MEMORIA PROYECTO JULIO

María Jesús Vega Vázquez

Índice:

1.- Entregable 1

En este primer entregable tenemos que hacer un proyecto con un Servlet que permite gestionar una petición GET y POST para un tipo de datos definido. Como repositorio de datos empleamos List.

En este entregable hago uso de dos constructores, uno para el sensor y otro para el actuador.

```java
package es.us.lsi.dad;

import java.math.BigInteger;

public class Actuador {

    private Integer idvalue;
    private Integer idactuador;
    private BigInteger timestamp;
    private Double value;
    private Integer idgrupo;
    private Integer idplaca;

    public Actuador(Integer idvalue, Integer idactuador, BigInteger timestamp, Double value, Int
        super();
        this.idvalue = idvalue;
        this.idactuador = idactuador;
        this.timestamp = timestamp;
        this.value = value;
        this.idgrupo = idgrupo;
        this.idplaca = idplaca;
    }

    public Integer getIdvalue() {
        return idvalue;
    }

    public void setIdvalue(Integer idvalue) {
        this.idvalue = idvalue;
    }

    public Integer getIdactuador() {
        return idactuador;
    }

    public void setIdactuador(Integer idactuador) {
        this.idactuador = idactuador;
    }
```

```java
package es.us.lsi.dad;

import java.math.BigInteger;

public class Sensor {

    private Integer idvalue;
    private Integer idsensor;
    private BigInteger timestamp;
    private Double valueTemp;
    private Double valueHum;
    private Integer idgrupo;
    private Integer idplaca;

    public Sensor(Integer idvalue, Integer idsensor, BigInteger timestamp, Double valueTemp, Double valueHum, Integer idgrupo, Integer idplaca) {
        super();
        this.idvalue = idvalue;
        this.idsensor = idsensor;
        this.timestamp = timestamp;
        this.valueTemp = valueTemp;
        this.valueHum = valueHum;
        this.idgrupo = idgrupo;
        this.idplaca = idplaca;
    }

    public Integer getIdvalue() {
        return idvalue;
    }

    public void setIdvalue(Integer idvalue) {
        this.idvalue = idvalue;
    }

    public Integer getIdsensor() {
        return idsensor;
    }

    public void setIdsensor(Integer idsensor) {
        this.idsensor = idsensor;
```

En nuestra clase Sevlet, hacemos uso de un toGet y un toPost para poder leer y poder postear los datos en nuestras lista.

```java
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    String tipo = req.getParameter("tipo");

    if (tipo == null) {
        PrintWriter out = resp.getWriter();
        out.println("<html>");
        out.println("<body>");
        out.println("<h1>Bienvenido al Servlet del entregable 1</h1><br>");
        out.println("<h2>Por favor, introduce en la url un tipo válido (sensor, actuador) y sus respectivos IDs</h2>")
        out.println("</body>");
        out.println("</html>");
    } else {
        Gson gson = new Gson();
        JsonObject respuesta = new JsonObject();

        switch (tipo) {

            case "sensor":
                Integer idSensor = Integer.valueOf(req.getParameter("idSensor"));
                Integer idPlacaSensor = Integer.valueOf(req.getParameter("idPlaca"));
                List<Sensor> listaSensor = new ArrayList<>();
                for (Sensor s : sensores) {
                    if (s.getIdsensor().equals(idSensor) && s.getIdplaca().equals(idPlacaSensor)) {
                        listaSensor.add(s);
                    }
                }
                respuesta.add("sensores", gson.toJsonTree(listaSensor));
                break;

            case "actuador":
                Integer idActuador = Integer.valueOf(req.getParameter("idActuador"));
                Integer idPlacaActuador = Integer.valueOf(req.getParameter("idPlaca"));
                List<Actuador> listaActuador = new ArrayList<>();
                for (Actuador a : actuadores) {
                    if (a.getIdactuador().equals(idActuador) && a.getIdplaca().equals(idPlacaActuador)) {
                        listaActuador.add(a);
                    }
                }
                respuesta.add("actuadores", gson.toJsonTree(listaActuador));
                break;

            default:
                resp.getWriter().println("Tipo no válido");
                resp.setStatus(400);
                return;
        }

        resp.getWriter().println(respuesta);
        resp.setStatus(201);
    }
}
```

```java
protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    BufferedReader reader = req.getReader();
    Gson gson = new Gson();
    String tipo = req.getParameter("tipo");

    if (tipo == null) {
        resp.getWriter().print("Tipo no especificado");
        resp.setStatus(400);
        return;
    }

    try {
        switch (tipo) {

            case "sensor":
                Sensor sensor = gson.fromJson(reader, Sensor.class);
                if (sensor.getIdplaca() == null || sensor.getIdsensor() == null || sensor.getIdgrupo() == null
                        || sensor.getTimestamp() == null || sensor.getValueTemp() == null || sensor.getValueHum() == null) {
                    resp.getWriter().print("Formulario incompleto, faltan datos");
                    resp.setStatus(400);
                } else {
                    sensores.add(sensor);
                    resp.getWriter().println(gson.toJson(sensor));
                    resp.setStatus(201);
                }
                break;

            case "actuador":
                Actuador actuador = gson.fromJson(reader, Actuador.class);
                if (actuador.getIdplaca() == null || actuador.getIdactuador() == null || actuador.getIdgrupo() == null
                        || actuador.getTimestamp() == null || actuador.getValue() == null) {
                    resp.getWriter().print("Formulario incompleto, faltan datos");
                    resp.setStatus(400);
                } else {
                    actuadores.add(actuador);
                    resp.getWriter().println(gson.toJson(actuador));
                    resp.setStatus(201);
                }
                break;

            default:
                resp.getWriter().println("Tipo no válido");
                resp.setStatus(400);
                break;
        }
    } catch (Exception e) {
        resp.getWriter().println("Los datos facilitados no son válidos.");
        resp.setStatus(500);
    }
}
```

2.- Entregable 2

En este entregable, empezamos a hacer los pilares de nuestro proyecto. Haremos una implementación mediante Vertx de los endpoints para la gestión de la API Rest, incluiremos peticiones GET y POST para el sensor y el actuador. Usaré Map para almacenar en memoria los datos, los constructores son los mismos en todos los entregables. A continuación, muestro capturas de parte del código que he usado para la creación de la api en esta práctica y que será modificada y usada en futuras entregas.

```java
public class RestServer extends AbstractVerticle {

    private Map<Integer, SensorEntity> sensores = new HashMap<>();
    private Map<Integer, ActuadorEntity> actuadores = new HashMap<>();
    private Gson gson;

    @Override
    public void start(Promise<Void> startPromise) {
        // Creating some synthetic data
        createSomeSensorData(25);
        createSomeActuadorData(25);

        // Instantiating a Gson serialize object using specific date format
        gson = new GsonBuilder().setDateFormat("yyyy-MM-dd").create();

        // Defining the router object
        Router router = Router.router(vertx);

        // Handling any server startup result
        vertx.createHttpServer().requestHandler(router).listen(8084, result -> {
            if (result.succeeded()) {
                startPromise.complete();
            } else {
                startPromise.fail(result.cause());
            }
        });

        // SENSOR
        router.route("/api/sensor*").handler(BodyHandler.create());
        router.get("/api/sensor").handler(this::getAllSensors);
        router.post("/api/sensor").handler(this::addOneSensor);
        router.get("/api/sensor/:sensorid").handler(this::getOneSensor);
        router.get("/api/sensor/last/:sensorid").handler(this::getLastSensorValue);
```

```java
// Sensor Methods

private void getAllSensors(RoutingContext routingContext) {
    routingContext.response().putHeader("content-type", "application/json; charset=utf-8").setStatusCode(200)
            .end(gson.toJson(sensores.values()));
}

private void addOneSensor(RoutingContext routingContext) {
    final SensorEntity sensor = gson.fromJson(routingContext.getBodyAsString(), SensorEntity.class);
    sensores.put(sensor.getIdSensor(), sensor);
    routingContext.response().setStatusCode(201).putHeader("content-type", "application/json; charset=utf-8")
            .end(gson.toJson(sensor));
}

private void getOneSensor(RoutingContext routingContext) {
    int id = Integer.parseInt(routingContext.request().getParam("sensorid"));
    if (sensores.containsKey(id)) {
        SensorEntity sensor = sensores.get(id);
        routingContext.response().putHeader("content-type", "application/json; charset=utf-8")
                .setStatusCode(200).end(gson.toJson(sensor));
    } else {
        routingContext.response().putHeader("content-type", "application/json; charset=utf-8")
                .setStatusCode(404).end();
    }
}

private void getLastSensorValue(RoutingContext routingContext) {
    int id = Integer.parseInt(routingContext.request().getParam("sensorid"));
    SensorEntity lastSensor = sensores.values().stream()
            .filter(sensor -> sensor.getIdSensor().equals(id))
            .max((s1, s2) -> s1.getTimestamp().compareTo(s2.getTimestamp()))
            .orElse(null);
    if (lastSensor != null) {
        routingContext.response().putHeader("content-type", "application/json; charset=utf-8").setStatusCode(200)
                .end(gson.toJson(lastSensor));
    } else {
        routingContext.response().setStatusCode(404).end();
    }
}
```

```java
// Actuador Methods

private void getAllActuadores(RoutingContext routingContext) {
    routingContext.response().putHeader("content-type", "application/json; charset=utf-8").setStatusCode(200)
            .end(gson.toJson(actuadores.values()));
}

private void addOneActuador(RoutingContext routingContext) {
    final ActuadorEntity actuador = gson.fromJson(routingContext.getBodyAsString(), ActuadorEntity.class);
    actuadores.put(actuador.getIdactuador(), actuador);
    routingContext.response().setStatusCode(201).putHeader("content-type", "application/json; charset=utf-8")
            .end(gson.toJson(actuador));
}

private void getOneActuador(RoutingContext routingContext) {
    int id = Integer.parseInt(routingContext.request().getParam("actuadorid"));
    if (actuadores.containsKey(id)) {
        ActuadorEntity actuador = actuadores.get(id);
        routingContext.response().putHeader("content-type", "application/json; charset=utf-8")
                .setStatusCode(200).end(gson.toJson(actuador));
    } else {
        routingContext.response().putHeader("content-type", "application/json; charset=utf-8")
                .setStatusCode(404).end();
    }
}

private void createSomeSensorData(int number) {
    Random rnd = new Random();
    IntStream.range(0, number).forEach(elem -> {
        int id = rnd.nextInt(1000);  // Use a range to avoid collision
        sensores.put(id, new SensorEntity(id, id, BigInteger.valueOf(System.currentTimeMillis()),
        rnd.nextDouble() * 100, rnd.nextDouble() * 100, rnd.nextInt(10), rnd.nextInt(10)));
    });
}

private void createSomeActuadorData(int number) {
    Random rnd = new Random();
    IntStream.range(0, number).forEach(elem -> {
        int id = rnd.nextInt(1000);  // Use a range to avoid collision
        actuadores.put(id, new ActuadorEntity(id, id, BigInteger.valueOf(System.currentTimeMillis()),
        rnd.nextDouble() * 100, rnd.nextInt(10), rnd.nextInt(10)));
    });
}
```

3.- Entregable 3

En este entregable, sustituimos los Map creados por la conexión con la base de datos. A continuación, vemos los cambios que hemos realizado:

```java
public class RestServer extends AbstractVerticle {

    public static MySQLPool mySqlClient;
    private Gson gson;

    public void start(Promise<Void> startFuture) {

        MySQLConnectOptions connectOptions = new MySQLConnectOptions().setPort(3306).setHost("127.0.0.1")
                .setDatabase("proyectodad").setUser("root").setPassword("root");

        PoolOptions poolOptions = new PoolOptions().setMaxSize(5);
        mySqlClient = MySQLPool.pool(vertx, connectOptions, poolOptions);

        mySqlClient.query("SELECT idValue from sensor last").execute(handler -> {
            System.out.println(handler.succeeded() ? "Todo OK" : "Error");
        });

        // Instantiating a Gson serialize object using specific date format
        gson = new GsonBuilder().setDateFormat("yyyy-MM-dd").create();

        // Defining the router object
        Router router = Router.router(vertx);

        // Handling any server startup result
        vertx.createHttpServer().requestHandler(router::handle).listen(8080, result -> {
            if (result.succeeded()) {
                startFuture.complete();
            } else {
                startFuture.fail(result.cause());
            }
        });

        // SENSOR
        router.route("/api/sensor*").handler(BodyHandler.create());
        router.get("/api/sensor").handler(this::getSensorWithAllParams);
        router.post("/api/sensor").handler(this::addOneSensor);
        router.get("/api/sensor/:sensorid").handler(this::getOneSensorValues);

        // ACTUADOR
        router.route("/api/actuador*").handler(BodyHandler.create());
        router.get("/api/actuador").handler(this::getActuadorWithAllParams);
        router.post("/api/actuador").handler(this::addOneActuador);
        router.get("/api/actuador/:actuadorid").handler(this::getOneActuadorValue);
        router.post("/api/actuador").handler(this::addOneActuadorValue);

    }
```

```java
// Sensor
@SuppressWarnings("unused")
private void getAllSensor() {
    mySqlClient.query("SELECT * FROM sensor").execute(res -> {
        if (res.succeeded()) {
            RowSet<Row> resultSet = res.result();
            JsonArray result = new JsonArray();
            for (Row elem : resultSet) {
                result.add(JsonObject.mapFrom(new SensorEntity(elem.getInteger("idvalue"),
                        elem.getInteger("idsensor"), BigInteger.valueOf(elem.getInteger("timestamp")), elem.getDouble("valueTemp"),
                        elem.getDouble("valueHum"), elem.getInteger("idgrupo"), elem.getInteger("idplaca"))));
            }
            System.out.println(result.toString());
            routingContext.response().putHeader("content-type", "application/json; charset=utf-8").setStatusCode(200)
            .end(gson.toJson(new SensorEntityListWrapper(sensores.values())));

        } else {
            System.out.println("Error: " + res.cause().getLocalizedMessage());
        }
    });
}

private void getSensorWithAllParams(RoutingContext routingContext) {
    Integer id = Integer.parseInt(routingContext.request().getParam("idsensor"));
    mySqlClient.getConnection(connection -> {
        if (connection.succeeded()) {
            connection.result().preparedQuery("SELECT * FROM sensor WHERE idSensor = ?").execute(Tuple.of(id),
                    res -> {
                        if (res.succeeded()) {
                            RowSet<Row> resultSet = res.result();
                            JsonArray result = new JsonArray();
                            for (Row elem : resultSet) {
                                result.add(JsonObject.mapFrom(new SensorEntity(elem.getInteger("idvalue"),
                                        elem.getInteger("idsensor"), BigInteger.valueOf(elem.getInteger("timestamp")),
                                        elem.getDouble("valueTemp"),
                                        elem.getDouble("valueHum"), elem.getInteger("idgrupo"), elem.getInteger("idplaca"))));
                            }
                            System.out.println(result.toString());
                            routingContext.response().setStatusCode(200)
                                    .putHeader("content-type", "application/json; charset=utf-8")
                                    .end(result.encodePrettily());
                        } else {
                            System.out.println("Error: " + res.cause().getLocalizedMessage());
                        }
                        connection.result().close();
                    });
        } else {
            System.out.println(connection.cause().toString());
        }
    });
}
```

```java
private void addOneSensor(RoutingContext routingContext) {
    SensorEntity sensor = new Gson().fromJson(routingContext.getBodyAsString(), SensorEntity.class);

    System.out.println(sensor.toString());
    mySqlClient
            .preparedQuery("INSERT INTO sensor (idSensor, timeStampSensor, valueTemp, valueHum, idGrupo, idPlaca) VALUES (?, ?, ?, ?, ?, ?);")
            .execute(Tuple.of(sensor.idsensor, sensor.timestamp, sensor.valueTemp, sensor.valueHum, sensor.idgrupo, sensor.idplaca), res -> {
                if (res.succeeded()) {
                    routingContext.response().setStatusCode(201)
                            .putHeader("content-type", "application/json; charset=utf-8")
                            .end(new Gson().toJson(sensor));
                } else {
                    System.out.println("Error: " + res.cause().getLocalizedMessage());
                }
            });
}

private void getOneSensorValues(RoutingContext routingContext) {

    int id = Integer.parseInt(routingContext.request().getParam("sensorid"));

    mySqlClient.getConnection(connection -> {
        if (connection.succeeded()) {
            connection.result().preparedQuery("SELECT * FROM sensor WHERE idSensor = ?").execute(Tuple.of(id),
                    res -> {
                        if (res.succeeded()) {
                            RowSet<Row> resultSet = res.result();
                            List<SensorEntity> result = new ArrayList<SensorEntity>();
                            for (Row elem : resultSet) {
                                result.add(new SensorEntity(elem.getInteger("idvalue"),
                                        elem.getInteger("idsensor"), BigInteger.valueOf(elem.getInteger("timestamp")), elem.getDouble("valueTemp"),
                                        elem.getDouble("valueHum"), elem.getInteger("idgrupo"), elem.getInteger("idplaca")));
                            }
                            routingContext.response().setStatusCode(200)
                                    .putHeader("content-type", "application/json; charset=utf-8")
                                    .end(gson.toJson(result));
                        } else {
                            System.out.println("Error: " + res.cause().getLocalizedMessage());
                        }
                        connection.result().close();
                    });
        } else {
            System.out.println(connection.cause().toString());
        }
    });
}
```

```java
// Actuador
@SuppressWarnings("unused")
private void getAllActuador(RoutingContext routingContext) {
    mySqlClient.query("SELECT * FROM actuador").execute(res -> {
        if (res.succeeded()) {
            RowSet<Row> resultSet = res.result();
            JsonArray result = new JsonArray();
            for (Row elem : resultSet) {
                result.add(JsonObject.mapFrom(new ActuadorEntity(elem.getInteger("idvalue"),
                        elem.getInteger("idactuador"), BigInteger.valueOf(elem.getInteger("timestamp")), elem.getDouble("value"),
                        elem.getInteger("idgrupo"), elem.getInteger("idplaca"))));
            }
            System.out.println(result.toString());
            routingContext.response().setStatusCode(200)
                    .putHeader("content-type", "application/json; charset=utf-8").end(result.encodePrettily());
        } else {
            System.out.println("Error: " + res.cause().getLocalizedMessage());
        }
    });
}

private void getActuadorWithAllParams(RoutingContext routingContext) {
    Integer id = Integer.parseInt(routingContext.request().getParam("idactuador"));

    mySqlClient.getConnection(connection -> {
        if (connection.succeeded()) {
            connection.result().preparedQuery("SELECT * FROM actuador WHERE idActuador = ?").execute(Tuple.of(id),
                    res -> {
                        if (res.succeeded()) {
                            RowSet<Row> resultSet = res.result();
                            JsonArray result = new JsonArray();
                            for (Row elem : resultSet) {
                                result.add(JsonObject.mapFrom(new ActuadorEntity(elem.getInteger("idvalue"),
                                        elem.getInteger("idactuador"), BigInteger.valueOf(elem.getInteger("timestamp")),
                                        elem.getDouble("value"), elem.getInteger("idgrupo"),
                                        elem.getInteger("idplaca"))));
                            }
                            System.out.println(result.toString());
                            routingContext.response().setStatusCode(200)
                                    .putHeader("content-type", "application/json; charset=utf-8")
                                    .end(result.encodePrettily());
                        } else {
                            System.out.println("Error: " + res.cause().getLocalizedMessage());
                        }
                        connection.result().close();
                    });
        } else {
            System.out.println(connection.cause().toString());
        }
    });
}
```

```java
private void getOneActuadorValue(RoutingContext routingContext) {
    int id = Integer.parseInt(routingContext.request().getParam("actuadorid"));
    mySqlClient.getConnection(connection -> {
        if (connection.succeeded()) {
            connection.result().preparedQuery("SELECT * FROM actuador WHERE idActuador = ?").execute(Tuple.of(id),
                    res -> {
                        if (res.succeeded()) {
                            RowSet<Row> resultSet = res.result();
                            JsonArray result = new JsonArray();
                            for (Row elem : resultSet) {
                                result.add(JsonObject.mapFrom(new ActuadorEntity(elem.getInteger("idvalue"),
                                        elem.getInteger("idactuador"), BigInteger.valueOf(elem.getInteger("timestamp")),
                                        elem.getDouble("value"), elem.getInteger("idgrupo"),
                                        elem.getInteger("idplaca"))));
                            }
                            System.out.println(result.toString());
                            routingContext.response().setStatusCode(200)
                                    .putHeader("content-type", "application/json; charset=utf-8")
                                    .end(result.encodePrettily());
                        } else {
                            System.out.println("Error: " + res.cause().getLocalizedMessage());
                        }
                        connection.result().close();
                    });
        } else {
            System.out.println(connection.cause().toString());
        }
    });
}

private void addOneActuador(RoutingContext routingContext) {
    ActuadorEntity actuador = new Gson().fromJson(routingContext.getBodyAsString(), ActuadorEntity.class);

    System.out.println(actuador.toString());
    mySqlClient
            .preparedQuery(
                    "INSERT INTO actuador (idActuador, timeStampActuador, valueActuador, idGrupo, idPlaca) VALUES (?, ?, ?, ?, ?);")
            .execute(Tuple.of(actuador.idactuador, actuador.timestamp, actuador.value, actuador.idgrupo, actuador.idplaca), res -> {
                if (res.succeeded()) {
                    routingContext.response().setStatusCode(201)
                            .putHeader("content-type", "application/json; charset=utf-8")
                            .end(new Gson().toJson(actuador));
                } else {
                    System.out.println("Error: " + res.cause().getLocalizedMessage());
                }
            });
}
```

```sql
CREATE TABLE actuador(
idValue INT NOT NULL AUTO_INCREMENT,
idActuador INT NOT NULL,
timeStampActuador BIGINT NOT NULL,
valueActuador INT NOT NULL,
idGrupo INT NOT NULL,
idPlaca INT NOT NULL,
PRIMARY KEY(idValue)
);

CREATE TABLE sensor(
idValue INT NOT NULL AUTO_INCREMENT,
idSensor INT NOT NULL,
timeStampSensor BIGINT NOT NULL,
valueTemp DOUBLE NOT NULL,
valueHum DOUBLE NOT NULL,
idGrupo INT NOT NULL,
idPlaca INT NOT NULL,
PRIMARY KEY(idValue)
);
```

Aquí podemos ver la creación de las tablas en la base de datos:

4.- Entregable 4

En este entregable he utilizado el Arduino.ide para hacer la conexión entre la placa, junto al sensor y el actuador, y la base de datos. Los datos recogidos por el sensor y el actuador son capaces de almacenarse en la base de datos mediante la petición POST. A continuación, muestro el código que he usado para hacer dicha conexión:

```ino
#include <WiFi.h>
#include <DHT.h>
#include <HTTPClient.h>
#include <ESP32Servo.h>  // Incluir la biblioteca ESP32Servo

// Definiciones de pines
#define DHTPIN 4         // Pin donde se conecta el DHT11
#define DHTTYPE DHT11    // DHT 11
#define SERVOPIN 18      // Pin donde se conecta el servomotor (en la ESP32)

// Credenciales de la red WiFi
const char* ssid = "Mj";          // Cambia por tu SSID
const char* password = "ComprateunputoWifi";  // Cambia por tu contraseña

// Inicializa el sensor DHT
DHT dht(DHTPIN, DHTTYPE);

// Inicializa el servomotor
Servo miServo;

// Umbrales de temperatura y humedad
const float umbralTemperatura = 30.0;  // Umbral de temperatura en grados Celsius
const float umbralHumedad = 70.0;      // Umbral de humedad en porcentaje

void setup() {
  Serial.begin(115200);
  dht.begin();
  miServo.attach(SERVOPIN);  // Adjuntar el servomotor al pin especificado

  // Conexión a la red WiFi
  WiFi.begin(ssid, password);
  Serial.print("Conectando a ");
  Serial.println(ssid);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("Conectado a la red WiFi");
  Serial.print("Dirección IP: ");
  Serial.println(WiFi.localIP());
}

void loop() {
  if ((WiFi.status() == WL_CONNECTED)) { // Verifica que esté conectado a la WiFi
    HTTPClient http;

    // Lee la temperatura y la humedad del DHT11
    float temperatura = dht.readTemperature();
    float humedad = dht.readHumidity();

    // Verifica si la lectura es válida
    if (isnan(temperatura) || isnan(humedad)) {
      Serial.println("Error al leer del DHT11!");
    } else {
      // Muestra la temperatura y la humedad en el monitor serial
      Serial.print("Temperatura: ");
      Serial.print(temperatura);
      Serial.print(" °C, Humedad: ");
      Serial.print(humedad);
      Serial.println(" %");

      // Verifica si se superan los umbrales y mueve el servomotor
      if (temperatura > umbralTemperatura || humedad > umbralHumedad) {
        miServo.write(90);  // Mover el servomotor a 90 grados
      } else {
        miServo.write(0);   // Regresar el servomotor a la posición inicial
      }

      // URL de la API local
      String serverName = "http://192.168.152.254:8080/api/sensor"; // Cambia a la IP y puerto correctos
      http.begin(serverName);
      http.addHeader("Content-Type", "application/json");

      // Cuerpo del POST en formato JSON
      String postData = "{\"idsensor\":1,\"timestamp\":";
      postData += String(time(NULL)); // Usa la hora actual
      postData += ",\"valueTemp\":";
      postData += String(temperatura);
      postData += ", \"valueHum\":";
      postData += String(humedad);
      postData += ", \"idgrupo\":1, \"idplaca\":1}";

      // Realiza la solicitud HTTP POST
      int httpResponseCode = http.POST(postData);

      // Imprime el código de respuesta
      if (httpResponseCode > 0) {
        String response = http.getString();
        Serial.println(httpResponseCode);
        Serial.println(response);
      } else {
        Serial.print("Error en la solicitud POST: ");
        Serial.println(httpResponseCode);
        Serial.println(http.errorToString(httpResponseCode).c_str());  // Mensaje de error más descriptivo
      }

      // Finaliza la conexión
      http.end();
    }
  }

  // Espera 10 segundos antes de la siguiente lectura
  delay(10000);
}
```

En este entregable, no he adjuntado nada del eclipse, debido a que no cambia nada con respecto al entregable anterior.

5.- Entregable 5

Hacemos los cambios necesarios para implementar el MQTT:

```
package es.us.lsi.dad;

import java.math.BigInteger;

public class RestServer extends AbstractVerticle {

    private static final int PLACA_1 = 1;
    private static final int PLACA_2 = 2;

    private double lastValueTemperaturaPlaca1 = 0.0;
    private double lastValueHumedadPlaca1 = 0.0;
    private boolean lastServoStatePlaca1 = false;

    private double lastValueTemperaturaPlaca2 = 0.0;
    private double lastValueHumedadPlaca2 = 0.0;
    private boolean lastServoStatePlaca2 = false;

    private MqttClient mqttClient;

    private void configureMqttClient() {
        MqttClientOptions options = new MqttClientOptions()
                .setClientId("restServer")
                .setUsername("mqtt").setPassword("mj")
                .setCleanSession(true);
        mqttClient = MqttClient.create(vertx, options);
        mqttClient.connect(1883, "192.168.229.254", s -> {
            if (s.succeeded()) {
                System.out.println("Connected to MQTT broker");
                mqttClient.subscribe("invernadero/sensor", MqttQoS.AT_LEAST_ONCE.value(), ar -> {
                    if (ar.succeeded()) {
                        System.out.println("Subscribed to topic invernadero/sensor");
                    } else {
                        System.out.println("Failed to subscribe to topic: " + ar.cause().getMessage());
                    }
                });

                mqttClient.subscribe("invernadero/actuador/placa1", MqttQoS.AT_LEAST_ONCE.value(), ar -> {
                    if (ar.succeeded()) {
                        System.out.println("Subscribed to topic invernadero/actuador/placa1");
                    } else {
                        System.out.println("Failed to subscribe to topic: " + ar.cause().getMessage());
                    }
                });
                mqttClient.subscribe("invernadero/actuador/placa2", MqttQoS.AT_LEAST_ONCE.value(), ar -> {
                    if (ar.succeeded()) {
                        System.out.println("Subscribed to topic invernadero/actuador/placa2");
                    } else {
                        System.out.println("Failed to subscribe to topic: " + ar.cause().getMessage());
                    }
                });
```

```
                System.out.println("Failed to connect to MQTT broker: " + s.cause().getMessage());
            }
        });

        mqttClient.publishHandler(message -> {
            String topic = message.topicName();
            String payload = message.payload().toString();
            System.out.println("Mensaje recibido en el tópico [" + topic + "]: " + payload);
            if (topic.equals("invernadero/sensor")) {
                // Procesa el mensaje recibido
                JsonObject json = new JsonObject(payload);
                SensorEntity sensor = new SensorEntity(
                        null,
                        json.getInteger("idsensor"),
                        BigInteger.valueOf(json.getLong("timestamp")),
                        json.getDouble("valueTemp"),
                        json.getDouble("valueHum"),
                        json.getInteger("idgrupo"),
                        json.getInteger("idplaca")
                );
                insertSensorData(sensor);
            }
        });
    }

    private void insertSensorData(SensorEntity sensor) {
        mySqlClient
            .preparedQuery("INSERT INTO sensor (idSensor, timeStampSensor, valueTemp, valueHum, idGrupo, idPlaca) VALUES (?, ?, ?, ?, ?, ?);")
            .execute(Tuple.of(sensor.idsensor, sensor.timestamp, sensor.valueTemp, sensor.valueHum, sensor.idgrupo, sensor.idplaca), res -> {
                if (res.succeeded()) {
                    System.out.println("Datos insertados correctamente en la base de datos.");

                    // Debugging: Mostrar valores para diagnóstico
                    System.out.println("Valor de sensor.valueTemp: " + sensor.valueTemp);
                    System.out.println("Valor de sensor.valueHum: " + sensor.valueHum);

                    if (sensor.idplaca == PLACA_1) {
                        handlePlaca1(sensor);
                    } else if (sensor.idplaca == PLACA_2) {
                        handlePlaca2(sensor);
                    }
                } else {
                    System.out.println("Error al insertar datos en la base de datos: " + res.cause().getLocalizedMessage());
                }
            });
    }

    private void handlePlaca1(SensorEntity sensor) {
```

```
private void handlePlaca1(SensorEntity sensor) {
    boolean currentServoState = (sensor.valueTemp > 35 || sensor.valueHum > 70);

    if (currentServoState != lastServoStatePlaca1) {
        String action = currentServoState ? "ServoOn" : "ServoOff";
        String topic = "invernadero/actuador/placa1";

        try {
            mqttClient.publish(topic, Buffer.buffer(action), MqttQoS.AT_LEAST_ONCE, false, false, ar -> {
                if (ar.succeeded()) {
                    insertActuadorData(1, action + " Placa 1", 1, 1);

                    System.out.println("Publicado " + action + " en " + topic);
                } else {
                    System.out.println("Error al publicar MQTT en " + topic + ": " + ar.cause().getMessage());
                }
            });
        } catch (Exception ex) {
            System.out.println("Excepción al publicar MQTT: " + ex.getMessage());
        }

        lastServoStatePlaca1 = currentServoState;
    }

    lastValueTemperaturaPlaca1 = sensor.valueTemp;
    lastValueHumedadPlaca1 = sensor.valueHum;
}


private void handlePlaca2(SensorEntity sensor) {
    boolean currentServoState = (sensor.valueTemp > 35 || sensor.valueHum > 70);

    if (currentServoState != lastServoStatePlaca2) {
        String action = currentServoState ? "ServoOn" : "ServoOff";
        String topic = "invernadero/actuador/placa2";

        try {
            mqttClient.publish(topic, Buffer.buffer(action), MqttQoS.AT_LEAST_ONCE, false, false, ar -> {
                if (ar.succeeded()) {
                    insertActuadorData(2, action + " Placa 2", 2, 2);

                    System.out.println("Publicado " + action + " en " + topic);
                } else {
                    System.out.println("Error al publicar MQTT en " + topic + ": " + ar.cause().getMessage());
                }
            });
        } catch (Exception ex) {
            System.out.println("Excepción al publicar MQTT: " + ex.getMessage());
        }
```

```
public static MySQLPool mySqlClient;

private Gson gson;

public void start(Promise<Void> startPromise) {
    MySQLConnectOptions connectOptions = new MySQLConnectOptions()
            .setPort(3306)
            .setHost("127.0.0.1")
            .setDatabase("proyectodaw")
            .setUser("root")
            .setPassword("root");

    PoolOptions poolOptions = new PoolOptions().setMaxSize(5);

    mySqlClient = MySQLPool.pool(vertx, connectOptions, poolOptions);

    mySqlClient.query("SELECT 1").execute(ar -> {
        if (ar.succeeded()) {
            System.out.println("Conectado a la base de datos exitosamente.");
            startPromise.complete();
        } else {
            System.out.println("Error al conectar a la base de datos: " + ar.cause().getMessage());
            startPromise.fail(ar.cause());
        }
    });

    gson = new GsonBuilder().setDateFormat("yyyy-MM-dd").create();

    Router router = Router.router(vertx);

    configureMqttClient();

    router.route("/api/sensor*").handler(BodyHandler.create());
    router.get("/api/sensor").handler(this::getSensorsWithAllParams);
    router.post("/api/sensor").handler(this::addOneSensor);
    router.get("/api/sensor/:sensorid").handler(this::getOneSensorValues);

    router.route("/api/actuador*").handler(BodyHandler.create());
    router.get("/api/actuador").handler(this::getActuadorsWithAllParams);
    router.post("/api/actuador").handler(this::addOneActuador);
    router.get("/api/actuador/:actuadorid").handler(this::getOneActuadorValue);

    vertx.createHttpServer().requestHandler(router).listen(8080, result -> {
        if (result.succeeded()) {
            System.out.println("Servidor HTTP corriendo en el puerto 8080");
            startPromise.complete();
        } else {
            System.out.println("Error al iniciar el servidor HTTP: " + result.cause().getMessage());
            startPromise.fail(result.cause());
        }
    });
}
```

```
private void insertActuadorData(int idActuador, String action, int idGrupo, int idPlaca) {
    int value = action.startsWith("ServoOn") ? 1 : 0;
    mySqlClient.preparedQuery(
            "INSERT INTO actuador (idActuador, timeStampActuador, valueActuador, idGrupo, idPlaca) VALUES (?, ?, ?, ?, ?);")
        .execute(Tuple.of(idActuador, System.currentTimeMillis() / 1000L, value, idGrupo, idPlaca), res -> {
            if (res.succeeded()) {
                System.out.println("Datos del actuador insertados correctamente en la base de datos. Acción: " + action);
            } else {
                System.out.println("Error al insertar datos del actuador en la base de datos: " + res.cause().getMessage());
            }
        });
}
```

```
private void getSensorsWithAllParams(RoutingContext routingContext) {
    Integer id = Integer.parseInt(routingContext.request().getParam("idsensor"));

    mySqlClient.getConnection(connection -> {
        if (connection.succeeded()) {
            connection.result().preparedQuery("SELECT * FROM sensor WHERE idSensor = ?").execute(Tuple.of(id),
                    res -> {
                        if (res.succeeded()) {
                            RowSet<Row> resultSet = res.result();
                            JsonArray result = new JsonArray();
                            for (Row elem : resultSet) {
                                result.add(JsonObject.mapFrom(new SensorEntity(elem.getInteger("idvalue"),
                                        elem.getInteger("idsensor"), BigInteger.valueOf(elem.getInteger("timestamp")), elem.getDouble("valueTemp"),
                                        elem.getDouble("valueHum"), elem.getInteger("idgrupo"), elem.getInteger("idplaca"))));
                            }
                            System.out.println(result.toString());
                            routingContext.response().setStatusCode(200)
                                    .putHeader("content-type", "application/json; charset=utf-8")
                                    .end(result.encodePrettily());
                        } else {
                            System.out.println("Error: " + res.cause().getLocalizedMessage());
                        }
                        connection.result().close();
                    });
        } else {
            System.out.println(connection.cause().toString());
        }
    });
}

private void getOneSensorValues(RoutingContext routingContext) {
    int id = Integer.parseInt(routingContext.request().getParam("sensorid"));

    mySqlClient.getConnection(connection -> {
        if (connection.succeeded()) {
            connection.result().preparedQuery("SELECT * FROM sensor WHERE idSensor = ?").execute(Tuple.of(id),
                    res -> {
                        if (res.succeeded()) {
                            RowSet<Row> resultSet = res.result();
                            List<SensorEntity> result = new ArrayList<SensorEntity>();
                            for (Row elem : resultSet) {
                                result.add(new SensorEntity(elem.getInteger("idvalue"),
                                        elem.getInteger("idsensor"), BigInteger.valueOf(elem.getInteger("timestamp")), elem.getDouble("valueTemp"),
                                        elem.getDouble("valueHum"), elem.getInteger("idgrupo"), elem.getInteger("idplaca")));
                            }
                            routingContext.response().setStatusCode(200)
                                    .putHeader("content-type", "application/json; charset=utf-8")
                                    .end(gson.toJson(result));
                        } else {
                            System.out.println("Error: " + res.cause().getLocalizedMessage());
                        }
                        connection.result().close();
                    });
        } else {
            System.out.println(connection.cause().toString());
        }
    });
```

Placa 1

EntregableJulio.ino

```
#include <WiFi.h>
#include <DHT.h>
#include <PubSubClient.h>
#include <ESP32Servo.h>  // Incluir la biblioteca ESP32Servo

// Definiciones de pines
#define DHTPIN 4          // Pin donde se conecta el DHT11
#define DHTTYPE DHT11     // DHT 11
#define SERVOPIN 18       // Pin donde se conecta el servomotor (en la ESP32)

// Credenciales de la red WiFi
const char* ssid = "Mj";
const char* password = "CompprateunputoWifi";

// Credenciales del MQTT Broker
const char* mqtt_server = "192.168.229.254";
const char* mqtt_user = "mqtt";
const char* mqtt_password = "mj";

// Inicializa el sensor DHT
DHT dht(DHTPIN, DHTTYPE);

// Inicializa el servomotor
Servo miServo;

// Umbrales de temperatura y humedad
const float umbralTemperatura = 35.0;
const float umbralHumedad = 70.0;

WiFiClient espClient;
PubSubClient client(espClient);

void setup() {
  Serial.begin(115200);
  dht.begin();
  miServo.attach(SERVOPIN);  // Adjuntar el servomotor al pin especificado

  // Conexión a la red WiFi
  Serial.println("Conectando a la red WiFi...");
  WiFi.begin(ssid, password);

  unsigned long startAttemptTime = millis();

  // Intentar conectar durante 30 segundos
  while (WiFi.status() != WL_CONNECTED && millis() - startAttemptTime < 30000) {
    delay(1000);
    Serial.print(".");
  }

  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("\nNo se pudo conectar a la red WiFi");
  } else {
    Serial.println("\nConectado a la red WiFi");
    Serial.print("Dirección IP: ");
    Serial.println(WiFi.localIP());
  }

  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);

  reconnect();
}
void reconnect() {
  while (!client.connected()) {
    Serial.print("Conectando al broker MQTT...");
    if (client.connect("ESP32Client", mqtt_user, mqtt_password)) {
      Serial.println("Conectado al broker MQTT");
      client.subscribe("invernadero/actuador"); // Tópico de control para
    } else {
      Serial.print("Falló la conexión, rc=");
      Serial.print(client.state());
      delay(5000);
    }
  }
}
void callback(char* topic, byte* payload, unsigned int length) {
  String messageTemp;
  for (unsigned int i = 0; i < length; i++) {
    messageTemp += (char)payload[i];
  }
  Serial.print("Mensaje recibido [");
  Serial.print(topic);
  Serial.print("]: ");
  Serial.println(messageTemp);

  if (String(topic) == "invernadero/actuador/placa1") {
    if (messageTemp.startsWith("ServoOn")) {
      miServo.write(90);  // Mover el servomotor a 90 grados
      Serial.println("Servo encendido");
    } else if (messageTemp.startsWith("ServoOff")) {
      miServo.write(0);   // Regresar el servomotor a la posición inicial
      Serial.println("Servo apagado");
    }
  }
}
void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  float temperatura = dht.readTemperature();
  float humedad = dht.readHumidity();

  if (isnan(temperatura) || isnan(humedad)) {
    Serial.println("Error al leer del DHT11!");
    return;
  }

  Serial.print("Temperatura: ");
  Serial.print(temperatura);
  Serial.print(" °C, Humedad: ");
  Serial.print(humedad);
  Serial.println(" %");

  String payload = "{\"idsensor\":1,\"timestamp\":";
  payload += String(time(NULL));
  payload += ",\"valueTemp\":";
  payload += String(temperatura);
  payload += ", \"valueHum\":";
  payload += String(humedad);
  payload += ", \"idgrupo\":1, \"idplaca\":1}";

  client.publish("invernadero/sensor", (char*) payload.c_str());
  delay(10000); // Ajusta este delay según sea necesario
}
```

Placa 2

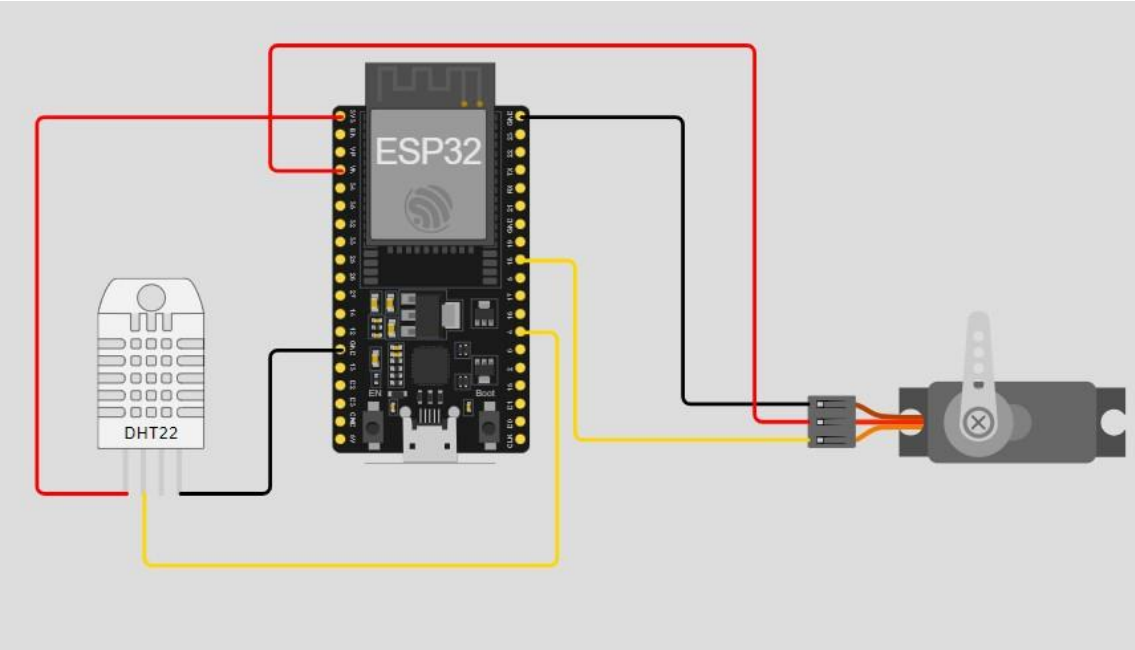
rduinoJulio2.ino

```
#include <WiFi.h>
#include <DHT.h>
#include <PubSubClient.h>
#include <ESP32Servo.h>  // Incluir la biblioteca ESP32Servo

// Definiciones de pines
#define DHTPIN 4          // Pin donde se conecta el DHT11
#define DHTTYPE DHT11     // DHT 11
#define SERVOPIN 18       // Pin donde se conecta el servomotor (en la ESP32)

// Credenciales de la red WiFi
const char* ssid = "Mj";
const char* password = "CompprateunputoWifi";

// Credenciales del MQTT Broker
const char* mqtt_server = "192.168.229.254";
const char* mqtt_user = "mqtt";
const char* mqtt_password = "mj";

// Inicializa el sensor DHT
DHT dht(DHTPIN, DHTTYPE);

// Inicializa el servomotor
Servo miServo;

// Umbrales de temperatura y humedad
const float umbralTemperatura = 35.0;
const float umbralHumedad = 70.0;

WiFiClient espClient;
PubSubClient client(espClient);

void setup() {
  Serial.begin(115200);
  dht.begin();
  miServo.attach(SERVOPIN);  // Adjuntar el servomotor al pin especificado

  // Conexión a la red WiFi
  Serial.println("Conectando a la red WiFi...");
  WiFi.begin(ssid, password);

  unsigned long startAttemptTime = millis();

  // Intentar conectar durante 30 segundos
  while (WiFi.status() != WL_CONNECTED && millis() - startAttemptTime < 30000) {
    delay(1000);
    Serial.print(".");
  }

  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("\nNo se pudo conectar a la red WiFi");
  } else {
    Serial.println("\nConectado a la red WiFi");
    Serial.print("Dirección IP: ");
    Serial.println(WiFi.localIP());
  }

  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);

  reconnect();
}

void reconnect() {
  while (!client.connected()) {
    Serial.print("Conectando al broker MQTT...");
    if (client.connect("ESP32Client", mqtt_user, mqtt_password)) {
      Serial.println("Conectado al broker MQTT");
      client.subscribe("invernadero/actuador"); // Tópico de control para el
    } else {
      Serial.print("Falló la conexión, rc=");
      Serial.print(client.state());
      delay(5000);
    }
  }
}

void callback(char* topic, byte* payload, unsigned int length) {
  String messageTemp;
  for (unsigned int i = 0; i < length; i++) {
    messageTemp += (char)payload[i];
  }
  Serial.print("Mensaje recibido [");
  Serial.print(topic);
  Serial.print("]: ");
  Serial.println(messageTemp);

  if (String(topic) == "invernadero/actuador/placa2") {
    if (messageTemp.startsWith("ServoOn")) {
      miServo.write(90);  // Mover el servomotor a 90 grados
      Serial.println("Servo encendido");
    } else if (messageTemp.startsWith("ServoOff")) {
      miServo.write(0);   // Regresar el servomotor a la posición inicial
      Serial.println("Servo apagado");
    }
  }
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  float temperatura = dht.readTemperature();
  float humedad = dht.readHumidity();

  if (isnan(temperatura) || isnan(humedad)) {
    Serial.println("Error al leer del DHT11!");
    return;
  }

  Serial.print("Temperatura: ");
  Serial.print(temperatura);
  Serial.print(" °C, Humedad: ");
  Serial.print(humedad);
  Serial.println(" %");

  String payload = "{\"idsensor\":2,\"timestamp\":";
  payload += String(time(NULL));
  payload += ",\"valueTemp\":";
  payload += String(temperatura);
  payload += ", \"valueHum\":";
  payload += String(humedad);
  payload += ", \"idgrupo\":2, \"idplaca\":2}";

  client.publish("invernadero/sensor", (char*) payload.c_str());
  delay(10000); // Ajusta este delay según sea necesario
}
```

6.- Materiales utilizados y montaje completo

Los materiales que he usado han sido:

- 2 placas ESP32 WROOM 32
- 2 servo motorres SG90
- 2 sensores de humedad y temperatura DHT11



En la foto adjunta aparece un DHT22, porque el programa usado no proporciona un DHT11, el montaje es el mismo cambiando el sensor. Y solo un montaje ya que los dos son el mismo.

7.- Problemas encontrados

En este nuevo entregable, me he encontrado con un problema de última hora y es que el día de antes de la entrega, todo el programa funcionaba a la perfección y la mañana del día de la entrega, dejó de funcionar y he tenido que estar trabajando todo el día para reparar errores que me han llevado una semana encontrar y solucionarlos. Al final, todo ha salido muy bien cómo se puede observar en el vídeo de presentación del proyecto final.

https://youtu.be/cbs52b87orU

8.- Conclusión

Este trabajo está pensado para una posible implementación en un invernadero, como ya expliqué en la presentación. Con ayuda del DHT11 podemos medir la temperatura y la humedad del entorno en el que nos encontramos y el servo nos ayudaría a mantenernos en los parámetros predefinidos, ya sea abriendo ventanas o activando un ventilador.