



Server MQTT (generic)

Coordonator,

Ș.l.dr.ing.Nicolae-Alexandru Botezatu

Tărăboanță Andreea

Vanzariuc Maria

Grupa: 1310A

Cuprins

1. Introducere.....	2
2. De ce sa utilizăm MQTT?	2
3. Topic-uri	2
4. Tipuri de mesaje MQTT	3
a) CONNECT	3
b) CONNACK	4
c) PUBLISH	5
d) SUBSCRIBE	5
e) SUBACK.....	6
f) UNSUBSCRIBE.....	6
g) UNSUBACK.....	6
5. Mecanism LastWill	7
6. Mecanism KeepAlive	7
7. Exemple de aplicații care folosesc MQTT.....	7
8. Interfata cu utilizatorul	8
9. Bibliografie	13

1. Introducere

MQTT (Message Queue Telemetry Transport) este un standard de comunicație prin internet, inventat și dezvoltat de IBM în 1999. Acesta definește un protocol de transport al mesajelor prin internet (prin TCP/IP în general), între un server, numit "broker" și mai mulți clienți, pe modelul "publicare și abonare" ("publish and subscribe").

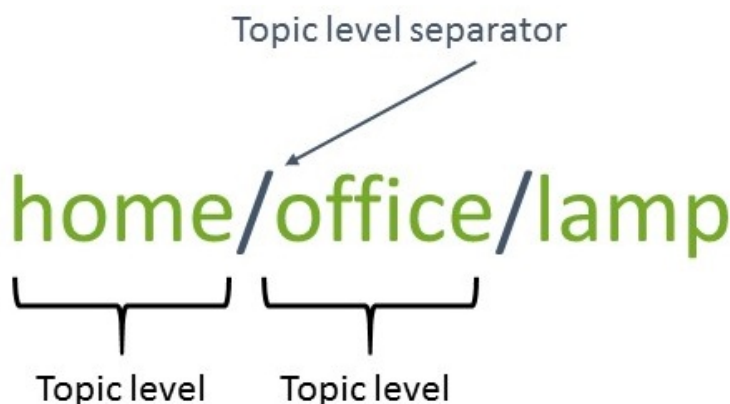
2. De ce sa utilizăm MQTT?

- Distribuirea informațiilor se realizează rapid și eficient
- Este ușor de implementat software, prin urmare se economisește timp în dezvoltare
- Utilizează puțină energie

3. Topic-uri

Topic-urile sunt string-uri pe care broker-ii le folosesc pentru a filtra mesajele pentru fiecare client conectat. Acestea pot avea mai multe nivele, fiind separate printr-un / (slash).

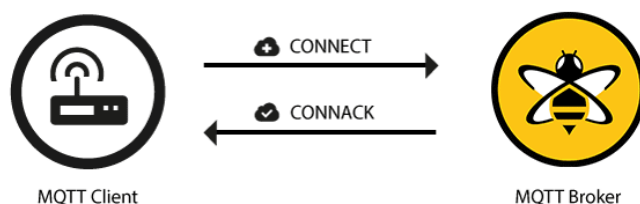
Clienții se abonează la un anumit topic pentru a putea primi mesajele dorite. Acest lucru înseamnă că vor primi mesajele corespunzătoare doar acelui topic, sau pot folosi wildecard-uri pentru a se abona la mai multe topic-uri simultan.



4. Tipuri de mesaje MQTT

a) CONNECT

Conexiunea MQTT este întotdeauna între un client și broker. Clienții nu se conectează niciodată unul cu celălalt în mod direct. Pentru a iniția o conexiune, clientul trimite mesajul CONNECT către broker.



MQTT-Packet: CONNECT		
contains:		Example
clientId		"client-1"
cleanSession		true
username (optional)		"hans"
password (optional)		"letmein"
lastWillTopic (optional)		"/hans/will"
lastWillQos (optional)		2
lastWillMessage (optional)		"unexpected exit"
lastWillRetain (optional)		false
keepAlive		60

ClientId: Identifică fiecare client care se conectează la un broker. Acest ID ar trebui să fie unic per client.

Clean Session: Acest flag comunică broker-ului dacă clientul dorește să stabilească o sesiune de lungă durată sau nu.

Username/Password: MQTT poate trimite un username și o parolă pentru autentificarea și autorizarea unui client.

Will Message: Acest mesaj notifică alți clienți când un client se deconectează forțat.

Will Retain: Flag ce indică dacă mesajul este salvat de către broker, pentru un topic specificat, ca ultimă valoare

KeepAlive: Reprezintă un interval de timp în secunde pe care clientul îl specifică brokerului atunci când conexiunea este stabilă.

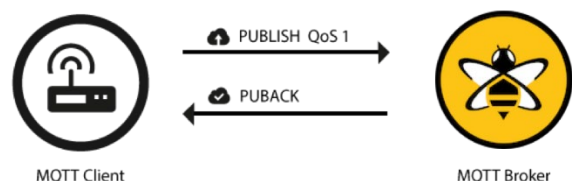
Qos (Quality of Service) este un acord între expeditorul unui mesaj și receptorul unui mesaj care definește garanția de livrare pentru un mesaj specific.

Există 3 niveluri QoS în MQTT:

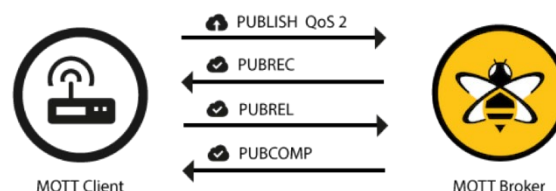
-> 0 – este nivelul minim de Qos. Aici nu există o garanție de livrare: destinatarul nu confirmă primirea mesajului, iar mesajul nu este stocat și re-transmis de către expeditor.



-> 1 - nivelul 1 garantează că un mesaj este livrat cel puțin o dată receptorului, expeditorul stochează mesajul până când receptorul confirmă primirea mesajului



-> 2 - este cel mai înalt nivel de serviciu din MQTT. Acest nivel garantează că fiecare mesaj este primit o singură dată de către destinatarii intenționați. QoS 2 este cel mai sigur și mai lent nivel al calității serviciului. Garanția este asigurată de cel puțin două fluxuri de solicitare / răspuns între expeditor și receptor.



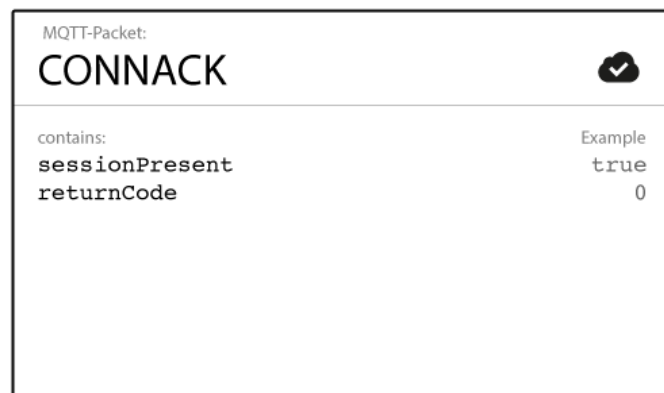
b) CONNACK

Când broker-ul primește mesajul CONNECT, trebuie să îi răspundă clientului cu mesajul CONNACK.

Session Present flag: Acest flag spune clientului dacă broker-ul deja are o sesiune de lungă durată disponibilă din interacțiunile trecute cu clientul.

Return code: Acest flag conține un cod de returnare care spune că încercarea clientului de a se conecta este cu succes sau nu.

- 0 – conexiune acceptată
- 1 – conexiune respinsă, versiune de protocol inacceptabilă
- 2 – conexiune respinsă, identificator respins
- 3 – conexiune refuzată, serverul nu este disponibil
- 4 – conexiune refuzată, nume/ parolă greșite
- 5 – conexiune refuzată, neautorizată



c) PUBLISH

Un client MQTT poate publica mesaje imediat ce se conectează la un broker. Fiecare mesaj trebuie să conțină un subiect (topic) pe care broker-ul îl poate folosi pentru a transmite mesajul către clienții interesați. De obicei, fiecare mesaj are un payload care conține datele de transmis în format de octeți.

MQTT-Packet:	
PUBLISH	
contains:	Example
packetId (always 0 for qos 0)	4314
topicName	"topic/1"
qos	1
retainFlag	false
payload	"temperature:32.5"
dupFlag	false

PacketId: Identificator unic pentru mesaje

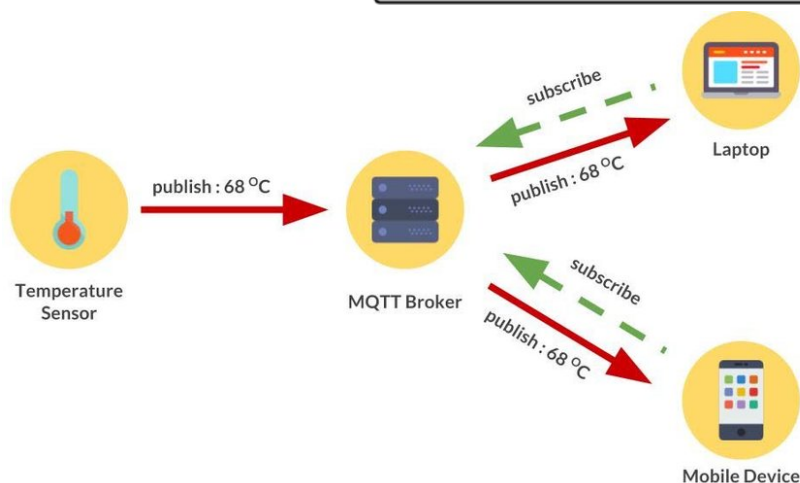
Payload: Conținutul efectiv al mesajului

DupFlag: Flag ce indică dacă mesajul a fost retrimis sau este duplicat

d) SUBSCRIBE

Publicarea unui mesaj nu are sens dacă nimeni nu îl primește vreodată. Cu alte cuvinte, dacă nu există clienți care să se aboneze la subiectele mesajelor. Pentru a primi mesaje pe subiecte de interes, clientul trimite un mesaj SUBSCRIBE către broker-ul MQTT. Acest mesaj de abonare este foarte simplu, conține un identificator unic de pachete și o listă de abonamente.

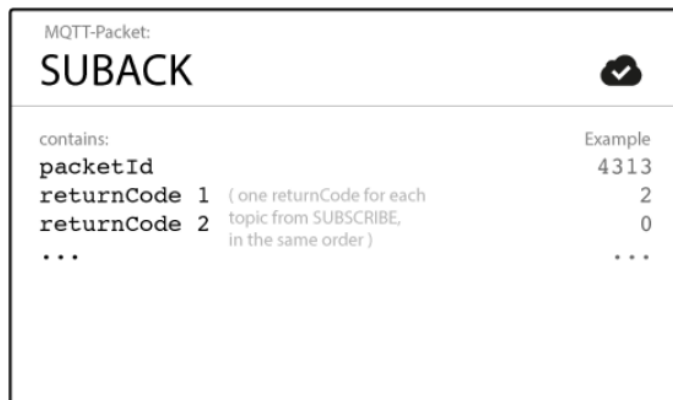
MQTT-Packet:	
SUBSCRIBE	
contains:	Example
packetId	4312
qos1 } (list of topic + qos)	1
topic1	"topic/1"
qos2 } (list of topic + qos)	0
topic2	"topic/2"
...	...



e) SUBACK

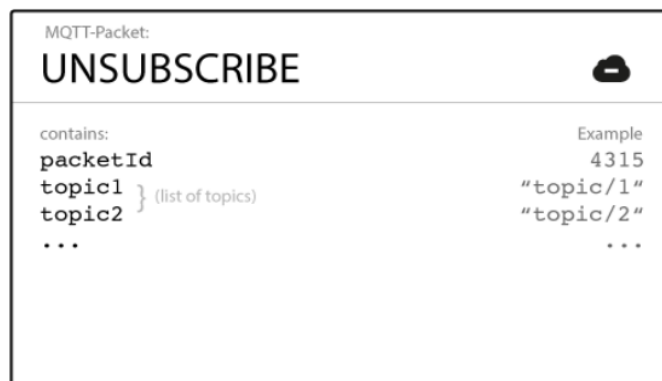
Pentru a confirma fiecare abonament, brokerul trimite mesajul SUBACK clientului. Acest mesaj conține identificatorul de pachet al mesajului original de abonare (pentru a identifica clar mesajul) și o listă de coduri de returnare.

- 0 – Succes: Maximum QoS 0
- 1 - Succes: Maximum QoS 1
- 2 - Succes: Maximum QoS 2
- 128 – Eșec



f) UNSUBSCRIBE

Acest mesaj șterge abonamentele existente ale unui client de la broker. Mesajul UNSUBSCRIBE este similar cu mesajul SUBSCRIBE și are un identificator de pachet și o listă de subiecte.



g) UNSUBACK

Pentru a confirma dezabonarea, brokerul trimite mesajul UNSUBACK clientului. Acest mesaj conține doar identificatorul de pachet al mesajului original de la UNSUBSCRIBE (pentru a identifica în mod clar mesajul).



5. Mecanism LastWill

În MQTT, se utilizează caracteristica Last Will and Testament (LWT) pentru a anunța alți clienți despre un client deconectat necorespunzător. Fiecare client își poate specifica ultimul mesaj de testament atunci când se conectează la un broker. Ultimul mesaj conține un subiect, flag-ul mesajului reținut, QoS-ul și topic-ul. Broker-ul stochează mesajul până când detectează că clientul nu s-a deconectat corect. Ca răspuns la această deconectare, broker-ul trimite ultimul mesaj către toți clienții abonați. Dacă clientul se deconectează corect (mesaj DISCONNECT), broker-ul renunță la mesajul LWT stocat.

6. Mecanism KeepAlive

KeepAlive este intervalul maxim de timp care este permis să treacă între punctul în care clientul termină transmiterea unui pachet de control și punctul pe care începe să îl trimită pe următorul. Clientul trebuie să aibă grijă ca intervalul dintre pachetele de control trimise să nu depășească valoarea Keep Alive.

Dacă nu circulă date pe o conexiune deschisă pentru o anumită perioadă de timp, atunci clientul va genera un PINGREQ și se așteaptă să primească un PINGRESP de la broker. Dacă broker-ul nu primește un PINGREQ sau orice alt pachet de la client, el închide conexiunea și trimite ultimul mesaj, bineînțeles dacă clientul a specificat un LWT.

7. Exemple de aplicații care folosesc MQTT

- Facebook Messenger
- Amazon Web Services
- OpenStack
- Adafruit
- Microsoft Azure
- Node-RED

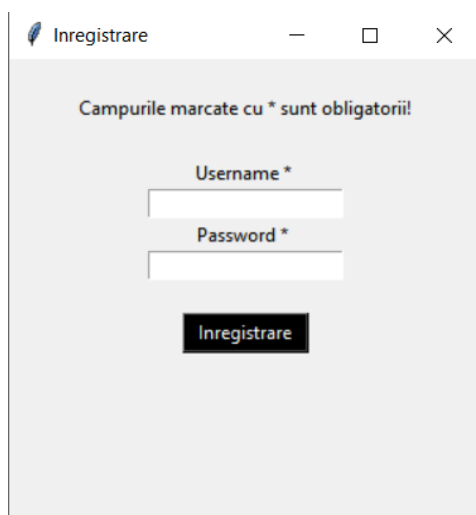
8. Interfața cu utilizatorul

La rularea programului se deschide fereastra de login, care conține 2 butoane:

- **Înregistrare**
- **Autentificare**



8.1. Înregistrare

A screenshot of a window titled 'Inregistrare'. The window has a light gray background. At the top, there is a message: 'Campurile marcate cu * sunt obligatorii!'. Below this, there are two input fields. The first is labeled 'Username *' and the second is labeled 'Password *'. Both fields are empty. Below the input fields, there is a dark gray button labeled 'Inregistrare' in white text.

Apăsarea butonului **Înregistrare** deschide o nouă fereastră în care trebuie introdus un nume și o parolă, ce vor fi salvate într-un fișier.

```

def register(self):
    global username
    global password
    global username_entry
    global password_entry

    username = StringVar()
    password = StringVar()

    self.register_screen = Toplevel(self.root)
    self.register_screen.title("Inregistrare")
    self.register_screen.geometry("300x300")

    Label(self.register_screen, text="").pack()
    Label(self.register_screen, text="Campurile marcate cu * sunt obligatorii!").pack()
    Label(self.register_screen, text="").pack()

    username_label = Label(self.register_screen, text="Username * ")
    username_label.pack()
    username_entry = Entry(self.register_screen, textvariable=username)
    username_entry.pack()

    password_label = Label(self.register_screen, text="Password * ")
    password_label.pack()
    password_entry = Entry(self.register_screen, textvariable=password, show='*')
    password_entry.pack()

    Label(self.register_screen, text="").pack()
    # buton inregistrare
    Button(self.register_screen, text="Inregistrare", width=10, height=1, bg="black", fg="white", command=self.register_user).pack()

def register_user(self):
    # get username and password
    username_info = username.get()
    password_info = password.get()

    file = open(username_info, "w")

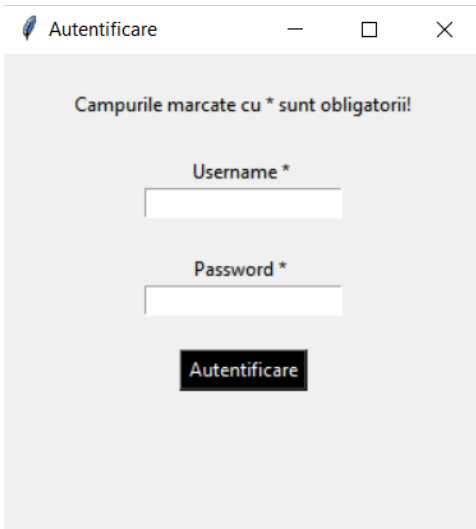
    file.write(username_info + "\n")
    file.write(password_info)
    file.close()

    username_entry.delete(0, END)
    password_entry.delete(0, END)

    Label(self.register_screen, text="Inregistrare realizata cu succes", fg="orange", font=("calibri", 11)).pack()

```

8.2. Autentificare



Apăsarea butonului **Autentificare** deschide o nouă fereastră în care trebuie introdus un nume și o parolă. Numele și parola vor fi comparate cu datele din fișierele create la înregistrare. Dacă nu există un utilizator cu numele respectiv se va afișa în fereastră mesajul “ Utilizator invalid”; iar dacă parola nu corespunde username-ului se va afișa în fereastră mesajul “Parolă incorectă”.

```
def login(self):

    global username_verify
    global password_verify
    global username_login_entry
    global password_login_entry

    self.login_screen = Toplevel(self.root)
    self.login_screen.title("Autentificare")
    self.login_screen.geometry("300x300")
    Label(self.login_screen, text="").pack()
    Label(self.login_screen, text="Campurile marcate cu * sunt obligatorii!").pack()
    Label(self.login_screen, text="").pack()

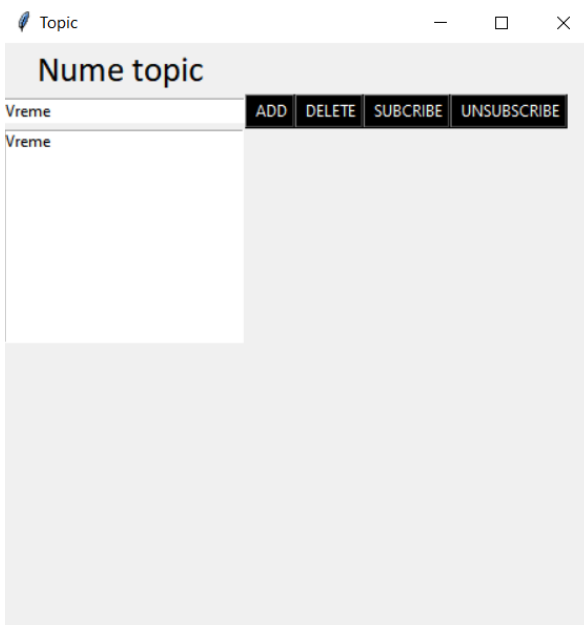
    username_verify = StringVar()
    password_verify = StringVar()

    Label(self.login_screen, text="Username * ").pack()
    username_login_entry = Entry(self.login_screen, textvariable=username_verify)
    username_login_entry.pack()
    Label(self.login_screen, text="").pack()
    Label(self.login_screen, text="Password * ").pack()
    password_login_entry = Entry(self.login_screen, textvariable=password_verify, show='*')
    password_login_entry.pack()
    Label(self.login_screen, text="").pack()
    Button(self.login_screen, text="Autentificare", bg="black", fg="white", width=10, height=1, command=self.login_verify).pack()
```

```
#functie care verifica daca exista un utilizator cu numele si parola aferenta
def login_verify(self):
    username1 = username_verify.get()
    password1 = password_verify.get()
    username_login_entry.delete(0, END)
    password_login_entry.delete(0, END)

    list_of_files = os.listdir()
    if username1 in list_of_files:
        file1 = open(username1, "r")
        verify = file1.read().splitlines()
        if password1 in verify:
            Label(self.login_screen, text="Autentificare realizata cu succes", fg="orange", font=("calibri", 11)).pack()
            self.login_screen.destroy()
            self.root.destroy()
            t = Topics()
        else:
            Label(self.login_screen, text="Parola incorecta", fg="red", font=("calibri", 11)).pack()
    else:
        Label(self.login_screen, text="Utilizator invalid", fg="red", font=("calibri", 11)).pack()
```

8.3. Topic



O dată ce autentificarea s-a realizat cu succes, fereastra principală și cea de autentificare se închid, iar în locul lor apare o fereastră numită **Topic**. Aici pot fi adăugate și șterse topic-urile.

```

from tkinter import *

class Topics:
    def __init__(self):
        global root
        self.v = []
        self.root = Tk()
        self.root.title('Topic')
        self.root.geometry('450x450')

        self.labelTopic = Label(self.root, text="Nume topic ", font=("calibri", 20))
        self.labelTopic.grid(row=0, column=0)

        #creare caseta text
        self.entryTopic = Entry(self.root, width="30")
        self.entryTopic.grid(row=2, column=0)

        #buton adaugare topic
        self.buttonAdd = Button(self.root, text=" ADD", bg="black", fg="white", command=self.add)
        self.buttonAdd.grid(row=2, column=2)

        #buton delete
        button = Button(self.root, text=" DELETE", bg="black", fg="white", command=self.delete_topic)
        button.grid(row=2, column=3)

        #buton subscribe
        button = Button(self.root, text=" SUBSCRIBE", bg="black", fg="white", command=self.subscribe_topic)
        button.grid(row=2, column=4)

        #buton unsubscribe
        button = Button(self.root, text=" UNSUBSCRIBE", bg="black", fg="white", command=self.unsubscribe_topic)
        button.grid(row=2, column=5)

        self.root.mainloop()

    #adaugare topic
    def add(self):
        self.v.append(self.entryTopic.get())
        self.textbox= Listbox(self.root, height=10, width=30)
        self.textbox.grid(row=5, column=0)

        for i in self.v:
            self.textbox.insert(END, i+'\n')

    #stergere topic
    def delete_topic(self):
        sel = self.textbox.curselection()
        for index in sel[::-1]:
            self.textbox.delete(index)

    #subscribe la topic
    def subscribe_topic(self):
        sel = self.textbox.curselection()

    #unsubscribe la topic
    def unsubscribe_topic(self):
        sel = self.textbox.curselection()

```

9. Bibliografie

- <https://mqtt.org/>
- <https://www.hivemq.com/mqtt-essentials/>
- <https://developer.ibm.com/articles/iot-mqtt-why-good-for-iot/>
- <https://www.automatizari-scada.ro/conectivitate-iot/ce-este-mqtt/?fbclid=IwAR163KNvj7j0rdfkmt8GjZatS1Sgj36chjIQVqVFRCAz841zZpJLKvMNlF4>