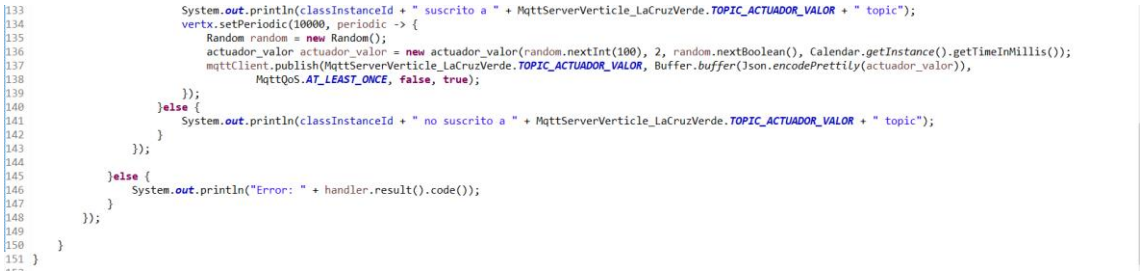
Explicación MQT

Metodo servidor:

```java
package ProyectoDad.LaCruzVerde;

import java.util.ArrayList;

public class MqttServerVerticle_LaCruzVerde extends AbstractVerticle{

    public static final String TOPIC_ACTUADOR = "actuador";
    public static final String TOPIC_ACTUADOR_VALOR = "actuador_valor";
    public static final String TOPIC_SENSOR = "sensor";
    public static final String TOPIC_SENSOR_VALOR = "sensor_valor";
    public static final String TOPIC_DISPOSITIVO = "dispositivo";
    public static final String TOPIC_PLANTA = "planta";

    private static final SetMultimap<String, MqttEndpoint> clients = LinkedHashMultimap.create();

    public void start (Promise<Void> promise) {

        MqttServerOptions options = new MqttServerOptions();
        options.setPort(1885);
        options.setClientAuth(ClientAuth.REQUIRED);
        MqttServer mqttServer = MqttServer.create(vertx, options);
        init(mqttServer);

    }

    public void init(MqttServer mqttSever) {
        mqttSever.endpointHandler(endpoint -> {
            System.out.println("Cliente MQTT [" + endpoint.clientIdentifier() +
                    "] solicita conexion. Sesion limpia = " + endpoint.isCleanSession());
            if(endpoint.auth().getUsername().contentEquals("mqttbroker") && endpoint.auth().getPassword().contentEquals("mqttbrokerpass")) {
                endpoint.accept();
                handleSubscription(endpoint);
                handleUnsubscription(endpoint);
                publishHandle(endpoint);
                handleClientDisconnect(endpoint);
            }else {
                endpoint.reject(MqttConnectReturnCode.CONNECTION_REFUSED_BAD_USER_NAME_OR_PASSWORD);
            }


        }).listen(ar -> {
            if(ar.succeeded()) {
                System.out.println("El servidor MQTT se esta ejecutando en el puerto: " + ar.result().actualPort());
            }else {
                System.out.println("Error al iniciar servidor MQTT");
                ar.cause().printStackTrace();
            }
        });
    }

    private void handleSubscription(MqttEndpoint endpoint) {
        endpoint.subscribeHandler(subscribe -> {
            List<MqttQoS> grantedQoSLevels = new ArrayList<>();
            for(MqttTopicSubscription s : subscribe.topicSubscriptions()) {
                System.out.println("Suscripcion de " + s.topicName() + " con QoS " + s.qualityOfService());
                grantedQoSLevels.add(s.qualityOfService());
                clients.put(s.topicName(), endpoint);
            }
            endpoint.subscribeAcknowledge(subscribe.messageId(), grantedQoSLevels);
        });
    }

    private void handleUnsubscription(MqttEndpoint endpoint) {
        endpoint.unsubscribeHandler(unsuscribe -> {
            for(String t : unsuscribe.topics()) {
                System.out.println("Desuscripcion de " + t);
                clients.remove(t, endpoint);
            }
            endpoint.unsubscribeAcknowledge(unsuscribe.messageId());
        });
    }

    private void publishHandle(MqttEndpoint endpoint) {
        endpoint.publishHandler(message -> {
            if(message.qosLevel() == MqttQoS.AT_LEAST_ONCE) {
                String topicName = message.topicName();
                System.out.println("Nuevo mensaje publicado en " + topicName);
                for(MqttEndpoint subscribed : clients.get(topicName)) {


                    subscribed.publish(message.topicName(), message.payload(), message.qosLevel(), message.isDup(), message.isRetain());
                }
                endpoint.publishAcknowledge(message.messageId());
            }else if(message.qosLevel() == MqttQoS.EXACTLY_ONCE){
                endpoint.publishRelease(message.messageId());
            }
        }).publishReleaseHandler(messageId -> {
            endpoint.publishComplete(messageId);
        });
    }

    private void handleClientDisconnect(MqttEndpoint endpoint) {
        endpoint.disconnectHandler(h -> {
            System.out.println("El cliente remoto a cerrado la conexion");
        });
    }

}
```

Metodo cliente:

```java
package ProyectoDad.LaCruzVerde;

import java.util.Calendar;

public class MqttClientVerticle_LaCruzVerde extends AbstractVerticle{

    private String classInstanceId;

    public void start (Promise<Void> promise) {
        MqttClientOptions mqttClientOptions = new MqttClientOptions();
        mqttClientOptions.setAutoKeepAlive(true);
        mqttClientOptions.setAutoGeneratedClientId(false);
        mqttClientOptions.setClientId(classInstanceId);
        mqttClientOptions.setConnectTimeout(10000);
        mqttClientOptions.setKeepAliveTimeSeconds(10);
        mqttClientOptions.setReconnectAttempts(10);
        mqttClientOptions.setReconnectInterval(5000);
        mqttClientOptions.setUsername("mqttbroker");
        mqttClientOptions.setPassword("mqttbrokerpass");
        MqttClient mqttClient = new MqttClientImpl(vertx, mqttClientOptions);

        mqttClient.publishHandler(messageReceivedHandler -> {
            System.out.println(messageReceivedHandler.payload().toString());
        });

        mqttClient.connect(1885, "localhost", handler -> {
            if(handler.result().code() == MqttConnectReturnCode.CONNECTION_ACCEPTED) {

                //TOPIC PLANTA --> Desde la app web deberiamos poder introducir en la bbdd una nueva planta para cuidar
                //                 la idea es cambiar que sea periodicamente para que sea simplemente cuando se lo mandemos desde
                //                 la app
                mqttClient.subscribe(MqttServerVerticle_LaCruzVerde.TOPIC_PLANTA, MqttQoS.AT_LEAST_ONCE.value(), handlerSubscribe -> {
                    if(handlerSubscribe.succeeded()) {
                        System.out.println(classInstanceId + " suscrito a " + MqttServerVerticle_LaCruzVerde.TOPIC_PLANTA + " topic");
                        vertx.setPeriodic(10000, periodic -> {
                            planta planta = new planta(1, "Aloevera", 22, 25, 5);
                            mqttClient.publish(MqttServerVerticle_LaCruzVerde.TOPIC_PLANTA, Buffer.buffer(Json.encodePrettily(planta)),
                                    MqttQoS.AT_LEAST_ONCE, false, true);


                        });
                    }else {
                        System.out.println(classInstanceId + " no suscrito a " + MqttServerVerticle_LaCruzVerde.TOPIC_PLANTA + " topic");
                    }
                });

                //TOPIC DISPOSITIVO --> cuando se conecte un nuevo dispositivo, deberiamos poder introducirle los parametros que necesita
                //                      para cuidar la planta, este canal introduce en la bbdd los parametros de este nuevo dispositivo
                //                      La idea es cambiar que sea periodicamente para que desde la app asignemos unn dispositivo a una
                //                      planta y lo pongamos en marcha

                mqttClient.subscribe(MqttServerVerticle_LaCruzVerde.TOPIC_DISPOSITIVO, MqttQoS.AT_LEAST_ONCE.value(), handlerSubscribe -> {
                    if(handlerSubscribe.succeeded()) {
                        System.out.println(classInstanceId + " suscrito a " + MqttServerVerticle_LaCruzVerde.TOPIC_DISPOSITIVO + " topic");
                        vertx.setPeriodic(10000, periodic -> {
                            Random random = new Random();
                            dispositivo dispositivo = new dispositivo(random.nextInt(100), random.nextInt(200)+"."+random.nextInt(200)+"."+random.nextInt(200)+"."+random.nextInt(
                                    "CruzVerde_Aloevera", 1, Calendar.getInstance().getTimeInMillis());
                            mqttClient.publish(MqttServerVerticle_LaCruzVerde.TOPIC_DISPOSITIVO, Buffer.buffer(Json.encodePrettily(dispositivo)),
                                    MqttQoS.AT_LEAST_ONCE, false, true);
                        });
                    }else {
                        System.out.println(classInstanceId + " no suscrito a " + MqttServerVerticle_LaCruzVerde.TOPIC_DISPOSITIVO + " topic");
                    }
                });

                //TOPIC SENSOR --> introduce los datos de un sensor en la bbdd
                //                 La idea es cambiar que sea periodicamente para que se añadan los sensores de un dispositivo,
                //                 unicamente una vez para cada dispositivo
                mqttClient.subscribe(MqttServerVerticle_LaCruzVerde.TOPIC_SENSOR, MqttQoS.AT_LEAST_ONCE.value(), handlerSubscribe -> {
                    if(handlerSubscribe.succeeded()) {
                        System.out.println(classInstanceId + " suscrito a " + MqttServerVerticle_LaCruzVerde.TOPIC_SENSOR + " topic");
                        vertx.setPeriodic(10000, periodic -> {
                            sensor sensor = new sensor(7, "temp_amb", "temp_amb_aloevera", 1);
                            mqttClient.publish(MqttServerVerticle_LaCruzVerde.TOPIC_SENSOR, Buffer.buffer(Json.encodePrettily(sensor)),
                                    MqttQoS.AT_LEAST_ONCE, false, true);
                        });
                    }else {


                        System.out.println(classInstanceId + " no suscrito a " + MqttServerVerticle_LaCruzVerde.TOPIC_SENSOR + " topic");
                    }
                });

                //TOPIC SENSOR_VALOR --> introduce las lecturas de un sensor en la bbdd cada cierto tiempo
                mqttClient.subscribe(MqttServerVerticle_LaCruzVerde.TOPIC_SENSOR_VALOR, MqttQoS.AT_LEAST_ONCE.value(), handlerSubscribe -> {
                    if(handlerSubscribe.succeeded()) {
                        System.out.println(classInstanceId + " suscrito a " + MqttServerVerticle_LaCruzVerde.TOPIC_SENSOR_VALOR + " topic");
                        vertx.setPeriodic(10000, periodic -> {
                            Random random = new Random();
                            sensor_valor sensor_valor = new sensor_valor(random.nextInt(100), 7, 30 + random.nextInt(8), random.nextInt(3), Calendar.getInstance().getTimeInMillis
                            mqttClient.publish(MqttServerVerticle_LaCruzVerde.TOPIC_SENSOR_VALOR, Buffer.buffer(Json.encodePrettily(sensor_valor)),
                                    MqttQoS.AT_LEAST_ONCE, false, true);
                        });
                    }else {
                        System.out.println(classInstanceId + " no suscrito a " + MqttServerVerticle_LaCruzVerde.TOPIC_SENSOR_VALOR + " topic");
                    }
                });

                //TOPIC ACTUADOR --> introduce los datos de un actuador en la bbdd
                //                   La idea es cambiar que sea periodicamente para que se añadan los sensores de un dispositivo,
                //                   unicamente una vez para cada dispositivo
                mqttClient.subscribe(MqttServerVerticle_LaCruzVerde.TOPIC_ACTUADOR, MqttQoS.AT_LEAST_ONCE.value(), handlerSubscribe -> {
                    if(handlerSubscribe.succeeded()) {
                        System.out.println(classInstanceId + " suscrito a " + MqttServerVerticle_LaCruzVerde.TOPIC_ACTUADOR + " topic");
                        vertx.setPeriodic(10000, periodic -> {
                            actuador actuador = new actuador(2, "luz", "luz aloevera", 1);
                            mqttClient.publish(MqttServerVerticle_LaCruzVerde.TOPIC_ACTUADOR, Buffer.buffer(Json.encodePrettily(actuador)),
                                    MqttQoS.AT_LEAST_ONCE, false, true);
                        });
                    }else {
                        System.out.println(classInstanceId + " no suscrito a " + MqttServerVerticle_LaCruzVerde.TOPIC_ACTUADOR + " topic");
                    }
                });

                //TOPIC ACTUADOR_VALOR --> introduce si esta funcionando un actuador en la bbdd cada cierto tiempo
                mqttClient.subscribe(MqttServerVerticle_LaCruzVerde.TOPIC_ACTUADOR_VALOR, MqttQoS.AT_LEAST_ONCE.value(), handlerSubscribe -> {
                    if(handlerSubscribe.succeeded()) {
```



```
                    System.out.println(classInstanceId + " suscrito a " + MqttServerVerticle_LaCruzVerde.TOPIC_ACTUADOR_VALOR + " topic");
                    vertx.setPeriodic(10000, periodic -> {
                        Random random = new Random();
                        actuador_valor actuador_valor = new actuador_valor(random.nextInt(100), 2, random.nextBoolean(), Calendar.getInstance().getTimeInMillis());
                        mqttClient.publish(MqttServerVerticle_LaCruzVerde.TOPIC_ACTUADOR_VALOR, Buffer.buffer(Json.encodePrettily(actuador_valor)),
                                MqttQoS.AT_LEAST_ONCE, false, true);
                    });
                }else {
                    System.out.println(classInstanceId + " no suscrito a " + MqttServerVerticle_LaCruzVerde.TOPIC_ACTUADOR_VALOR + " topic");
                }
            });

        }else {
            System.out.println("Error: " + handler.result().code());
        }
    });

    }
}
```

Entidad sensor_valor:

Da información al servidor sobre la entidad sensor_valor. Proporciona el siguiente Json:

{

“id_sensor_valor” = 4,

"id_sensor" : 1,

"valor" : 40,

"precisión_valor" : 3,

"tiempo" : 1234234562

}

Entidad sensor:

Da información al servidor sobre la entidad sensor. Proporciona el siguiente Json:

{

“id_sensor” = 2,

"tipo" : "temp_amb",

"nombre" : "temp_amb_tomatera",

"id_dispositivo" : 3

}

Entidad dispositivo:

Da información al servidor sobre la entidad dispositivo. Proporciona el siguiente Json:

{

“id_dispositivo” = 1,

"ip" : "192.101.5.5",

"nombre" : "CruzVerde_Tomatera",

"id_planta" : 4,

"tiempoInicial" : 12345754323

}

Entidad actuador:

Da información al servidor sobre la entidad actuador. Proporciona el siguiente Json:

{

“id_actuador” = 2,

“tipo” = “bomba”,

“nombre” = “bomba_aloevera”,

“id_dispositivo” = 1

}

Entidad actuador_valor:

Da información al servidor sobre la entidad actuador_valor. Proporciona el siguiente Json:

{

“id_actuador_valor” = 2,

"id_actuador" : 1,

“on” = true,

"tiempo" : 1234234562

}

Entidad planta:

Da información al servidor sobre la entidad planta. Proporciona el siguiente Json:

{

"nombre_planta" : "Tomatera",

"temp_amb_planta" : 32,

"humed_tierra_planta" : 40,

"humed_amb_planta" : 35

}

Captura consola:

<terminated> New_configuration [Java Application] C:\Program Files\Java\jre1.8.0_251\bin\javaw.exe (24 abr. 2020 19:27:26)

```
Cliente MQTT [093a9b80-c970-443b-8ec8-c9e3b32d4e17] solicita conexion. Sesion limpia = true
null suscrito a planta topic
null suscrito a dispositivo topic
null suscrito a sensor topic
null suscrito a sensor_valor topic
null suscrito a actuador topic
null suscrito a actuador_valor topic
Suscripcion de planta con QoS AT_LEAST_ONCE
Suscripcion de dispositivo con QoS AT_LEAST_ONCE
Suscripcion de sensor con QoS AT_LEAST_ONCE
Suscripcion de sensor_valor con QoS AT_LEAST_ONCE
Suscripcion de actuador con QoS AT_LEAST_ONCE
Suscripcion de actuador_valor con QoS AT_LEAST_ONCE
Nuevo mensaje publicado en planta
Nuevo mensaje publicado en dispositivo
Nuevo mensaje publicado en sensor
Nuevo mensaje publicado en sensor_valor
Nuevo mensaje publicado en actuador
Nuevo mensaje publicado en actuador_valor
{
  "id_planta" : 1,
  "nombre_planta" : "Aloevera",
  "temp_amb_planta" : 22.0,
  "humed_tierra_planta" : 25.0,
  "humed_amb_planta" : 5.0
}
{
  "id_dispositivo" : 41,
  "ip" : "140.35.45.64",
  "nombre" : "CruzVerde_Aloevera",
  "id_planta" : 1,
  "tiempoInicial" : 1587749257810
}
{
  "id_sensor" : 7,
```