     private String name;
37     private Image img;
38     private JFrame fenetre ;
39     private JButton bouton_ok ;
```

```

40     private ButtonDoW bouton_ok_channel ;
41     private JTextField pseudo ;
42     private JLabel label_pseudal ;
43
44     private JTextField pass ;
45     private JLabel label_pass ;
46     private JLabel affichage ;
47     private String username ;
48     private String password ;
49
50     private XmppManager xmppManager;
51     private String ProblemeCourant;
52     private boolean isRunning ;
53     private ArrayList<HostedRoom> ListeRoom;
54     private JComboBox<String> comboPrb;
55
56     public ButtonWorker(String str){
57         super(str);
58         this.name = str;
59         this.addMouseListener(this);
60         this.fenetre = new JFrame();
61         this.bouton_ok = new JButton("OK");
62         this.bouton_ok_channel = new ButtonDoW("OK");
63         this.pseudo = new JTextField("pseudo");
64         this.label_pseudal = new JLabel("Pseudo");
65         this.pass = new JTextField("Password");
66         this.label_pass = new JLabel("Mot de passe");
67         this.affichage = new JLabel("Selection du salon:");
68         this.comboPrb = new JComboBox<String>();
69         this.ListeRoom = new ArrayList<HostedRoom>();
70         this.xmppManager = new XmppManager(xmppManager.NOM_HOTE, 5222);
71     }
72
73     @Override
74     public void mouseClicked(MouseEvent arg0) {
75         // TODO Auto-generated method stub
76         GridLayout gl;
77         //fenetre.setLayout(gl=new GridLayout(3, 2));
78         fenetre.setLayout(new FlowLayout());
79         fenetre.add(label_pseudal);
80         fenetre.add(pseudo);

```

```

81
82     fenetre.add(label_pass);
83     fenetre.add(pass);
84     fenetre.setTitle("Travail sur des taches");
85     fenetre.add(bouton_ok);
86
87     fenetre.setSize(400, 500);
88     //gl.setColumns(2);
89     //gl.setRows(3);
90
91     fenetre.setLocationRelativeTo(null);
92
93     fenetre.setVisible(true);
94
95
96
97     bouton_ok.addMouseListener(new MouseListener(){
98
99         @SuppressWarnings("deprecation")
100         public void mouseClicked(MouseEvent e) {
101
102             try {
103
104                 xmppManager.init();
105                 String username = pseudo.getText();
106                 String password = pass.getText();
107                 xmppManager.performLogin(username, password);
108                 xmppManager.setStatus(true, "YOLO");
109                 xmppManager.setProvider(false);
110
111
112                 System.out.println("Etablir une liste de room");
113
114                 ListeRoom =(ArrayList<HostedRoom>)MultiUserChat.getHostedRooms(xmppManager.getConn
115                 System.out.println("Fin de etablir une liste de room");
116                 Iterator it = ListeRoom.iterator();
117
118                 while(it.hasNext())
119                 {
120                     HostedRoom tmp = (HostedRoom) it.next();
121                     System.out.println("ChatRoom : "+tmp.getName());

```

```

122         comboPrb.addItem(tmp.getName());
123     }
124
125     fenetre.remove(label_pseudal);
126     fenetre.remove(pseudo);
127     fenetre.remove(label_pass);
128     fenetre.remove(pass);
129     fenetre.remove(bouton_ok);
130     fenetre.setVisible(false);
131
132     //On rafraichit
133     fenetre.add(affichage);
134     fenetre.add(comboPrb);
135     fenetre.add(bouton_ok_channel);
136
137
138     bouton_ok_channel.setProblemeCourant(comboPrb.getSelectedItem().toString());
139     bouton_ok_channel.setUsername(username);
140     bouton_ok_channel.setPassword(password);
141
142     fenetre.setVisible(true);
143     xmppManager.destroy();
144 } catch (XMPPException exc) {
145     // TODO Auto-generated catch block
146     exc.printStackTrace();
147 }
148 }
149
150 public void mousePressed(MouseEvent e) {
151
152 }
153
154 public void mouseReleased(MouseEvent e) {
155
156 }
157
158 public void mouseEntered(MouseEvent e) {
159
160 }
161
162 public void mouseExited(MouseEvent e) {

```

```

163     }
164     });
165 }
166
167 @Override
168 public void mouseEntered(MouseEvent arg0) {
169     // TODO Auto-generated method stub
170
171 }
172
173 @Override
174 public void mouseExited(MouseEvent arg0) {
175     // TODO Auto-generated method stub
176
177 }
178
179 @Override
180 public void mousePressed(MouseEvent arg0) {
181     // TODO Auto-generated method stub
182
183 }
184
185 @Override
186 public void mouseReleased(MouseEvent arg0) {
187     // TODO Auto-generated method stub
188
189 }
190
191 public String getName() {
192     return name;
193 }
194
195 public void setName(String name) {
196     this.name = name;
197 }
198
199 public Image getImg() {
200     return img;
201 }
202
203 public void setImg(Image img) {

```

```

204     this.img = img;
205 }
206
207 public JFrame getFenetre() {
208     return fenetre;
209 }
210
211 public void setFenetre(JFrame fenetre) {
212     this.fenetre = fenetre;
213 }
214
215 public JButton getBouton_ok() {
216     return bouton_ok;
217 }
218
219 public void setBouton_ok(JButton bouton_ok) {
220     this.bouton_ok = bouton_ok;
221 }
222
223 public ButtonDoW getBouton_ok_channel() {
224     return bouton_ok_channel;
225 }
226
227 public void setBouton_ok_channel(ButtonDoW bouton_ok_channel) {
228     this.bouton_ok_channel = bouton_ok_channel;
229 }
230
231 public JTextField getPseudo() {
232     return pseudo;
233 }
234
235 public void setPseudo(JTextField pseudo) {
236     this.pseudo = pseudo;
237 }
238
239 public JLabel getLabel_pseudal() {
240     return label_pseudal;
241 }
242
243 public void setLabel_pseudal(JLabel label_pseudal) {
244     this.label_pseudal = label_pseudal;

```



```

245     }
246
247     public JTextField getPass() {
248         return pass;
249     }
250
251     public void setPass(JTextField pass) {
252         this.pass = pass;
253     }
254
255     public JLabel getLabel_pass() {
256         return label_pass;
257     }
258
259     public void setLabel_pass(JLabel label_pass) {
260         this.label_pass = label_pass;
261     }
262
263     public JLabel getAffichage() {
264         return affichage;
265     }
266
267     public void setAffichage(JLabel affichage) {
268         this.affichage = affichage;
269     }
270
271     public String getUsername() {
272         return username;
273     }
274
275     public void setUsername(String username) {
276         this.username = username;
277     }
278
279     public String getPassword() {
280         return password;
281     }
282
283     public void setPassword(String password) {
284         this.password = password;
285     }

```

```

286
287     public XmppManager getXmppManager() {
288         return xmppManager;
289     }
290
291     public void setXmppManager(XmppManager xmppManager) {
292         this.xmppManager = xmppManager;
293     }
294
295     public String getProblemeCourant() {
296         return ProblemeCourant;
297     }
298
299     public void setProblemeCourant(String problemeCourant) {
300         ProblemeCourant = problemeCourant;
301     }
302
303     public boolean isRunning() {
304         return isRunning;
305     }
306
307     public void setRunning(boolean isRunning) {
308         this.isRunning = isRunning;
309     }
310
311     public ArrayList<HostedRoom> getListeRoom() {
312         return ListeRoom;
313     }
314
315     public void setListeRoom(ArrayList<HostedRoom> listeRoom) {
316         ListeRoom = listeRoom;
317     }
318
319     public JComboBox<String> getComboPrb() {
320         return comboPrb;
321     }
322
323     public void setComboPrb(JComboBox<String> comboPrb) {
324         this.comboPrb = comboPrb;
325     }
326

```


Identity.java :

```
1 public class Identity {
2
3
4     private String id;
5     private String node;
6     private String name;
7
8     public Identity() {
9         super();
10    }
11
12    public Identity(String id, String node, String name) {
13        super();
14        this.id = id;
15        this.node = node;
16        this.name = name;
17    }
18
19    public String getId() {
20        return id;
21    }
22    public void setId(String id) {
23        this.id = id;
24    }
25    public String getNode() {
26        return node;
27    }
28    public void setNode(String node) {
29        this.node = node;
30    }
31    public String getName() {
32        return name;
33    }
34    public void setName(String name) {
35        this.name = name;
36    }
37 }
```

main.java :

```
1  import javax.swing.*;
2  import java.awt.*;
3  import java.io.File;
4  import java.io.IOException;
5  import javax.imageio.ImageIO;
6  import java.awt.Graphics;
7  import java.awt.Image;
8  import javax.swing.JPanel;
9
10
11  @SuppressWarnings("unused")
12  public class main {
13
14      public static void main(final String[] args) {
15          JFrame fenetre = new JFrame();
16          ImageIcon icone = new ImageIcon("/home/apocalypzer/workspace/TER/src/images/urca.jpg");
17          JLabel image = new JLabel(icone);
18
19          fenetre.setTitle("Calcul Grid XMPP");
20          fenetre.setSize(250, 310);
21          fenetre.setResizable(false);
22          fenetre.setLocationRelativeTo(null);
23          fenetre.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
24          fenetre.setVisible(true);
25
26
27          fenetre.setContentPane(new Panneau());
28
29          fenetre.setLayout(new FlowLayout());
30
31
32          fenetre.getContentPane().add(image, BorderLayout.WEST);
33          fenetre.getContentPane().add(new ButtonLaunch("Lancer une tache"), BorderLayout.EAST);
34          fenetre.getContentPane().add(new ButtonWorker("Client d'une tache"), BorderLayout.EAST);
35          fenetre.getContentPane().add(new ButtonAjout("Ajouter une tache a la base"), BorderLayout.EAST);
36          fenetre.getContentPane().add(new ButtonDel("Retirer une tache de la base"), BorderLayout.EAST);
37          fenetre.getContentPane().add(new ButtonContact("Contact"), BorderLayout.EAST);
38          fenetre.setVisible(true);
39      }
```

40

41 }

Xmppmanager.java :

```
1  import org.jivesoftware.smack.*;
2  import org.jivesoftware.smack.Chat;
3  import org.jivesoftware.smack.ChatManager;
4  import org.jivesoftware.smack.ConnectionConfiguration;
5  import org.jivesoftware.smack.MessageListener;
6  import org.jivesoftware.smack.Roster;
7  import org.jivesoftware.smack.SmackConfiguration;
8  import org.jivesoftware.smack.XMPPConnection;
9  import org.jivesoftware.smack.XMPPException;
10 import org.jivesoftware.smack.ConnectionConfiguration.SecurityMode;
11 import org.jivesoftware.smack.packet.Message;
12 import org.jivesoftware.smack.packet.Presence;
13 import org.jivesoftware.smack.packet.Presence.Type;
14
15 import java.io.*;
16 import java.io.BufferedWriter;
17 import java.io.File;
18 import java.io.FileInputStream;
19 import java.io.FileOutputStream;
20 import java.io.IOException;
21 import java.io.PrintWriter;
22 import java.nio.charset.Charset;
23 import java.nio.file.Files;
24 import java.nio.file.Path;
25
26 import javax.xml.parsers.DocumentBuilder;
27 import javax.xml.parsers.DocumentBuilderFactory;
28 import javax.xml.parsers.ParserConfigurationException;
29 import javax.xml.transform.Transformer;
30 import javax.xml.transform.TransformerFactory;
31 import javax.xml.transform.dom.DOMSource;
32 import javax.xml.transform.stream.StreamResult;
33 import javax.xml.xpath.XPath;
34 import javax.xml.xpath.XPathConstants;
35 import javax.xml.xpath.XPathExpressionException;
36 import javax.xml.xpath.XPathFactory;
37
38 import org.w3c.dom.Document;
39 import org.w3c.dom.Element;
```

```

40 import org.w3c.dom.NodeList;
41 import org.xml.sax.SAXException;
42
43 @SuppressWarnings("unused")
44 public class XmppManager {
45
46     private static final int packetReplyTimeout = 500; // millis
47
48     private String server;
49     private int port;
50
51     private ConnectionConfiguration config;
52     private XMPPConnection connection;
53     static String NOM_HOTE="apocalypzer-lg-gram";
54     private ChatManager chatManager;
55     private MessageListener messageListener;
56     private boolean provider;
57     private boolean travail_terminer=false;
58     private int retour_Providing =0;
59     private int envoyer =0;
60     private int recu =0;
61
62     public XmppManager(String server, int port) {
63         this.server = server;
64         this.port = port;
65         this.provider=true;
66     }
67
68     public boolean isProvider() {
69         return provider;
70     }
71
72     public void setProvider(boolean provider) {
73         this.provider = provider;
74     }
75
76     public void init() throws XMPPException {
77
78         System.out.println(String.format("Initializing connection to server %1$s port %2$d",
79
80         SmackConfiguration.setPacketReplyTimeout(packetReplyTimeout);

```



```

81
82     config = new ConnectionConfiguration(server, port);
83     config.setSASLAuthenticationEnabled(false);
84     config.setSecurityMode(SecurityMode.disabled);
85
86     connection = new XMPPConnection(config);
87     connection.connect();
88
89     System.out.println("Connected: " + connection.isConnected());
90
91     chatManager = connection.getChatManager();
92     messageListener = new MyMessageListener();
93
94 }
95
96 public XMPPConnection getConnection() {
97     return connection;
98 }
99
100 public void setConnection(XMPPConnection connection) {
101     this.connection = connection;
102 }
103
104 public boolean performLogin(String username, String password) throws XMPPException {
105     if (connection!=null && connection.isConnected()) {
106         connection.login(username, password);
107     }
108     return true;
109 }
110
111 public void setStatus(boolean available, String status) {
112
113     Presence.Type type = available? Type.available: Type.unavailable;
114     Presence presence = new Presence(type);
115
116     presence.setStatus(status);
117     connection.sendPacket(presence);
118
119 }
120
121 public void destroy() {

```

```

122         if (connection!=null && connection.isConnected()) {
123             connection.disconnect();
124         }
125     }
126
127     public void sendMessage(String message, String buddyJID) throws XMPPException {
128         System.out.println(String.format("Sending message '%1$s' to user %2$s", message, buddyJID));
129         Chat chat = chatManager.createChat(buddyJID, messageListener);
130         chat.sendMessage(message);
131     }
132
133     public void createEntry(String user, String name) throws Exception {
134         System.out.println(String.format("Creating entry for buddy '%1$s' with name %2$s", user, name));
135         Roster roster = connection.getRoster();
136         roster.createEntry(user, name, null);
137     }
138
139     class MyMessageListener implements MessageListener {
140
141         @Override
142         public void processMessage(Chat chat, Message message) {
143
144             if(provider){
145                 //Modification de reaction si provider ou non
146                 String from = message.getFrom();
147                 String body = message.getBody();
148                 System.out.println(String.format("Received message '%1$s' from %2$s", body, from));
149                 // on regarde l'en tete du message apparence a un message xml reponse tel que
150                 //si impossible a executer dans le cas echeant un troisieme champ correspond a
151                 //et 0 si envoie travail
152
153                 // on procede donc au build un script ici un peu fictif mais pour prolonger po
154                 // ici on va juste sommer les results choses assez simple
155
156                 String regex="[,]";
157                 String[] en_tete = body.split(regex);
158
159                 // indice 0 = en tete indice 1 = res
160                 if(Integer.parseInt(en_tete[0])==1){
161                     retour_Providing+=Integer.parseInt(en_tete[1]);
162                     recu++;

```

```

163         if(recu==envoyer)
164         {
165             travail_terminer=true;
166         }
167     }
168     else
169     {
170         System.out.println("Lexecution n'as pas ete possible on redistribue aleatoirement");
171     }
172 }
173 else
174 {
175
176     String from = message.getFrom();
177     String body = message.getBody();
178     System.out.println(String.format("Received message '%1$s' from %2$s", body, from));
179     System.out.println("c'est partie on va executer ce qu'il faut");
180     // creation de l'architecture necessaire
181     File dir = new File ("JOB_REC/DATA_EXTRACT");
182     dir.mkdirs();
183     try{
184
185         System.out.println("Ecriture du XML dans un fichier xml receive");
186         File file = new File("JOB_REC/xml_receive.xml");
187         file.createNewFile();
188
189
190
191         PrintWriter writer = new PrintWriter(file);
192         writer.write(body);
193         writer.close();
194
195         //On a maintenant cherché le fichier on va le parser pour récupérer le fichier précis
196         DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
197         DocumentBuilder builder = factory.newDocumentBuilder();
198         Document xml = builder.parse(file);
199         Element root = xml.getDocumentElement();
200         XPathFactory xpf = XPathFactory.newInstance();
201         XPath path = xpf.newXPath();
202         String expression = "/JOB/exec";
203         String strexec = (String)path.evaluate(expression, root);

```

```

204
205
206     expression = "/JOB/contraintes";
207
208     String strperl = (String)path.evaluate(expression, root);
209
210     /* On récupère tous les noeuds répondant au chemin //patient */
211
212
213     // On ajouter a sa la bonne ligne de commande a executer
214     expression = "/JOB/cmd";
215     String strcmd = (String)path.evaluate(expression, root);
216     //Creation du fichier de contrainte en PERL
217
218     System.out.println("Ecriture du XML dans un fichier contrainte");
219     File file_con = new File("JOB_REC/DATA_EXTRACT/contraintes.pl");
220     file.createNewFile();
221     writer = new PrintWriter(file_con);
222     writer.write(strperl);
223     writer.close();
224
225     //Creation du fichier de exec en PERL et son execution
226
227     System.out.println("Ecriture du XML dans un fichier calcul.pl");
228     File file_exec = new File("JOB_REC/DATA_EXTRACT/calcul.pl");
229     file.createNewFile();
230     writer = new PrintWriter(file_exec);
231     writer.write(strexec);
232     writer.close();
233     // execution
234     Runtime runtime = Runtime.getRuntime();
235     System.out.println("execution du fichier de contrainte ");
236     // on execute le fichier de contrainte 3 = GOOD different = NOGOOD
237
238     Process p_cunt =runtime.exec("perl JOB_REC/DATA_EXTRACT/contraintes.pl");
239     int resultat_con=p_cunt.waitFor();
240     System.out.println("execution termine du fichier de contrainte ");
241
242     System.out.println("On a recupere la valeur de sortie de contraintes");
243     System.out.println("Resultat contraintes = "+resultat_con);
244

```

```

245         if(resultat_con==3){
246             //execution
247             System.out.println("La contrainte a valider,on va commencer l'execution");
248
249             Process p_cmd =runtime.exec(strcmd);
250             int resultat=p_cmd.waitFor();
251             System.out.println("Resultats du calcul = "+resultat);
252             // on n'as plus que a renvoyer le resultats
253
254             getCurrent().sendMessage("1,"+resultat, "provider@"+NOM_HOTE);
255             System.out.println("message envoyer = 1,"+resultat);
256         }else
257         {
258             //on a pas pu lexecuter on renvoi un message
259         }
260     }
261     catch(IOException ioe){
262         System.out.println("Erreur IO");
263         ioe.printStackTrace();
264     } catch (InterruptedException e) {
265         // TODO Auto-generated catch block
266         e.printStackTrace();
267     } catch (ParserConfigurationException e) {
268         // TODO Auto-generated catch block
269         e.printStackTrace();
270     } catch (SAXException e) {
271         // TODO Auto-generated catch block
272         e.printStackTrace();
273     } catch (XPathExpressionException e) {
274         // TODO Auto-generated catch block
275         e.printStackTrace();
276     } catch (XMPPEException e) {
277         // TODO Auto-generated catch block
278         e.printStackTrace();
279     }
280     }
281 }
282
283 }
284
285 public boolean isTravail_terminer() {

```

```

286     return travail_terminer;
287 }
288
289 public void setTravail_terminer(boolean travail_terminer) {
290     this.travail_terminer = travail_terminer;
291 }
292
293 public int getEnvoyer() {
294     return envoyer;
295 }
296
297 public void setEnvoyer(int envoyer) {
298     this.envoyer = envoyer;
299 }
300
301 public int getRecu() {
302     return reçu;
303 }
304
305 public void setRecu(int reçu) {
306     this.reçu = reçu;
307 }
308
309 public XmppManager getCurrent() {
310     return this;
311 }
312
313 public String getServer() {
314     return server;
315 }
316
317 public void setServer(String server) {
318     this.server = server;
319 }
320
321 public int getPort() {
322     return port;
323 }
324
325 public void setPort(int port) {
326     this.port = port;

```

```

327     }
328
329     public ConnectionConfiguration getConfig() {
330         return config;
331     }
332
333     public void setConfig(ConnectionConfiguration config) {
334         this.config = config;
335     }
336
337     public ChatManager getChatManager() {
338         return chatManager;
339     }
340
341     public void setChatManager(ChatManager chatManager) {
342         this.chatManager = chatManager;
343     }
344
345     public MessageListener getMessageListener() {
346         return messageListener;
347     }
348
349     public void setMessageListener(MessageListener messageListener) {
350         this.messageListener = messageListener;
351     }
352
353     public int getRetour_Providing() {
354         return retour_Providing;
355     }
356
357     public void setRetour_Providing(int retour_Providing) {
358         this.retour_Providing = retour_Providing;
359     }
360
361     public static int getPacketreplytimeout() {
362         return packetReplyTimeout;
363     }
364
365 }

```
