PECL2

Alvaro Gomez Martinez - 03223127E
Denisa Gabriela Moldovan - X8613167N
Angel Oroquieta Gómez - 25206584X
David Marquez Minguez - 47319570Z

PARTE ARDUINO

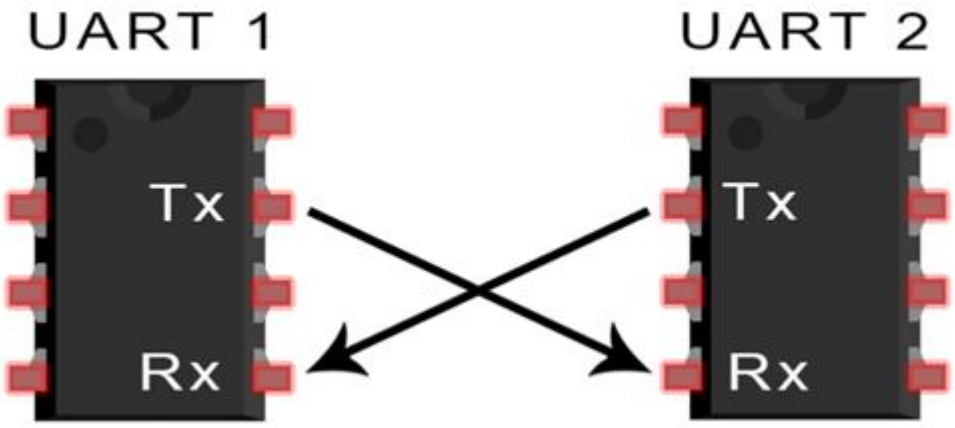
A continuación, se va a realizar una detallada explicación sobre el montaje y desarrollo, de la lectura y envío de datos. Así como la explicación de la comunicación serial entre Arduino y ESP8266.

En primer lugar, vamos a exponer los componentes que hemos empleado para el desarrollo de esta parte:

- NodeMCU ESP8266
- Placa de Arduino UNO
- Sensor de pulso
- Sensor de temperatura LM35
- Sensor de temperatura y humedad DHT11
- Cable USB Arduino UNO
- Cable USB de 1M

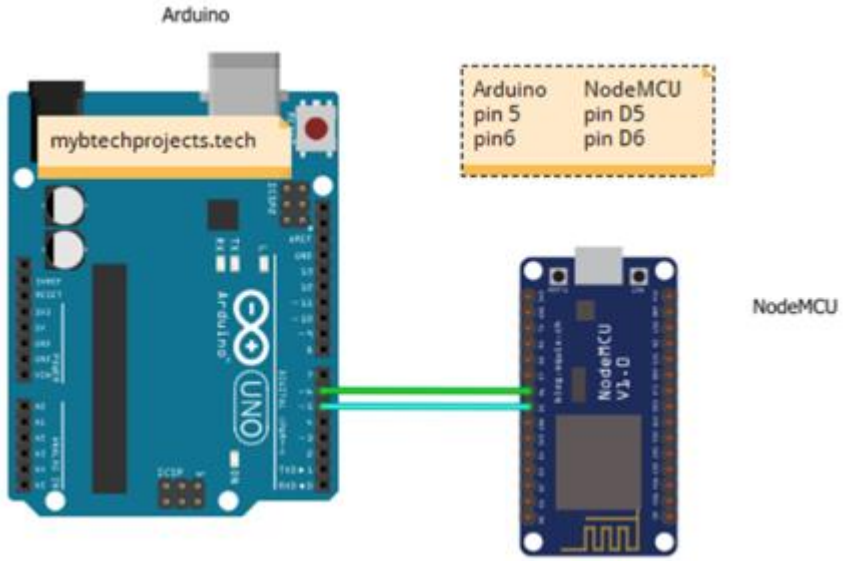
En primer lugar, para poder realizar la lectura de dos o más sensores, debíamos utilizar una placa de Arduino y realizar una comunicación serial entre la placa de Arduino y nuestro NodeMCU. Con ello la placa de Arduino solo se encarga de leer los sensores y el módulo wifi(en nuestro caso ESP8266), se encargaría de enviar los datos al servidor.

En cuanto a la comunicación serial, la realizamos de la siguiente manera. En primer lugar debemos decir, que todas las placas de Arduino poseen un puerto serie(UART o USART). La comunicación entre nuestra placa de Arduino y el módulo wifi se realiza mediante los pines digitales 0(RX) y 1(TX). Para establecer dicha comunicación en serie, los dispositivos, deben conectarse como se muestra en la siguiente figura:



Los datos enviados desde el dispositivo 1, en nuestro caso la placa de Arduino, deben recibirse en el dispositivo 2, en nuestro caso el módulo wifi. ¿Cómo sabe Arduino cuando tiene que enviar los datos?. ESP8266 solicita los datos de Arduino enviando un carácter. Una vez que el Arduino detecta que hay datos entrantes, envía los datos en serie como respuesta.

A continuación, se muestra una imagen de la conexión en serie realizada entre Arduino y ESP8266, así como fragmentos de código de ambos programas, donde se irá explicando la funcionalidad de cada uno.



Código parte ARDUINO

En la placa de Arduino, consideramos el pin 5 como RX y el pin 6 como TX. Para emplear los pines GPIO para la comunicación en serie, se puede utilizar la librería SoftwareSerial. En este fragmento de código, hemos creado un puerto serie llamado ‘s’ con el pin 5 como RX y con el pin 6 cómo TX.

```
2   #include <SoftwareSerial.h>
3   SoftwareSerial s(5,6);
```

Código parte NODEMCU

En el módulo wifi, configuraremos el pin 5 como TX y el pin 6 como RX. Ahora los datos enviados desde Arduino serán leídos por el módulo wifi y viceversa.

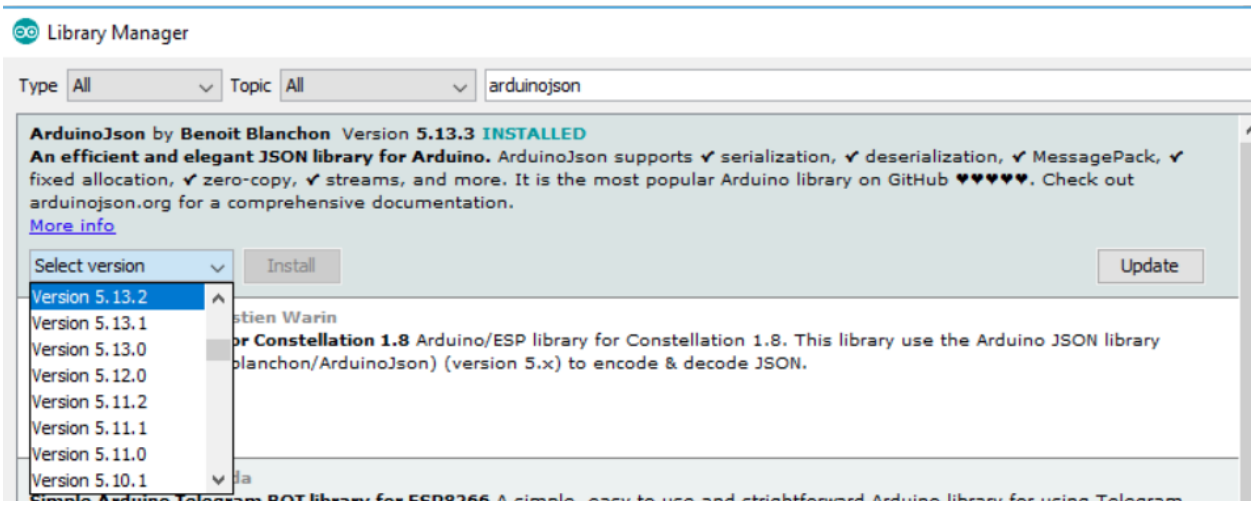
```
2   #include <SoftwareSerial.h>
3   SoftwareSerial s(D6,D5);
```

Como hemos mencionado antes, el módulo wifi envia un caracter de reconocimiento, y este comienza a leer los datos procedentes de Arduino. El fragmento de código que se encarga de estos es el siguiente:

```
38    s.write("s");
39    if (s.available()>0)
40    {
41      data=s.read();
42    }
43    else{
44      Serial.println("El caracter de reconocimiento no se ha enviado correctamente");
45    }
46
```

Una vez que conocemos la comunicación entre NodeMCU y Arduino, ahora solo nos queda una pregunta. ¿Cómo se envían los datos entre Arduino y NodeMCU?. Lo que acabamos de ver es cómo se comunica la placa de arduino con el módulo wifi, pero, ¿cuál es la forma en a que se envían los datos?. Para enviar varios datos en serie, nosotros hemos utilizado. JSON es un formato ligero de intercambio de datos. JSON se basa en pares clave-valor. La clave siempre es una cadena, donde el valor puede ser un entero, una cadena o una matriz

Para poder trabajar, debemos descargarnos una biblioteca nueva, en este caso ArduinoJson.h. En cuanto a la versión a descargar, la versión deseable es la 5.13.2, puesto que a partir de ahí se emplean otro tipo de funciones que no nos interesan:



El fragmento de código de creación del buffer es el siguiente, dicho buffer como es lógico, está implementado tanto en la parte de Arduino, como en la parte del módulo wifi.

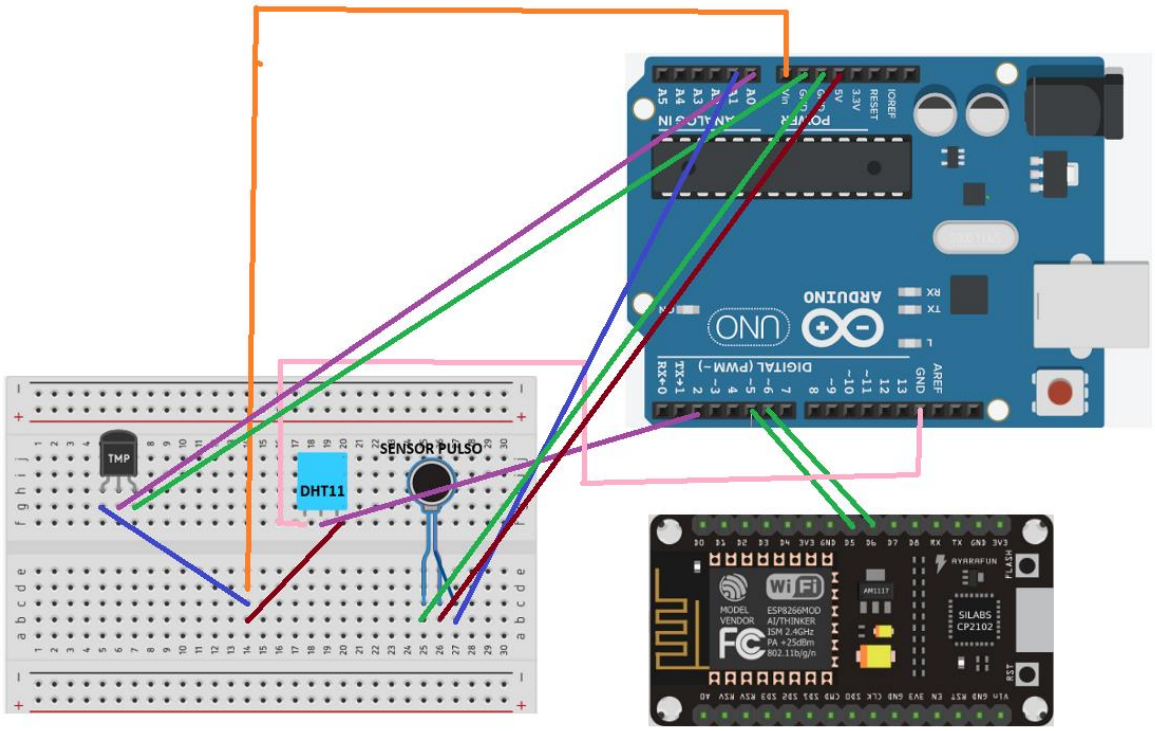
```
15   StaticJsonBuffer<1000> jsonBuffer;
16   JsonObject& root = jsonBuffer.createObject();
```

JsonBuffer, realiza la parte de gestión de memoria, y puede contener dos tipos, DynamicJsonBuffer o StaticJsonBuffer. DynamicJsonBuffer actualiza la memoria automáticamente según el requisito, mientras que StaticJsonBuffer asigna memoria fija y no cambiará según el requisito. Aquí hemos creado un StaticJsonBuffer con un tamaño de 1000.

```
39   if (root == JsonObject::invalid())
40   {
41     return;
42   }
```

Si los datos JSON analizados desde el puerto serie no están en el formato válido, deben ignorarse y la función de bucle se repite nuevamente.

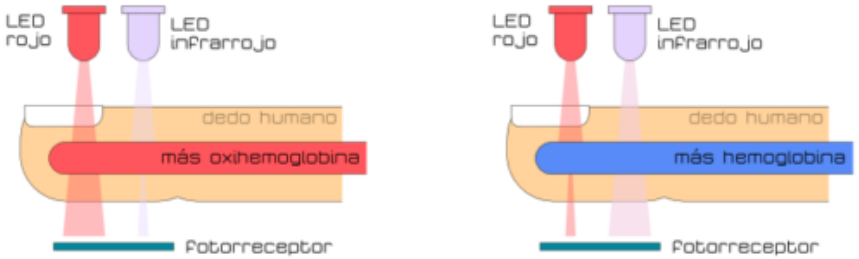
Una vez que conocemos la forma en la que se intercambian datos entre Arduino y nuestro módulo wifi, ahora procedemos a la lectura de datos. En primer lugar mostraremos como tenemos conectado nuestro circuito y a continuación, mostraremos la parte de código necesaria para leer y enviar al ESP8266 dichos datos.



En cuanto al código para la lectura de los sensores, es el siguiente:

```
20  int lecturaTemp = analogRead(sensorTemp);
21  int lecturaPulso = analogRead(sensorPulso);
22
23  //--------codigo para leer temperatura y humedad
24  byte temperature = 0;
25  byte humidity = 0;
26  int err = SimpleDHTErrSuccess;
27  if((err = dht11.read(pinDHT11, &temperature, &humidity, NULL)) != SimpleDHTErrSuccess){
28    Serial.print("Error al leer el dht");
29    Serial.println(err);
30    delay(1000);
31    return;
32  }
33
34  //-----codigo para leer la temperatura
35  float temp = lecturaTemp * 0.48828125;
```

En cuanto al sensor de pulso, la medición se realiza de la siguiente manera. Se realiza una medición óptica con un diodo de emisión de luz(LED). Un LED tiene una longitud de onda de 660 nm (rojo) y el otro tiene una longitud de onda de 940 nm (infrarrojo).



En nuestro fragmento de código, nosotros no medimos los latidos por minuto, sino la cantidad de hemoglobina. Si quisiéramos medir los pulsos por minuto, deberíamos crear un contador que midiese la longitud de onda por segundo, si dicha longitud supera 660 nm, se considera un latido. Por ello deberíamos esperar 1 minuto para calcular cada medición, algo que no queremos contemplar de momento.

En cuanto al sensor de temperatura LM35 la conexión es bastante simple, en cuanto a la forma de procesar los datos, se realiza de la siguiente manera:

Temperatura = temp * 5000 /1024/10

¿Porque 5000/1024/10?En primer lugar 5000, porque el sensor está conectado a un voltaje de 5v, y 1024 por qué estamos leyendo bits y lo queremos convertir a una medida de temperatura real, es decir, en grados centígrados. Por ultimo, dividimos entre 10, porque si observamos el datasheet veremos que cada 10 mV es directamente proporcional a 1 Celcius.

La medida de nuestro sensor de temperatura, en ocasiones puede distar mucho de la realidad, es decir, nos proporciona valores que difieren mucho con la temperatura real. Esto puede ser porque el sensor venia defectuoso.

En cuanto al DHT11, no tenemos mucho que explicar, simplemente añadir que es recomendable poner un cierto delay para que al sensor le dé tiempo a medir la temperatura y humedad. Como nosotros no podemos ponerle dicho delay puesto que afectaría al envío de datos de otros sensores, decidimos incluir un fragmento de código donde contemplamos la posibilidad de que el sensor nos devuelva valores nulos.

```
27    int err = SimpleDHTErrSuccess;
28    if((err = dht11.read(pinDHT11, &temperature, &humidity, NULL)) != SimpleDHTErrSuccess){
29      Serial.print("Error al leer el dht");
30      Serial.println(err);
31      delay(1000);
32      return;
33    }
```

Una vez que los datos han sido leídos por la placa de arduino, son enviados al módulo wifi mediante el siguiente fragmento de código:

```
38    root["temp"] = temp;
39    root["pulso"] = lecturaPulso;
40    root["humedad"] = ((int)humidity);
41    root["temp_dht"] = ((int)temperature);
42
43    if(s.available()>0){
44      root.printTo(s);
45    }
```

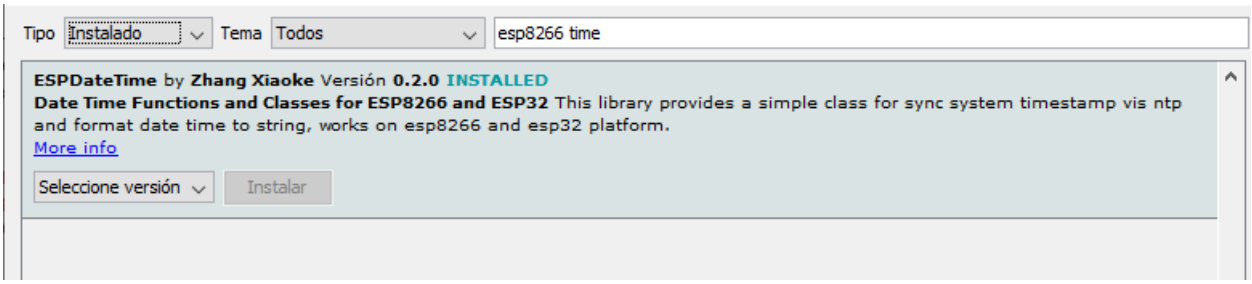
Aquí lo que hacemos es almacenar en el json los valores que queremos enviar al módulo wifi. Una vez que los datos han sido almacenamos los enviamos con la solicitud root.printTo(s); donde s es el SoftwareSerial.

Una vez que los datos han sido enviados desde la placa de arduino, ¿como recoge la placa nodemcu dicho json y guarda los valores de cada sensor?, mediante el siguiente fragmento de código. *"Ahora estamos en el código del módulo wifi".*

```
53    //Print the data in the serial monitor
54    Serial.println("JSON received and parsed");
55    root.prettyPrintTo(Serial);
56    int data1=root["temp"];
57    int data2=root["pulso"];
58    int data3=root["humedad"];
59    int data4=root["temp_dht"];
60
61    Serial.println("");
62    Serial.print("Temperatura ");
63    Serial.println(data1);
64    Serial.print("Pulso ");
65    Serial.println(data2);
66    Serial.print("Humedad ");
67    Serial.println(data3);
68    Serial.print("Temperatura del DHT ");
69    Serial.println(data4);
```

Como se puede observar, se recibe el json enviado por la placa arduino y se procesa cada dato por separado.

Por otro lado, otro dato que nos interesaba enviar era un timestamp de la fecha en la que se enviaban los datos desde la placa al servidor. Esto lo hemos realizado mediante la siguiente librería.



Con dicha librería seremos capaces de enviar al servidor la fecha que nos interesa, en nuestro caso día-hora-minuto-segundo. El siguiente fragmento de código se encarga de ello.

```
102  if (!DateTime.isTimeValid()) {
103      Serial.println("Failed to get time from server, retry.");
104      DateTime.begin();
105   }else {
106  DateTimeParts p = DateTime.getParts();
107
108  char dia = p.getMonthDay();
109  char horas = p.getHours();
110  char minutos = p.getMinutes();
111  char segundos = p.getSeconds();
112  time_t t = DateTime.now();
113  String fecha = DateFormatter::format("%d-%H-%M-%S", t);
```

Cabe decir que para que la fecha se lea de manera correcta, previamente nuestro NodeMCU debe estar conectado a una red wifi.

Finalmente una vez se han procesado todos los datos necesarios, debemos conectarnos con el servidor y enviar dichos datos. El siguiente fragmento de código se encarga de ello.

```
char server[] = "192.168.43.118";
WiFiClient client;

String valores = "https://192.168.43.118:8080/v1.0obi/Servlet_arduino_medicion1_in?" + String(data4) + "-" +
  String(data3) + "-" + String(data1) + "-" + String(data2) + "-" + fecha;

  Serial.print(valores);
  Serial.println("");

  if(client.connect(server, 8080)){
      Serial.println("- connected");
      Serial.println("Me conecto al servidor");
      client.println("POST "+ valores +" HTTP/1.1");
      client.print("Host: 192.168.43.118");
      client.print("User-Agent: Arduino/1.0");
      client.print("Connection: close");
      client.print("Content-Type: application/x-www-form-urlencoded");
      client.print("Content-Length: ");
      client.println(valores.length());
      client.println();
      client.print(valores);
      Serial.println("- done");
    }else{
      Serial.print("No se ha podido conectar con el sevidor");
    }
    Serial.println("Disconnecting from server...");
    client.stop();
    Serial.println("- bye!");
  }
```

En el anexo se muestra el código al completo para una mejor visualización y comprensión del mismo. Tanto el código de la parte de la placa de arduino como el código correspondiente al módulo wifi.

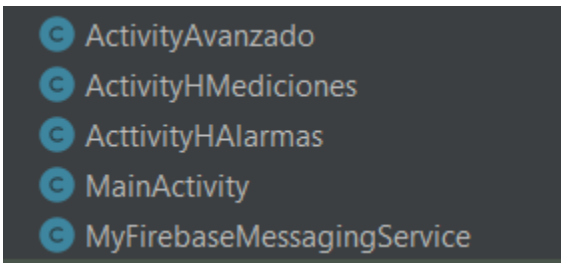
Finalmente se muestra un ejemplo de ejecucion del codigo para poder observar y corroborar su funcionamiento, así como comprobar que la conexión con el servidor se realiza correctamente. Debemos mencionar que para la ejecución de ambos códigos, primero debemos compilar el código correspondiente a la placa arduino y después el código correspondiente al ESP.

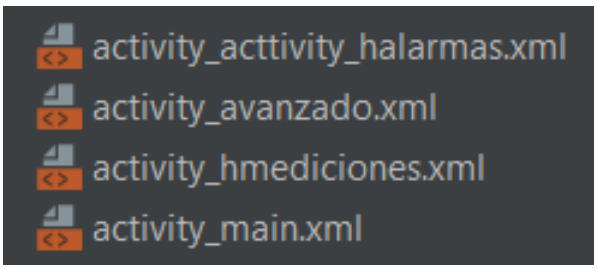
```
13:38:39.994 -> JSON received and parsed
13:38:39.994 -> {
13:38:39.994 ->   "temp": 49.80469,
13:38:39.994 ->   "pulso": 507,
13:38:39.994 ->   "humedad": 15,
13:38:39.994 ->   "temp_dht": 23
13:38:39.994 -> }
13:38:39.994 -> Temperatura 49
13:38:39.994 -> Pulso 507
13:38:39.994 -> Humedad 15
13:38:39.994 -> Temperatura del DHT 23
13:38:40.028 ->
13:38:40.028 -> ----------------------xxxxx----------------------
13:38:40.028 ->
13:38:40.028 -> 08 13:38:39
13:38:40.028 -> https://192.168.43.118:8080/v1.0obi/Servlet_arduino_medicion1_in?23-15-49-507-08-13-38-39
13:38:40.028 -> - connected
13:38:40.028 -> Me conecto al servidor
13:38:40.028 -> - done
13:38:40.028 -> Disconnecting from server...
13:38:40.127 -> - bye!
```

PARTE APP

La aplicación ha sido elaborada en Android Studio, y se compone de cuatro actividades distintas. La primera actividad, que es la principal y la que se lanza nada más iniciar la aplicación, contiene el código y la interfaz gráfica de lo que es la parte de consultar las últimas mediciones. Al pulsar el botón de CONSULTAR, la aplicación mostrará la temperatura y pulso del niño, y la temperatura y humedad de la habitación, también la fecha y hora en la que han sido tomadas estas mediciones.
A parte del botón anterior, hay tres botones más: AVANZADO, HISTORIAL MEDICIONES, HISTORIAL ALARMAS.
Al pulsar el primer botón se iniciará una nueva actividad, que mostrará una tabla con las últimas mediciones, las mediciones anteriores a estas, la medición media de las últimas treinta almacenadas en la base de datos, la tendencia de cada parámetro (a aumentar, disminuir o quedarse igual) y un resumen.
El segundo botón inicia una actividad nueva en la que se muestran las treinta últimas mediciones.
El tercer botón inicia también una actividad nueva, mostrando las treinta últimas alarmas guardadas en la base de datos.

En cuanto al código de la aplicación, tenemos una clase java y un xml por cada actividad. En los xml se configura la interfaz, mientras que en la clases java se crea la conexión con el servidor y se cambia el texto de la interfaz por los datos que se reciben del servidor.





Aparte de las clases de las actividades, tenemos la clase MyFirebaseMessagingService, que en teoría debería servirnos para conectarnos con el servidor, pero por los problemas con los certificados lo único que hace es conectarse con Firebase. Desde Firebase se pueden enviar notificaciones a la app, pero no nos es útil puesto que solo sirve para enviar notificaciones programadas, no para enviar las queremos nosotros. Esta clase se comunica con el servlet Servlet_app_enviarNotificacion, pasándole el token que serviría para identificar a nuestra aplicación y así poder enviarle las notificaciones.
Cada vez que se pulsa un botón se abre una conexión hacia el servidor mediante la clase HttpURLConnection y se pide los datos que se necesiten, luego se cierra.

```
HttpURLConnection con= null;
Scanner sc = null;
String cadena="";

try{
    URL url = new URL( spec: "http://192.168.43.118:8080/v1.0obi/Servlet_app_historiaAlarmas");
    con = (HttpURLConnection)url.openConnection();
    con.setRequestMethod("GET");
    con.setDoInput(true);
    con.connect();

    int responseCode = con.getResponseCode();

    if(responseCode == HttpsURLConnection.HTTP_OK){
        InputStream in = new BufferedInputStream(con.getInputStream());
        InputStreamReader ir = new InputStreamReader(in);
        BufferedReader br=new BufferedReader(ir);
        int i;
        while((i=br.read())!=-1){
            cadena+=((char)i);
        }
        br.close();
        in.close();
        con.disconnect();
        return cadena;
    }else{
        return ("ERROR");
    }
}catch(Exception e){
    return ("ERROR");
}
```

Estas conexiones las hacemos en una clase que hereda de AsyncTask y que tiene dos métodos: doInBackground, que es donde se produce la conexión y la toma de datos; y onPostExecute, que es el que se encarga de mostrar el texto por pantalla. Esta clase se ejecuta llamando: new nombreClase().execute(); en el método onCreate() de cada actividad.

PARTE BBDD

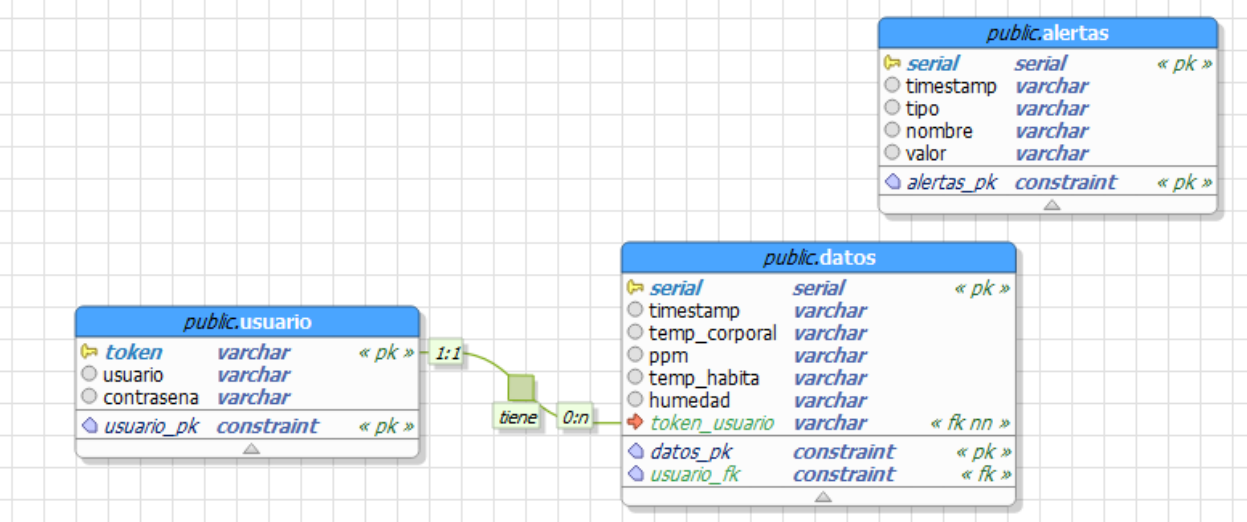
La idea inicial que teníamos para la realización de nuestra base de datos se basaba en el uso de postgreSQL. Nuestra aplicación no necesitaba muchas tablas para su correcto funcionamiento y barajamos diferentes ideas.

En primer lugar, decidimos que hubiera una tabla usuario, con su usuario y su contraseña para iniciar sesión en la aplicación (en el prototipo no es necesario). Una tabla para los datos recibidos de la pulsera, y otra para los datos recibidos del sensor. El problema en este diseño aparecía a la hora de introducir los datos desde el arduino, siendo necesarias dos inserciones (una para cada tabla).

Para simplificar este diseño decidimos unificar las tablas en un único conjunto de datos, relacionados con el usuario. Este usuario descubrimos que necesitaba un token por el cual se conecta a nuestra aplicación Android. Este token será clave del usuario y a su vez clave externa de los datos, relacionándolos así con su usuario.

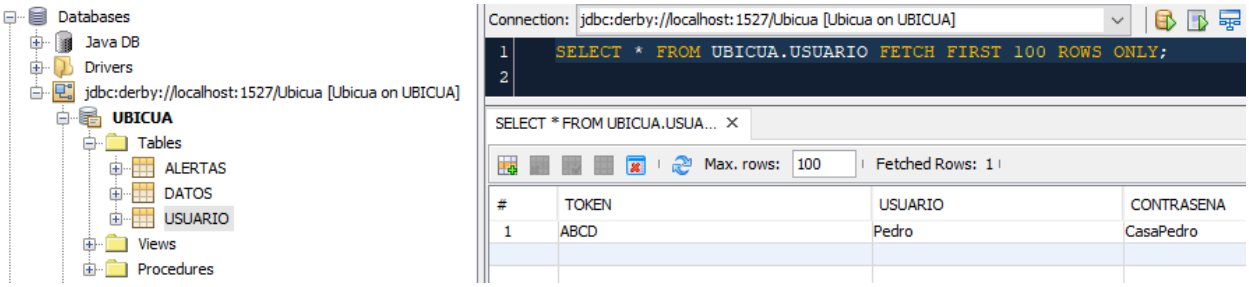
Nuestra aplicación nos da la posibilidad, entonces, de crear una serie de usuarios, cada uno con su token de aplicación, que tienen acceso a sus datos de forma independiente al resto de usuarios.

En la última actualización de nuestra base de datos añadimos una tabla de alertas, para almacenar todas las que pudieran surgir en la ejecución y recibirlas en el móvil.



A lo largo del desarrollo de nuestro proyecto nos encontramos con un problema. Pese a que la conexión Java-PostgreSQL funcionaba sin problema en una aplicación normal, en un servidor Java no conseguimos hallar la manera de introducir el driver JDBC de Postgres, que habilita la conexión y ejecución de queries desde Java. Tras largos intentos decidimos trasladar nuestra base de datos a Derby, dentro del propio NetBeans, que nos permitía la conexión con el servidor.

Traducimos las queries de creación de la base de datos de PostgreSQL a Derby, y usamos las mismas consultas, pues esta base de datos también está basada en SQL. Estas consultas están disponibles en un .txt anexo a este documento. Contienen, además, un usuario ejemplo con el que trabajamos en el prototipo, todos los datos están relacionados con el y sus dispositivos de medición.



La dirección de la base de datos se usa para enviar los datos desde el arduino, pudiéndose conectar a nuestro GlassFish server (explicado en la parte servidor web) haciendo uso de la dirección IP del equipo en el que se encuentra el servidor y la base de datos, seguida del puerto, dirección de la base de datos, y por último los datos a insertar en la tabla. La misma URL sirve para conectarse desde el Android y solicitar los datos para mostrarlos en la aplicación.

http://192.168.43.118:8080/v1.0obi/Servlet_arduino_medicion1_in?x-x-x-x-x-x-x-x

Por último a destacar, cada vez que se envía o se solicita un dato, ha de abrirse la conexión con la base de datos, y cerrarse al finalizar la acción. De esta manera evitamos que la base de datos se quede conectada permanentemente a un cliente (arduino por ejemplo) y no pueda conectarse desde la aplicación. Las conexiones han de ser intermitentes permitiendo el acceso alterno de quien necesita acceder a los datos. El arduino mandará datos continuamente, mientras que la aplicación esperará a que se soliciten para hacerse con la conexión.

PARTE SERVIDOR WEB

Para montar el servidor web, utilizamos GlassFish server integrado en Netbeans 11.2, con JDK 1.8 sobre java EE 7.

La estructura del proyecto en netbeans tiene esta estructura:

-una clase Inteligencia, encargada de disparar alarmas

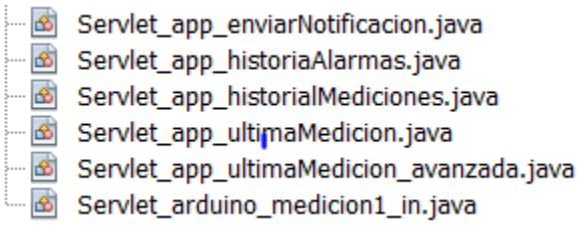
-una clase ModeloDatos, encargada del acceso a la base de datos, (insertar datos, recuperarlos y operar con ellos).

-una clase FCMNotificacion, que en teoría se encargaría de la conexión entre la aplicación y el servidor por medio de Firebase de google, y el envío de las notificaciones.

-6 servlets para gestionar la conexión con la base de datos y la aplicación con el servidor.

Clases servlets

Hay un servlet para cada operación sobre la base de datos, el servidor de arduino para insertar datos, y el resto de servlets app para tratar las peticiones de la aplicación y devolverle los datos requeridos.



La estructura del servlet es siempre la misma: se abre la conexión con la base de datos, se ejecuta la query determinada y se cierra la conexión al terminar.

En el caso de insertar datos a la base de datos no se devuelve nada, se insertan los datos y se comprueban las alarmas, con posibilidad de insertar nuevas alarmas.

En el caso de solicitar datos, se realiza las operaciones correspondientes a la función determinada en la clase modelodatos, y después de cerrar la conexión, se envían los datos obtenidos y se devuelven a la app.

El servlet de enviarNotificacion recogería el token que envía la aplicación al instalarse y se la mandaría a ModeloDatos.

Para conectarnos a un servlet es necesario conocer la url del propio servlet. El ejemplo siguiente es insertar datos desde arduino.

http://192.168.43.236:8080/prueba1obi/Servlet_arduino_medicion1_in?32-15-35-512-08-01-40-11

Para enviar datos al servidor, se obtienen los datos tras la ? del servlet, se realiza un tokenizer por guión, y recogemos los datos.

```
String temperaturaAmbiente = String.valueOf(params[0]);
String humedadAmbiente = String.valueOf(params[1]);
String temperaturaBebe = String.valueOf(params[2]);
String pulsoBebe = String.valueOf(params[3]);
String dia = String.valueOf(params[4]);
String hora = String.valueOf(params[5]);
String minuto = String.valueOf(params[6]);
String segundo = String.valueOf(params[7]);
```

La llamada de un servlet lleva implícita una respuesta del servidor. De esta forma podemos devolver un resultado en la misma llamada.

Se nos ocurre que para devolver las cosas podemos utilizar la misma estructura de guiones entre valores y tokens. Se hace así en la clase modeloDatos, las cadenas que se devuelven son formadas por valores entre guiones.

La clase inteligencia

Se le llama desde modelo datos.

```
public boolean gestionarAlarmaConDatosNuevos(float tempbebe, float pulsobebe, float temphabita, float humhabita, String timestamp)
{
```

Tras insertar un dato, debemos comprobar si esos datos insertados cumplen las condiciones de alarma. Si se cumplen esas condiciones, se inserta en la base de datos dicha alarma, y se lanzaría una notificación a la aplicación móvil. No se ha podido implementar las notificaciones push en la aplicación móvil por problemas con en el servidor glassfish.

```
boolean flagAlarmaIndividual = false;
flagAlarmaIndividual = gestionarAlarmaIndividual(tempbebe, pulsobebe, temphabita, humhabita, timestamp);

boolean flagAlarmaDerivada = false;
flagAlarmaDerivada = gestionarAlarmaDerivada(timestamp);
```

Hay dos tipos de alarmas que gestionamos, alarmas individuales y alarmas derivadas.

Las alarmas individuales gestionan si un dato determinado está fuera de los rangos aceptados.

```
if(alarma_valor_temperatura_bebe_low(tempbebe)){
    flag =true;
    ModeloDatos.insertarAlarma(timestamp, "Individual", "tempbebe_low", String.valueOf(tempbebe));
}
if(alarma_valor_temperatura_bebe_high(tempbebe)){
    flag =true;
    ModeloDatos.insertarAlarma(timestamp, "Individual", "tempbebe_high", String.valueOf(tempbebe));
}
```

Las alarmas derivadas, por su parte, tratan de verificar si un tipo de dato determinado tiene un ritmo de crecimiento continuado. Tras 5 valores de un mismo dato crecientes o decrecientes seguidos, se guardaría una alarma nueva y se inserta en la base de datos. Igual para los 4 datos distintos.

```
if(ModeloDatos.es_temperatura_bebe_DEcreciente()==true){
    alarma = true;
    ModeloDatos.insertarAlarma(timestamp, "Derivada", "tempbebe", "decreciente");
    //lanza alarma
}

if(ModeloDatos.es_temperatura_bebe_creciente()==true){
    alarma = true;
    ModeloDatos.insertarAlarma(timestamp, "Derivada", "tempbebe", "creciente");
    //lanza alarma
}
```

Para consultar estas alarmas, desde la aplicación pulsando un botón se llamará al servlet adecuado que devuelva el historial de alarmas.

No hemos sido capaces de implementar las notificaciones push por problemas con los certificados, aun así, se pueden consultar desde la app.

La clase ModeloDatos

La clase Modelo datos fue originalmente diseñada para solamente realizar querys solicitadas desde un servlet, y poder devolver datos a través de esta conexión. Esta idea se respeta, para mantener el patrón modelo vista controlador, pero en la clase modelodatos también se realiza la parte del controlador, operando con los valores y devolviendo cadenas complejas modificadas aquí mismo.

Las funciones a destacar de esta clase son las de abrir conexion, cerrar conexión, que han de ser llamadas antes y después de cada acceso a la bbdd, para no entrar en conflicto con otras peticiones de servlets.

```
    public static void abrirConexion() {
        String sURL = "jdbc:odbc:mvc";
        try {
//Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Class.forName("org.apache.derby.jdbc.ClientDriver");
//     con = DriverManager.getConnection(sURL,"","");
            con = DriverManager.getConnection("jdbc:derby://localhost:1527/Ubicua", "Ubicua", "UbicuaBBDD");
            System.out.println("Se ha conectado");
        } catch (Exception e) {
            System.out.println("No se ha conectado"+e.toString());
        }
    }
```

```
public static void cerrarConexion() {
    try {
        con.close();
    } catch (Exception e) {
    }
}
```

También es muy usada insertar datos, que es una muy simple.

```
public static void insertaDatos(String v1, String v2, String v3, String v4, String v5){
    try {
        //Timestamp timestamp = new Timestamp(System.currentTimeMillis());
        set = con.createStatement();
        set.executeUpdate("INSERT INTO datos(timestamp, temp_corporal, ppm, temp_habita, humedad, token_usuario)"
                + " VALUES ('"+v5+"', '"+v1+"', '"+v2+"', '"+v3+"', '"+v4+"', 'ABCD')");
        //rs.close();

    } catch (Exception e) {
        System.out.println("No inserta en la tabla insertardatos: "+e.toString());
    }
    //set.close();
    ia.gestionarAlarmaConDatosNuevos(Float.valueOf(v1), Float.valueOf(v2), Float.valueOf(v3), Float.valueOf(v4), v5);
    ia.gestionarAlarmaDerivada(v5);
}
```

Como se puede ver, se ejecuta la query de inserción y se llama a la clase de inteligencia, que a su vez contiene funciones que insertan, de manera similar a los datos del sensor, datos de alarma. Esto activaría las notificaciones push, pero hemos tenido problemas de certificados.

Las otra más importantes son la de obtener datos avanzados. Esta función devuelve la última medición registrada en la bbdd, la penúltima, la media de entre las últimas 30 mediciones, **y lo más importante, valores complejos: la tendencia de los valores (creciente, decreciente o equilibradas), y una predicción de futuro: si los valores aumentarán, disminuirán o no cambiarán.**

```
String mediaDatos = getMedia30UltimasLecturas();//SIN TIMESTAMP//tal cual
String cadenaFutura = ModeloDatos.estimarDatosFuturos();//SIN TIMESTAMP//comparas actual con anterior y ante
String cadenaResumen = ModeloDatos.resumirDatosPasado();//SIN TIMESTAMP//contiene: medicion30 - medicion1, y



cadenaCompleja = tempbebe+"-"+pulsobebe+"-"+tempcasa+"-"+humcasa+"-"+timestamp+"-";//el actual
cadenaCompleja =cadenaCompleja+ tempbebe_anterior+"-"+pulsobebe_anterior+"-"+tempcasa_anterior+"-"+humcasa_a

cadenaCompleja = cadenaCompleja+mediaDatos+"-";//media de los ultimos 30 datos medidos
cadenaCompleja = cadenaCompleja+cadenaFutura+"-";/*tempbebe_futura+"-"+pulsobebe_futura+"-"+tempcasa_futura+
cadenaCompleja = cadenaCompleja+cadenaResumen;
//String cadena = "hola prueba numero 1";
return cadenaCompleja;
```

Todos estas querys tendrán reflejo en la aplicación, que mostrará los resultados a través de la respuesta del servlet.

EJEMPLO DE FUNCIONAMIENTO CON CAPTURAS

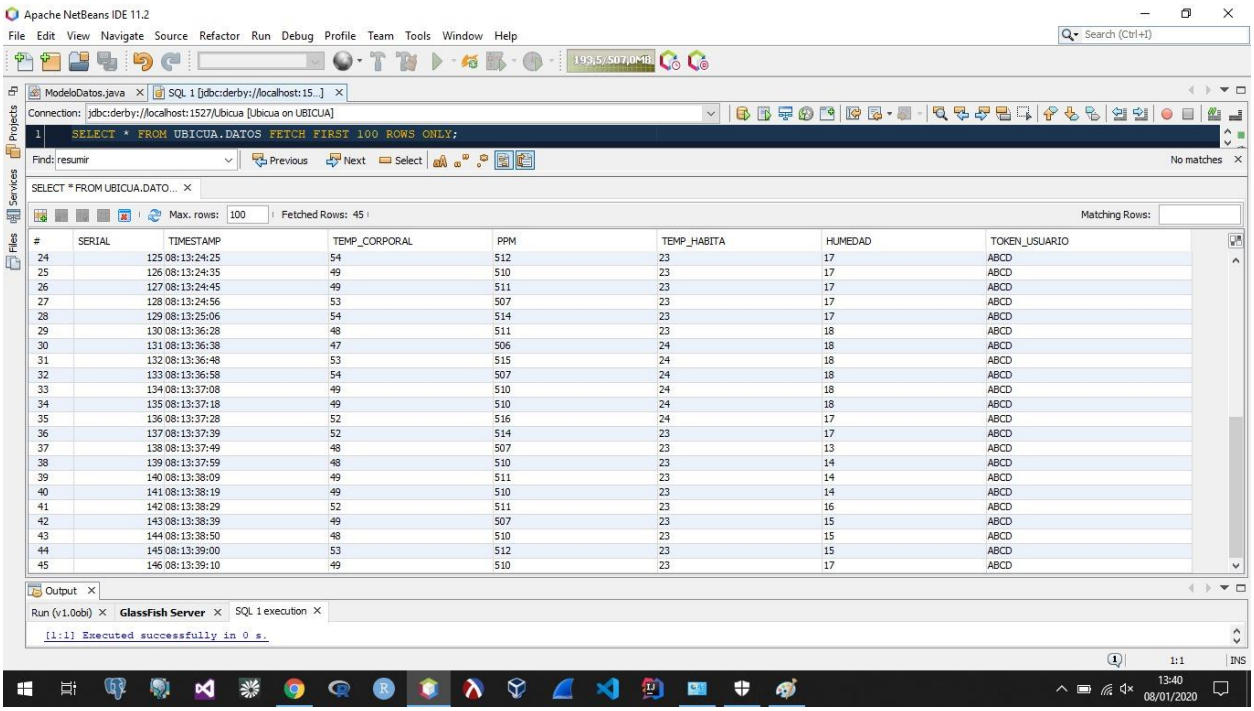
En este ejemplo vamos a escribir e ilustrar el funcionamiento de nuestra aplicación de cuidado infantil.

Por un lado, se pone en marcha arduino, y envía una serie de datos al servidor, con una frecuencia que nosotros determinamos, en este caso, cada 10 segundos sale una tanda de datos.

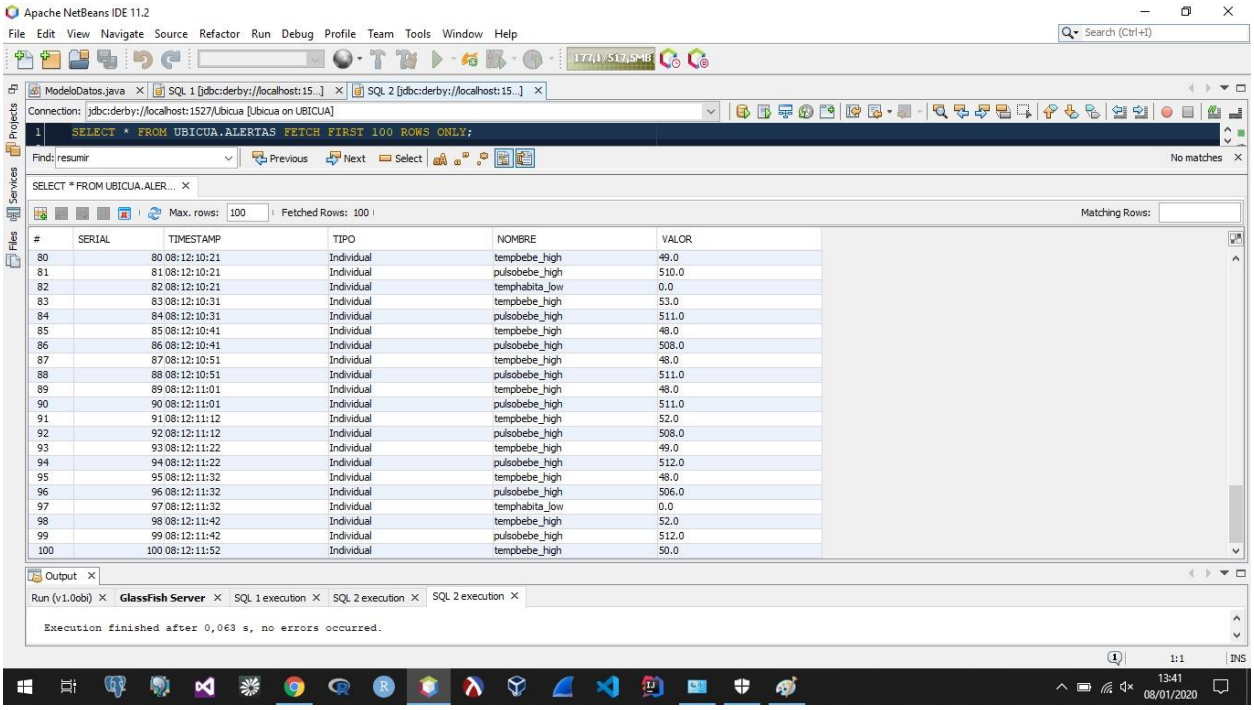
```
13:36:28.143 -> JSON received and parsed
13:36:28.177 -> {
13:36:28.177 ->   "temp": 48.82813,
13:36:28.177 ->   "pulso": 511,
13:36:28.177 ->   "humedad": 18,
13:36:28.177 ->   "temp_dht": 23
13:36:28.177 -> }
13:36:28.177 -> Temperatura 48
13:36:28.177 -> Pulso 511
13:36:28.177 -> Humedad 18
13:36:28.177 -> Temperatura del DHT 23
13:36:28.177 ->
13:36:28.177 -> ----------------------xxxxx---------------------
13:36:28.177 ->
13:36:28.177 -> 08 13:36:28
13:36:28.177 -> https://192.168.43.118:8080/v1.0obi/Servlet_arduino_medicionl_in?23-18-48-511-08-13-36-28
13:36:28.177 -> - connected
13:36:28.177 -> Me conecto al servidor
13:36:28.177 -> - done
13:36:28.177 -> Disconnecting from server...
13:36:28.313 -> - bye!


13:38:39.994 -> JSON received and parsed
13:38:39.994 -> {
13:38:39.994 ->   "temp": 49.80469,
13:38:39.994 ->   "pulso": 507,
13:38:39.994 ->   "humedad": 15,
13:38:39.994 ->   "temp_dht": 23
13:38:39.994 -> }
13:38:39.994 -> Temperatura 49
13:38:39.994 -> Pulso 507
13:38:39.994 -> Humedad 15
13:38:39.994 -> Temperatura del DHT 23
13:38:40.028 ->
13:38:40.028 -> ----------------------xxxxx---------------------
13:38:40.028 ->
13:38:40.028 -> 08 13:38:39
13:38:40.028 -> https://192.168.43.118:8080/v1.0obi/Servlet_arduino_medicionl_in?23-15-49-507-08-13-38-39
13:38:40.028 -> - connected
13:38:40.028 -> Me conecto al servidor
13:38:40.028 -> - done
13:38:40.028 -> Disconnecting from server...
13:38:40.127 -> - bye!
```

Los datos entran mediante el servlet a la base de datos, se ejecuta el insertar datos nuevos

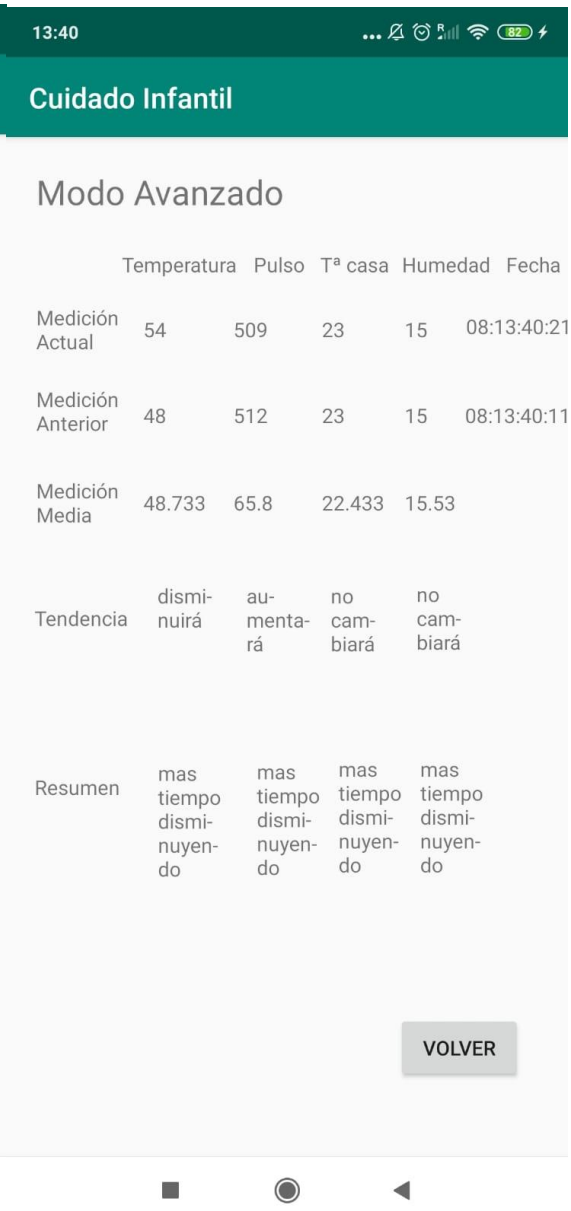


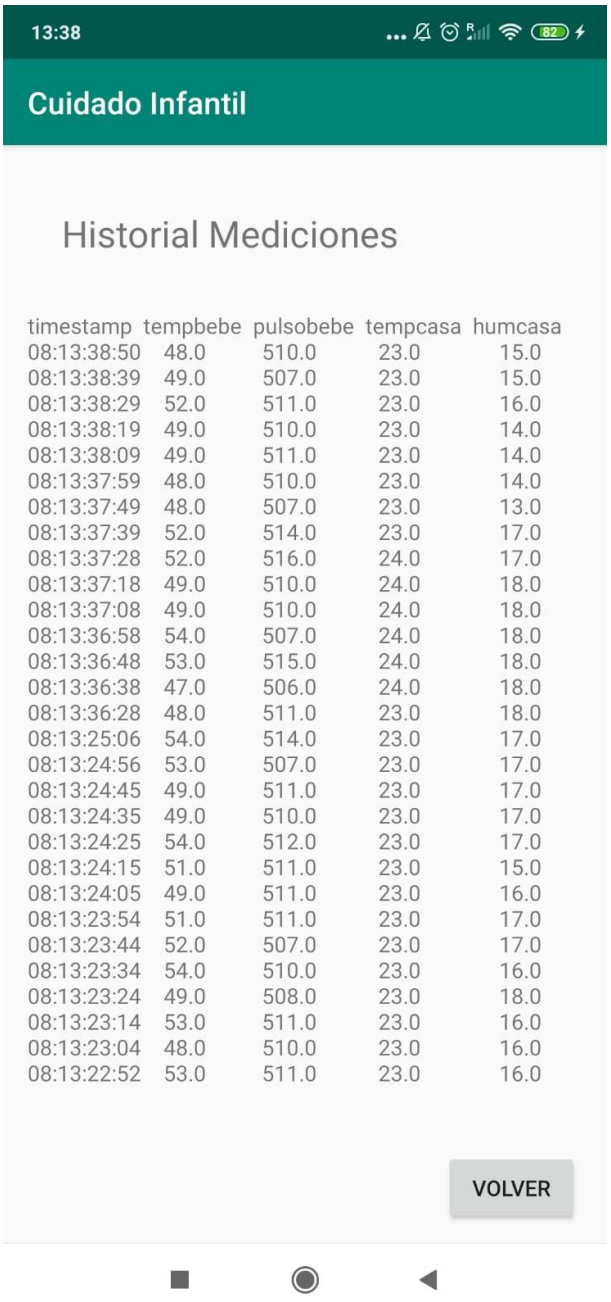
Y a su vez se ejecuta el insertar alarmas si procede (alarmas/alertas, de ambas formas vale).



Y finalmente, en la aplicación se visualizan los datos a tiempo real (hay que pulsar cada botón para recargar los datos que se quieran ver.


13:38
Cuidado Infantil
Últimos parámetros
08:13:38:39
Pulsera
Temperatura
49
ºC
Pulso
507
lpm
Dispositivo
Temperatura
23
ºC
Humedad
15
%
AVANZADO
CONSULTAR
HISTORIAL MEDICIONES
HISTORIAL ALARMAS


13:40
Cuidado Infantil
Modo Avanzado
Temperatura
Pulso
Tª casa
Humedad
Fecha
Medición Actual
54
509
23
15
08:13:40:21
Medición Anterior
48
512
23
15
08:13:40:11
Medición Media
48.733
65.8
22.433
15.53
Tendencia
dismi-nuirá
au-menta-rá
no cam-biará
no cam-biará
Resumen
mas tiempo dismi-nuyen-do
mas tiempo dismi-nuyen-do
mas tiempo dismi-nuyen-do
mas tiempo dismi-nuyen-do
VOLVER



ANEXO

Codigo Arduino completo

```
//Arduino code
#include <SoftwareSerial.h>
#include <ArduinoJson.h>
#include <SimpleDHT.h>
```

```
int pinDHT11 = 2;
SimpleDHT11 dht11;

SoftwareSerial s(5,6);
int sensorTemp = A0;
int sensorPulso = A1;
//DHT dht(DHTPIN,DHTTYPE);

void setup() {
s.begin(115200);
pinMode(sensorTemp, INPUT);
pinMode(sensorPulso, INPUT);
}

//Creamos un json en el que meter nuestras mediciones
StaticJsonBuffer<1000> jsonBuffer;
JsonObject& root = jsonBuffer.createObject();

void loop() {

  int lecturaTemp = analogRead(sensorTemp);
  int lecturaPulso = analogRead(sensorPulso);

  //--------codigo para leer temperatura y humedad
  byte temperature = 0;
  byte humidity = 0;
  int err = SimpleDHTErrSuccess;
  if((err = dht11.read(pinDHT11, &temperature, &humidity, NULL)) != SimpleDHTErrSuccess){
    Serial.print("Error al leer el dht");
    Serial.println(err);
    delay(1000);
    return;
  }

  //-----codigo para leer la temperatura
  float temp = lecturaTemp * 0.48828125;

  root["temp"] = temp;
  root["pulso"] = lecturaPulso;
  root["humedad"] = ((int)humidity);
  root["temp_dht"] = ((int)temperature);

  if(s.available()>0){
    root.printTo(s);
```

```
  }

}
```

Codigo NodeMCU completo

```
#include <SoftwareSerial.h>
#include <ArduinoJson.h>
#include <ESP8266WiFi.h>
#include <ESPDateTime.h>

#define WIFI_SSID "Wifidavid"
#define WIFI_PASS "david123"

char server[] = "192.168.43.118";

WiFiClient client;

SoftwareSerial s(D6,D5);
int data;
void setup() {
  s.begin(115200);
  Serial.begin(115200);
  setupWifi();
  setupDate();
  while (!Serial) continue;
}

void setupWifi(){
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASS);
  Serial.print("WiFi Connecting");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println();
}

void setupDate(){
  DateTime.setTimeZone(1);
  DateTime.begin();
  if (!DateTime.isTimeValid()) {
    Serial.println("Failed to get time from server.");
```

```
  }
}


void loop() {
  StaticJsonBuffer<1000> jsonBuffer;
  JsonObject& root = jsonBuffer.parseObject(s);
  s.write("s");

  if (root == JsonObject::invalid())
  {
    return;
  }
  if (s.available()>0)
  {
    data=s.read();
  }
  else{
    Serial.println("No va");
  }

  //Print the data in the serial monitor
  Serial.println("JSON received and parsed");
  root.prettyPrintTo(Serial);
  int data1=root["temp"];
  int data2=root["pulso"];
  int data3=root["humedad"];
  int data4=root["temp_dht"];

  Serial.println("");
  Serial.print("Temperatura ");
  Serial.println(data1);
  Serial.print("Pulso ");
  Serial.println(data2);
  Serial.print("Humedad ");
  Serial.println(data3);
  Serial.print("Temperatura del DHT ");
  Serial.println(data4);

  Serial.println("");
  Serial.println("---------------------xxxxx--------------------");
 Serial.println("");

    if (!DateTime.isTimeValid()) {
```

```
    Serial.println("Failed to get time from server, retry.");
    DateTime.begin();
   }else {
  DateTimeParts p = DateTime.getParts();

  char dia = p.getMonthDay();
  char horas = p.getHours();
  char minutos = p.getMinutes();
  char segundos = p.getSeconds();

  //String fecha = DateTime.toString().c_str();
  time_t t = DateTime.now();
  String fecha = DateFormatter::format("%d-%H-%M-%S", t);


  //String fecha = String(horas) + String(minutos) + String(segundos);
 Serial.printf("%02d %02d:%02d:%02d\n",dia, horas,minutos,segundos);

 String valores = "https://192.168.43.118:8080/v1.0obi/Servlet_arduino_medicion1_in?" +
String(data4) + "-" + String(data3) + "-" + String(data1) + "-" + String(data2) + "-" + fecha;
 Serial.print(valores);
 Serial.println("");

 if(client.connect(server, 8080)){
    Serial.println("- connected");
    Serial.println("Me conecto al servidor");
    client.println("POST "+ valores +" HTTP/1.1");
    client.print("Host: 192.168.43.118");
    client.print("User-Agent: Arduino/1.0");
    client.print("Connection: close");
    client.print("Content-Type: application/x-www-form-urlencoded");
    client.print("Content-Length: ");
    client.println(valores.length());
    client.println();
    client.print(valores);
    Serial.println("- done");
  }else{
    Serial.print("No se ha podido conectar con el sevidor");
  }
  Serial.println("Disconnecting from server...");
  client.stop();
  Serial.println("- bye!");
 }
  delay(10000);
```

}

Queries

Creo la base de datos con nombre Ubicua, usuario Ubicua, y contraseña UbicuaBBDD

```
CREATE TABLE usuario (
    token Varchar(100) PRIMARY KEY,
    usuario Varchar(20),
    contrasena Varchar(20)
);

CREATE TABLE datos (
    serial INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1),
    timestamp Varchar(20),
    temp_corporal Varchar(20),
    ppm Varchar(20),
    temp_habita Varchar(20),
    humedad Varchar(20),
    token_usuario Varchar(100),
    PRIMARY KEY (serial),
    FOREIGN KEY (token_usuario) references usuario(token) ON DELETE CASCADE
);

CREATE TABLE alertas (
    serial INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1),
    timestamp Varchar(20),
    tipo Varchar(20),
    nombre Varchar(20),
    valor Varchar(20),
    PRIMARY KEY (serial)
);

INSERT INTO usuario VALUES ('ABCD', 'Pedro', 'CasaPedro');
```

FCMNotification.java

```
import com.google.api.client.googleapis.auth.oauth2.GoogleCredential;
import java.io.FileInputStream;
import java.io.IOException;
```

```
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.Arrays;
import org.json.JSONException;
import org.json.JSONObject;


public class FCMNotification {
        public final static String AUTH_KEY_FCM = "AAAA0-0Lb-
Y:APA91bGgVrPgn6uYZ8a1BFN9gZp9qogC5zVOlrC0E8OO_mAV22QxFFaF-
saZk6eabKnupyKww9dYA49oJndSSt8GO2Pxy0nCn-d4zDnLcKJ2JwRlBfF-
zpMw1wiz3GWN2m9W6q7CRx9P";
        public final static String API_URL_FCM =
"https://fcm.googleapis.com/fcm/send";
        private static final String MESSAGING_SCOPE =
"https://www.googleapis.com/auth/firebase.messaging";
        private static final String[] SCOPES = { MESSAGING_SCOPE };
        private static final String PROJECT_ID = "cuidadoinfantil-188a3";
        private static final String BASE_URL =
"https://fcm.googleapis.com";
        private static final String FCM_SEND_ENDPOINT = "/v1/projects/" +
PROJECT_ID + "/messages:send";

    private static String getAccessToken() throws IOException {
        GoogleCredential googleCredential = GoogleCredential
        .fromStream(new
FileInputStream("C:\\Users\\Denis\\Desktop\\uni\\Ubicua\\20190103obi\\serv
ice-account.json"))
        .createScoped(Arrays.asList(SCOPES));
        googleCredential.refreshToken();
        return googleCredential.getAccessToken();
}

    public static void pushFCMNotification(String deviceToken, String
title, String body)
        throws IOException, JSONException {
    String authKey = AUTH_KEY_FCM;   // You FCM AUTH key
    String FMCurl = API_URL_FCM;
    //URL url = new URL(BASE_URL + FCM_SEND_ENDPOINT);
```

```
    //HttpURLConnection conn = (HttpURLConnection) url.openConnection();
    //conn.setRequestProperty("Authorization",      "Bearer      "      +
getAccessToken());
    //conn.setRequestProperty("Content-Type", "application/json; UTF-8");

    URL url = new URL(FMCurl);
    HttpURLConnection conn = (HttpURLConnection) url.openConnection();

    conn.setUseCaches(false);
    conn.setDoInput(true);
    conn.setDoOutput(true);
    conn.setRequestMethod("POST");
    conn.setRequestProperty("Authorization","key="+authKey);
    conn.setRequestProperty("Content-Type","application/json");

    JSONObject json = new JSONObject();
    json.put("to",deviceToken.trim());
    JSONObject info = new JSONObject();
    info.put("title", title);   // Notification title
    info.put("body", body); // Notification body
    json.put("notification", info);

   OutputStreamWriter wr = new OutputStreamWriter(conn.getOutputStream());
    wr.write(json.toString());
    wr.flush();
    conn.getInputStream();
  }
}
```

Inteligencia.java

```
public class Inteligencia {
    float tempMinBebe;
    float tempMaxBebe;
    float pulsoMinBebe;
    float pulsoMaxBebe;
    float tempMinCasa;
    float tempMaxCasa;
    float humMinCasa;
```

```
    float humMaxCasa;

    public Inteligencia(){
        //temperaturas      http://www.disfrutatuembarazo.com/temperatura-
descanso
        //humedad        https://www.bebepolis.es/blog/cual-es-la-humedad-
perfecta-para-la-habitacion-de-tu-bebe/
        //pulso
https://medlineplus.gov/spanish/ency/article/003399.htm
        /*
        Resultados normales

        Para la frecuencia cardíaca en reposo:

        Recién nacidos de 0 a 1 mes de edad: 70 a 190 latidos por minuto
        Bebés de 1 a 11 meses de edad: 80 a 160 latidos por minuto
        Niños de 1 a 2 años de edad: 80 a 130 latidos por minuto
        Niños de 3 a 4 años de edad: 80 a 120 latidos por minuto
        Niños de 5 a 6 años de edad: 75 a 115 latidos por minuto
        Niños de 7 a 9 años de edad: 70 a 110 latidos por minuto
        Niños de 10 años o más y adultos (incluso ancianos): 60 a 100 latidos
por minuto
        Atletas bien entrenados: de 40 a 60 latidos por minuto

        */

        tempMinBebe = 36.0f;
        tempMaxBebe = 37.5f;
        pulsoMinBebe = 0.80f;
        pulsoMaxBebe = 0.50f;
        tempMinCasa = 18.0f;
        tempMaxCasa = 22.0f;
        humMinCasa = 0.30f;
        humMaxCasa = 0.50f;
    }

    //DECLARACIONES
    //individuales
    public boolean alarma_valor_temperatura_bebe_low;
    public boolean alarma_valor_temperatura_bebe_high;
```

```
    public boolean alarma_valor_pulso_bebe_low;
    public boolean alarma_valor_pulso_bebe_high;
    public boolean alarma_valor_temperatura_habitacion_low;
    public boolean alarma_valor_temperatura_habitacion_high;
    public boolean alarma_valor_humedad_habitacion_low;
    public boolean alarma_valor_humedad_habitacion_high;
    //general
    public boolean gestionarAlarmaConDatosNuevos;
    //IMPLEMENTACIONES
    //individuales
    public boolean alarma_valor_temperatura_bebe_low(float v){
        if(v < tempMinBebe){
            //llamar a excepcion
            return true;
        }
        else{
            return false;
        }
    }
    public boolean alarma_valor_temperatura_bebe_high(float v){
        if(v > tempMaxBebe){
            //llamar a excepcion
            return true;
        }
        else{
            return false;
        }
    }
    public boolean alarma_valor_pulso_bebe_low(float v){
        if(v < pulsoMinBebe){
            //llamar a excepcion
            return true;
        }
        else{
            return false;
        }
    }
    public boolean alarma_valor_pulso_bebe_high(float v){
        if(v > pulsoMaxBebe){
            //llamar a excepcion
```

```
            return true;
        }
        else{
            return false;
        }
    }
    public boolean alarma_valor_temperatura_habitacion_low(float v){
        if(v < tempMinCasa){
            //llamar a excepcion
            return true;
        }
        else{
            return false;
        }
    }
    public boolean alarma_valor_temperatura_habitacion_high(float v){
        if(v > tempMaxCasa){
            //llamar a excepcion
            return true;
        }
        else{
            return false;
        }
    }
    public boolean alarma_valor_humedad_habitacion_low(float v){
        if(v < humMinCasa){
            //llamar a excepcion
            return true;
        }
        else{
            return false;
        }
    }
    public boolean alarma_valor_humedad_habitacion_high(float v){
        if(v > humMaxCasa){
            //llamar a excepcion
            return true;
        }
        else{
            return false;
```

```
        }
    }




    public boolean gestionarAlarmaConDatosNuevos(float tempbebe, float
pulsobebe, float temphabita, float humhabita, String timestamp)
    {
        boolean flagAlarmaIndividual = false;
        flagAlarmaIndividual     =     gestionarAlarmaIndividual(tempbebe,
pulsobebe, temphabita, humhabita, timestamp);

        boolean flagAlarmaDerivada = false;
        flagAlarmaDerivada = gestionarAlarmaDerivada(timestamp);
        //combinar temperatura con humedad, a ver si hay algun margen
distinto


       /* boolean flagAlarmaConjunta = false;
        flagAlarmaConjunta = gestionarAlarmaConjunta();
       */

      /*  boolean flagAvisoHistorial = false;
        if(contador==1)
            flagAvisoHistorial = gestionarAvisoHistorial();
        else{
            //nada
        }*/

        //mirar los ultimos x valores, ver variaciones y pronosticar cosas.

        return flagAlarmaIndividual;
    }

    /*public boolean gestionarAvisoHistorial(){
        boolean aviso = false;
        String cadenaSalida ="";

        cadenaSalida = ModeloDatos.getMedia30UltimasLecturas();
```

```
        //alarma con cadenaSalida

        if(!cadenaSalida.equals(""))
            aviso=true;

        return aviso;
    }*/

    public boolean gestionarAlarmaDerivada(String timestamp){
        //otro tipo de alarma, si una medicion aunmenta o disminuye 5 tics
seguidos, salta una alarma.
        //es mas para que funcione que para otra cosa: si cambia de 21.0 a
21.5, no es un cambio muy notable,
        //pero salta alarma para ver que funciona
        boolean alarma = false;

        if(ModeloDatos.es_temperatura_bebe_DEcreciente()==true){
            alarma = true;
            ModeloDatos.insertarAlarma(timestamp, "Derivada", "tempbebe",
"decreciente");
            //lanza alarma
        }

        if(ModeloDatos.es_temperatura_bebe_creciente()==true){
            alarma = true;
            ModeloDatos.insertarAlarma(timestamp, "Derivada", "tempbebe",
"creciente");
            //lanza alarma
        }
        if(ModeloDatos.es_pulso_bebe_DEcreciente()==true){
            alarma = true;
            ModeloDatos.insertarAlarma(timestamp, "Derivada", "pulsobebe",
"decreciente");

            //lanza alarma
        }
        if(ModeloDatos.es_pulso_bebe_creciente()==true){
            alarma = true;
            ModeloDatos.insertarAlarma(timestamp, "Derivada", "pulsobebe",
"creciente");
```

```
            //lanza alarma
        }
        if(ModeloDatos.es_temperatura_casa_DEcreciente()==true){
            alarma = true;
            ModeloDatos.insertarAlarma(timestamp, "Derivada", "tempcasa",
"decreciente");
            //lanza alarma
        }
        if(ModeloDatos.es_temperatura_casa_creciente()==true){
            alarma = true;
            ModeloDatos.insertarAlarma(timestamp, "Derivada", "tempcasa",
"creciente");
            //lanza alarma
        }
        if(ModeloDatos.es_humedad_casa_DEcreciente()==true){
            alarma = true;
            ModeloDatos.insertarAlarma(timestamp, "Derivada", "humcasa",
"decreciente");
            //lanza alarma
        }
        if(ModeloDatos.es_humedad_casa_creciente()==true){
            alarma = true;
            ModeloDatos.insertarAlarma(timestamp, "Derivada", "humcasa",
"creciente");
            //lanza alarma
        }
        return alarma;
    }

    public boolean gestionarAlarmaIndividual(float tempbebe, float
pulsobebe, float temphabita, float humhabita, String timestamp){
        boolean flag = false;
        if(alarma_valor_temperatura_bebe_low(tempbebe)){
            flag =true;
            ModeloDatos.insertarAlarma(timestamp, "Individual",
"tempbebe_low", String.valueOf(tempbebe));
        }
        if(alarma_valor_temperatura_bebe_high(tempbebe)){
            flag =true;
```

```
            ModelosDatos.insertarAlarma(timestamp,              "Individual",
"tempbebe_high", String.valueOf(tempbebe));
        }
        if(alarma_valor_pulso_bebe_low(pulsobebe)){
            flag =true;
            ModelosDatos.insertarAlarma(timestamp,              "Individual",
"pulsobebe_low", String.valueOf(pulsobebe));
        }
        if(alarma_valor_pulso_bebe_high(pulsobebe)){
            flag =true;
            ModelosDatos.insertarAlarma(timestamp,              "Individual",
"pulsobebe_high", String.valueOf(pulsobebe));
        }
        if(alarma_valor_temperatura_habitacion_low(temphabita)){
            flag =true;
            ModelosDatos.insertarAlarma(timestamp,              "Individual",
"temphabita_low", String.valueOf(temphabita));
        }
        if(alarma_valor_temperatura_habitacion_high(temphabita)){
            flag =true;
            ModelosDatos.insertarAlarma(timestamp,              "Individual",
"temphabita_high", String.valueOf(temphabita));
        }
        if(alarma_valor_humedad_habitacion_low(humhabita)){
            flag =true;
            ModelosDatos.insertarAlarma(timestamp,              "Individual",
"humhabita_low", String.valueOf(humhabita));
        }
        if(alarma_valor_humedad_habitacion_high(humhabita)){
            flag =true;
        }
        return flag;
    }
}
```

ModeloDatos.java

```
import java.sql.*;
```

```
public class ModeloDatos {


    private static Connection con;
    private static Statement set;
    private static ResultSet rs;
    private static Inteligencia ia = new Inteligencia();


    public static void abrirConexion() {
        String sURL = "jdbc:odbc:mvc";
        try {
//Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Class.forName("org.apache.derby.jdbc.ClientDriver");
//    con = DriverManager.getConnection(sURL,"","");
            con                                                    =
DriverManager.getConnection("jdbc:derby://localhost:1527/Ubicua",
"Ubicua", "UbicuaBBDD");
            System.out.println("Se ha conectado");
        } catch (Exception e) {
            System.out.println("No se ha conectado"+e.toString());
        }
    }

    public static String prueba(){
        String cad="";
        try {
            //Timestamp             timestamp             =             new
Timestamp(System.currentTimeMillis());
            set = con.createStatement();
            rs=set.executeQuery("Select * from usuario");
            rs.next();

            cad = rs.getString("token"); //el nombre de la columna dentro
del select *, devuelve el primer valor
            cad = cad.trim();//usar trim para dividir y comparar

            rs.close();
            set.close();
        } catch (Exception e) {
```

```
            System.out.println("No inserta en la tabla usuario: "+e.toString());
        }

        return cad;
        //ia.gestionarAlarmaConDatosNuevos(Float.valueOf(v1), Float.valueOf(v2), Float.valueOf(v3), Float.valueOf(v4), v5);
       // ia.gestionarAlarmaDerivada(v5);
    }


    public static void insertaDatos(String v1, String v2, String v3, String v4, String v5){
        try {
            //Timestamp timestamp = new Timestamp(System.currentTimeMillis());
            set = con.createStatement();
            set.executeUpdate("INSERT INTO datos(timestamp, temp_corporal, ppm, temp_habita, humedad, token_usuario)"
                    + " VALUES ('"+v5+"', '"+v1+"', '"+v2+"', '"+v3+"', '"+v4+"', 'ABCD')");
            //rs.close();

        } catch (Exception e) {
            System.out.println("No inserta en la tabla insertardatos: "+e.toString());
        }
        //set.close();
        ia.gestionarAlarmaConDatosNuevos(Float.valueOf(v1), Float.valueOf(v2), Float.valueOf(v3), Float.valueOf(v4), v5);
        ia.gestionarAlarmaDerivada(v5);
    }



    public static void insertarAlarma(String timestamp, String tipo, String nombre, String valor){
        try {
            //Timestamp timestamp = new Timestamp(System.currentTimeMillis());
```

```
            set = con.createStatement();
            set.executeUpdate("INSERT INTO alertas(timestamp, tipo, nombre,
valor)"
                    + " VALUES ('"+timestamp+"', '"+tipo+"', '"+nombre+"',
'"+valor+"')");
            rs.close();
            set.close();
        } catch (Exception e) {
            System.out.println("No   inserta   en   la   tabla   UNTITLED
"+e.toString());
        }
    }

    public static String getUltimaCadena(){
        // String nombre = "hola";
        //boolean existe = false;

        String cadena;
        String cad;

        String tempbebe="-10";
        String pulsobebe="-10";
        String tempcasa="-10";
        String humcasa="-10";
        String timestamp="-10";

        try {

            set = con.createStatement();
            rs  =  set.executeQuery("SELECT  *  FROM  datos  ORDER  BY  serial
DESC"); //ORDEN DE MAS NUEVO A MAS
            //while (rs.next()) { //SIEMPRE HAY QUE PONER ESTO AUNQUE SEA
PARA UNO SOLO, PODEMOS HACER OREDER BY
            rs.next();

            cad = rs.getString("temp_corporal"); //el nombre de la columna
dentro del select *, devuelve el primer valor
            cad = cad.trim();//usar trim para dividir y comparar
            tempbebe = cad;
```

```
            cad = rs.getString("ppm"); //el nombre de la columna dentro del
select *, devuelve el primer valor
            cad = cad.trim();//usar trim para dividir y comparar
            pulsobebe = cad;

            cad = rs.getString("temp_habita"); //el nombre de la columna
dentro del select *, devuelve el primer valor
            cad = cad.trim();//usar trim para dividir y comparar
            tempcasa = cad;

            cad = rs.getString("humedad"); //el nombre de la columna dentro
del select *, devuelve el primer valor
            cad = cad.trim();//usar trim para dividir y comparar
            humcasa = cad;

            cad = rs.getString("timestamp"); //el nombre de la columna
dentro del select *, devuelve el primer valor
            cad = cad.trim();//usar trim para dividir y comparar
            timestamp = cad;

            /*if (cad.compareTo(nombre.trim()) == 0) {
                    existe = true;
                    break;//salimos del bucle si existe
                }*/
                //break;//a la primera, fuera

            //}
            rs.close();
            set.close();//puede que sobre una de estas
        } catch (Exception e) {
            System.out.println("No lee de la tabla"+e.toString());
        }


        cadena          =tempbebe+"-"+pulsobebe+"-"+tempcasa+"-"+humcasa+"-
"+timestamp;


        //String cadena = "hola prueba numero 1";
        return cadena;
```

```
    }

    public static String getUltimaCadena_avanzada(){
        // String nombre = "hola";
        //boolean existe = false;

        String cadenaCompleja;
        String cad;

        String tempbebe="-10";
        String pulsobebe="-10";
        String tempcasa="-10";
        String humcasa="-10";
        String timestamp="-10";

        String tempbebe_anterior="-10";
        String pulsobebe_anterior="-10";
        String tempcasa_anterior="-10";
        String humcasa_anterior="-10";
        String timestamp_anterior="-10";


        try {

            set = con.createStatement();
            rs = set.executeQuery("SELECT * FROM datos ORDER BY serial
DESC"); //ORDEN DE MAS NUEVO A MAS
            //while (rs.next()) { //SIEMPRE HAY QUE PONER ESTO AUNQUE SEA
PARA UNO SOLO, PODEMOS HACER OREDER BY
            rs.next();

            //ACTUALES
            cad = rs.getString("temp_corporal"); //el nombre de la columna
dentro del select *, devuelve el primer valor
            cad = cad.trim();//usar trim para dividir y comparar
            tempbebe = cad;

            cad = rs.getString("ppm"); //el nombre de la columna dentro del
select *, devuelve el primer valor
            cad = cad.trim();//usar trim para dividir y comparar
```

```
            pulsobebe = cad;

            cad = rs.getString("temp_habita"); //el nombre de la columna
dentro del select *, devuelve el primer valor
            cad = cad.trim();//usar trim para dividir y comparar
            tempcasa = cad;

            cad = rs.getString("humedad"); //el nombre de la columna dentro
del select *, devuelve el primer valor
            cad = cad.trim();//usar trim para dividir y comparar
            humcasa = cad;

            cad = rs.getString("timestamp"); //el nombre de la columna
dentro del select *, devuelve el primer valor
            cad = cad.trim();//usar trim para dividir y comparar
            timestamp = cad;

            //ANTERIOR
            rs.next();

            cad = rs.getString("temp_corporal"); //el nombre de la columna
dentro del select *, devuelve el primer valor
            cad = cad.trim();//usar trim para dividir y comparar
            tempbebe_anterior = cad;

            cad = rs.getString("ppm"); //el nombre de la columna dentro del
select *, devuelve el primer valor
            cad = cad.trim();//usar trim para dividir y comparar
            pulsobebe_anterior = cad;

            cad = rs.getString("temp_habita"); //el nombre de la columna
dentro del select *, devuelve el primer valor
            cad = cad.trim();//usar trim para dividir y comparar
            tempcasa_anterior = cad;

            cad = rs.getString("humedad"); //el nombre de la columna dentro
del select *, devuelve el primer valor
            cad = cad.trim();//usar trim para dividir y comparar
            humcasa_anterior = cad;
```

```
            cad = rs.getString("timestamp"); //el nombre de la columna
dentro del select *, devuelve el primer valor
            cad = cad.trim();//usar trim para dividir y comparar
            timestamp_anterior = cad;
            /*if (cad.compareTo(nombre.trim()) == 0) {
                    existe = true;
                    break;//salimos del bucle si existe
                }*/
                //break;//a la primera, fuera

            //}
            rs.close();
            set.close();//puede que sobre una de estas




        } catch (Exception e) {
            System.out.println("No lee de la tabla");
        }


         //FUTURO
        /*String tempbebe_futura="-10";
        String pulsobebe_futura="-10";
        String tempcasa_futura="-10";
        String humcasa_futura="-10";*/

        String     mediaDatos     =     getMedia30UltimasLecturas();//SIN
TIMESTAMP//tal cual
        String   cadenaFutura   =   ModeloDatos.estimarDatosFuturos();//SIN
TIMESTAMP//comparas actual con anterior y anterior, y si es creciente
creciente, decreciente decreciente, y tal.
        String   cadenaResumen   =   ModeloDatos.resumirDatosPasado();//SIN
TIMESTAMP//contiene:    medicion30    -    medicion1,    y    string    de
creciente/decreciente/igual
```

```
        cadenaCompleja       =       tempbebe+"-"+pulsobebe+"-"+tempcasa+"-
"+humcasa+"-"+timestamp+"-";//el actual
        cadenaCompleja        =cadenaCompleja+        tempbebe_anterior+"-
"+pulsobebe_anterior+"-"+tempcasa_anterior+"-"+humcasa_anterior+"-
"+timestamp_anterior+"-";

        cadenaCompleja   =   cadenaCompleja+mediaDatos+"-";//media   de   los
ultimos 30 datos medidos
        cadenaCompleja           =           cadenaCompleja+cadenaFutura+"-
";/*tempbebe_futura+"-"+pulsobebe_futura+"-"+tempcasa_futura+"-
"+humcasa_futura;*/
        cadenaCompleja = cadenaCompleja+cadenaResumen;
        //String cadena = "hola prueba numero 1";
        return cadenaCompleja;
    }

    public static String estimarDatosFuturos(){
        String cadena="";
        String tmp ="";
        float valor_actual = 0;
        float tempbebe_anterior =0;
        float tempcasa_anterior = 0;
        float pulsobebe_anterior=0;
        float humcasa_anterior=0;

        float diferencia;
        float diferencia_tempbebe=0;
        float diferencia_pulsobebe=0;
        float diferencia_tempcasa=0;
        float diferencia_humcasa=0;

        float ultimtempbebe =0;
        float ultimpulsobebe=0;
        float ultimtempcasa=0;
        float ultimhumcasa=0;

        try{
            set = con.createStatement();
```

```
            rs = set.executeQuery("SELECT * FROM datos ORDER BY serial
DESC");

            //primera iteracion, asignaciones
            rs.next();

            tmp = rs.getString("temp_corporal"); //el nombre de la columna
dentro del select *, devuelve el primer valor
            tmp = tmp.trim();
            valor_actual = Float.parseFloat(tmp);
            tempbebe_anterior=valor_actual;
            ultimtempbebe=valor_actual;

            tmp = rs.getString("ppm"); //el nombre de la columna dentro del
select *, devuelve el primer valor
            tmp = tmp.trim();
            valor_actual = Float.parseFloat(tmp);
            pulsobebe_anterior=valor_actual;
            ultimpulsobebe=valor_actual;

            tmp = rs.getString("temp_habita"); //el nombre de la columna
dentro del select *, devuelve el primer valor
            tmp = tmp.trim();
            valor_actual = Float.parseFloat(tmp);
            tempcasa_anterior= valor_actual;
            ultimtempcasa=valor_actual;

            tmp = rs.getString("humedad"); //el nombre de la columna dentro
del select *, devuelve el primer valor
            tmp = tmp.trim();
            valor_actual = Float.parseFloat(tmp);
            humcasa_anterior=valor_actual;
            ultimhumcasa=valor_actual;


            for(int i = 1;i<=2;i++){//tenemos el 0 de antes, ahora miramos
el 1 y el 2.
                rs.next();
```

```
                tmp  =  rs.getString("temp_corporal");  //el  nombre  de  la
columna dentro del select *, devuelve el primer valor
                tmp = tmp.trim();
                valor_actual = Float.parseFloat(tmp);

                diferencia = valor_actual - ultimtempbebe;
                if(i==1){
                    diferencia_tempbebe=diferencia;
                }
                else{//)
                    diferencia_tempbebe=diferencia_tempbebe+diferencia;
                }

                tempbebe_anterior=valor_actual;




                tmp = rs.getString("ppm"); //el nombre de la columna dentro
del select *, devuelve el primer valor
                tmp = tmp.trim();
                valor_actual = Float.parseFloat(tmp);
                diferencia = valor_actual - ultimpulsobebe;
                if(i==1){
                    diferencia_pulsobebe=diferencia;
                }
                else{//)
                    diferencia_pulsobebe=diferencia_pulsobebe+diferencia;
                }
                pulsobebe_anterior=valor_actual;



                tmp = rs.getString("temp_habita"); //el nombre de la columna
dentro del select *, devuelve el primer valor
                tmp = tmp.trim();
                valor_actual = Float.parseFloat(tmp);
                diferencia = valor_actual - ultimtempcasa;
                if(i==1){
                    diferencia_tempcasa=diferencia;
```

```
                }
                else{//)
                    diferencia_tempcasa=diferencia_tempcasa+diferencia;
                }
                tempcasa_anterior=valor_actual;


                tmp = rs.getString("humedad"); //el nombre de la columna
dentro del select *, devuelve el primer valor
                tmp = tmp.trim();
                valor_actual = Float.parseFloat(tmp);
                diferencia = valor_actual - ultimhumcasa;
                if(i==1){
                    diferencia_humcasa=diferencia;
                }
                else{//)
                    diferencia_humcasa=diferencia_humcasa+diferencia;
                }
                humcasa_anterior=valor_actual;
            }

        rs.close();
        set.close();//puede que sobre una de estas
        } catch (Exception e) {
            System.out.println("No lee de la tabla");
        }

        String tmp1="";
        String tmp2="";
        String tmp3="";
        String tmp4="";

        if(diferencia_tempbebe>0){
            tmp1="aumentará";
        }
        else if(diferencia_tempbebe<0){
            tmp1="disminuirá";
        }
        else{
            tmp1="no cambiará";
```

```
        }
        if(diferencia_pulsobebe>0){
            tmp2="aumentará";
        }
        else if(diferencia_pulsobebe<0){
            tmp2="disminuirá";
        }
        else{
            tmp2="no cambiará";
        }
        if(diferencia_tempcasa>0){
            tmp3="aumentará";
        }
        else if(diferencia_tempcasa<0){
            tmp3="disminuirá";
        }
        else{
            tmp3="no cambiará";
        }
        if(diferencia_humcasa>0){
            tmp4="aumentará";
        }
        else if(diferencia_humcasa<0){
            tmp4="disminuirá";
        }
        else{
            tmp4="no cambiará";
        }


        cadena = tmp1+"-"+tmp2+"-"+tmp3+"-"+tmp4;

        return cadena;

    }
    public static String resumirDatosPasado(){
        String cadena="";
        String tmp ="";
        float valor_actual = 0;
        float tempbebe_anterior =0;
```

```
        float tempcasa_anterior = 0;
        float pulsobebe_anterior=0;
        float humcasa_anterior=0;


        float ultimtempbebe =0;
        float ultimpulsobebe=0;
        float ultimtempcasa=0;
        float ultimhumcasa=0;


        int tempbebe_creciente =0;
        int tempcasa_creciente = 0;
        int pulsobebe_creciente=0;
        int humcasa_creciente=0;

        int tempbebe_decreciente =0;
        int tempcasa_decreciente = 0;
        int pulsobebe_decreciente=0;
        int humcasa_decreciente=0;

        try{
            set = con.createStatement();
            rs = set.executeQuery("SELECT * FROM datos ORDER BY serial
DESC");

            //primera iteracion, asignaciones
            rs.next();

            tmp = rs.getString("temp_corporal"); //el nombre de la columna
dentro del select *, devuelve el primer valor
            tmp = tmp.trim();
            valor_actual = Float.parseFloat(tmp);
            tempbebe_anterior=valor_actual;
            ultimtempbebe=valor_actual;

            tmp = rs.getString("ppm"); //el nombre de la columna dentro del
select *, devuelve el primer valor
            tmp = tmp.trim();
            valor_actual = Float.parseFloat(tmp);
```

```
            pulsobebe_anterior=valor_actual;
            ultimpulsobebe=valor_actual;

            tmp = rs.getString("temp_habita"); //el nombre de la columna
dentro del select *, devuelve el primer valor
            tmp = tmp.trim();
            valor_actual = Float.parseFloat(tmp);
            tempcasa_anterior= valor_actual;
            ultimtempcasa=valor_actual;

            tmp = rs.getString("humedad"); //el nombre de la columna dentro
del select *, devuelve el primer valor
            tmp = tmp.trim();
            valor_actual = Float.parseFloat(tmp);
            humcasa_anterior=valor_actual;
            ultimhumcasa=valor_actual;


            for(int i = 0;i<29;i++){
                rs.next();


                tmp = rs.getString("temp_corporal"); //el nombre de la
columna dentro del select *, devuelve el primer valor
                tmp = tmp.trim();
                valor_actual = Float.parseFloat(tmp);

                if(comparativa(tempbebe_anterior,"mayor",valor_actual)){
                    tempbebe_creciente++;
                }
                else{
                    tempbebe_decreciente++;
                }
                tempbebe_anterior=valor_actual;




                tmp = rs.getString("ppm"); //el nombre de la columna dentro
del select *, devuelve el primer valor
```

```
                tmp = tmp.trim();
                valor_actual = Float.parseFloat(tmp);
                if(comparativa(pulsobebe_anterior,"mayor",valor_actual)){
                    pulsobebe_creciente++;
                }
                else{
                    pulsobebe_decreciente++;
                }
                pulsobebe_anterior=valor_actual;



                tmp = rs.getString("temp_habita"); //el nombre de la columna
dentro del select *, devuelve el primer valor
                tmp = tmp.trim();
                valor_actual = Float.parseFloat(tmp);
                if(comparativa(tempcasa_anterior,"mayor",valor_actual)){
                    tempcasa_creciente++;
                }
                else{
                    tempcasa_decreciente++;
                }
                tempcasa_anterior=valor_actual;


                tmp = rs.getString("humedad"); //el nombre de la columna
dentro del select *, devuelve el primer valor
                tmp = tmp.trim();
                valor_actual = Float.parseFloat(tmp);
                if(comparativa(humcasa_anterior,"mayor",valor_actual)){
                    humcasa_creciente++;
                }
                else{
                    humcasa_decreciente++;
                }
                humcasa_anterior=valor_actual;
            }

        rs.close();
        set.close();//puede que sobre una de estas
```

```
        } catch (Exception e) {
            System.out.println("No lee de la tabla");
        }

        //cadena           =           (tempbebe_anterior-ultimtempbebe)+"-
"+(pulsobebe_anterior-ultimpulsobebe)+"-"+(tempcasa_anterior-
ultimtempcasa)+"-"+(humcasa_anterior-ultimhumcasa)+"-";

        if(tempbebe_creciente>tempbebe_decreciente){
            cadena=cadena+"mas tiempo aumentando-";
        }
        else if(tempbebe_creciente<tempbebe_decreciente){
            cadena=cadena+"mas tiempo disminuyendo-";
        }else{
            cadena=cadena+"equilibradas en tiempo-";
        }
        if(pulsobebe_creciente>pulsobebe_decreciente){
            cadena=cadena+"mas tiempo aumentando-";
        }
        else if(pulsobebe_creciente<pulsobebe_decreciente){
            cadena=cadena+"mas tiempo disminuyendo-";
        }else{
            cadena=cadena+"equilibradas en tiempo-";
        }

        if(tempcasa_creciente>tempcasa_decreciente){
            cadena=cadena+"mas tiempo aumentando-";
        }
        else if(tempcasa_creciente<tempcasa_decreciente){
            cadena=cadena+"mas tiempo disminuyendo-";
        }else{
            cadena=cadena+"equilibradas en tiempo-";
        }

        if(humcasa_creciente>humcasa_decreciente){
            cadena=cadena+"mas tiempo aumentando";
        }
        else if(humcasa_creciente<humcasa_decreciente){
            cadena=cadena+"mas tiempo disminuyendo";
        }else{
```

```
            cadena=cadena+"equilibradas en tiempo";
        }

        return cadena;

    }


    public static String getHistorialMediciones(){
        String cadena="";
        String tmp ="";
        String timestamp;
        float valor_actual = 0;
        float tempbebe =0;
        float tempcasa = 0;
        float pulsobebe=0;
        float humcasa=0;

        cadena += "timestamp  tempbebe  pulsobebe  tempcasa  humcasa\n";

        try{
            set = con.createStatement();
            rs  =  set.executeQuery("SELECT  *  FROM  datos  ORDER  BY  serial
DESC");
            for(int i = 0;i<29;i++){
                rs.next();


                tmp = rs.getString("timestamp"); //el nombre de la columna
dentro del select *, devuelve el primer valor
                tmp = tmp.trim();
                timestamp= tmp;

                tmp  =  rs.getString("temp_corporal");  //el  nombre  de  la
columna dentro del select *, devuelve el primer valor
                tmp = tmp.trim();
                valor_actual = Float.parseFloat(tmp);
                tempbebe= valor_actual;
```

```
                tmp = rs.getString("ppm"); //el nombre de la columna dentro
del select *, devuelve el primer valor
                tmp = tmp.trim();
                valor_actual = Float.parseFloat(tmp);
                pulsobebe= valor_actual;

                tmp = rs.getString("temp_habita"); //el nombre de la columna
dentro del select *, devuelve el primer valor
                tmp = tmp.trim();
                valor_actual = Float.parseFloat(tmp);
                tempcasa= valor_actual;

                tmp = rs.getString("humedad"); //el nombre de la columna
dentro del select *, devuelve el primer valor
                tmp = tmp.trim();
                valor_actual = Float.parseFloat(tmp);
                humcasa= valor_actual;

                cadena+=timestamp+"    "+tempbebe+"          "+pulsobebe+"
"+tempcasa+"              "+humcasa+"\n";
            }

        rs.close();
        set.close();//puede que sobre una de estas
        } catch (Exception e) {
            System.out.println("No lee de la tabla");
        }

        return cadena;
    }

    public static String getHistorialAlarmas(){
        String cadena="";
        String tmp ="";
        String timestamp = "";
        String valor_actual = "";
        String tipoalarma ="";
        String tipodato = "";
        String valordato="";
```

```
        cadena += "timestamp     tipoalarma     tipodato     valordato\n";

        try{
            set = con.createStatement();
            rs = set.executeQuery("SELECT * FROM alertas ORDER BY serial
DESC");
            for(int i = 0;i<29;i++){
                rs.next();


                tmp = rs.getString("timestamp"); //el nombre de la columna
dentro del select *, devuelve el primer valor
                tmp = tmp.trim();
                timestamp= tmp;

                tmp = rs.getString("tipo"); //el nombre de la columna dentro
del select *, devuelve el primer valor
                tmp = tmp.trim();
                tipoalarma=tmp;

                tmp = rs.getString("nombre"); //el nombre de la columna
dentro del select *, devuelve el primer valor
                tmp = tmp.trim();
                tipodato= tmp;

                tmp = rs.getString("valor"); //el nombre de la columna
dentro del select *, devuelve el primer valor
                tmp = tmp.trim();
                valordato= tmp;


                cadena+=timestamp+"       "+tipoalarma+"       "+tipodato+"
"+valordato+"\n";
            }

        rs.close();
        set.close();//puede que sobre una de estas
        } catch (Exception e) {
            System.out.println("No lee de la tabla alarma");
        }
```

```
        return cadena;
    }



    public static String getMedia30UltimasLecturas(){
        String cadena="";
        String tmp ="";
        float valor_actual = 0;
        float tempbebe =0;
        float tempcasa = 0;
        float pulsobebe=0;
        float humcasa=0;

        try{
            set = con.createStatement();
            rs = set.executeQuery("SELECT * FROM datos ORDER BY serial
DESC");
            for(int i = 0;i<29;i++){
                rs.next();


                tmp = rs.getString("temp_corporal"); //el nombre de la
columna dentro del select *, devuelve el primer valor
                tmp = tmp.trim();
                valor_actual = Float.parseFloat(tmp);
                tempbebe= tempbebe+valor_actual;

                tmp = rs.getString("ppm"); //el nombre de la columna dentro
del select *, devuelve el primer valor
                tmp = tmp.trim();
                valor_actual = Float.parseFloat(tmp);
                pulsobebe= tempbebe+valor_actual;

                tmp = rs.getString("temp_habita"); //el nombre de la columna
dentro del select *, devuelve el primer valor
                tmp = tmp.trim();
                valor_actual = Float.parseFloat(tmp);
                tempcasa= tempcasa+valor_actual;
```

```
                tmp = rs.getString("humedad"); //el nombre de la columna
dentro del select *, devuelve el primer valor
                tmp = tmp.trim();
                valor_actual = Float.parseFloat(tmp);
                humcasa= humcasa+valor_actual;
            }

        rs.close();
        set.close();//puede que sobre una de estas
        } catch (Exception e) {
            System.out.println("No lee de la tabla");
        }
        tempbebe = tempbebe/30;
        tempcasa = tempcasa/30;
        humcasa = humcasa/30;
        pulsobebe=pulsobebe/30;

        cadena =tempbebe+"-"+pulsobebe+"-"+tempcasa+"-"+humcasa;

        return cadena;
    }


    public static boolean comparativa(float anterior, String comparador,
float actual){
        //tipo == false quiere decir decreciente
        //tipo == true quiere decir creciente
        if(comparador.equals("menor")){//comparar decreciente, actual mas
pequeño que anterior
            if(anterior<actual){
                return true;
            }
            else
                return false;
        }
        else{//comparar creciente, actual mayor que anterior
            if(anterior>actual){
                return true;
            }
```

```
            else
                return false;
        }
    }

    public static boolean es_temperatura_bebe_creciente(){
        boolean monotona = false;
        String cad;
        float valor_anterior = 999;
        float valor_actual = 999;



        try {
            set = con.createStatement();
            rs = set.executeQuery("SELECT * FROM datos");
            //si ordenamos de mas nuevo a mas antiguo
            //y estamos buscando valores crecientes
            //eso quiere decir que el primer valor que encontremos debe ser
MAYOR que el segundo

            for(int i = 0; i<5;i++){
                rs.next();
                cad = rs.getString("temp_corporal");
                cad = cad.trim();
                valor_actual = Float.parseFloat(cad);

if(comparativa(valor_anterior,"mayor",valor_actual)==true){
                   //el valor mirado anteriormente es mayor que el actual.
                   //el ultimo valor registrado es mayor que el penultimo,
creciente.
                   valor_anterior=valor_actual;
                }
                else{
                    rs.close();
                    set.close();
                    return false;
                }

            }
```

```
        rs.close();
        set.close();

        } catch (Exception e) {
            System.out.println("No lee de la tabla");
        }

            return true;
    }
    public static boolean es_temperatura_bebe_DEcreciente(){
        boolean monotona = false;
        String cad;
        float valor_anterior = -999;
        float valor_actual = -999;



        try {
            set = con.createStatement();

            rs = set.executeQuery("SELECT * FROM datos");

            for(int i = 0; i<5;i++){
                rs.next();
                cad = rs.getString("temp_corporal");
                cad = cad.trim();
                valor_actual = Float.parseFloat(cad);

if(comparativa(valor_anterior,"menor",valor_actual)==false){
                    //el valor mirado anteriormente es menor que el actual.
                    //el ultimo valor registrado es menor que el penultimo,
decreciente.
                    valor_anterior=valor_actual;
                }
                else{
                    rs.close();
                    set.close();
                    return false;
                }
```

```
            }
        rs.close();
        set.close();

        } catch (Exception e) {
            System.out.println("No lee de la tabla");
        }

            return true;
    }

    //*************************
public static boolean es_pulso_bebe_creciente(){
        boolean monotona = false;
        String cad;
        float valor_anterior = 999;
        float valor_actual = 999;



        try {
            set = con.createStatement();
            rs = set.executeQuery("SELECT * FROM datos");
            //si ordenamos de mas nuevo a mas antiguo
            //y estamos buscando valores crecientes
            //eso quiere decir que el primer valor que encontremos debe ser
MAYOR que el segundo

            for(int i = 0; i<5;i++){
                rs.next();
                cad = rs.getString("ppm");
                cad = cad.trim();
                valor_actual = Float.parseFloat(cad);

if(comparativa(valor_anterior,"mayor",valor_actual)==true){
                    //el valor mirado anteriormente es mayor que el actual.
                    //el ultimo valor registrado es mayor que el penultimo,
creciente.
                    valor_anterior=valor_actual;
                }
```

```
                else{
                    rs.close();
                    set.close();
                    return false;
                }

            }
        rs.close();
        set.close();

        } catch (Exception e) {
            System.out.println("No lee de la tabla");
        }

            return true;
    }
    public static boolean es_pulso_bebe_DEcreciente(){
        boolean monotona = false;
        String cad;
        float valor_anterior = -999;
        float valor_actual = -999;



        try {
            set = con.createStatement();

            rs = set.executeQuery("SELECT * FROM datos");

            for(int i = 0; i<5;i++){
                rs.next();
                cad = rs.getString("ppm");
                cad = cad.trim();
                valor_actual = Float.parseFloat(cad);

if(comparativa(valor_anterior,"menor",valor_actual)==false){
                    //el valor mirado anteriormente es menor que el actual.
                    //el ultimo valor registrado es menor que el penultimo,
decreciente.
                    valor_anterior=valor_actual;
```

```
                }
                else{
                    rs.close();
                    set.close();
                    return false;
                }

            }
        rs.close();
        set.close();

        } catch (Exception e) {
            System.out.println("No lee de la tabla");
        }

            return true;
    }

//**************************
public static boolean es_temperatura_casa_creciente(){
        boolean monotona = false;
        String cad;
        float valor_anterior = 999;
        float valor_actual = 999;



        try {
            set = con.createStatement();
            rs = set.executeQuery("SELECT * FROM datos");
            //si ordenamos de mas nuevo a mas antiguo
            //y estamos buscando valores crecientes
            //eso quiere decir que el primer valor que encontremos debe ser
MAYOR que el segundo

            for(int i = 0; i<5;i++){
                rs.next();
                cad = rs.getString("temp_habita");
                cad = cad.trim();
                valor_actual = Float.parseFloat(cad);
```

```
if(comparativa(valor_anterior,"mayor",valor_actual)==true){
                    //el valor mirado anteriormente es mayor que el actual.
                    //el ultimo valor registrado es mayor que el penultimo,
creciente.
                    valor_anterior=valor_actual;
                }
                else{
                    rs.close();
                    set.close();
                    return false;
                }

            }
        rs.close();
        set.close();

        } catch (Exception e) {
            System.out.println("No lee de la tabla");
        }

            return true;
    }
    public static boolean es_temperatura_casa_DEcreciente(){
        boolean monotona = false;
        String cad;
        float valor_anterior = -999;
        float valor_actual = -999;



        try {
            set = con.createStatement();

            rs = set.executeQuery("SELECT * FROM datos");

            for(int i = 0; i<5;i++){
                rs.next();
                cad = rs.getString("temp_habita");
                cad = cad.trim();
```

```
                valor_actual = Float.parseFloat(cad);

if(comparativa(valor_anterior,"menor",valor_actual)==false){
                    //el valor mirado anteriormente es menor que el actual.
                    //el ultimo valor registrado es menor que el penultimo,
decreciente.
                    valor_anterior=valor_actual;
                }
                else{
                    rs.close();
                    set.close();
                    return false;
                }

            }
        rs.close();
        set.close();

        } catch (Exception e) {
            System.out.println("No lee de la tabla");
        }

            return true;
    }

//*************************************
public static boolean es_humedad_casa_creciente(){
        boolean monotona = false;
        String cad;
        float valor_anterior = 999;
        float valor_actual = 999;



        try {
            set = con.createStatement();
            rs = set.executeQuery("SELECT * FROM datos");
            //si ordenamos de mas nuevo a mas antiguo
            //y estamos buscando valores crecientes
```

```
            //eso quiere decir que el primer valor que encontremos debe ser
MAYOR que el segundo

            for(int i = 0; i<5;i++){
                rs.next();
                cad = rs.getString("humedad");
                cad = cad.trim();
                valor_actual = Float.parseFloat(cad);

if(comparativa(valor_anterior,"mayor",valor_actual)==true){
                   //el valor mirado anteriormente es mayor que el actual.
                   //el ultimo valor registrado es mayor que el penultimo,
creciente.
                   valor_anterior=valor_actual;
                }
                else{
                    rs.close();
                    set.close();
                    return false;
                }

            }
        rs.close();
        set.close();

        } catch (Exception e) {
            System.out.println("No lee de la tabla");
        }

            return true;
    }
    public static boolean es_humedad_casa_DEcreciente(){
        boolean monotona = false;
        String cad;
        float valor_anterior = -999;
        float valor_actual = -999;



        try {
```

```
            set = con.createStatement();

            rs = set.executeQuery("SELECT * FROM datos");

            for(int i = 0; i<5;i++){
                rs.next();
                cad = rs.getString("humedad");
                cad = cad.trim();
                valor_actual = Float.parseFloat(cad);

if(comparativa(valor_anterior,"menor",valor_actual)==false){
                   //el valor mirado anteriormente es menor que el actual.
                   //el ultimo valor registrado es menor que el penultimo,
decreciente.
                   valor_anterior=valor_actual;
                }
                else{
                    rs.close();
                    set.close();
                    return false;
                }

            }
        rs.close();
        set.close();

        } catch (Exception e) {
            System.out.println("No lee de la tabla");
        }

            return true;
    }
    public static void cerrarConexion() {
        try {
            con.close();
        } catch (Exception e) {
        }
    }

}
```

Servlet_app_enviarNotificacion.java

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 *
 * @author Denis
 */
@WebServlet(urlPatterns = {"/Servlet_app_enviarNotificacion"})
public class Servlet_app_enviarNotificacion extends HttpServlet {

    public String token = "";

    protected      void      processRequest(HttpServletRequest      request,
HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
             String query = request.getQueryString();
            this.token= query; //cogemos el token
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet
Servlet_app_enviarNotificacion</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Servlet Servlet_app_enviarNotificacion at " +
request.getContextPath() + "</h1>");
            out.println("<h2> token "+token+" </h2>");
            //ModeloDatos.insertarToken(token);
            out.println("</body>");
```

```
            out.println("</html>");
        }
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods.
Click on the + sign on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
```

```
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

Servlet_app_historiaAlarmas.java

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.net.*;

/**
 *
 * @author dcc
 */
@WebServlet(urlPatterns = {"/Servlet_app_historiaAlarmas"})
public class Servlet_app_historiaAlarmas extends HttpServlet {

    /**
     *   Processes   requests   for   both   HTTP   <code>GET</code>   and
<code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected      void      processRequest(HttpServletRequest      request,
HttpServletResponse response)
            throws ServletException, IOException {
```

```
        response.setContentType("text/html; charset=UTF-8");
        response.setCharacterEncoding("UTF-8");

        try (
            PrintWriter out = response.getWriter()) {
            //out.println("valor1 valor2 valor3 valor4");
            ModeloDatos.abrirConexion();
            String cadenaTexto = ModeloDatos.getHistorialAlarmas();
            ModeloDatos.cerrarConexion();
            out.println(cadenaTexto);
        }

    }
    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods.
Click on the + sign on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
```

```
    protected void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>
}
```

Servlet_app_historialMediciones.java

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.net.*;

/**
 *
 * @author dcc
 */
@WebServlet(urlPatterns = {"/Servlet_app_historialMediciones"})
public class Servlet_app_historialMediciones extends HttpServlet {

    /**
     *   Processes   requests   for   both   HTTP   <code>GET</code>   and
<code>POST</code>
     * methods.
     *
```

```
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {



        response.setContentType("text/html; charset=UTF-8");
        response.setCharacterEncoding("UTF-8");

        try (
            PrintWriter out = response.getWriter()) {
            //out.println("valor1 valor2 valor3 valor4");
            ModeloDatos.abrirConexion();
            String cadenaTexto = ModeloDatos.getHistorialMediciones();
            ModeloDatos.cerrarConexion();
            out.println(cadenaTexto);
        }

    }
    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods.
Click on the + sign on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
```

```
    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>
}
```

Servlet_app_ultimaMedicion.java

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.net.*;

/**
 *
 * @author dcc
```

```
 */
@WebServlet(urlPatterns = {"/Servlet_app_ultimaMedicion"})
public class Servlet_app_ultimaMedicion extends HttpServlet {

    /**
     *   Processes   requests   for   both   HTTP   <code>GET</code>   and
<code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected     void     processRequest(HttpServletRequest     request,
HttpServletResponse response)
            throws ServletException, IOException {



        response.setContentType("text/html; charset=UTF-8");
        response.setCharacterEncoding("UTF-8");

        try (



            PrintWriter out = response.getWriter()) {
            //out.println("valor1 valor2 valor3 valor4");
            ModeloDatos.abrirConexion();
            String cadenaTexto = ModeloDatos.getUltimaCadena();
            ModeloDatos.cerrarConexion();
            out.println(cadenaTexto);



        }

    }
    //  <editor-fold  defaultstate="collapsed"  desc="HttpServlet  methods.
Click on the + sign on the left to edit the code.">
```

```
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>
}
```

Servlet_app_ultimaMedicion_avanzada.java

```java
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.net.*;

/**
 *
 * @author dcc
 */
@WebServlet(urlPatterns = {"/Servlet_app_ultimaMedicion_avanzada"})
public class Servlet_app_ultimaMedicion_avanzada extends HttpServlet {

    /**
     *   Processes   requests   for   both   HTTP   <code>GET</code>   and
<code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected     void     processRequest(HttpServletRequest     request,
HttpServletResponse response)
            throws ServletException, IOException {



        response.setContentType("text/html; charset=UTF-8");
        response.setCharacterEncoding("UTF-8");

        try (



            PrintWriter out = response.getWriter()) {
```

```
            //out.println("valor1 valor2 valor3 valor4");
            ModeloDatos.abrirConexion();
            String cadenaTexto = ModeloDatos.getUltimaCadena_avanzada();
            ModeloDatos.cerrarConexion();
            out.println(cadenaTexto);


        }

    }
    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods.
Click on the + sign on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        processRequest(request, response);
```

```
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>
}
```

Servlet_arduino_medicion1_in.java

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 *
 * @author dcc
 */
@WebServlet(urlPatterns = {"/Servlet_arduino_medicion1_in"})
public class Servlet_arduino_medicion1_in extends HttpServlet {

//http://192.168.43.118:8080/v1.0obi/Servlet_arduino_medicion1_in?40-40-
40-40-x-x-x-x
    /**
     *   Processes   requests   for   both   HTTP   <code>GET</code>   and
<code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
```

```
     */
    protected      void      processRequest(HttpServletRequest      request,
HttpServletResponse response)
            throws ServletException, IOException {

        response.setContentType("text/html; charset=UTF-8");
        response.setCharacterEncoding("UTF-8");

        try (



            PrintWriter out = response.getWriter()) {

            /*String[]ids = request.getParameterValues("id");
            request.getQueryString();

            String valor = request.getParameter("t1");
            String valor2 = request.getParameter("t2");*/

            String query = request.getQueryString();
            String[] params = query.split("-");
            /**/



            /* TODO output your page here. You may use following sample
code. */
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>     Nuevo     Servlet_arduino_medicion1_in
</title>");
           //            out.println("<character-encoding>utf-8</character-
encoding>");
            out.println("</head>");

            out.println("<body>");
            out.println("<h1> Nueva medicion1 registrada: </h1>");
            //out.println("<h3> "+ params.length+" </h3>");
```

```
            for (int i = 0; i < params.length; i++) {
                out.println("<h3> "+ params[i]+" </h3>");
                //out.println("<h3> newval </h3>");
            }
            out.println("<h2> Registro valido </h2>");
            //out.println("<h1>Servlet      NewServlet      at      "      +
request.getContextPath() + "</h1>");
            out.println("<h2> VariablesTransformadas: </h2>");
            String temperaturaAmbiente = String.valueOf(params[0]);
            String humedadAmbiente = String.valueOf(params[1]);
            String temperaturaBebe = String.valueOf(params[2]);
            String pulsoBebe = String.valueOf(params[3]);
            String dia = String.valueOf(params[4]);
            String hora = String.valueOf(params[5]);
            String minuto = String.valueOf(params[6]);
            String segundo = String.valueOf(params[7]);

            out.println("<h3> temperaturaAmbiente: "+ temperaturaAmbiente+"
</h3>");
            out.println("<h3>    humedadAmbiente:    "+    humedadAmbiente+"
</h3>");
            out.println("<h3>    temperaturaBebe:    "+    temperaturaBebe+"
</h3>");
            out.println("<h3> pulsoBebe: "+ pulsoBebe+" </h3>");

            String timestamp = dia+":"+hora+":"+minuto+":"+segundo;

            ModeloDatos.abrirConexion();
            //System.out.println(ModeloDatos.prueba());
            //ModeloDatos.insertarDatos2();
            ModeloDatos.insertaDatos(temperaturaBebe,           pulsoBebe,
temperaturaAmbiente, humedadAmbiente, timestamp);

            ModeloDatos.cerrarConexion();

            out.println("</body>");
            out.println("</html>");
        }

    }
```

```
    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods.
Click on the + sign on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
```

```
    }// </editor-fold>
}
```

CÓDIGO APLICACIÓN ANDROID
MAINACTIVITY

```
public class MainActivity extends AppCompatActivity {
    TextView myAwesomeTextView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void consultar(View view) {
        new getMediciones().execute();
    }

    private class getMediciones extends AsyncTask<Void, Void, String> {

        protected String doInBackground (Void... params){

            HttpURLConnection con= null;
            Scanner sc = null;
            String cadena;

            try{
            URL                 url                 =                 new
URL("http://192.168.43.118:8080/v1.0obi/Servlet_app_ultimaMedicion");
```

```
            con = (HttpURLConnection)url.openConnection();
            con.setRequestMethod("GET");
            con.setDoInput(true);
            con.connect();

            int responseCode = con.getResponseCode();

            if(responseCode == HttpsURLConnection.HTTP_OK){
                InputStream in = new
BufferedInputStream(con.getInputStream());
                sc = new Scanner(in);
                cadena = sc.next();
                in.close();
                con.disconnect();
                return cadena;
            }else{
                return ("ERROR");
            }
            }catch(Exception e){
                return ("ERROR");
            }
        }
    @Override
     protected void onPostExecute(String s){
            super.onPostExecute(s);
            ArrayList listaTextViews = new ArrayList<>();
            listaTextViews.add(findViewById(R.id.textView1));
            listaTextViews.add(findViewById(R.id.textView2));
            listaTextViews.add(findViewById(R.id.textView3));
            listaTextViews.add(findViewById(R.id.textView4));
            listaTextViews.add(findViewById(R.id.textView8));
            int contador = 0;
            StringTokenizer tokens = new StringTokenizer(s,"-");
            while(tokens.hasMoreTokens()){
                myAwesomeTextView = (TextView)
listaTextViews.get(contador);
                myAwesomeTextView.setText(tokens.nextToken());
                contador++;
            }

        }
        }
    public void avanzado(View view){
        Intent intent = new Intent(this, ActivityAvanzado.class);
        startActivity(intent);
```

```
    }

    public void historialMediciones(View view){
        Intent intent = new Intent(this, ActivityHMediciones.class);
        startActivity(intent);
    }
    public void historialHAlarmas(View view){
        Intent intent = new Intent(this, ActtivityHAlarmas.class);
        startActivity(intent);
    }
    }
```

ACTIVITYAVANZADO

```
public class ActivityAvanzado extends AppCompatActivity {

    TextView myAwesomeTextView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_avanzado);
        new getAvanzado().execute();
    }

    public void volver(View view) {
        Intent intent = new Intent(this, MainActivity.class);
        startActivity(intent);
    }

    private class getAvanzado extends AsyncTask<Void, Void, String> {

        protected String doInBackground (Void... params){

            HttpURLConnection con= null;
            Scanner sc = null;
            String cadena="";

            try{
                URL url = new
URL("http://192.168.43.118:8080/v1.0obi/Servlet_app_ultimaMedicion_ava
nzada");
                con = (HttpURLConnection)url.openConnection();
                con.setRequestMethod("GET");
                con.setDoInput(true);
```

```
                con.connect();

                int responseCode = con.getResponseCode();

                if(responseCode == HttpsURLConnection.HTTP_OK){
                    InputStream in = new
BufferedInputStream(con.getInputStream());
                    InputStreamReader ir = new InputStreamReader(in);
                    BufferedReader br=new BufferedReader(ir);
                    int i;
                    while((i=br.read())!=-1){
                        cadena+=((char)i);
                    }
                    br.close();
                    in.close();
                    in.close();
                    Log.d("CADENA : ", "" + cadena);
                    con.disconnect();
                    return cadena;
                }else{
                    return ("ERROR");
                }
            }catch(Exception e){
                return ("ERROR");
            }
        }
        @Override
        protected void onPostExecute(String s){
            super.onPostExecute(s);
            ArrayList listaTextViews = new ArrayList<>();
            listaTextViews.add(findViewById(R.id.textView16));
            listaTextViews.add(findViewById(R.id.textView21));
            listaTextViews.add(findViewById(R.id.textView22));
            listaTextViews.add(findViewById(R.id.textView23));
            listaTextViews.add(findViewById(R.id.textView24));
            listaTextViews.add(findViewById(R.id.textView26));
            listaTextViews.add(findViewById(R.id.textView27));
            listaTextViews.add(findViewById(R.id.textView28));
            listaTextViews.add(findViewById(R.id.textView29));
            listaTextViews.add(findViewById(R.id.textView30));
            listaTextViews.add(findViewById(R.id.textView36));
            listaTextViews.add(findViewById(R.id.textView37));
            listaTextViews.add(findViewById(R.id.textView38));
            listaTextViews.add(findViewById(R.id.textView39));
            listaTextViews.add(findViewById(R.id.textView41));
```

```
            listaTextViews.add(findViewById(R.id.textView42));
            listaTextViews.add(findViewById(R.id.textView43));
            listaTextViews.add(findViewById(R.id.textView44));
            listaTextViews.add(findViewById(R.id.textView46));
            listaTextViews.add(findViewById(R.id.textView47));
            listaTextViews.add(findViewById(R.id.textView48));
            listaTextViews.add(findViewById(R.id.textView49));


            int contador = 0;
            StringTokenizer tokens = new StringTokenizer(s,"-");
            while(tokens.hasMoreTokens()){
                myAwesomeTextView               =               (TextView)
listaTextViews.get(contador);
                String token = tokens.nextToken();
                Log.d("TOKEN: ", "" + token);
                myAwesomeTextView.setText(token);
                contador++;
            }

        }
    }
}
```

ACTIVITYHMEDICIONES

```
public class ActivityHMediciones extends AppCompatActivity {

    TextView myAwesomeTextView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_hmediciones);
        new getHistorialMediciones().execute();
    }
    public void volver(View view) {
        Intent intent = new Intent(this, MainActivity.class);
        startActivity(intent);

    }

    private class getHistorialMediciones extends AsyncTask<Void, Void,
String> {
```

```
        protected String doInBackground (Void... params){

            HttpURLConnection con= null;
            Scanner sc = null;
            String cadena="";

            try{
                URL                url                =                new
URL("http://192.168.43.118:8080/v1.0obi/Servlet_app_historialMedicione
s");
                con = (HttpURLConnection)url.openConnection();
                con.setRequestMethod("GET");
                con.setDoInput(true);
                con.connect();

                int responseCode = con.getResponseCode();

                if(responseCode == HttpsURLConnection.HTTP_OK){
                    InputStream            in            =            new
BufferedInputStream(con.getInputStream());
                    InputStreamReader ir = new InputStreamReader(in);
                    BufferedReader br=new BufferedReader(ir);
                    int i;
                    while((i=br.read())!=-1){
                        cadena+=((char)i);
                    }
                    br.close();
                    in.close();
                    con.disconnect();
                    return cadena;
                }else{
                    return ("ERROR");
                }
            }catch(Exception e){
                return ("ERROR");
            }
        }
        @Override
        protected void onPostExecute(String s){
            super.onPostExecute(s);
                myAwesomeTextView               =               (TextView)
(findViewById(R.id.textView18));
                myAwesomeTextView.setMovementMethod(new
ScrollingMovementMethod());
                myAwesomeTextView.setText(s);
```

```
        }

    }
}
```

ACTIVITYHALARMAS

```
public class ActtivityHAlarmas extends AppCompatActivity {

    TextView myAwesomeTextView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_acttivity_halarmas);
        new getHistorialAlarmas().execute();
    }

    public void volver(View view) {
        Intent intent = new Intent(this, MainActivity.class);
        startActivity(intent);

    }

    private class getHistorialAlarmas extends AsyncTask<Void, Void,
String> {

        protected String doInBackground (Void... params){

            HttpURLConnection con= null;
            Scanner sc = null;
            String cadena="";

            try{
                URL url = new
URL("http://192.168.43.118:8080/v1.0obi/Servlet_app_historiaAlarmas");
                con = (HttpURLConnection)url.openConnection();
                con.setRequestMethod("GET");
                con.setDoInput(true);
                con.connect();

                int responseCode = con.getResponseCode();

                if(responseCode == HttpsURLConnection.HTTP_OK){
```

```
                    InputStream            in            =            new
BufferedInputStream(con.getInputStream());
                    InputStreamReader ir = new InputStreamReader(in);
                    BufferedReader br=new BufferedReader(ir);
                    int i;
                    while((i=br.read())!=-1){
                        cadena+=((char)i);
                    }
                    br.close();
                    in.close();
                    con.disconnect();
                    return cadena;
                }else{
                    return ("ERROR");
                }
            }catch(Exception e){
                return ("ERROR");
            }
        }
        @Override
        protected void onPostExecute(String s){
            super.onPostExecute(s);
            myAwesomeTextView                =                (TextView)
(findViewById(R.id.textView20));
            myAwesomeTextView.setText(s);
        }

    }
}

MYFIREBASEMESSAGINGSERVICE
public          class          MyFirebaseMessagingService          extends
FirebaseMessagingService {

    private static final String TAG = "MyFirebaseMsgService";

    @Override
    public void onMessageReceived(RemoteMessage remoteMessage) {

        super.onMessageReceived(remoteMessage);
        Log.i(TAG,"FROM: "+ remoteMessage.getFrom());
        Log.i(TAG, "BODY: "+remoteMessage.getNotification().getBody());

        Intent intent = new Intent(this, MainActivity.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
```

```
        PendingIntent pendingIntent = PendingIntent.getActivity(this, 0
/* Request code */, intent,
                PendingIntent.FLAG_ONE_SHOT);

        String channelId = "app_channel";
        Uri                        defaultSoundUri                        =
RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
        NotificationCompat.Builder notificationBuilder =
                new NotificationCompat.Builder(this, channelId)
                        .setContentTitle(getString(R.string.app_name))

.setContentText(remoteMessage.getNotification().getBody())
                        .setAutoCancel(true)
                        .setSound(defaultSoundUri)
                        .setContentIntent(pendingIntent);

        NotificationManager notificationManager =
                (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);

        // Since android Oreo notification channel is needed.
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            NotificationChannel          channel          =          new
NotificationChannel(channelId,
                    "Channel human readable title",
                    NotificationManager.IMPORTANCE_DEFAULT);
            notificationManager.createNotificationChannel(channel);
        }

        notificationManager.notify(0   /*   ID   of   notification   */,
notificationBuilder.build());
    }

    @Override
    public void onNewToken(String token) {
        Log.d(TAG, "Refreshed token: " + token);
        sendRegistrationToServer(token);
    }

    private void sendRegistrationToServer(String token) {
        new enviarToken().execute(token);
    }

    public class enviarToken extends AsyncTask<String, Void, Void>{
        @Override
```

```
        protected Void doInBackground(String... strings){
            HttpURLConnection con = null;
            try{
                URL                url                =                new
URL("http://192.168.43.118:8080/20190103obi/Servlet_app_enviarNotifica
cion?"+strings[0]);
                Log.d(TAG, "URL ENVIADA: " + url);

                con = (HttpURLConnection)url.openConnection();
                con.setDoOutput(true);

con.setFixedLengthStreamingMode(strings[0].getBytes().length);


               OutputStream             out             =             new
BufferedOutputStream(con.getOutputStream());


                out.write(strings[0].getBytes());
                out.flush();
                out.close();
                final int responseCode = con.getResponseCode();

                Log.d(TAG, "Response Code: " + responseCode);

            }catch(Exception e){
                e.printStackTrace();
            }finally{
                if (con!=null) con.disconnect();
            }
            return null;
        }

        @Override
        protected void onPostExecute(Void s){
            Toast.makeText(getBaseContext(),"TOKEN
ENVIADO",Toast.LENGTH_LONG).show();
        }
    }

}
```