PROGRAMMATION RÉSEAUX, WEB ET MOBILES

Evaluation 2

SEEL Océane, FINK Jérôme
[Email address]

Table of Contents

Code du serveur trafic

1) Partie TRAMAP :

```
1.  public class RunnableTraitementEntree implements Runnable
2.  {
3.      private Socket CSocket = null;
4.      private DataInputStream dis = null;
5.      private DataOutputStream dos = null;
6.      private BeanBDAccess beanOracle;
7.
8.      boolean first = true;
9.
10.     public RunnableTraitementEntree(Socket s)
11.     {
12.         CSocket = s;
13.
14.         try
15.         {
16.             dis = new DataInputStream(new BufferedInputStream(CSocket.getInputStream()));
17.             dos = new DataOutputStream(new BufferedOutputStream(CSocket.getOutputStream()));
18.         }
19.         catch(IOException e)
20.         {
21.             System.err.println("RunnableTraitement : Host non trouvÃ© : " + e);
22.         }
23.
24.         beanOracle = new BeanBDAccess();
25.         try {
26.             beanOracle.connexionOracle("localhost", 1521, "TRAFIC", "TRAFIC", "XE");
27.         } catch (ClassNotFoundException ex) {
28.             System.err.println("Class not found " + ex.getMessage());
29.         } catch (SQLException ex) {
30.             System.err.println("SQL Exception (oracle)" + ex.getMessage());
31.         } catch (connexionException ex) {
32.             System.err.println(ex.getNumException() + " -- " + ex.getMessage());
33.         }
34.     }
35.
36.     @Override
37.     public void run()
38.     {
39.         String[] parts = (ReceiveMsg()).split("#");
40.
41.         if(parts[0].equals("LOGIN"))
42.         {
43.             if(!login(parts))
44.                 return;
45.         }
46.         else
47.         {
48.             SendMsg("ERR#RequÃ¨te invalide");
49.             return;
50.         }
51.
52.         boolean terminer = false;
53.
54.         while(!terminer)
55.         {
```

```
56.             parts = ReceiveMsg().split("#");
57.             switch (parts[0])
58.             {
59.                 case "INPUT_LORRY" :
60.                         inputLorry(parts);
61.                         break;
62.                 case "INPUT_LORRY_WITHOUT_RESERVATION" :
63.                         inputLorryWithoutReserv(parts);
64.                         break;
65.                 case "LIST_OPERATIONS" :
66.                         listOperation(parts);
67.                         break;
68.                 case "LOGOUT" :
69.                     terminer = true;
70.                     break;
71.
72.                 default :
73.                     terminer = true;
74.                     break;
75.             }
76.         }
77.
78.         System.err.println("fin du runnable");
79.     }
80.
81.     private boolean login(String[] part)
82.     {
83.         ResultSet rs = null;
84.
85.         try
86.         {
87.             rs = beanOracle.selection("PASSWORD", "UTILISATEURS", "LOGIN = '" + part[1]+"'");
88.         }
89.         catch(SQLException e){
90.             System.err.println(e.getStackTrace());
91.         }
92.
93.         String pwd = null;
94.
95.         try {
96.             if(!rs.next())
97.             {
98.                 SendMsg("ERR#Login invalide");
99.             }
100.            else
101.                pwd = rs.getString("PASSWORD");
102.        } catch (SQLException ex) {
103.            System.err.println(ex.getStackTrace());
104.        }
105.
106.        if(pwd.equals(part[2]))
107.        {
108.            SendMsg("ACK");
109.            return true;
110.        }
111.        else
112.            SendMsg("ERR#Mot de passe incorrecte");
113.
114.        return false;
115.    }
116.
```

```
117.    private void inputLorry(String[] request)
118.    {
119.
120.        ResultSet rs = null;
121.
122.        try {
123.            rs = beanOracle.selection("ID_CONTAINER", "CONTAINERS", "RESERVATION = '"+request[1]+"
    '");
124.        } catch (SQLException ex) {
125.            SendMsg("ERR#Base de donnÃ©e inaccessible");
126.            System.err.println("Erreur SQL exception input lorry");
127.            return;
128.        }
129.
130.        boolean isResultEmpty = true;
131.        int nbrElemParc = 0;
132.
133.        String[] idList =  request[2].split("@");
134.        System.out.println(request[2]);
135.        try {
136.            while(rs.next())
137.            {
138.                nbrElemParc++;
139.                isResultEmpty = false;
140.                String curId = null;
141.                String id = rs.getString("ID_CONTAINER");
142.                boolean invalidContainerID = true;
143.                for(String s : idList)
144.                {
145.                    System.out.println(s +"---"+id);
146.                    curId = s;
147.                    if(s.equals(id))
148.                    {
149.                        invalidContainerID = false;
150.                        break;
151.                    }
152.                }
153.                if(invalidContainerID)
154.                {
155.                    SendMsg("ERR#Le container " + curId +" ne fait pas partie de la reservation" )
    ;
156.                    return;
157.                }
158.
159.                if(nbrElemParc >= idList.length)
160.                    break;
161.            }
162.        } catch (SQLException ex) {
163.            SendMsg("ERR#Base de donnÃ©e inaccessible");
164.            System.err.println("Erreur SQL exception input lorry resultat");
165.            return;
166.        }
167.
168.        if(isResultEmpty)
169.        {
170.            SendMsg("ERR#Le numero de reservation demande n'existe pas");
171.            return;
172.        }
173.
174.        try {
175.            rs = beanOracle.selection("X, Y", "PARC", "ETAT=1");
```

```
176.         } catch (SQLException ex) {
177.             SendMsg("ERR#Base de donnÃ©e inaccessible");
178.             System.err.println("Erreur SQL exception input lorry" + ex.getStackTrace());
179.             return;
180.         }
181.
182.         String reponse = "ACK#";
183.
184.         try
185.         {
186.             for(int i = 0; i < idList.length ; i++)
187.             {
188.                 if(rs.next())
189.                 {
190.                     reponse = reponse + idList[i] + "==>("+rs.getString("X")+";"+rs.getString("Y")
    +")@";
191.                 }
192.                 else
193.                 {
194.                     SendMsg("ERR#Erreur pas assez de places reservees");
195.                     return;
196.                 }
197.             }
198.         }
199.         catch(SQLException ex){
200.             SendMsg("ERR#Base de donnee inaccessible");
201.             System.err.println("Erreur SQL exception input lorry" + ex.getStackTrace());
202.             return;
203.         }
204.
205.         SendMsg(reponse);
206.     }
207.
208.
209.     private void listOperation(String[] request)
210.     {
211.         String Select = "ID_MOUVEMENT, MOUVEMENTS.ID_CONTAINER, ID_TRANSPORTEUR_ENTRANT, DATE_ARRI
    VEE, ID_TRANSPORTEUR_SORTANT, POIDS, DATE_DEPART, DESTINATION, ID_SOCIETE";
212.         String From = "MOUVEMENTS INNER JOIN CONTAINERS ON MOUVEMENTS.ID_CONTAINER = CONTAINERS.ID
    _CONTAINER";
213.         String Where = null;
214.
215.         if(request[1].equals("societe"))
216.             Where = "CONTAINERS.ID_SOCIETE = '"+request[2]+"'";
217.         if(request[1].equals("destination"))
218.             Where = "DESTINATION = '"+request[2]+"'";
219.         if(request[1].equals("date"))
220.             Where = "To_date(DATE_ARRIVEE, 'DD/MM/YYYY') BETWEEN To_date('"+request[2]+"', 'DD/MM/
    YYYY') AND To_date('"+request[3]+"', 'DD/MM/YYYY')";
221.
222.         //To_date(madate, 'DD/MM/YYYY')
223.
224.
225.         if(Where == null)
226.         {
227.             SendMsg("ERR#Recherche impossible sur ce critere");
228.         }
229.
230.         ResultSet rs = null;
231.
232.         try {
```

```
233.            rs = beanOracle.selection(Select, From, Where);
234.        } catch (SQLException ex) {
235.            SendMsg("ERR#Base de donnee inaccessible");
236.            System.err.println("Erreur SQL exception input lorry" + ex.getStackTrace());
237.            return;
238.        }
239.
240.        boolean empty = true;
241.        String message = "";
242.
243.        try {
244.            while(rs.next())
245.            {
246.                empty = false;
247.                message = message + rs.getString("ID_MOUVEMENT") + "  --
    -  " + rs.getString("ID_CONTAINER") + "  ---  " + rs.getString("ID_TRANSPORTEUR_ENTRANT") + "  --
    -  ";
248.                message = message + rs.getString("DATE_ARRIVEE") + "  --
    -  " + rs.getString("ID_TRANSPORTEUR_SORTANT") + "  ---  " + rs.getString("POIDS") + "  ---  ";
249.                message = message + rs.getString("DATE_DEPART") + "  --
    -  " + rs.getString("DESTINATION") + "  ---  " + rs.getString("ID_SOCIETE")+"#";
250.            }
251.        } catch (SQLException ex) {
252.            SendMsg("ERR#Base de donnee inaccessible");
253.            System.err.println("Erreur SQL exception input lorry" + ex.getStackTrace());
254.            return;
255.        }
256.        if(empty)
257.        {
258.            SendMsg("ERR#Aucun resultats pour la societe " + request[2]);
259.            return;
260.        }
261.
262.        SendMsg("ACK#"+message);
263.    }
264.
265.
266.    private void inputLorryWithoutReserv(String[] request)
267.    {
268.        ResultSet rs = null;
269.
270.        String[] idList =  request[2].split("@");
271.
272.        try {
273.            rs = beanOracle.selection("X, Y", "PARC", "ETAT=0");
274.        } catch (SQLException ex) {
275.            SendMsg("ERR#Base de donnÃ©e inaccessible");
276.            System.err.println("Erreur SQL exception input lorry" + ex.getStackTrace());
277.            return;
278.        }
279.
280.        String reponse = "ACK#";
281.        ArrayList emplacement = new ArrayList();
282.
283.        try// on regarde si y'a assez de place et on recupere l'id de ces places.
284.        {
285.            for(int i = 0; i < idList.length ; i++)
286.            {
287.                if(rs.next())
288.                {
```

```
289.                     reponse = reponse + idList[i] + "==>("+rs.getString("X")+";"+rs.getString("Y")
    +")@";
290.                     emplacement.add(rs.getString("X")+";"+rs.getString("Y"));
291.                 }
292.                 else
293.                 {
294.                     SendMsg("ERR#Erreur pas assez de places");
295.                     return;
296.                 }
297.             }
298.         }
299.         catch(SQLException ex){
300.             SendMsg("ERR#Base de donnÃ©e inaccessible");
301.             System.err.println("Erreur SQL exception input lorry" + ex.getStackTrace());
302.             return;
303.         }
304.
305.         //On insert les containers ajoutÃ©s dans la BD et on leur met un numÃ©ro de rÃ©servation +
    on rÃ©serve leurs places
306.         Random rand =  new Random();
307.         int resID = rand.nextInt(999999);
308.         for(int i = 0; i < idList.length; i++)
309.         {
310.             String[] coord = emplacement.get(i).toString().split(";");
311.             HashMap<String, String> insertion = new HashMap();
312.             HashMap<String, String> update = new HashMap();
313.
314.             insertion.put("ID_CONTAINER", idList[i]);
315.             insertion.put("RESERVATION", Integer.toString(resID));
316.
317.             update.put("ETAT", "1");
318.
319.             try {
320.                 beanOracle.ecriture("CONTAINERS", insertion);
321.                 beanOracle.miseAJour("PARC", update, "X="+coord[0]+" AND Y=" + coord[1]);
322.             } catch (requeteException ex) {
323.                 System.err.println("Erreur d'insertion ");
324.             }
325.         }
326.
327.         SendMsg(reponse);
328.     }
329.
330.     /* Envoi d'un message au client */
331.     public void SendMsg(String msg)
332.     {
333.         String chargeUtile = msg;
334.         int taille = chargeUtile.length();
335.         StringBuffer message = new StringBuffer(String.valueOf(taille) + "#" + chargeUtile);
336.
337.         try
338.         {
339.             dos.write(message.toString().getBytes());
340.             dos.flush();
341.         }
342.         catch(IOException e)
343.         {
344.             System.err.println("RunnableTraitement : Erreur d'envoi de msg (IO) : " + e);
345.         }
346.     }
347.
```

```
348.    /* RÃ©ception d'un message du client */
349.    public String ReceiveMsg()
350.    {
351.        byte b;
352.        StringBuffer taille = new StringBuffer();
353.        StringBuffer message = new StringBuffer();
354.
355.        try
356.        {
357.            while ((b = dis.readByte()) != (byte)'#')
358.            {
359.                if (b != (byte)'#')
360.                    taille.append((char)b);
361.            }
362.
363.            for (int i = 0; i < Integer.parseInt(taille.toString()); i++)
364.            {
365.                b = dis.readByte();
366.                message.append((char)b);
367.            }
368.        }
369.        catch(IOException e)
370.        {
371.            System.err.println("RunnableTraitement : Erreur de reception de msg (IO) : " + e);
372.        }
373.
374.        return message.toString();
375.    }
376.}
```

2) Partie BOOMAP

```
3)  public class RunnableBOOMAP implements Runnable{
4)      private Socket CSocket = null;
5)      private DataInputStream dis = null;
6)      private DataOutputStream dos = null;
7)      private BeanBDAccess beanOracle;
8)
9)      boolean first = true;
10)
11)     public RunnableBOOMAP(Socket s)
12)     {
13)         CSocket = s;
14)
15)         try
16)         {
17)             dis = new DataInputStream(new BufferedInputStream(CSocket.getInputStream())
    );
18)             dos = new DataOutputStream(new BufferedOutputStream(CSocket.getOutputStream
    ()));
19)         }
20)         catch(IOException e)
21)         {
22)             System.err.println("RunnableTraitement : Host non trouvÃ© : " + e);
23)         }
24)
25)         beanOracle = new BeanBDAccess();
26)         try {
27)             beanOracle.connexionOracle("localhost", 1521, "TRAFIC", "TRAFIC", "XE");
28)         } catch (ClassNotFoundException ex) {
29)             System.err.println("Class not found " + ex.getMessage());
30)         } catch (SQLException ex) {
31)             System.err.println("SQL Exception (oracle)" + ex.getMessage());
32)         } catch (connexionException ex) {
33)             System.err.println(ex.getNumException() + " -- " + ex.getMessage());
34)         }
35)     }
36)
37)     @Override
38)     public void run()
39)     {
40)         String[] parts = (ReceiveMsg()).split("#");
41)
42)         if(parts[0].equals("LOGIN"))
43)         {
44)             if(!login(parts))
45)                 return;
46)         }
47)         else
48)         {
49)             SendMsg("ERR#RequÃ¨te invalide");
50)             return;
51)         }
52)
53)         boolean terminer = false;
54)
55)         while(!terminer)
56)         {
57)             parts = ReceiveMsg().split("#");
58)             System.err.println(parts[0]);
59)             switch (parts[0])
```

```
60)             {
61)                 case "LOGOUT" :
62)                     terminer = true;
63)                     break;
64)
65)                 case "GET_XY" :
66)                     get_xy(parts);
67)                     break;
68)
69)                 case "SEND_WEIGHT" :
70)                     send_weight(parts);
71)                     break;
72)
73)                 case "GET_LIST" :
74)                     get_list(parts);
75)                     break;
76)
77)                 case "SIGNAL_DEP" :
78)                     signal_dep(parts);
79)                     break;
80)
81)                 default :
82)                     terminer = true;
83)                     break;
84)             }
85)         }
86)
87)         try {
88)             CSocket.close();
89)         } catch (IOException ex) {
90)             System.err.println("Erreur de close : " + ex.getStackTrace());
91)         }
92)
93)     }
94)
95)     private boolean login(String[] part)
96)     {
97)         ResultSet rs = null;
98)
99)         try
100)                {
101)                    rs = beanOracle.selection("PASSWORD", "UTILISATEURS", "LOGIN = '" +
    part[1]+"'");
102)                }
103)                catch(SQLException e){
104)                    System.err.println(e.getStackTrace());
105)                }
106)
107)                String pwd = null;
108)
109)                try {
110)                    if(!rs.next())
111)                    {
112)                        SendMsg("ERR#Login invalide");
113)                    }
114)                    else
115)                        pwd = rs.getString("PASSWORD");
116)                } catch (SQLException ex) {
117)                    System.err.println(ex.getStackTrace());
118)                }
119)
```

```
120)                if(pwd.equals(part[2]))
121)                {
122)                    SendMsg("ACK");
123)                    return true;
124)                }
125)                else
126)                    SendMsg("ERR#Mot de passe incorrecte");
127)
128)                return false;
129)            }
130)
131)            private void get_xy(String[] request)
132)            {
133)                String message = "ACK#";
134)                ResultSet rs = null;
135)
136)                try {
137)                    rs = beanOracle.selection("X, Y", "PARC", "ETAT = 1");
138)                } catch (SQLException ex) {
139)                    SendMsg("ERR#Probleme SQL");
140)                    System.err.println(ex.getStackTrace());
141)                    return;
142)                }
143)                System.err.println(request[3]);
144)                String[] listContainer = request[3].split("\\@");
145)
146)                for(String s : listContainer)
147)                {
148)                    System.err.println(s);
149)                    try {
150)                        if(rs.next())
151)                        {
152)                            if(!message.equals("ACK#"))
153)                                message = message + "@";
154)
155)                            String[] infoContainer = s.split(";");
156)
157)
158)                            //insert societe
159)                            HashMap<String, String> insertSociete = new HashMap<>();
160)
161)                            insertSociete.put("ID_SOCIETE", request[2]);
162)                            try {
163)                                beanOracle.ecriture("SOCIETES", insertSociete);
164)                            } catch (requeteException ex) {
165)                                System.err.println("La societe existe deja");
166)                            }
167)
168)                            //Insert transporteur
169)
170)                            HashMap<String, String> insertTransporteur = new HashMap<>()
    ;
171)
172)                            insertTransporteur.put("ID_TRANSPORTEUR", request[1]);
173)                            insertTransporteur.put("ID_SOCIETE", request[2]);
174)                            try {
175)                                beanOracle.ecriture("TRANSPORTEURS", insertTransporteur)
    ;
176)                            } catch (requeteException ex) {
177)                                System.err.println("Le transporteur existe deja");
178)                            }
```

```
179)
180)
181)                            //INSERTION DANS PARC
182)                            HashMap<String, String> updateParc = new HashMap();
183)                            updateParc.put("ETAT", "2");
184)                            updateParc.put("ID_CONTAINER", infoContainer[0]);
185)                            Calendar cal = Calendar.getInstance();
186)                            SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/YYYY");
187)                            updateParc.put("DATE_ARRIVEE", sdf.format(cal.getTime()));
188)                            updateParc.put("DESTINATION", infoContainer[1]);
189)
190)                            System.err.println(infoContainer[0] + "  " + infoContainer[1
    ] + "  " + sdf.format(cal.getTime()));
191)                            try {
192)                                beanOracle.miseAJour("PARC", updateParc, "X = "+rs.getSt
    ring("X")+" AND Y = " + rs.getString("Y"));
193)                            } catch (requeteException ex) {
194)                                System.err.println("echec update parc : " + ex.getMessag
    e());
195)                            }
196)
197)                            //Insert mouvement
198)
199)                            HashMap<String, String> insertMouvement = new HashMap<>();
200)                            Random rand = new Random();
201)                            int idMouvement = rand.nextInt(89892);
202)                            insertMouvement.put("ID_MOUVEMENT", Integer.toString(idMouve
    ment));
203)                            insertMouvement.put("ID_CONTAINER", infoContainer[0]);
204)                            insertMouvement.put("ID_TRANSPORTEUR_ENTRANT", request[1]);

205)                            insertMouvement.put("DATE_ARRIVEE", sdf.format(cal.getTime()
    ));
206)                            insertMouvement.put("DESTINATION", infoContainer[1]);
207)
208)                            try {
209)                                beanOracle.ecriture("MOUVEMENTS", insertMouvement);
210)                            } catch (requeteException ex) {
211)                                System.err.println("Le mouvement existe deja");
212)                            }
213)
214)                            //UPDATE CONTAINERS
215)
216)                            HashMap<String, String> updateContainer = new HashMap<>();
217)
218)                            updateContainer.put("ID_SOCIETE", request[2]);
219)
220)                            try {
221)                                beanOracle.miseAJour("CONTAINERS", updateContainer, "ID_
    CONTAINER = '"+infoContainer[0]+"'");
222)                            } catch (requeteException ex) {
223)                                System.err.println("Erreur MAJ container");
224)                            }
225)
226)                            message = message + rs.getString("X")+";"+rs.getString("Y");

227)
228)                        }
229)                        else
230)                        {
231)                            System.err.println("pas assez de resultat");
```

```
232)                            SendMsg("ERR#Pas assez de resultats");
233)                        }
234)                    } catch (SQLException ex) {
235)                        SendMsg("ERR#Probleme SQL");
236)                        System.err.println(ex.getStackTrace());
237)                        return;
238)                    }
239)                }
240)
241)                SendMsg(message);
242)            }
243)
244)            private void get_list(String[] request)
245)            {
246)                ResultSet rs = null;
247)                String transport = null;
248)
249)                if(request[2].equals("1"))
250)                    transport = "TRAIN";
251)                else
252)                    transport = "BATEAU";
253)
254)                try {
255)                    rs = beanOracle.selection("ID_CONTAINER, X, Y", "PARC", "TRANSPORT =
    '"+transport+"' AND UPPER(DESTINATION) = UPPER('"+request[3]+"')");
256)                } catch (SQLException ex) {
257)                    SendMsg("ERR#Acces a la BD impossible");
258)                    System.err.println("erreur: " + ex.getStackTrace().toString());
259)                    return;
260)                }
261)
262)                String message = "";
263)
264)                try {
265)                    while(rs.next())
266)                    {
267)                        if(!message.isEmpty())
268)                            message = message + "#";
269)
270)                        message = message + rs.getString("ID_CONTAINER") + "@" + rs.getS
    tring("X") + ";" + rs.getString("Y");
271)                    }
272)                } catch (SQLException ex) {
273)                    SendMsg("ERR#Acces a la BD impossible");
274)                    System.err.println("test" + ex.getStackTrace().toString());
275)                    return;
276)                }
277)
278)                if(message.isEmpty())
279)                {
280)                    SendMsg("ERR#Aucun container pour la destination");
281)                    return;
282)                }
283)
284)                SendMsg("ACK#"+message);
285)            }
286)
287)            private void signal_dep(String[] requete)
288)            {
289)                //Insertion transporteur :
290)
```

```
291)                //Insert transporteur
292)
293)                HashMap<String, String> insertTransporteur = new HashMap<>();
294)
295)                insertTransporteur.put("ID_TRANSPORTEUR", requete[1]);
296)                try {
297)                    beanOracle.ecriture("TRANSPORTEURS", insertTransporteur);
298)                } catch (requeteException ex) {
299)                    System.err.println("Le transporteur existe deja");
300)                }
301)
302)                for(String coord : requete)
303)                {
304)                    if(coord.equals(requete[1]) || coord.equals(requete[0]))
305)                        continue;
306)
307)                    //MAJ etat parc
308)                    String[] splitCoord = coord.split(";");
309)                    HashMap<String, String> updateParc = new HashMap();
310)                    updateParc.put("ETAT", "0");
311)
312)                    try {
313)                            beanOracle.miseAJour("PARC", updateParc, "X = "+splitCoord[0
    ]+" AND Y = " + splitCoord[1]);
314)                    } catch (requeteException ex) {
315)                        System.err.println("echec update parc : " + ex.getMessage());
316)                    }
317)
318)
319)                    //MAJ mouvement
320)
321)                    //Insert mouvement
322)
323)                    HashMap<String, String> insertMouvement = new HashMap<>();
324)                    Calendar cal = Calendar.getInstance();
325)                    SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/YYYY");
326)                    insertMouvement.put("ID_TRANSPORTEUR_SORTANT", requete[1]);
327)                    insertMouvement.put("DATE_DEPART", sdf.format(cal.getTime()));
328)
329)                    String where = "ID_CONTAINER = (SELECT ID_CONTAINER FROM PARC WHERE
    X = "+splitCoord[0]+" AND Y = " + splitCoord[1]+")";
330)                    try {
331)                        beanOracle.miseAJour("MOUVEMENTS", insertMouvement, where);
332)                    } catch (requeteException ex) {
333)                        System.err.println("Le mouvement existe deja");
334)                    }
335)                }
336)
337)
338)
339)                SendMsg("ACK#");
340)            }
341)
342)            private void send_weight(String[] requete)
343)            {
344)
345)                for(String s : requete)
346)                {
347)
348)                    System.err.println(s);
349)                    if(s.equals("SEND_WEIGHT"))
```

```
350)                         continue;
351)
352)                     String[] contTraite = s.split(";");
353)                     String transport;
354)
355)                     if(contTraite[3].equals("1"))
356)                         transport = "TRAIN";
357)                     else
358)                         transport = "BATEAU";
359)
360)                     HashMap<String,String> updateMouvement = new HashMap<>();
361)                     System.err.println(contTraite[2]);
362)                     updateMouvement.put("POIDS", contTraite[2]);
363)
364)                     try {
365)                         beanOracle.miseAJour("MOUVEMENTS", updateMouvement, "ID_CONTAINE
    R = '"+contTraite[0]+"'");
366)                     } catch (requeteException ex) {
367)                         System.err.println("Aucun mouvement trouve");
368)                     }
369)
370)                     HashMap<String, String> updateParc = new HashMap<>();
371)                     updateParc.put("TRANSPORT", transport);
372)
373)                     try {
374)                         beanOracle.miseAJour("PARC", updateParc, "ID_CONTAINER = '"+cont
    Traite[0]+"'");
375)                     } catch (requeteException ex) {
376)                         System.err.println("Aucun parc trouve");
377)                     }
378)
379)                 }
380)                 SendMsg("ACK#");
381)             }
382)
383)
384)             /* Envoi d'un message au client */
385)             public void SendMsg(String msg)
386)             {
387)                 String chargeUtile = msg;
388)                 int taille = chargeUtile.length();
389)                 StringBuffer message = new StringBuffer(String.valueOf(taille) + "#" + c
    hargeUtile);
390)
391)                 try
392)                 {
393)                     dos.write(message.toString().getBytes());
394)                     dos.flush();
395)                 }
396)                 catch(IOException e)
397)                 {
398)                     System.err.println("RunnableTraitement : Erreur d'envoi de msg (IO)
    : " + e);
399)                 }
400)             }
401)
402)             /* RÃ©ception d'un message du client */
403)             public String ReceiveMsg()
404)             {
405)                 byte b;
406)                 StringBuffer taille = new StringBuffer();
```

```
407)                StringBuffer message = new StringBuffer();
408)
409)                try
410)                {
411)                    while ((b = dis.readByte()) != (byte)'#')
412)                    {
413)                        if (b != (byte)'#')
414)                            taille.append((char)b);
415)                    }
416)
417)                    for (int i = 0; i < Integer.parseInt(taille.toString()); i++)
418)                    {
419)                        b = dis.readByte();
420)                        message.append((char)b);
421)                    }
422)                }
423)                catch(IOException e)
424)                {
425)                    System.err.println("RunnableTraitement : Erreur de reception de msg
    (IO) : " + e);
426)                }
427)
428)                return message.toString();
429)            }
430)        }
```

Code administrateur serveur container

```
1.  public class GUIAdmin extends javax.swing.JFrame {
2.
3.
4.      private Socket cliSocket;
5.
6.      private DataInputStream dis;
7.      private DataOutputStream dos;
8.
9.      /**
10.      * Creates new form GUIAdmin
11.      */
12.     public GUIAdmin() {
13.         initComponents();
14.     }
15.
16.
17.
18.     private void connexionButtonActionPerformed(java.awt.event.ActionEvent evt) {

19.
20.         if(connexionButton.getText().equals("Connexion"))
21.         {
22.             try
23.             {
24.                 cliSocket = new Socket(ipTextField.getText(), Integer.parseInt(portText
    Field.getText()));
25.                 dis = new DataInputStream(new BufferedInputStream(cliSocket.getInputStr
    eam()));
26.                 dos = new DataOutputStream(new BufferedOutputStream(cliSocket.getOutput
    Stream()));
27.             }
28.             catch (IOException ex)
29.             {
30.                 System.err.println("Erreur gui admin connexion : " + ex);
31.             }
32.
33.             String message = protocoleCSA.LOGIN + "#"+LoginTextField.getText()+"#"+pass
    wordField.getText();
34.
35.             SendMsg(message);
36.
37.             System.out.println(ReceiveMsg());
38.
39.             connexionButton.setText("DÃ©connexion");
40.         }
41.         else
42.         {
43.             connexionButton.setText("Connexion");
44.             deconnexion();
45.         }
46.     }
47.
48.     private void listerButtonActionPerformed(java.awt.event.ActionEvent evt) {

49.         if(cliSocket == null)
50.             return;
51.
52.         SendMsg(protocoleCSA.LISTCLIENT + "#");
```

```
53.
54.         String str = ReceiveMsg();
55.
56.         String split[] = str.split("#");
57.
58.         listTextArea.setText("");
59.         for(int i = 1; i < split.length; i++)
60.             listTextArea.append(split[i] + "\n");
61.
62.     }
63.
64.     private void pauseButton1ActionPerformed(java.awt.event.ActionEvent evt) {
65.         SendMsg(protocoleCSA.PAUSE + "#");
66.
67.         String str = ReceiveMsg();
68.     }
69.
70.     private void continuerButtonActionPerformed(java.awt.event.ActionEvent evt) {
71.         SendMsg(protocoleCSA.CONTINUER + "#");
72.
73.         String str = ReceiveMsg();
74.     }
75.
76.     private void stopButtonActionPerformed(java.awt.event.ActionEvent evt) {
77.
78.         int sec = 0;
79.
80.         try
81.         {
82.             sec = Integer.parseInt(secondesTextField.getText());
83.         }
84.         catch(NumberFormatException nfe)
85.         {
86.             sec = 0;
87.         }
88.
89.         if(sec > 1000 || sec < 0)
90.             sec = 0;
91.
92.         SendMsg(protocoleCSA.STOP + "#" + sec);
93.
94.         String str = ReceiveMsg();
95.     }
96.
97.     public void deconnexion()
98.     {
99.         SendMsg(protocoleCSA.LOGOUTCSA + "#");
100.                ReceiveMsg();
101.                try
102.                {
103.                    dos.close();
104.                    dis.close();
105.                    cliSocket.close();
106.                    cliSocket = null;
107.                    System.out.println("ClientServeurBateau : Client dÃ©connectÃ©");
108.                }
109.                catch(IOException e)
110.                {
```

```
111.                     System.err.println("ClientServeurBateau : Erreur de dÃ©connexion : "
    + e);
112.                 }
113.             }
114.
115.
116.             public void SendMsg(String chargeUtile)
117.             {
118.                 int taille = chargeUtile.length();
119.                 String message = String.valueOf(taille) + "#" + chargeUtile;
120.
121.                 try
122.                 {
123.                     dos.write(message.getBytes());
124.                     dos.flush();
125.                 }
126.                 catch(IOException e)
127.                 {
128.                     System.err.println("ClientServeurBateu : Erreur d'envoi de msg (IO)
   : " + e);
129.                 }
130.             }
131.
132.             public String ReceiveMsg()
133.             {
134.                 byte b;
135.                 StringBuffer taille = new StringBuffer();
136.                 StringBuffer message = new StringBuffer();
137.
138.                 try
139.                 {
140.                     while ((b = dis.readByte()) != (byte)'#')
141.                     {
142.                         if (b != (byte)'#')
143.                             taille.append((char)b);
144.                     }
145.
146.                     for (int i = 0; i < Integer.parseInt(taille.toString()); i++)
147.                     {
148.                         b = dis.readByte();
149.                         message.append((char)b);
150.                     }
151.                 }
152.                 catch(IOException e)
153.                 {
154.                     System.err.println("ClientServeurBateau : Erreur de reception de msg
    (IO) : " + e);
155.                 }
156.
157.                 reponseLabel.setText("reponse serveur : " + message.toString()); // Ã  r
   etirer
158.                 return message.toString();
159.             }
160.
161.
162.             /**
163.              * @param args the command line arguments
164.              */
165.             public static void main(String args[]) {
166.                 /* Set the Nimbus look and feel */
```

```
167.                //<editor-
    fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
168.                /* If Nimbus (introduced in Java SE 6) is not available, stay with the d
    efault look and feel.
169.                 * For details see http://download.oracle.com/javase/tutorial/uiswing/lo
    okandfeel/plaf.html
170.                 */
171.                try {
172.                    for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIMana
    ger.getInstalledLookAndFeels()) {
173.                        if ("Nimbus".equals(info.getName())) {
174.                            javax.swing.UIManager.setLookAndFeel(info.getClassName());
175.                            break;
176.                        }
177.                    }
178.                } catch (ClassNotFoundException ex) {
179.                    java.util.logging.Logger.getLogger(GUIAdmin.class.getName()).log(jav
    a.util.logging.Level.SEVERE, null, ex);
180.                } catch (InstantiationException ex) {
181.                    java.util.logging.Logger.getLogger(GUIAdmin.class.getName()).log(jav
    a.util.logging.Level.SEVERE, null, ex);
182.                } catch (IllegalAccessException ex) {
183.                    java.util.logging.Logger.getLogger(GUIAdmin.class.getName()).log(jav
    a.util.logging.Level.SEVERE, null, ex);
184.                } catch (javax.swing.UnsupportedLookAndFeelException ex) {
185.                    java.util.logging.Logger.getLogger(GUIAdmin.class.getName()).log(jav
    a.util.logging.Level.SEVERE, null, ex);
186.                }
187.                //</editor-fold>
188.
189.                /* Create and display the form */
190.                java.awt.EventQueue.invokeLater(new Runnable() {
191.                    public void run() {
192.                        new GUIAdmin().setVisible(true);
193.                    }
194.                });
195.            }
196.
197.            // Variables declaration - do not modify
198.            private javax.swing.JTextField LoginTextField;
199.            private javax.swing.JButton connexionButton;
200.            private javax.swing.JButton continuerButton;
201.            private javax.swing.JLabel ipLabel;
202.            private javax.swing.JTextField ipTextField;
203.            private javax.swing.JScrollPane jScrollPane1;
204.            private javax.swing.JTextArea listTextArea;
205.            private javax.swing.JButton listerButton;
206.            private javax.swing.JLabel loginLabel;
207.            private javax.swing.JPasswordField passwordField;
208.            private javax.swing.JLabel passwordTextField;
209.            private javax.swing.JButton pauseButton1;
210.            private javax.swing.JLabel portLabel;
211.            private javax.swing.JTextField portTextField;
212.            private javax.swing.JLabel reponseLabel;
213.            private javax.swing.JLabel secondesLabel;
214.            private javax.swing.JTextField secondesTextField;
215.            private javax.swing.JButton stopButton;
216.            // End of variables declaration
217.        }
```

Schéma relationnel BD_COMPTA

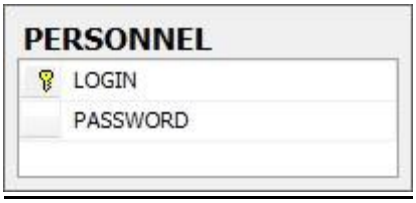
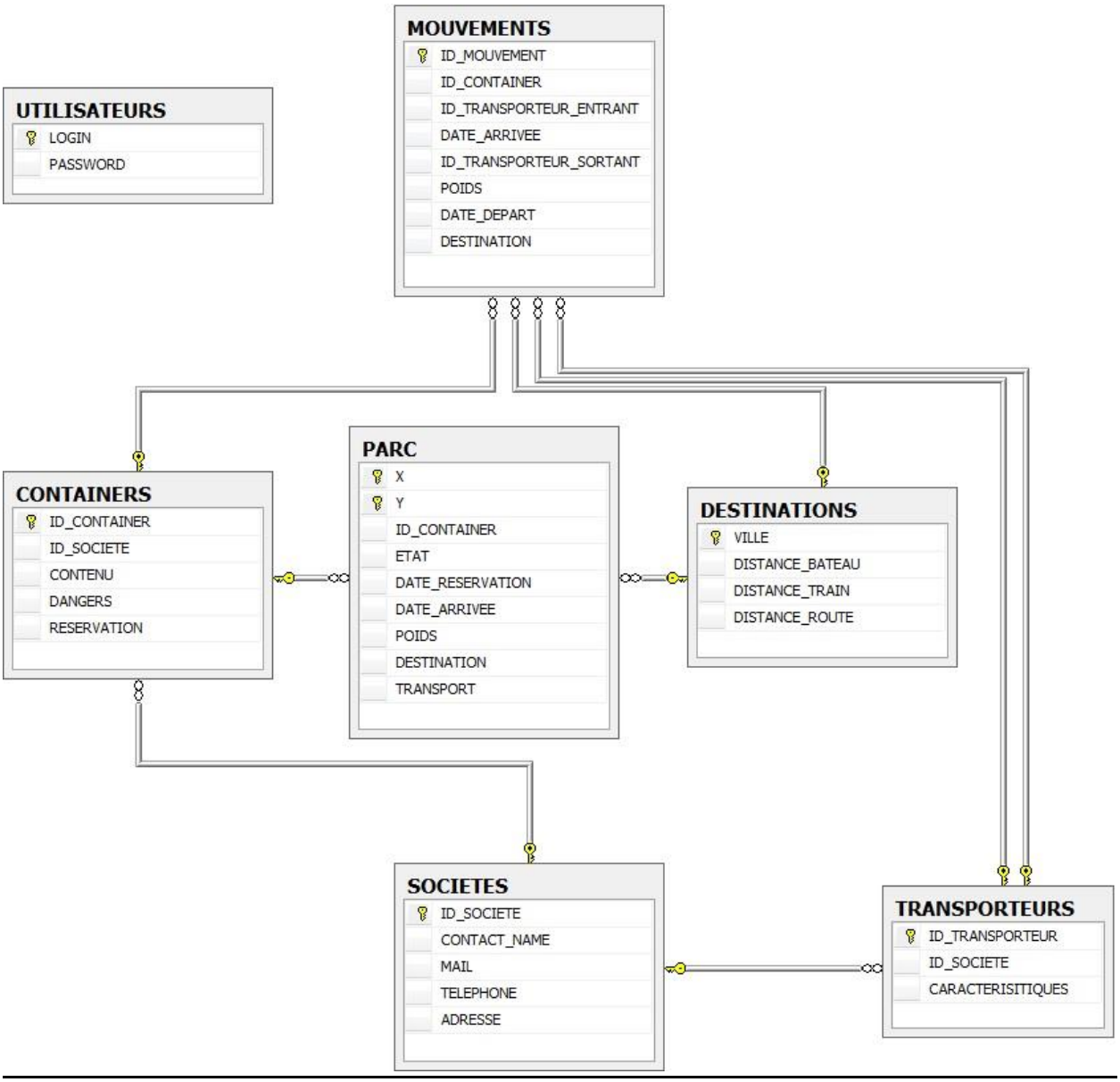


Schéma relationnel BD_TRAFIC



Trame chat UDP

Question :

```
27038 207.6093790(192.168.1.8          224.42.42.1          UDP        76 Source port: 31049  Destination port: 31049
 01 00 5e 2a 2a 01 40 16  7e ad 79 3d 08 00 45 00   ..^**.@. ~.y=..E.
 00 3e 3f be 00 00 01 11  ae 15 c0 a8 01 08 e0 2a   .>?..... .......*
 2a 01 79 49 79 49 00 2a  cc 17 6a 65 72 6f 6d 65   *.yIyI.* ..jerome
 23 51 39 38 32 33 33 23  6e 6f 75 76 65 6c 6c 65   #Q98233# nouvelle
 20 71 75 65 73 74 69 6f  6e 23 35 32                questio n#52
```

Infos :

```
  26037 198.2982180(192.168.1.8          224.42.42.1          UDP        62 Source port: 31049  Destination port: 31049
0000  01 00 5e 2a 2a 01 40 16  7e ad 79 3d 08 00 45 00   ..^**.@. ~.y=..E.
0010  00 30 3d cf 00 00 01 11  b0 12 c0 a8 01 08 e0 2a   .0=..... .......*
0020  2a 01 79 49 79 49 00 1c  cc 09 6a 65 72 6f 6d 65   *.yIyI.. ..jerome
0030  23 49 6e 66 6f 73 23 62  6f 6e 6a 6f 75 72         #Infos#b onjour
```