# Laporan Praktikum

**MQTT SERVER**


## Internet Of Things

**Diva Lorenza Marbun**
**11323043**
**D3 Teknologi Informasi**

**INSTITUT TEKNOLOGI DEL**
**FAKULTAS VOKASI**

# Tugas

## 1. Imports

```
import json
from topic import Topic
from data_classes import BrokerSettings, ClientSettings
```

json: Digunakan untuk memproses file konfigurasi yang berbentuk JSON.
Topic: Kelas yang mengelola setiap topik dalam sistem.
BrokerSettings, ClientSettings: Kelas untuk menyimpan pengaturan broker dan klien (seperti URL, port, QoS, dll).

2. Konstruktor _init_

```
def __init__(self, settings_file):
    self.default_client_settings = ClientSettings(
        clean=True,
        retain=False,
        qos=2,
        time_interval=10
    )
    self.topics = self.load_topics(settings_file)
```

Di dalam konstruktor, kita membuat objek default_client_settings yang berisi pengaturan default untuk klien MQTT.
Kemudian memanggil load_topics untuk memuat topik-topik dari file konfigurasi yang diberikan.

3. Fungsi read_client_settings

```
def read_client_settings(self, settings_dict: dict, default: ClientSettings):
    return ClientSettings(
        clean=settings_dict.get('CLEAN_SESSION', default.clean),
        retain=settings_dict.get('RETAIN', default.retain),
        qos=settings_dict.get('QOS', default.qos),
        time_interval= settings_dict.get('TIME_INTERVAL', default.time_interval)
```

Fungsi ini digunakan untuk mengambil pengaturan klien dari dictionary (settings_dict). Jika pengaturan tersebut tidak ada, maka nilai default akan digunakan.
Mengembalikan objek ClientSettings yang berisi pengaturan-pengaturan seperti clean, retain, qos, dan time_interval.

4. Fungsi load_topics

```python
def load_topics(self, settings_file):
    topics = []
    with open(settings_file) as json_file:
        config = json.load(json_file)
        broker_settings = BrokerSettings(
            url=config.get('BROKER_URL', 'localhost'),
            port=config.get('BROKER_PORT', 1883),
            protocol=config.get('PROTOCOL_VERSION', 4) # mqtt.MQTTv311
        )
        broker_client_settings = self.read_client_settings(config, default=self.default_client_settings)
        # read each configured topic
        for topic in config['TOPICS']:
            topic_data = topic['DATA']
            topic_payload_root = topic.get('PAYLOAD_ROOT', {})
            topic_client_settings = self.read_client_settings(topic, default=broker_client_settings)
            if topic['TYPE'] == 'single':
                # create single topic with format: /{PREFIX}
                topic_url = topic['PREFIX']
                topics.append(Topic(broker_settings, topic_url, topic_data, topic_payload_root, topic_client_settings))
            elif topic['TYPE'] == 'multiple':
                # create multiple topics with format: /{PREFIX}/{id}
                for id in range(topic['RANGE_START'], topic['RANGE_END']+1):
                    topic_url = topic['PREFIX'] + '/' + str(id)
                    topics.append(Topic(broker_settings, topic_url, topic_data, topic_payload_root, topic_client_settings))
            elif topic['TYPE'] == 'list':
                # create multiple topics with format: /{PREFIX}/{item}
                for item in topic['LIST']:
                    topic_url = topic['PREFIX'] + '/' + str(item)
                    topics.append(Topic(broker_settings, topic_url, topic_data, topic_payload_root, topic_client_settings))
    return topics
```

- Fungsi ini membuka file konfigurasi JSON dan membaca pengaturan broker serta klien.
- Kemudian mengiterasi setiap topik di dalam file dan membuat objek Topic sesuai dengan jenis topik (single, multiple, atau list).
- Topik ditambahkan ke dalam daftar topics berdasarkan format yang sesuai.

5. Fungsi run

```python
def run(self):
    for topic in self.topics:
        print(f'Starting: {topic.topic_url} ...')
        topic.start()
    for topic in self.topics:
        # workaround for Python 3.12
        topic.join()
```

- Fungsi ini memulai setiap topik dengan memanggil metode start() pada objek Topic.
- Setelah itu, join() dipanggil untuk memastikan bahwa eksekusi setiap topik selesai sebelum melanjutkan.

6. Fungsi stop :

```python
def stop(self):
    for topic in self.topics:
        print(f'Stopping: {topic.topic_url} ...')
        topic.stop()
```

- Fungsi ini menghentikan setiap topik dengan memanggil metode stop() pada objek Topic.

7. Pengaturan Topik :
    - **Single**: Topik hanya memiliki satu URL dengan format /PREFIX.
    - **Multiple**: Membuat beberapa topik dengan format /PREFIX/id di mana id berada dalam rentang yang ditentukan.
    - **List**: Membuat beberapa topik dengan format /PREFIX/item di mana item diambil dari daftar.
    - Contoh implementasi topik berdasarkan jenisnya:

```
if topic['TYPE'] == 'single':
    topic_url = topic['PREFIX']
    topics.append(Topic(broker_settings, topic_url, topic_data, topic_payload_root, topic_client_settings))
elif topic['TYPE'] == 'multiple':
    for id in range(topic['RANGE_START'], topic['RANGE_END'] + 1):
        topic_url = f"{topic['PREFIX']}/{id}"
        topics.append(Topic(broker_settings, topic_url, topic_data, topic_payload_root, topic_client_settings))
elif topic['TYPE'] == 'list':
    for item in topic['LIST']:
        topic_url = f"{topic['PREFIX']}/{item}"
        topics.append(Topic(broker_settings, topic_url, topic_data, topic_payload_root, topic_client_settings))
```

**Kesimpulan :**

- Kode ini digunakan untuk mensimulasikan sistem yang bekerja dengan banyak topik MQTT.
- Berdasarkan file konfigurasi JSON, simulator akan:
- Membaca pengaturan klien dan broker.
- Membuat dan memulai topik-topik berdasarkan jenis yang ditentukan.
- Menyediakan fungsi untuk memulai dan menghentikan topik-topik tersebut.