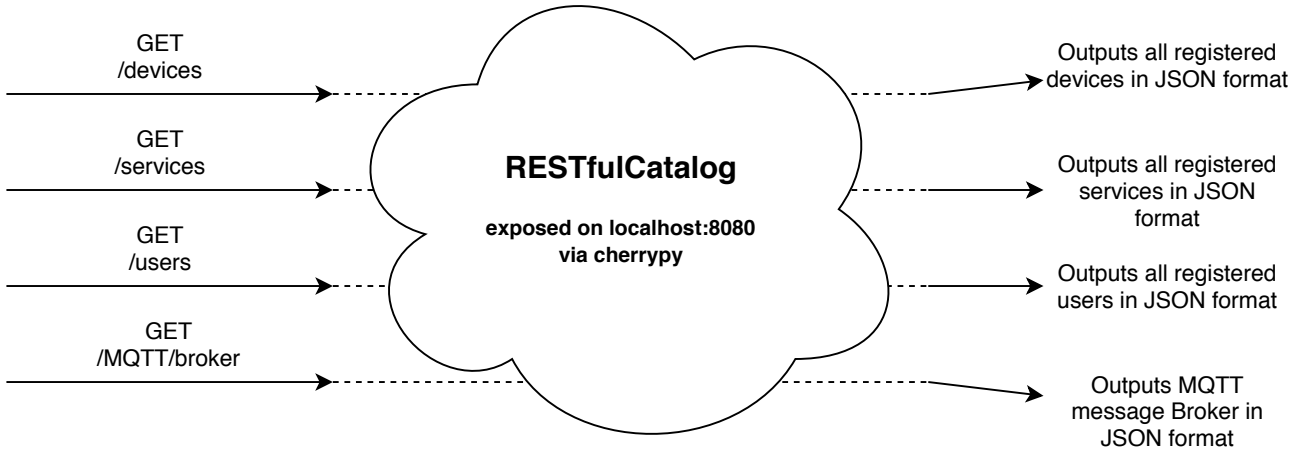
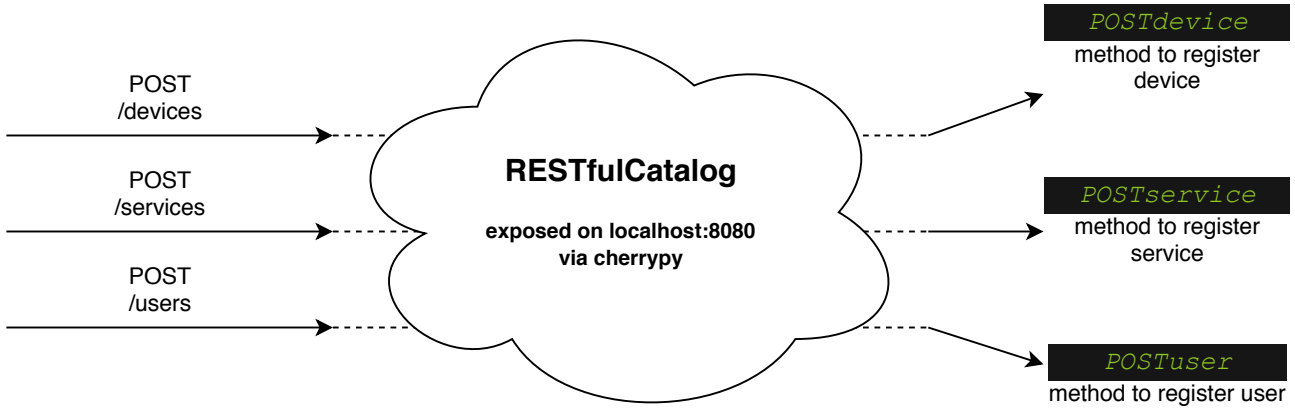
GET REQUESTS SCHEMA



Additionally, every get request accepts **id** parameter to which the Catalog will respond with the corresponding entry.

POST REQUESTS SCHEMA



Every post request must contain the header `Content-Type: application/json` to be successfully processed.

Body of the request must contain the data needed to be registered in the correct JSON format.

DATA FORMATS

DEVICE data format INPUT (POST BODY)

```
{
    "resources": [],
    "end-points": {
        "rest": [],
        "mqtt-
topics": []
    }
}
```

DEVICE data format OUTPUT (GET RESPONSE)

```
{
    "ip_address": {
        "resources": [],
        "end-points": {
            "rest": [],
            "mqtt-topics": []
        },
        "insert-timestamp":
0.0
    }
}
```

Internally devices are identified by their own **IP address** on the local network

SERVICE data format INPUT (POST BODY)

```
{

"description": "Service",
        "end-points": {
            "rest": [],
            "mqtt-
topics": []
        }
}
```

SERVICE data format OUTPUT (GET RESPONSE)

```
{
    "id": {
        "description": "Service",
        "end-points": {
            "rest": [],
            "mqtt-topics": []
        },
        "insert-timestamp": 0.0
    }
}
```

Internally services are identified by a progressive integer, **id** of deleted services is reused.

USER data format INPUT (POST BODY)

```
{
    "name": "mario",
    "surname": "rossi",
    "email": [
        "email1@g.com",
        "email@g.com"
    ]
}
```

USER data format OUTPUT (GET RESPONSE)

```
{
    "id": {
        "0": {
            "name": "mario",
            "surname": "rossi",
            "email": [
                "email1@g.com",
                "email@g.com"
            ]
        }
    }
}
```

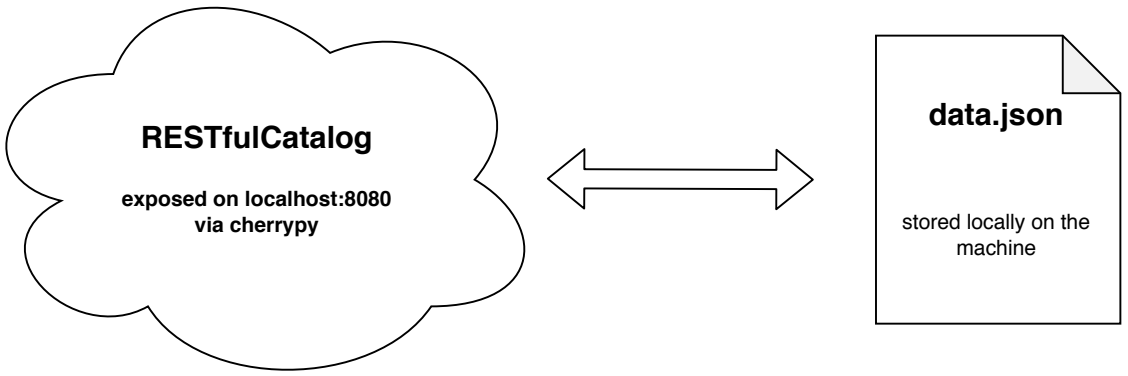
Internally users are identified by a progressive integer.

Registration of users with at least one email already existing in the database is negated, error code **`409 Conflict`** is raised.

MQTT message Broker data format OUTPUT (GET RESPONSE)

```
{
    "broker": {
        "address": "test.mosquitto.org",
        "port": 1883
    }
}
```

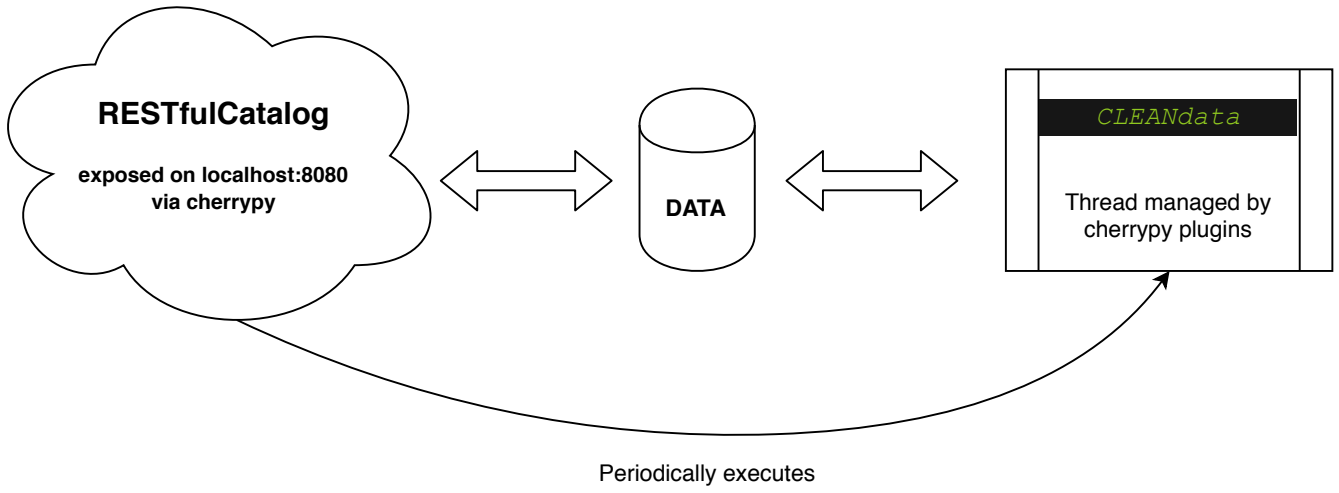
INTERNAL DATA MANAGEMENT



Devices, services and users data is saved in a **python dictionary structure**.

Whenever those data are updated (POST request, Timeout cleanup) the entire content of the dictionary is written to the local file.

CLEANUP



The cleaner thread periodically checks insertion times of services and devices, deleting every entry older than a certain threshold.

Access to the data is mutually exclusive between the catalog and the cleaner thread through a semaphore, to avoid incongruencies.

DEBUG CLIENT

The debug client is developed to add, on user request, random entries to the catalog through POST method. It also fetches data through GET method, providing the possibility to fetch single entries by their ID.