```java
package Serveur.server;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.Socket;
import java.net.SocketTimeoutException;

import Serveur.Main;
import Serveur.maths.Fonction;
import Serveur.triangulation.Capteur;

public class ClientTriangulation implements Runnable {
    private volatile boolean active = false, running = false;
    private final Socket socket;
    private final ServerTriangulation server;
    private InputStream in = null;
    private OutputStream out = null;
    public Capteur capteur = null;
    public static final double ZOOM = 2;

    public ClientTriangulation(Socket socket, ServerTriangulation server)
            throws IOException {
        this.socket = socket;
        this.server = server;
        in = socket.getInputStream();
        out = socket.getOutputStream();
    }

    public void start() {
        new Thread(this).start();
    }

    private String receive() throws IOException {
        byte[] buf = new byte[1024];
        StringBuffer strBuf = new StringBuffer();
        int len, i;

        do {
            len = in.read(buf);

            if (len > 0) {
                for (i = 0; i < len; i++)
                    strBuf.append((char) buf[i]);
            }
        } while (len == buf.length);

        String msg = strBuf.toString();
        return msg == "" ? null : msg;
    }

    private void send(String msg) throws IOException {
        byte[] buffer = msg.getBytes("UTF-8");
        out.write(buffer);
        out.flush();
    }

    private boolean setup() {
        String msg = null;

        try {
            msg = receive();
```

```
63          } catch (IOException e) {
64              if (!(e instanceof SocketTimeoutException))
65                  e.printStackTrace();
66          }
67
68          boolean clientIsCarteArduino = false;
69
70          if (msg == null)
71              clientIsCarteArduino = true;
72          else if (msg.startsWith("GET / HTTP/1.1"))
73              clientIsCarteArduino = false;
74          else if (msg.startsWith("arduino"))
75              clientIsCarteArduino = true;
76
77          if (!clientIsCarteArduino) {
78              try {
79                  envoyerLeSite();
80              } catch (IOException e) {
81                  e.printStackTrace();
82              }
83
84              try {
85                  socket.close();
86              } catch (IOException e) {
87                  e.printStackTrace();
88              }
89          }
90
91          return clientIsCarteArduino;
92      }
93
94      @Override
95      public void run() {
96          active = running = true;
97          boolean clientIsCarteArduino = setup();
98
99          if (clientIsCarteArduino) {
100             System.out.println(" => arduino");
101         } else {
102             System.out.println(" => pas arduino");
103             active = false;
104         }
105
106         while (active) {
107             try {
108                 String msg = receive();
109
110                 if (Main.main.DEBUG)
111                     System.out.println(socket.getInetAddress() + " > " + msg);
112
113                 performRequest(msg);
114             } catch (IOException e) {
115                 if (!(e instanceof SocketTimeoutException))
116                     e.printStackTrace();
117                 else
118                     active = false;
119             }
120         }
121
122         System.out.println(
123                 "Le client " + socket.getInetAddress() + " est déconnecté.("
124                         + server.clients.size() + " clients restants)");
```

```
125
126             server.doModifications(() -> {
127                 server.clients.remove(this);
128                 return null;
129             });
130
131             if (clientIsCarteArduino)
132             {
133                 Main.main.doModifications(() -> {
134                     Main.main.map.removeCapteur(capteur);
135                     Main.main.resetBalles();
136                     return null;
137                 });
138
139                 capteur.close();
140             }
141
142         running = false;
143     }
144
145     public static boolean isDigit(char c) {
146         return c >= 45 && c <= 57 && c != 47;
147     }
148
149     /**
150      * renvoie un vecteur (x = le nombre, y = l'indice de fin + 1)
151      */
152     public static double nextNumber(String str, int index) {
153         int length = str.length();
154         int start = index;
155
156         while (start < length && !isDigit(str.charAt(start)))
157             start++;
158
159         int end = start;
160
161         while (end < length && isDigit(str.charAt(end)))
162             end++;
163
164         return new Double(str.substring(start, end));
165     }
166
167     protected void performRequest(String msg) {
168         String msgUpper = msg.toUpperCase();
169
170         if (msgUpper.contains("CARD_ID")) {
171             capteur = new Capteur(0, 0);
172
173             Main.main.doModifications(() -> {
174                 Main.main.map.addCapteur(capteur);
175                 return null;
176             });
177
178             Main.main.map.doModifications(() -> {
179                 Main.main.resetBalles();
180                 return null;
181             });
182         }
183
184         if (msgUpper.contains("RSSI")) {
185             capteur.setRSSI(nextNumber(msgUpper, msgUpper.indexOf("RSSI")));
186         }
```

```
187
188         if (msgUpper.contains("DISTANCE")) {
189             capteur.setDistance(
190                     nextNumber(msgUpper, msgUpper.indexOf("DISTANCE")));
191         }
192
193         if (msgUpper.contains("X")) {
194             double x = nextNumber(msgUpper, msgUpper.indexOf("X")) * ZOOM;
195             capteur.setPosition(x, capteur.getPosition().y);
196         }
197
198         if (msgUpper.contains("Y")) {
199             double y = nextNumber(msgUpper, msgUpper.indexOf("Y")) * ZOOM;
200             capteur.setPosition(capteur.getPosition().x, y);
201         }
202
203         if (msgUpper.contains("STOP"))
204             active = false;
205     }
206
207     public void envoyerLeSite() throws IOException {
208         String code = server.constructHtml();
209         send(code);
210     }
211
212     public boolean isRunning() {
213         return running;
214     }
215
216     public void close() {
217         active = false;
218     }
219
220     public synchronized Object doModifications(Fonction f) {
221         return f.method();
222     }
223 }
```