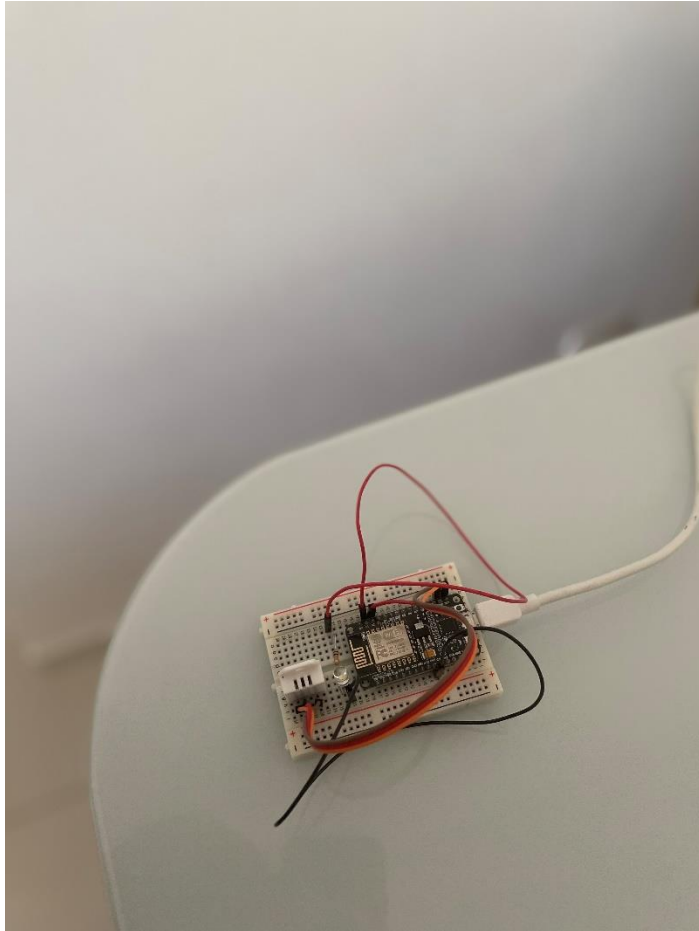
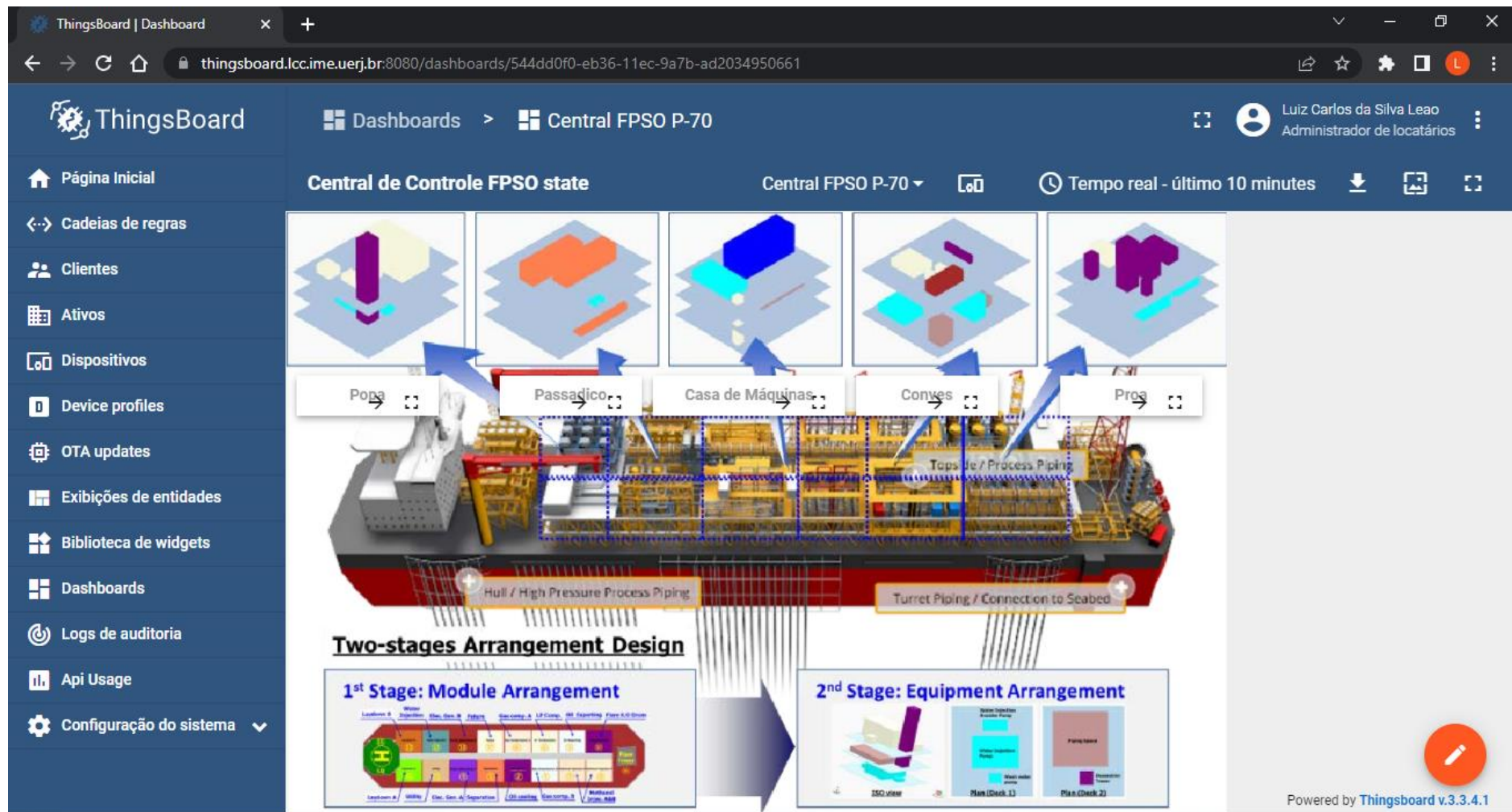


Resultados do Trabalho Prático no ThingsBoard do aluno Luiz Carlos – Data de entrega: 08/07/2022 – Prof. Alexandre Sztajnberg

a) Foto do dispositivo NodeMCU + o sensor DHT 22 + o LED para acendimento via RPC:



b) Dashboard – Pannel principal (Áreas da FPSO P-70: Popa, Passadiço, Casa de Máquinas, Convés e a Proa – cada uma com 3 dispositivos)



c) Dashboard – Popa 1 (com a apresentação das médias e média móvel)



d) Dashboard Popa

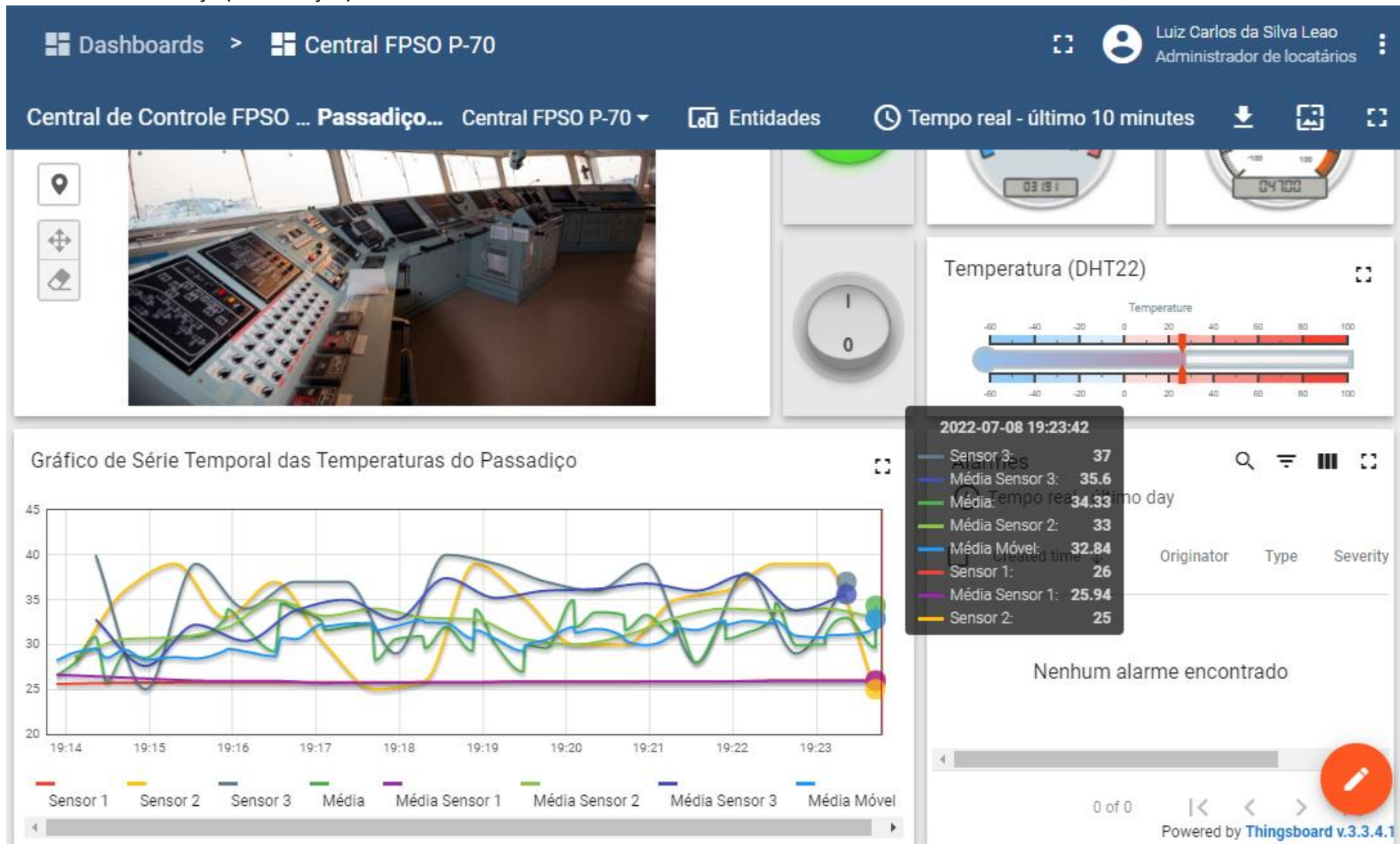




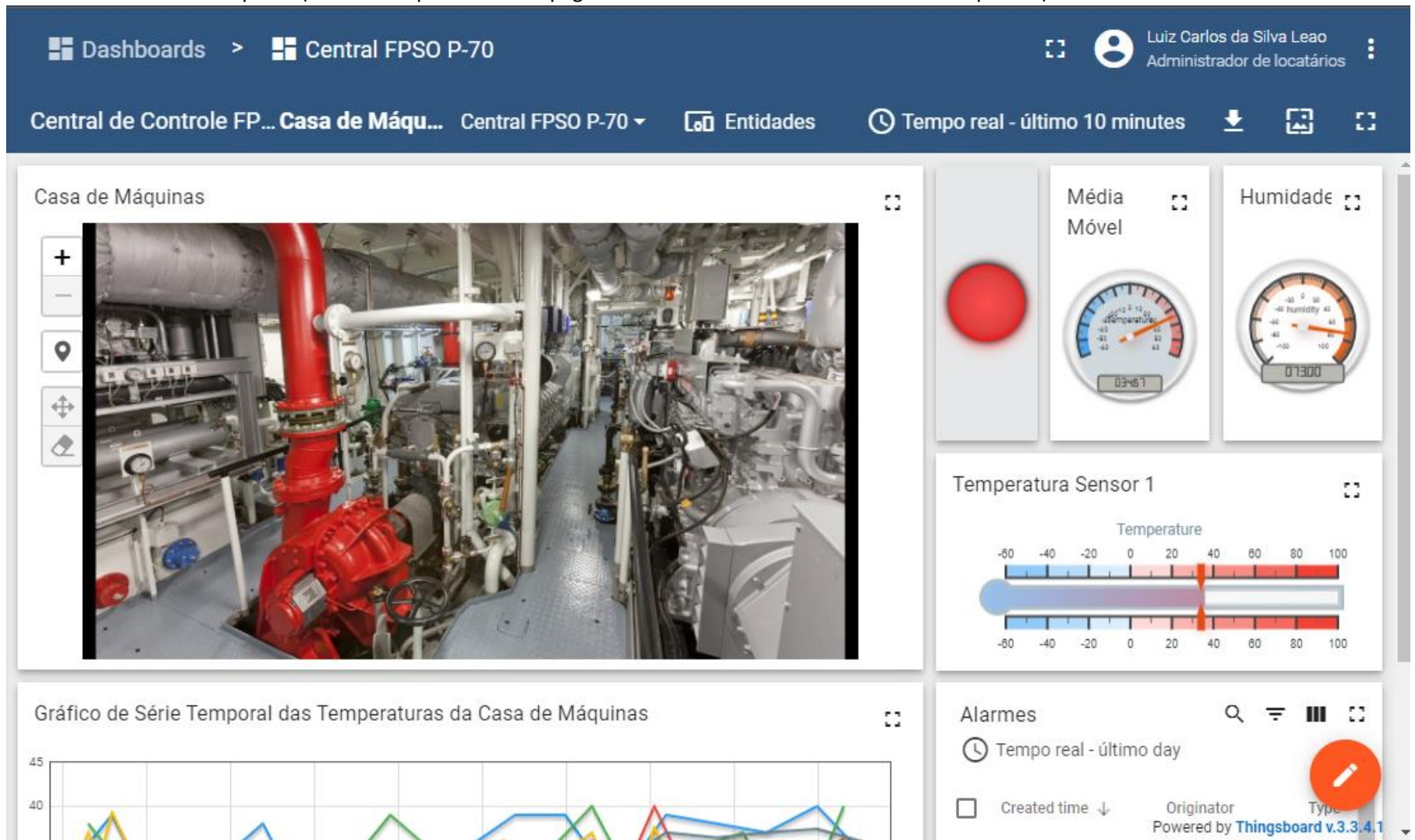
e) Dashboard – Passadiço (com o atuador que liga e desliga via RPC o LED do dispositivo)



f) Dashboard – Passadiço (continuação):

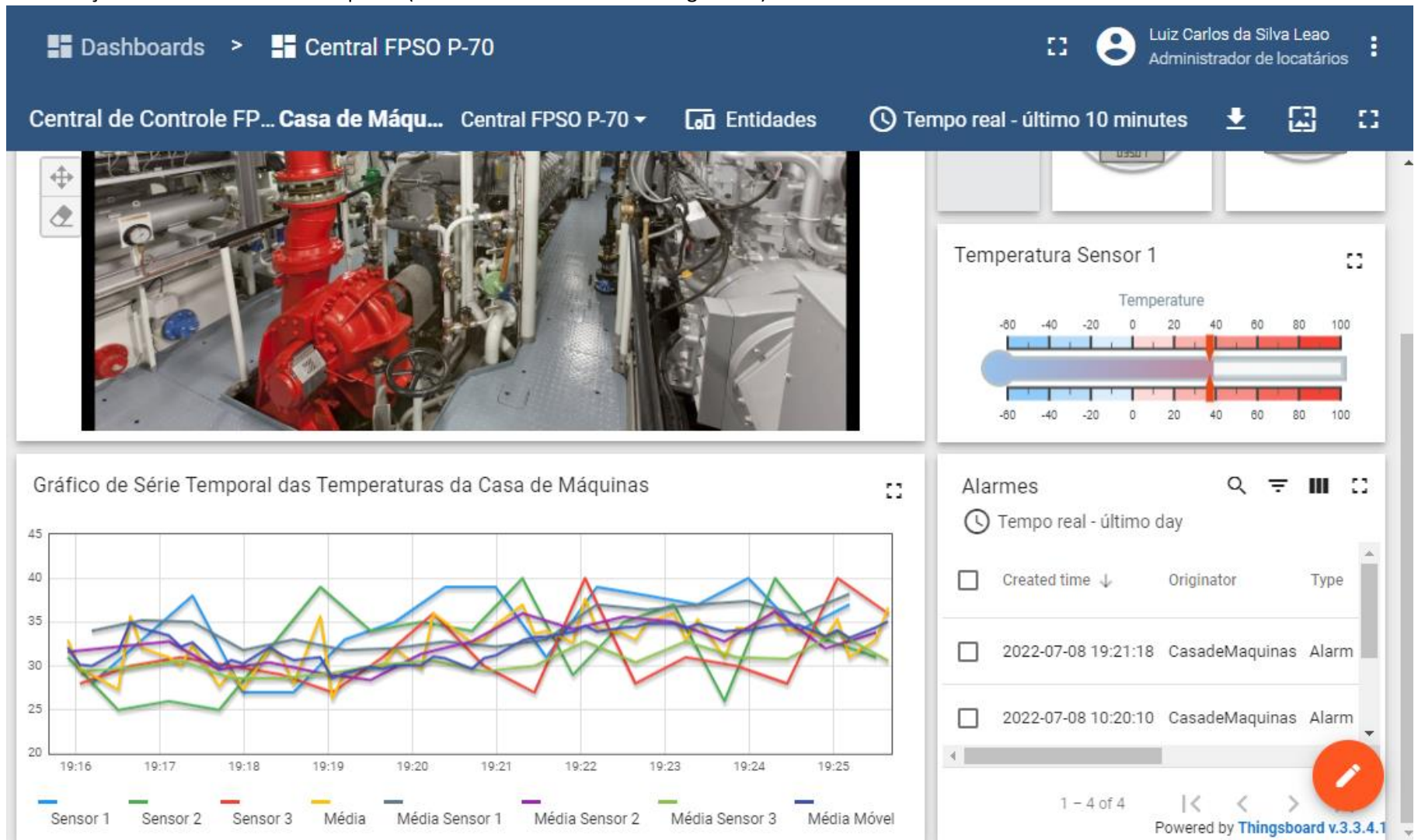


g) Dashboard – Casa de Máquinas (com o LED que acende e apaga conforme o alarme da média móvel é disparado)



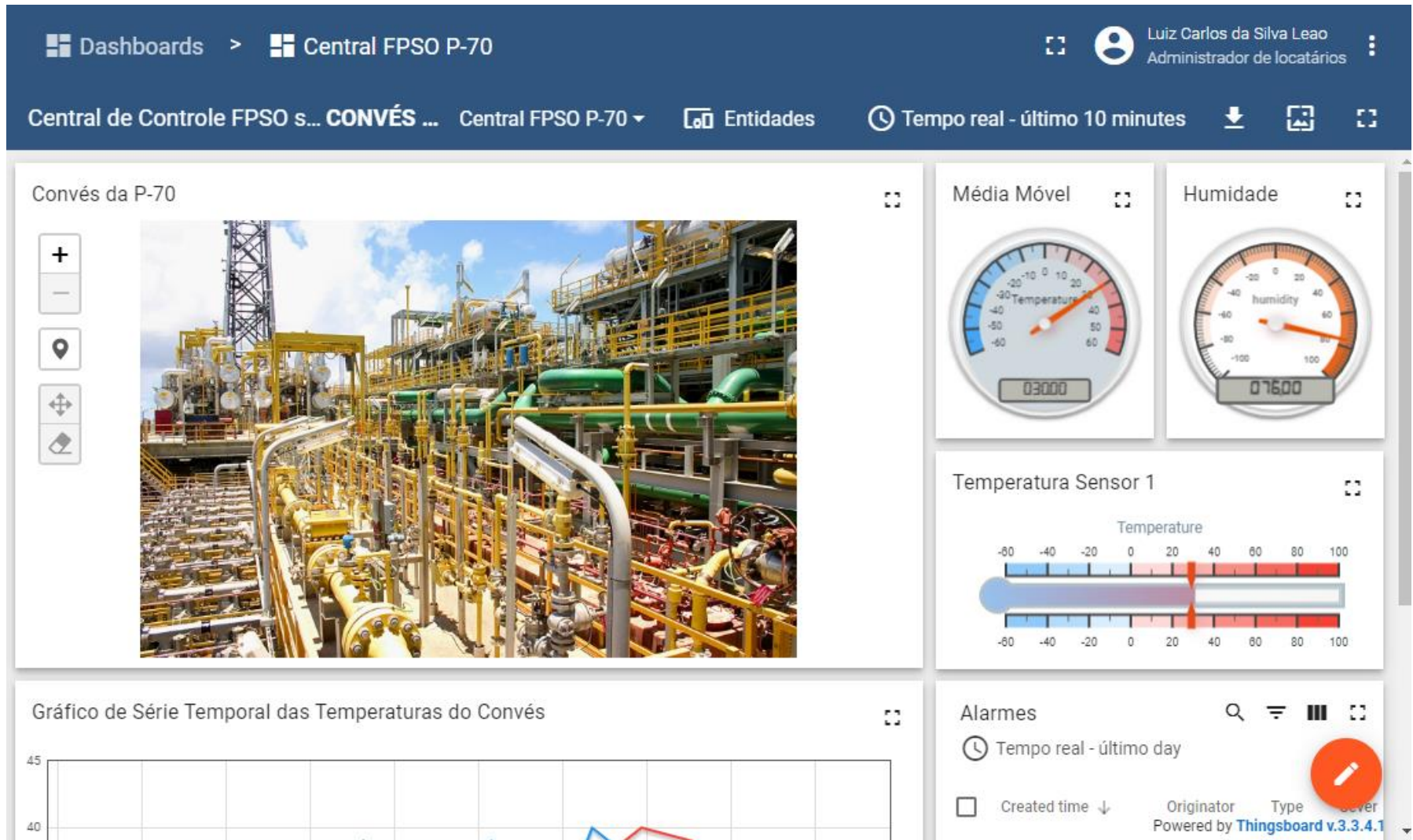


h) Continuação Dashboard Casa de Máquinas (com a indicativo dos alarmes gerados):





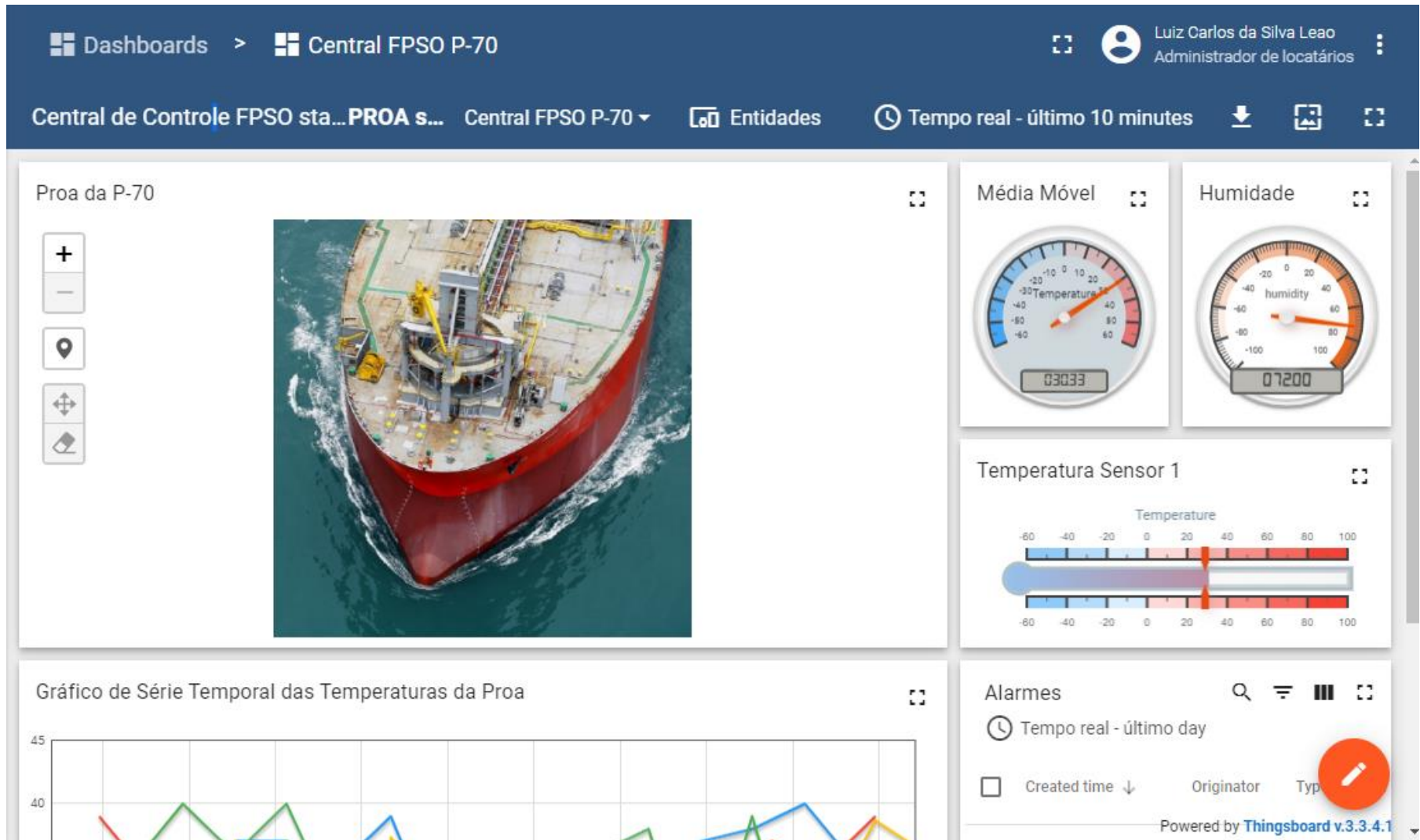
i) Dashboard Convés:



j) Dashboard Convés (continuação):

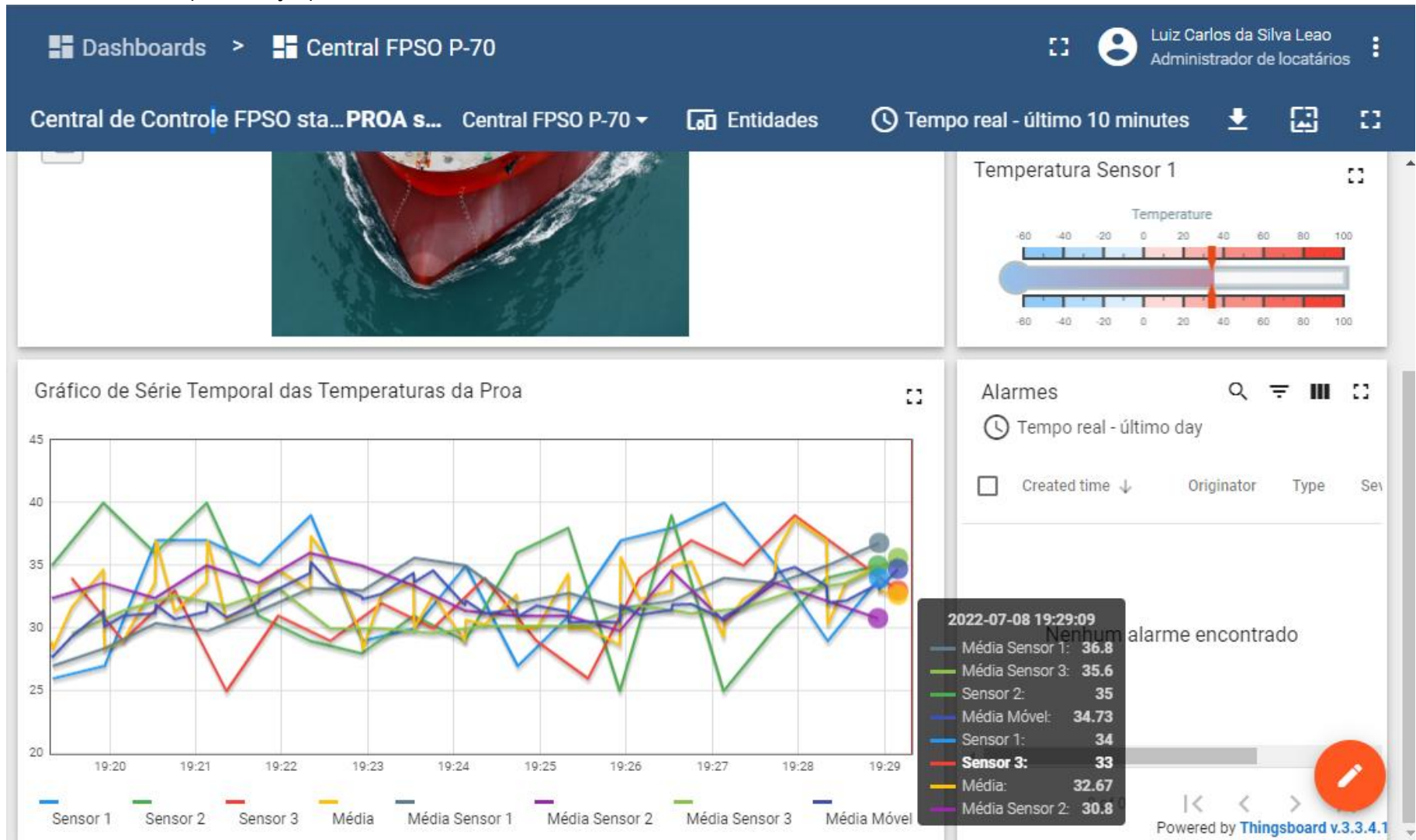


k) Dashboard da Proa:








































I) Dashboard da Proa (continuação):



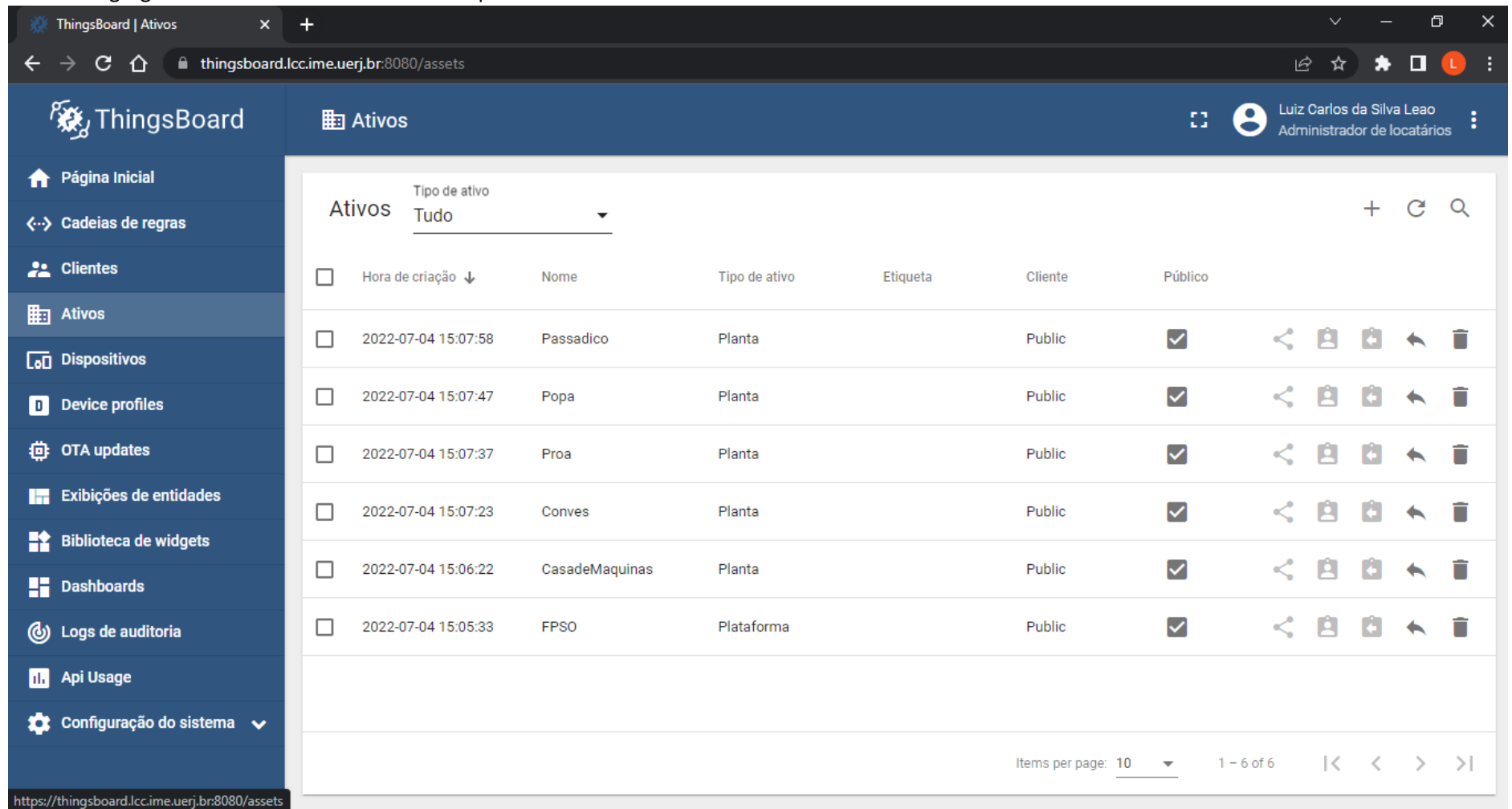
m) Dispositivos (um total de 14 simulados por um Raspberry Pi e 1 NodeMCU + DHT 22) – medidas: temperatura e humidade

The screenshot shows the ThingsBoard web interface. The left sidebar contains navigation links: Página Inicial, Cadeias de regras, Clientes, Ativos, Dispositivos (selected), Device profiles, OTA updates, Exibições de entidades, Biblioteca de widgets, Dashboards, Logs de auditoria, Api Usage, and Configuração do sistema. The main content area is titled 'Dispositivos' and shows a list of devices under the 'All' profile filter. The table lists 10 devices, each with a checkbox, creation time, name, device profile, label, client, public status, gateway status, and action icons. The devices are: conves-3, conves-2, proa-3, proa-2, popa-3, popa-2, and casademaquinas-3. The bottom of the page shows 'Items per page: 10' and '1 - 10 of 16'.

<input type="checkbox"/>	Hora de criação ↓	Nome	Device profile	Etiqueta	Cliente	Público	É gateway	
<input type="checkbox"/>	2022-07-05 07:48:04	conves-3	default	CONVES - Sensor 3 - RaspberryPi4	Public	<input checked="" type="checkbox"/>	<input type="checkbox"/>	    
<input type="checkbox"/>	2022-07-05 07:47:53	conves-2	default	CONVES - Sensor 2 - RaspberryPi4	Public	<input checked="" type="checkbox"/>	<input type="checkbox"/>	    
<input type="checkbox"/>	2022-07-05 07:46:38	proa-3	default	PROA - Sensor 3 - RaspberryPi4	Public	<input checked="" type="checkbox"/>	<input type="checkbox"/>	    
<input type="checkbox"/>	2022-07-05 07:46:21	proa-2	default	PROA - Sensor 2 - RaspberryPi4	Public	<input checked="" type="checkbox"/>	<input type="checkbox"/>	    
<input type="checkbox"/>	2022-07-05 07:45:51	popa-3	default	POPA - Sensor 3 - RaspberryPi4	Public	<input checked="" type="checkbox"/>	<input type="checkbox"/>	    
<input type="checkbox"/>	2022-07-05 07:45:26	popa-2	default	POPA - Sensor 2 - RaspberryPi4	Public	<input checked="" type="checkbox"/>	<input type="checkbox"/>	    
<input type="checkbox"/>	2022-07-05 07:44:58	casademaquinas-3	default	CASA DE MÁQUINAS - Sensor 3 - RaspberryPi4	Public	<input checked="" type="checkbox"/>	<input type="checkbox"/>	    

Items per page: 10 1 - 10 of 16

n) Assets: 1 agregando os sensores de cada área da plataforma



The screenshot shows the ThingsBoard web interface. The left sidebar contains navigation links: Página Inicial, Cadeias de regras, Clientes, Ativos (selected), Dispositivos, Device profiles, OTA updates, Exibições de entidades, Biblioteca de widgets, Dashboards, Logs de auditoria, Api Usage, and Configuração do sistema. The main content area is titled 'Ativos' and displays a table of assets. The table has columns: Hora de criação, Nome, Tipo de ativo, Etiqueta, Cliente, and Público. There are 6 assets listed, all with a 'Public' status and a 'Planta' or 'Plataforma' type. The bottom of the page shows pagination information: 'Items per page: 10' and '1 - 6 of 6'.

	Tipo de ativo					
Ativos	Tudo					
<input type="checkbox"/>	Hora de criação ↓	Nome	Tipo de ativo	Etiqueta	Cliente	Público
<input type="checkbox"/>	2022-07-04 15:07:58	Passadico	Planta		Public	<input checked="" type="checkbox"/>
<input type="checkbox"/>	2022-07-04 15:07:47	Popa	Planta		Public	<input checked="" type="checkbox"/>
<input type="checkbox"/>	2022-07-04 15:07:37	Proa	Planta		Public	<input checked="" type="checkbox"/>
<input type="checkbox"/>	2022-07-04 15:07:23	Conves	Planta		Public	<input checked="" type="checkbox"/>
<input type="checkbox"/>	2022-07-04 15:06:22	CasadeMaquinas	Planta		Public	<input checked="" type="checkbox"/>
<input type="checkbox"/>	2022-07-04 15:05:33	FPSO	Plataforma		Public	<input checked="" type="checkbox"/>



o) Relações dos Assets:

ThingsBoard | Ativos

thingsboard.lcc.ime.uerj.br:8080/assets

Luiz Carlos da Silva Leao  
Administrador de locatários

**Ativos**

Tipo de ativo: Tudo

	Hora de criação ↓	Nome
<input type="checkbox"/>	2022-07-04 15:07:58	Passadico
<input type="checkbox"/>	2022-07-04 15:07:47	Popa
<input type="checkbox"/>	2022-07-04 15:07:37	Proa
<input type="checkbox"/>	2022-07-04 15:07:23	Conves
<input type="checkbox"/>	2022-07-04 15:06:22	CasadeMaquinas
<input type="checkbox"/>	2022-07-04 15:05:33	FPSO

**CasadeMaquinas**  
Detalhes do ativo

Relações

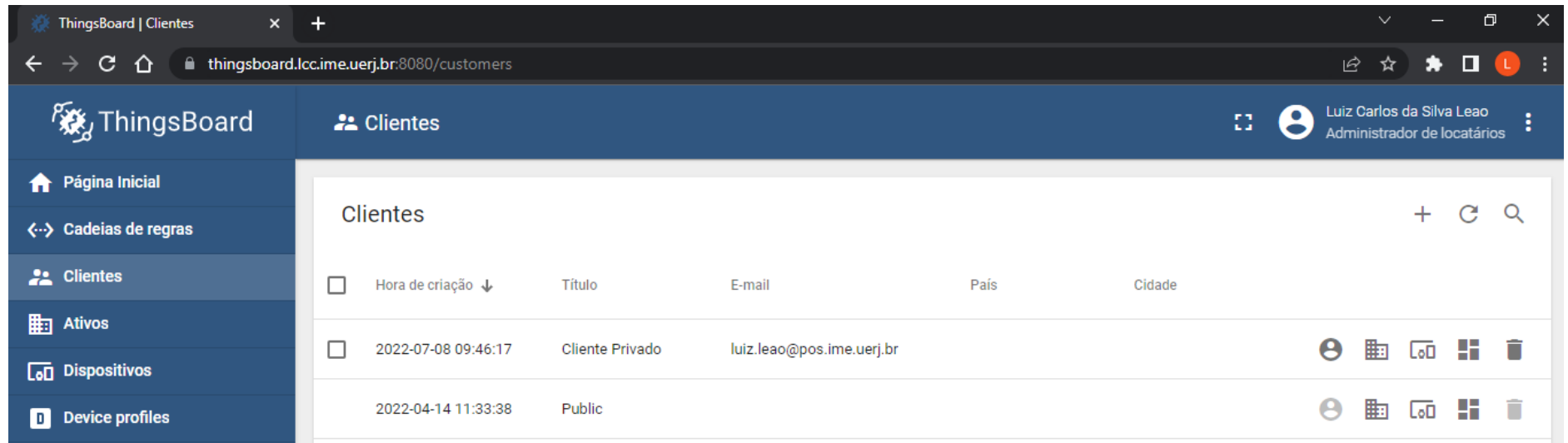
Relações de saída

Direção: De

	Tipo ↑	Para tipo de entidade	Para nome de entidade	
<input type="checkbox"/>	Contains	Dispositivo	casademaquinas-1	
<input type="checkbox"/>	Contains	Dispositivo	casademaquinas-2	
<input type="checkbox"/>	Contains	Dispositivo	casademaquinas-3	

Items per page: 10 1 - 3 of 3

p) Clientes:













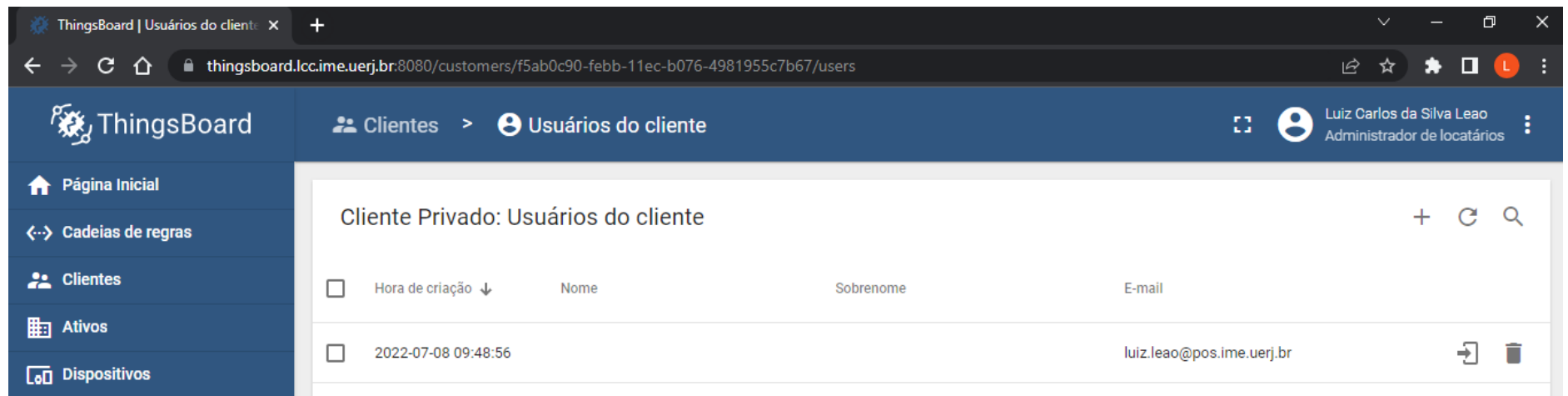
ThingsBoard | Clientes

Clientes

Luiz Carlos da Silva Leao  
Administrador de locatários

Clientes

<input type="checkbox"/>	Hora de criação ↓	Título	E-mail	País	Cidade	
<input type="checkbox"/>	2022-07-08 09:46:17	Cliente Privado	luiz.leao@pos.ime.uerj.br			    
	2022-04-14 11:33:38	Public				    





ThingsBoard | Usuários do cliente

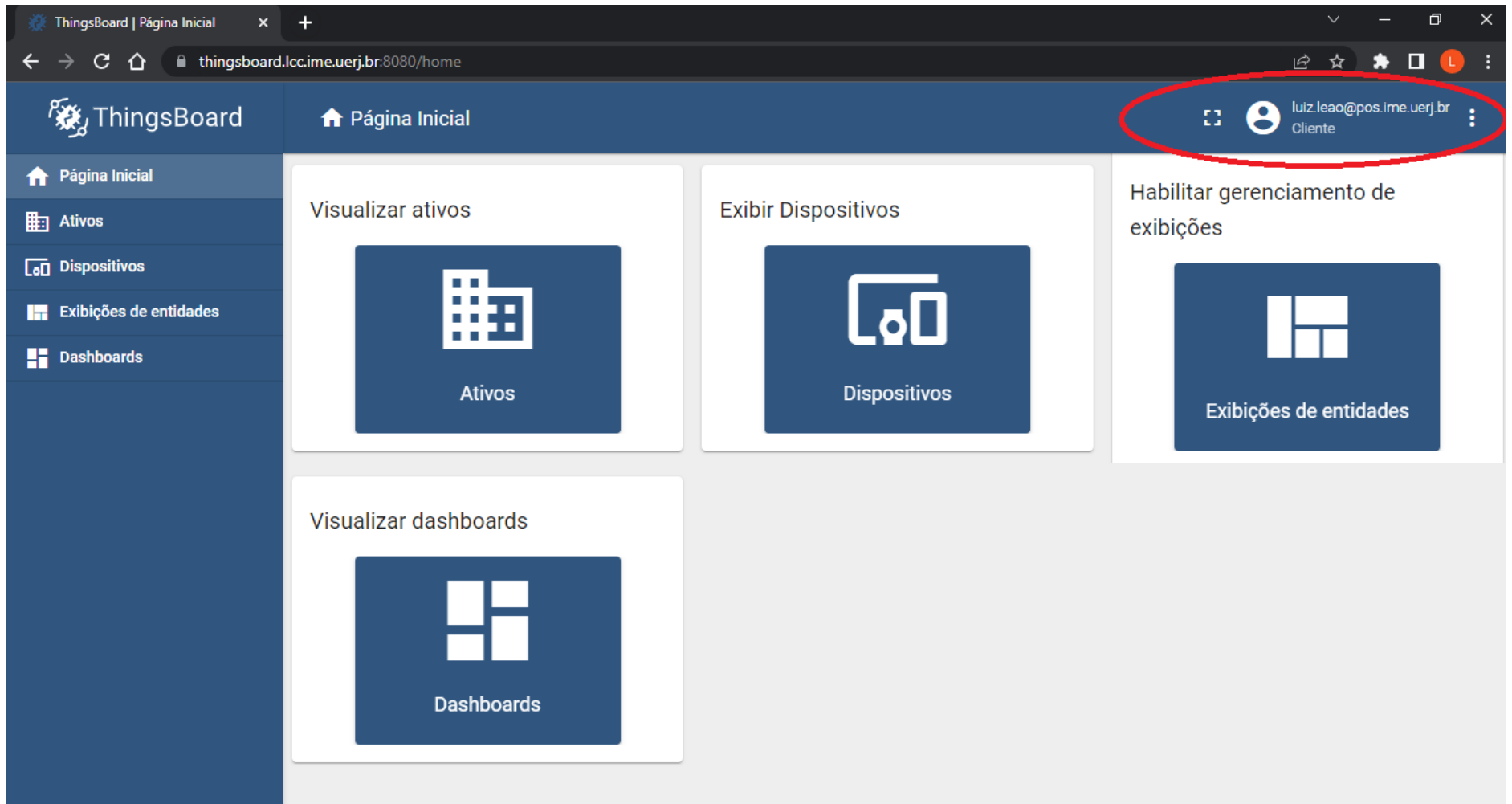
Clientes > Usuários do cliente

Luiz Carlos da Silva Leao  
Administrador de locatários

Cliente Privado: Usuários do cliente

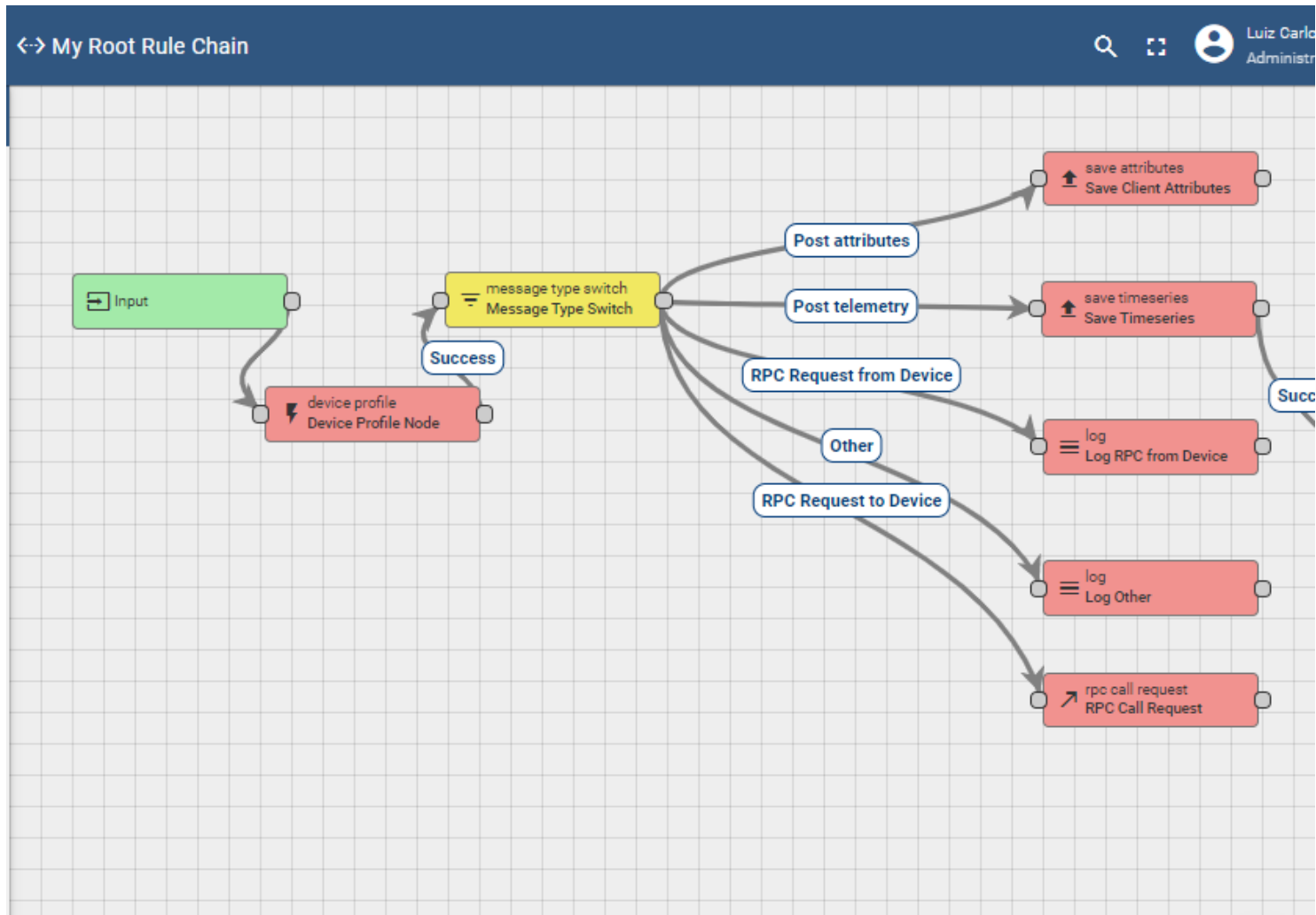
<input type="checkbox"/>	Hora de criação ↓	Nome	Sobrenome	E-mail	
<input type="checkbox"/>	2022-07-08 09:48:56			luiz.leao@pos.ime.uerj.br	 

q) Criação de cliente privado que só tem perfil de leitura do dashboard:

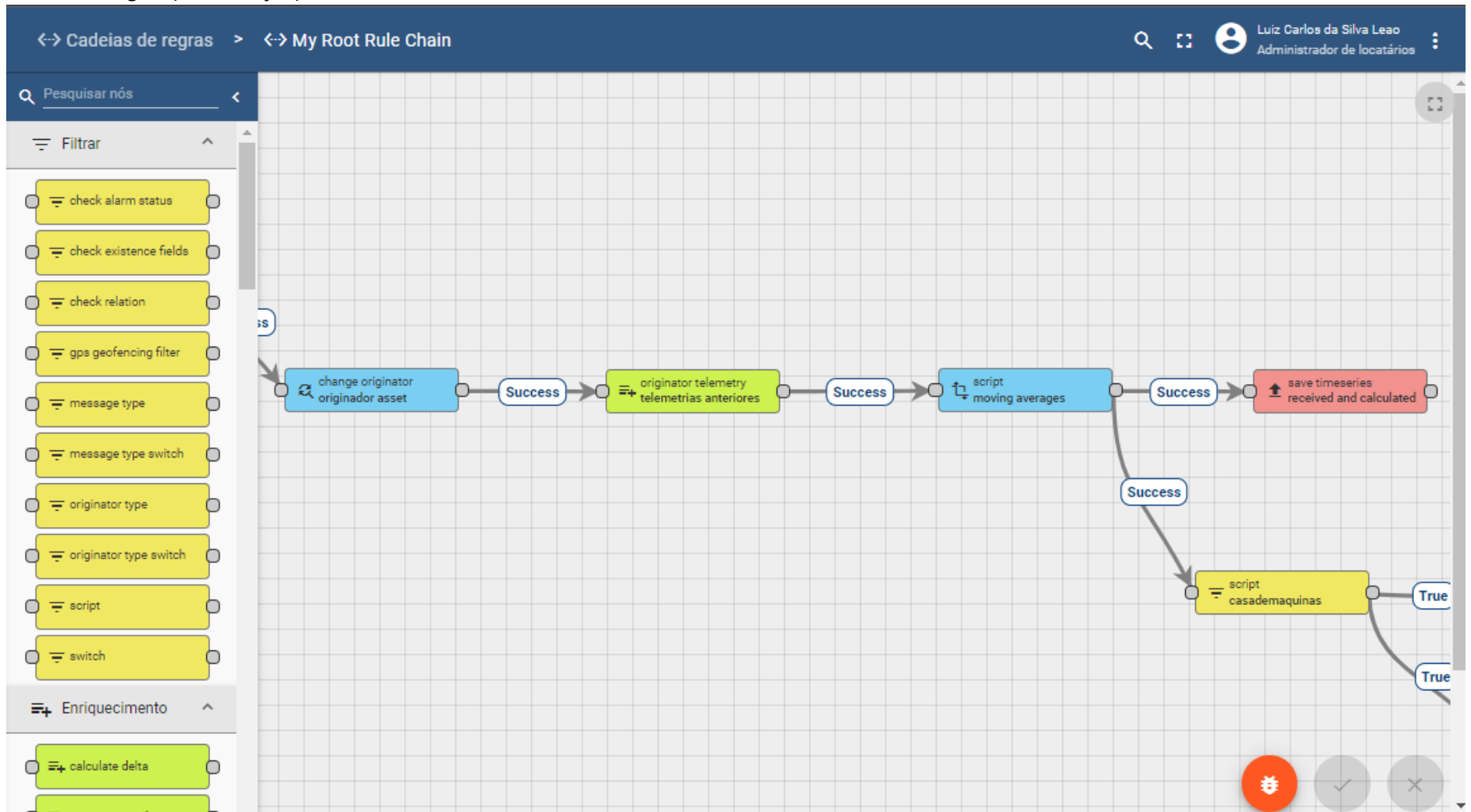




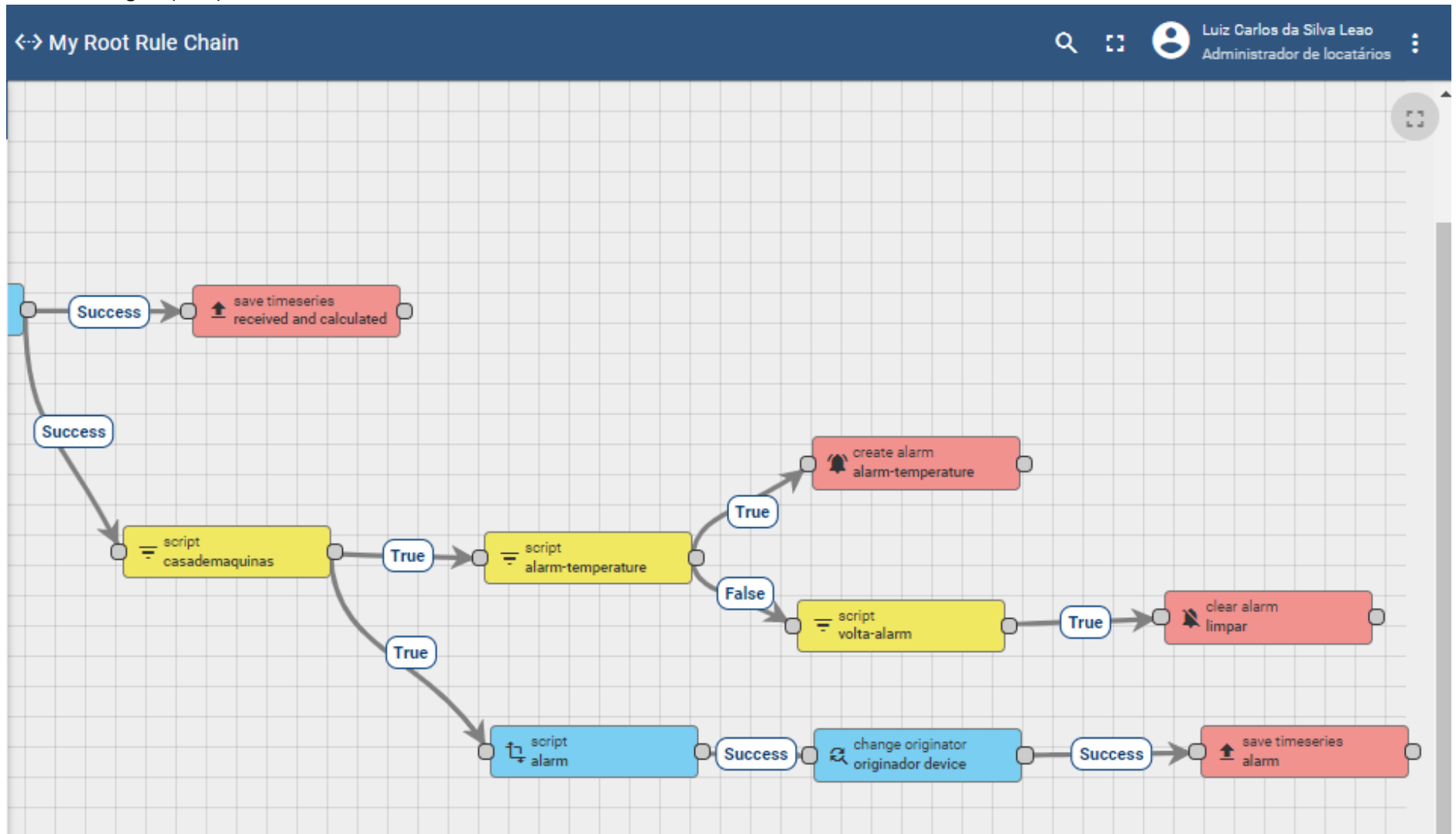
r) Cadeia de Regras (My Root Rule Chains) – uma cadeia de regras única para todo o projeto:



s) Cadeia de Regras (Continuação):



t) Cadeia de Regras (final):



u) Scripts (1 – Média e Média Móvel):

```
//return {msg: msg, metadata: metadata, msgType: msgType};
var newMsg = {};
var tname = "";
var tempArray = [];
var newMsg = {}
var media = 0.0;
var average = 0.0;
var i =0;
media = parseFloat((msg.temperature).toFixed(2));
tname = "Temperature" +
metadata.deviceName.slice(metadata.deviceName.indexOf("-"));
newMsg[tname] = media;
tempArray = [];
average = media;
if(metadata.hasOwnProperty(tname)){
    tempArray = JSON.parse(metadata[tname]);
    //calcula a média das 5 primeiras amostras de cada sensor e da
    média dos sensores
    for(var j = 0;j<tempArray.length && j<4;j++){
        average = (average*(j+1) +
parseFloat(tempArray[j].value))/(j+2);
    }
}
newMsg["media" + tname] =
parseFloat(average.toFixed(2));//adiciona a média móvel do
sensor como nova telemetria
```

```
for(var key in metadata){
    if(key.startsWith("Temperature")){
        if(key !== tname){
            tempArray = [];
            tempArray = JSON.parse(metadata[key]);
            media = (media*(i+1) +
parseFloat(tempArray[0].value))/(i+2);
            i++;
        }
    }
}
media = parseFloat(media.toFixed(2));
newMsg.media = media;
tempArray = [];
average = media;
if(metadata.hasOwnProperty("media")){
    tempArray = JSON.parse(metadata.media);
    //calcula a média das 5 primeiras amostras de cada sensor e da
    média dos sensores
    for(var j = 0;j<tempArray.length && j<4;j++){
        average = (average*(j+1) +
parseFloat(tempArray[j].value))/(j+2);
    }
}
newMsg.mediaMovel = parseFloat(average.toFixed(2));//adiciona
a média móvel do sensor como nova telemetria
var newMeta = {};
```



```

for(var key in metadata) {
    //mantém os metadados do sensor e o timestamp da
    mensagem
    if(key.startsWith("device") || key.startsWith("ts")){

```

```

        newMeta[key] = metadata[key];
    }
}
return {msg: newMsg, metadata: newMeta, msgType: msgType};

```

v) Scripts (Alarme):

```

return metadata.deviceName.startsWith("casademaquinas");

var newMsg = {"alarm" : (msg.mediaMovel > 32)};
metadata.ts = new Date().getTime();
return {msg: newMsg, metadata: metadata, msgType: msgType};

return msg.mediaMovel > 32;

```

w) Script (arquivo do Arduino \*.ino para envio da telemetria do NodeMCU + DHT22 e conexão RPC para liga/desliga do LED)

https-NodeMCU-DHT22.ino:

```

#include <Arduino.h>
#include <ArduinoJson.h>
#include <ESP8266WiFi.h>
#include <ESP8266WiFiMulti.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClientSecureBearSSL.h>
#include <DHT.h>

#define DHTPIN 4

```

```

#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);

ESP8266WiFiMulti WiFiMulti;

const char* REDE_WIFI = "****";
const char* SENHA_WIFI = "****";

```

```

//UERJ
String SERVER_URL =
"https://thingsboard.lcc.ime.uerj.br:8080/api/v1/";
String TOKEN = "***";

// fingerprint do servidor Thingsboard da UERJ para conexão
HTTPS
const uint8_t FINGERPRINT[20] =
{0xd8,0xae,0xeb,0x77,0x30,0xe3,0xa8,0x2a,0x38,0x4b,0x56,0xd8,
0x6d,0xa8,0xf9,0x0a,0x39,0x41,0x5d,0x0f};

float temp = 0.0;
float umid = 0.0;

int porta = D1; //porta do LED

unsigned long timeSinceLastRead = 0;//timestamp da última
leitura dos sensores
unsigned long timeSinceLastSend = 0;//timestamp do último
envio das telemetrias
unsigned long timeSinceLastRpc = 0;//timestamp da última
verificação dos comandos RPC
const long intervalSend = 30000;//envio a cada 30 segundos

void setup() {
  Serial.begin(115200);
  Serial.setTimeout(2000);

  while(!Serial) { }

```

```

dht.begin();

// espera 4s para a conexão
for (uint8_t t = 4; t > 0; t--) {
  Serial.printf("[SETUP] Aguarde %d...\n", t);
  Serial.flush();
  delay(1000);
}

//conexão wifi
WiFi.mode(WIFI_STA);
//WiFiMulti.addAP("RapNet_Virtua", "a0abea1234");
WiFiMulti.addAP(REDE_WIFI, SENHA_WIFI);
//client.setCallback(on_message);
pinMode(porta,OUTPUT);
digitalWrite(porta,HIGH);//começa com o LED ligado
while(WiFiMulti.run() != WL_CONNECTED){}
Serial.println("Dispositivo iniciado e WIFI conectada");
}

void loop() {
  unsigned long timeCurrent = millis();
  //Somente captura os valores dos sensores
  if((timeCurrent - timeSinceLastRead) >= 2000) {
    float h = dht.readHumidity();
    float t = dht.readTemperature();
    float f = dht.readTemperature(true);

```

```

if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println("Failed to read from DHT sensor!");
    timeSinceLastRead = timeCurrent - 1000;
    return;
}
temp=t;
umid=h;
/*
if(temp<20){
    temp=20;
}else if(temp>=5){
    temp=20;
}else{
    temp+=1;
}
*/
timeSinceLastRead = timeCurrent;
Serial.println("Telemetrias atualizadas");
}

// wait for WiFi connection
if ((WiFiMulti.run() == WL_CONNECTED)) {
    std::unique_ptr<BearSSL::WiFiClientSecure>client(new
BearSSL::WiFiClientSecure);

    client->setFingerprint(FINGERPRINT);
    //client->setInsecure();

    HTTPClient https;

```

```

String payload = "";
String url_conexao;
int httpCode;
if((timeCurrent - timeSinceLastRpc) >= 5000){
    //Verificação dos comandos RPC
    //concatena o token na url de conexão
    url_conexao = SERVER_URL + TOKEN + "/rpc?timeout=5000";
    //url_conexao.replace("$TOKEN$",TOKEN);
    Serial.print("url_conexao:" + url_conexao + "\n");

    //if (https.begin(*client,
"https://thingsboard.lcc.ime.uerj.br:8080/api/v1/sHxI9MKNW8O
BqneP3caV/telemetry")) { // HTTPS
        if (https.begin(*client, url_conexao)) { // HTTPS

            Serial.print("[HTTPS] GET Payload: " + payload + "\n");

            httpCode = https.GET();

            if (httpCode > 0) {

                Serial.printf("[HTTPS] GET... code: %d\n", httpCode);

                if (httpCode == HTTP_CODE_OK || httpCode ==
HTTP_CODE_MOVED_PERMANENTLY) {
                    payload = https.getString();
                    https.end();
                    Serial.println(payload);

```

```

        DynamicJsonDocument
        json(uint32_t(float(payload.length() + 10)/10)*10);
        char payloadchar[payload.length()+1];
        strncpy(payloadchar,payload.c_str(),payload.length());
        DeserializationError error =
        deserializeJson(json,payloadchar);
        if(error){
            Serial.println("Payload não está no formato JSON!");
        }else{
            if(json["method"].isNull()){
                Serial.println("Método não encontrado");
            }else{
                if(json["method"].as<String>()=="getValor"){
                    if(!json["id"].isNull()){
                        int id = json["id"].as<int>();
                        //Retorna
                        url_conexao = SERVER_URL + TOKEN + "/rpc/" +
String(id);
                        if (https.begin(*client, url_conexao)) { // HTTPS

                            httpCode = https.POST(digitalRead(porta) ? "true" :
"false");

                            if (httpCode > 0) {

                                Serial.printf("[HTTPS] POST... code: %d\n",
httpCode);

```

```

                            if (httpCode == HTTP_CODE_OK || httpCode ==
HTTP_CODE_MOVED_PERMANENTLY) {
                                payload = https.getString();
                                Serial.println(payload);
                            }
                        } else {
                            Serial.printf("[HTTPS] GET... failed, error: %s\n",
https.errorToString(httpCode).c_str());
                        }
                        https.end();
                    }
                } else if(json["method"].as<String>()=="setValor"){
                    if(json["params"].isNull()){
                        Serial.println("Parâmetros não encontrados");
                    }else{

digitalWrite(porta,json["params"].as<String>()=="true"?HIGH:LOW);

                            timeSinceLastSend = 0;
                        }
                    }
                }
            } else{
                Serial.printf("[HTTPS] GET... failed, error: %s\n",
https.errorToString(httpCode).c_str());
                https.end();
            }
        }
    }
}

```

```

    } else {
        Serial.printf("[HTTPS] GET... failed, error: %s\n",
https.errorToString(httpCode).c_str());
    }
    } else {
        Serial.printf("[HTTPS] Unable to connect\n");
    }
    timeSinceLastRpc = timeCurrent;
}

if((timeCurrent - timeSinceLastSend) >= intervalSend){
    //Envio das telemetrias a cada intervalSend
    //concatena o token na url de conexão
    url_conexao = SERVER_URL + TOKEN + "/telemetry";
    //url_conexao.replace("$TOKEN$",TOKEN);
    Serial.print("url_conexao:" + url_conexao + "\n");

    //if (https.begin(*client,
"https://thingsboard.lcc.ime.uerj.br:8080/api/v1/sHxI9MKNW8O
BqneP3caV/telemetry")) { // HTTPS
        if (https.begin(*client, url_conexao)) { // HTTPS

            Serial.print("[HTTPS] POST...\n");

            payload = "{\"temperature\":\"" + String(temp) + ",
\"humidity\":\"" + String(umid) + "\",\"port\":\"" + (digitalRead(porta) ?
"true" : "false") + "\"}";

```

```

Serial.print("[HTTPS] Payload: " + payload + "\n");

httpCode = https.POST(payload);

if (httpCode > 0) {

    Serial.printf("[HTTPS] POST... code: %d\n", httpCode);

    if (httpCode == HTTP_CODE_OK || httpCode ==
HTTP_CODE_MOVED_PERMANENTLY) {
        payload = https.getString();
        Serial.println(payload);
    }
    } else {
        Serial.printf("[HTTPS] GET... failed, error: %s\n",
https.errorToString(httpCode).c_str());
    }

    https.end();

} else {
    Serial.printf("[HTTPS] Unable to connect\n");
}
timeSinceLastSend = timeCurrent;
}
}
}

```



- x) Script Python para rodar no Raspberry Pi e simular o envio de telemetria para o ThingsBoard (replicou-se o mesmo para os demais sensores):

```
import os
import time
import sys
import paho.mqtt.client as mqtt
import json
from random import randint
import random
THINGSBOARD_HOST = 'thingsboard.lcc.ime.uerj.br'
ACCESS_TOKEN = 'xyz'

# configure aqui o intervalo
INTERVAL=36

sensor_data = {'temperature': 0, 'humidity': 0}

next_reading = time.time()

client = mqtt.Client()

# Set access token
client.username_pw_set(ACCESS_TOKEN)

# Connect to ThingsBoard using default MQTT port and 60
seconds keepalive interval
client.connect(THINGSBOARD_HOST, 1883, 60)
```

```
client.loop_start()

try:
    while True:
        # gera um valor de umidade aleatório
        humidity = randint(70,80)
        humidity = round(humidity, 2)
        # gera um valor de temperatura aleatório
        temperature = randint(25,40)
        temperature = round(temperature, 2)

        print(u"Temperatura: {:g}\u00b0C, Umidade:
{:g}%".format(temperature, humidity))
        sensor_data['temperature'] = temperature
        sensor_data['humidity'] = humidity

        # Sending humidity and temperature data to ThingsBoard
        client.publish('v1/devices/me/telemetry',
            json.dumps(sensor_data), 1)

        next_reading += INTERVAL
        sleep_time = next_reading-time.time()
        if sleep_time > 0:
            time.sleep(sleep_time)
```

```
except KeyboardInterrupt:  
    pass
```

```
client.loop_stop()  
client.disconnect()
```