package com.dai.webServer.Mqtt; import java.sql.PreparedStatement; import java.sql.SQLException;

import com.dai.db.AnalyticsDB; import com.dai.webServer.Conexao.Conexao; import com.dai.webServer.Conexao.Email; import com.dai.webServer.Mqtt.*; import org.json.simple.parser.ParseException; import org.apache.commons.text.RandomStringGenerator;*
/***

import org.eclipse.paho.client.mqttv3.IMqttDeliveryToken; import org.eclipse.paho.client.mqttv3.MqttCallback; import org.eclipse.paho.client.mqttv3.MqttClient; import org.eclipse.paho.client.mqttv3.MqttConnectOptions; import org.eclipse.paho.client.mqttv3.MqttException; import org.eclipse.paho.client.mqttv3.MqttMessage; import org.springframework.beans.factory.annotation.Autowired; import java.util.Random; import static org.apache.commons.text.CharacterPredicates.LETTERS;

public class AlarmReceive implements MqttCallback {

```
@Autowired

/** The broker url. */
private static final String brokerUrl = "tcp://alvesvitor.ddns.net:80";

/** The client id. */
Random rand = new Random();

/** The topic. */
public static final String topic = "alarm/#";
//private AnalyticsDB db = new AnalyticsDB();

public String random() {
RandomStringGenerator generator = new RandomStringGenerator.Builder()
        .withinRange('0', 'z') .filteredBy(LETTERS)
        .build();
String a = generator.toString();
return a;
}


public void subscribe(String clientId) {
    try {
String a = random();
        MqttClient sampleClient = new MqttClient(brokerUrl, a);
        MqttConnectOptions connOpts = new MqttConnectOptions();
        connOpts.setMqttVersion(MqttConnectOptions.MQTT_VERSION_3_1_1);
        connOpts.setUserName("dai");
        String password = "12345678";
        connOpts.setPassword(password.toCharArray());

        connOpts.setCleanSession(true);

        System.out.println("checking");

        System.out.println("Mqtt Connecting to broker: " + brokerUrl);
        sampleClient.connect(connOpts);
        System.out.println("Mqtt Connected");

        sampleClient.setCallback(this);
        sampleClient.subscribe(topic);

        System.out.println("Subscribed");
        System.out.println("Listening");

    } catch (MqttException me) {

        System.out.println("Mqtt reason " + me.getReasonCode());
        System.out.println("Mqtt msg " + me.getMessage());
        System.out.println("Mqtt loc " + me.getLocalizedMessage());
        System.out.println("Mqtt cause " + me.getCause());
        System.out.println("Mqtt excep " + me);
    }
}




public void connectionLost(Throwable arg0) {
    try {
        wait(1000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
  this.subscribe(topic);
}
```

```
public void deliveryComplete(IMqttDeliveryToken arg0) {

}

public void messageArrived(String topic, MqttMessage message) throws Exception  {

    System.out.println("Mqtt topic : " + topic);

    System.out.println("Mqtt msg : " + message.toString());
    byte[] hey = message.getPayload();
    String str = new String(hey, "UTF-8"); // for UTF-8 encoding
    insert(str, topic);
}


public void insert(String message , String topic) throws ParseException, SQLException  {

    System.out.println(message);


    String correctedTopic = topic.substring(topic.lastIndexOf('/')+1);
    AnalyticsDB db2 = new AnalyticsDB();

    AnalyticsDB email2 = new AnalyticsDB();
    System.out.println("It's here now");
    System.out.println(correctedTopic);
    String outcome = db2.armed(correctedTopic);


    System.out.println("outcome" + outcome);
    String responseTopic = "response/" + correctedTopic;






    if (outcome != null && outcome.equals("false")){
    }else if(outcome != null && outcome.equals("true")){





        String email = email2.readEmail(correctedTopic);
        System.out.println(email);
        Email b = new Email();

        b.sendEmail(email, "<img src='https://i.imgur.com/dbsHhsJ.png' alt='Por favor faça


    }else{

    System.out.println("fodeu");
    }

}

}
```