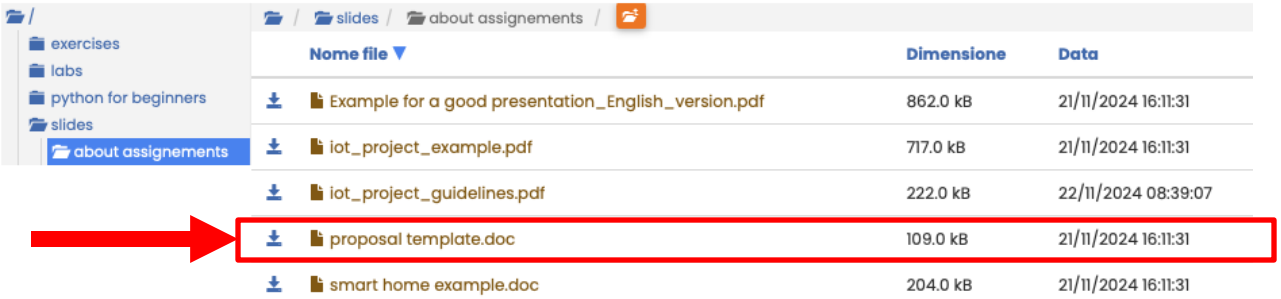
Projects Good Practices

Project Proposal

Projects Proposals

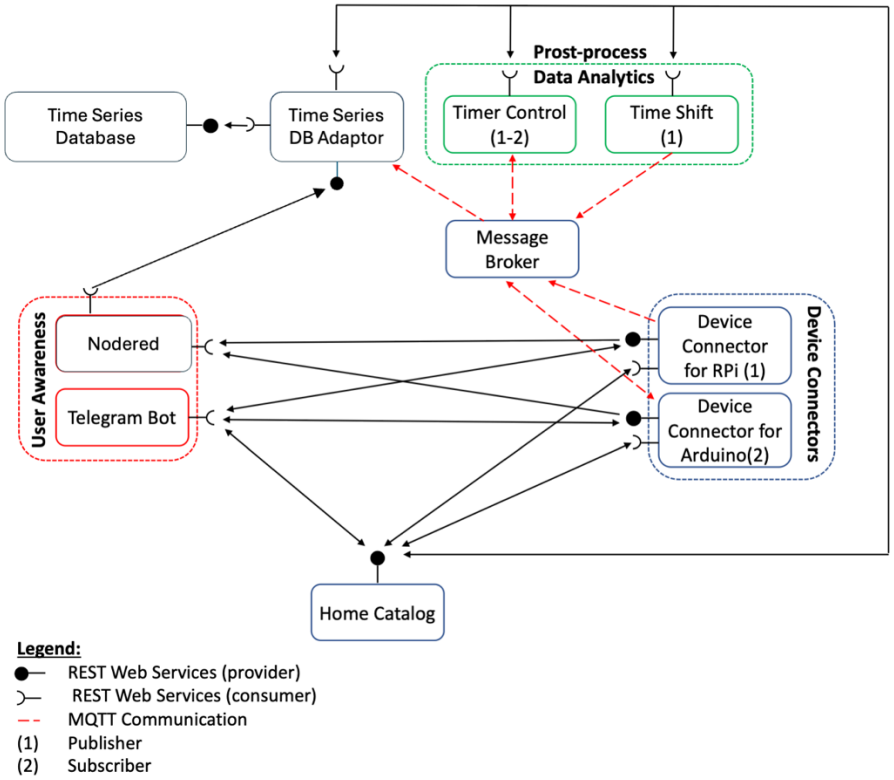
- Use the shared Template to fill out the proposal



- Example of Proposal as guideline

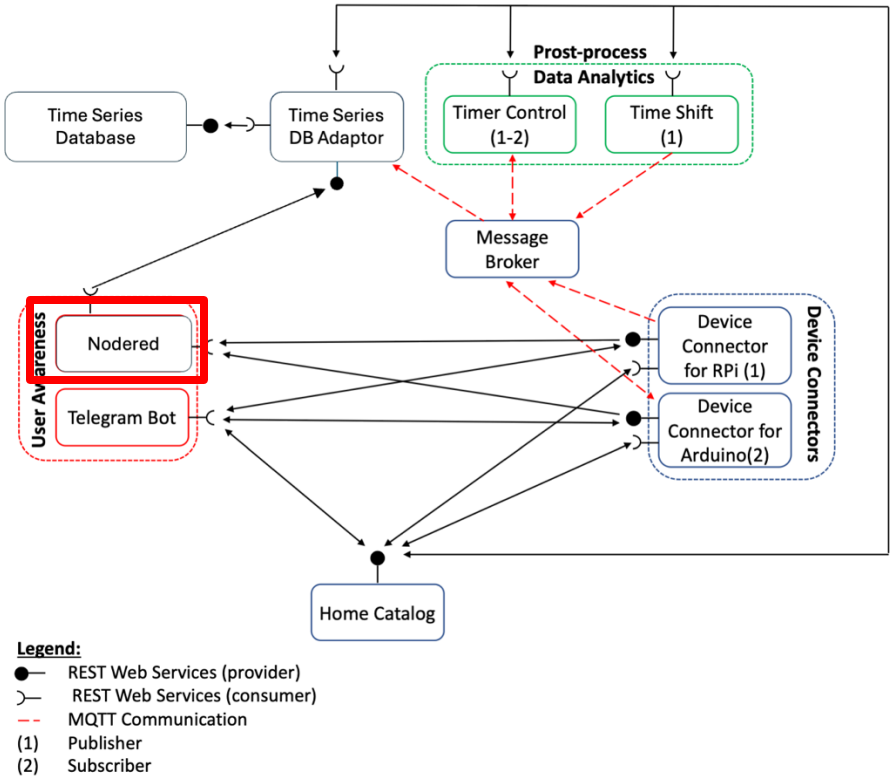
Projects Proposals

- Example of Proposal as guideline



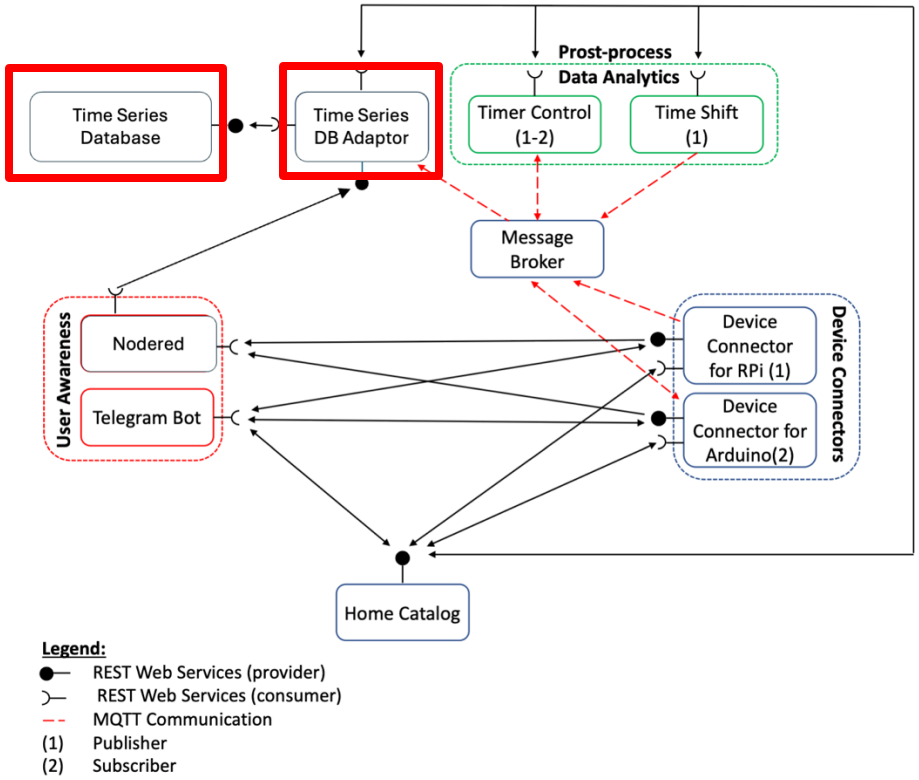
Projects Proposals

- Example of Proposal as guideline
  - Nodered instead of Freeboard



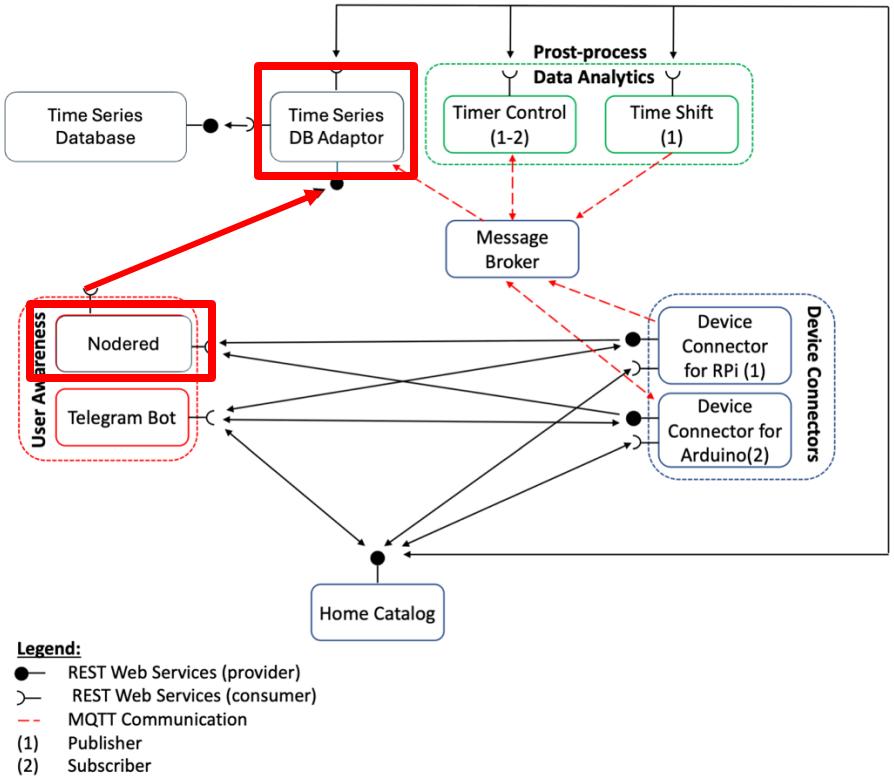
Projects Proposals

- Example of Proposal as guideline
  - Nodered instead of Freeboard
  - Third-Party services with connector



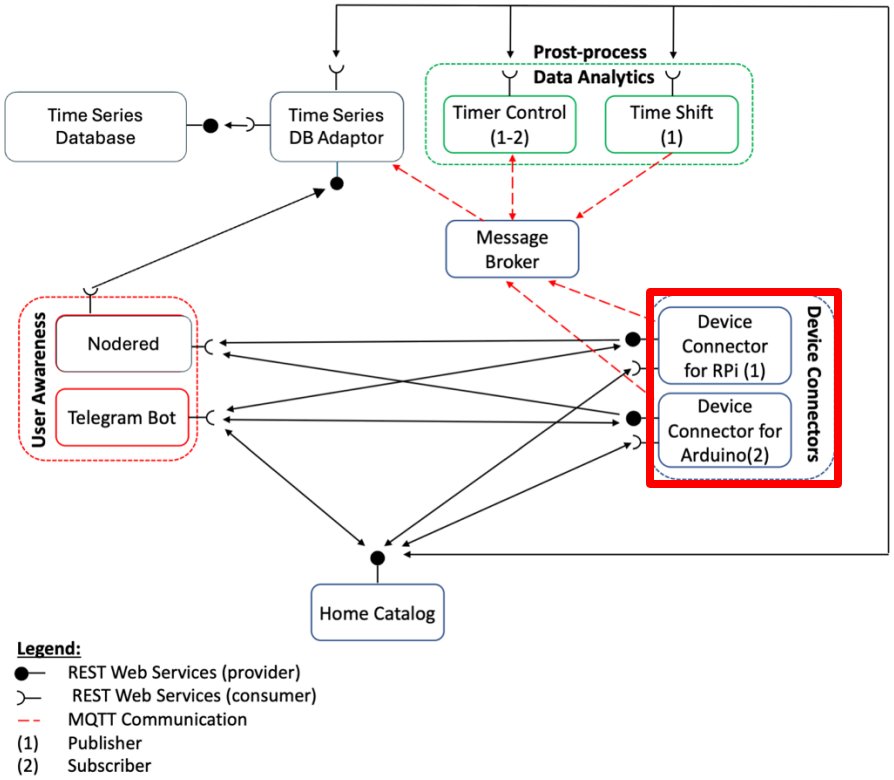
Projects Proposals

- Example of Proposal as guideline
  - Nodered instead of Freeboard
  - Third-Party services with connector
    - Connections of other services to the adaptor



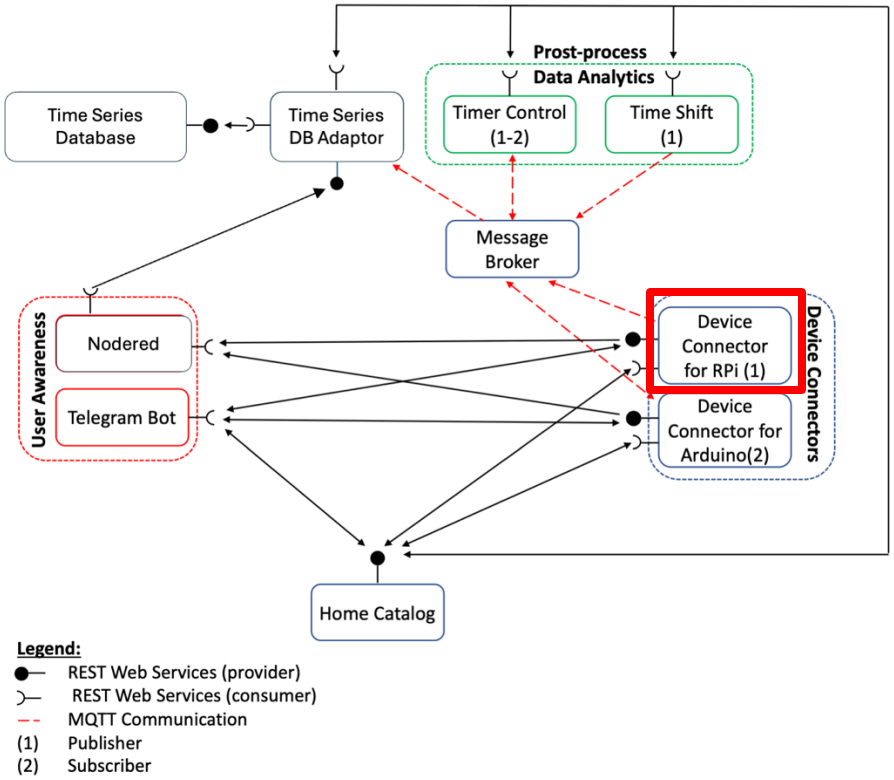
Projects Proposals

- Example of Proposal as guideline
  - Nodered instead of Freeboard
  - Third-Party services with connector
    - Connections of other services to the adaptor
  - Device vs Device Connector



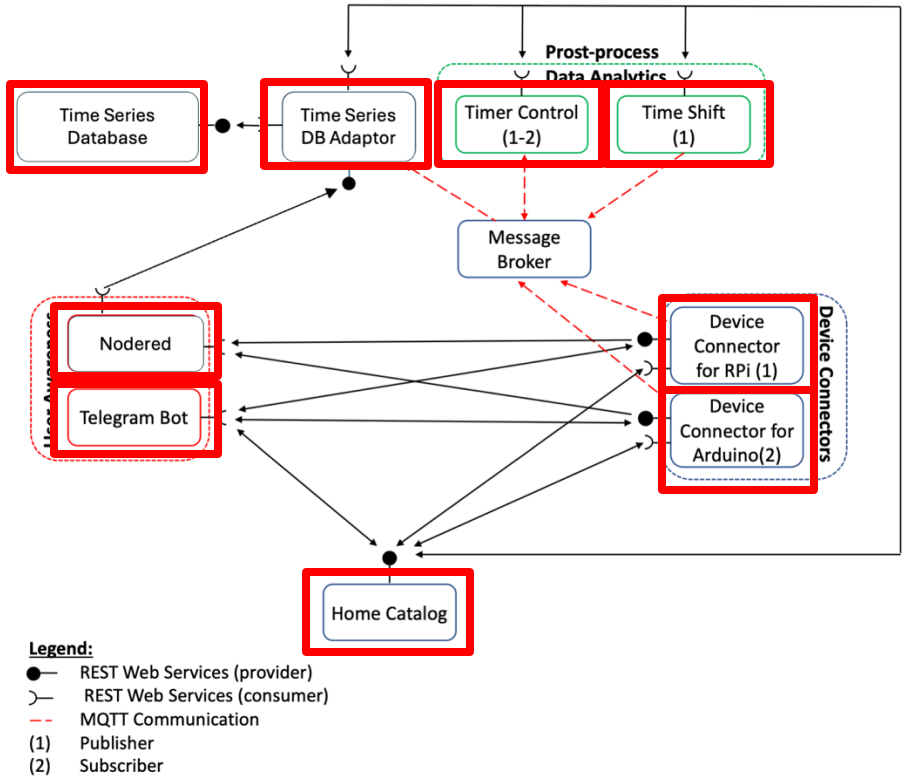
Projects Proposals

- Example of Proposal as guideline
  - Nodered instead of Freeboard
  - Third-Party services with connector
    - Connections of other services to the adaptor
  - Device vs Device Connector
    - Raspberry Pi as a DEVICE CONNECTOR
      - NOT for running the other microservices
      - CAN use other HARDWARE (e.g. Arduino)



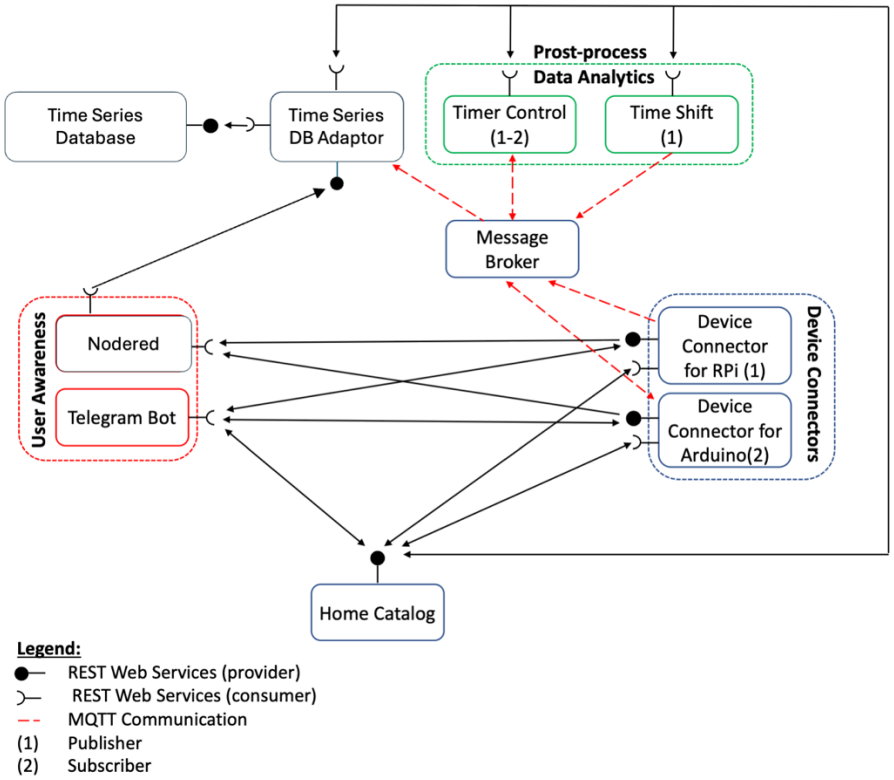
Projects Proposals

- Example of Proposal as guideline
  - ⚠️ ⚠️ DESCRIBE ALL MICROSERVICES !! ⚠️ ⚠️
    - Input -> data and communication protocol
    - Process of the information (if any)
    - Output -> data and communication protocol



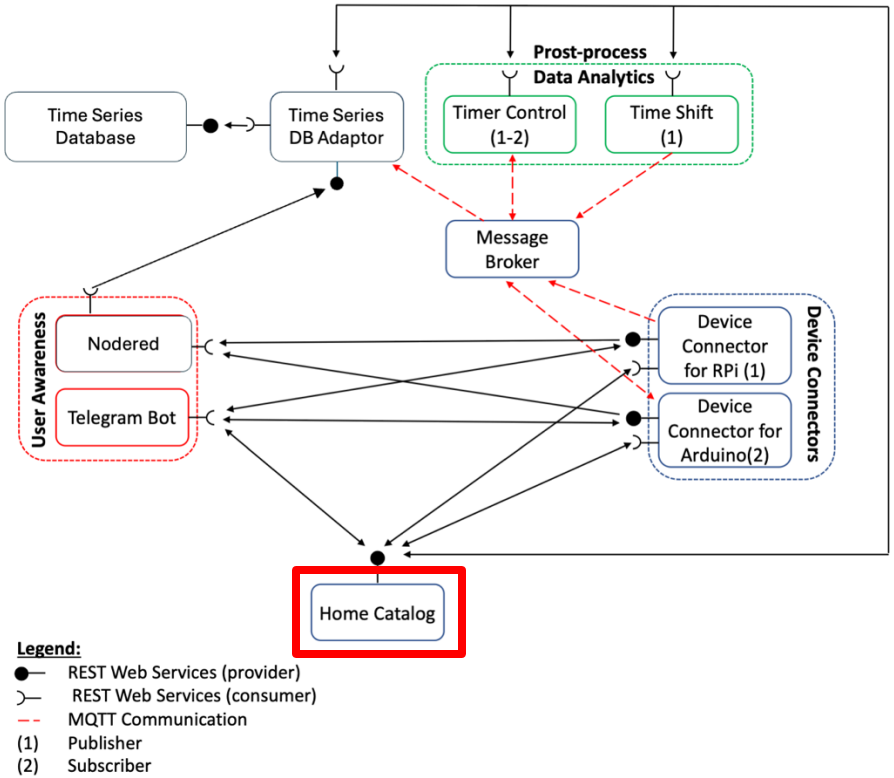
Projects Proposals

- Example of Proposal as guideline
  - ⚠️ ⚠️ DESCRIBE ALL MICROSERVICES !! ⚠️ ⚠️
    - Input -> data and communication protocol
    - Process of the information (if any)
    - Output -> data and communication protocol
  - MAKE SURE correct description according to USE CASE DIAGRAM
    - Corresponding REST and MQTT Connections



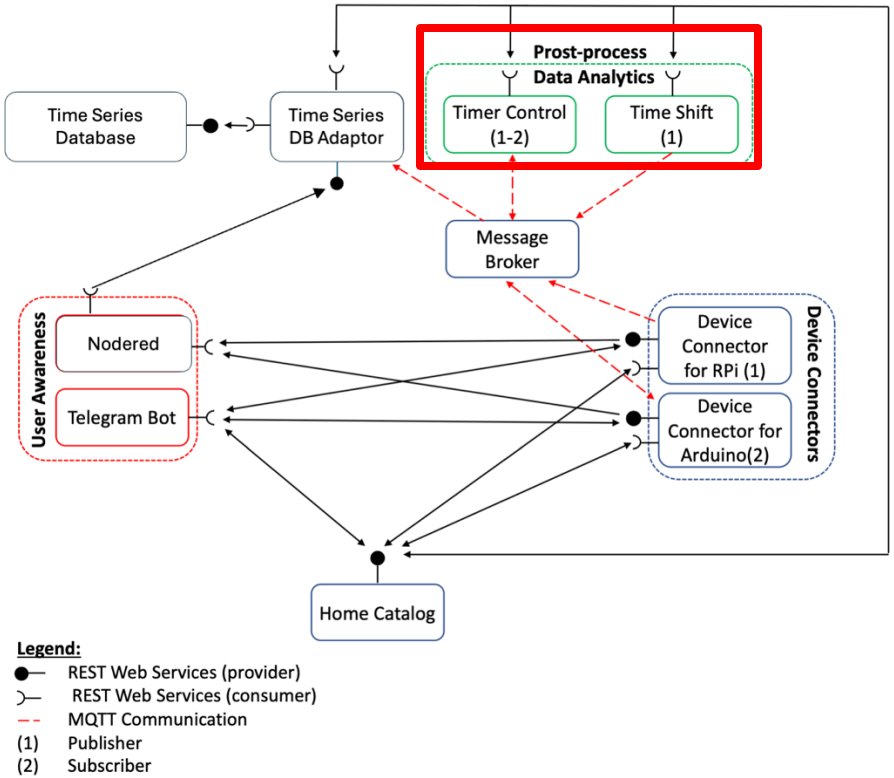
Projects Proposals

- Example of Proposal as guideline
  - ⚠️ ⚠️ DESCRIBE ALL MICROSERVICES !! ⚠️ ⚠️
    - Input -> data and communication protocol
    - Process of the information (if any)
    - Output -> data and communication protocol
  - MAKE SURE correct description according to USE CASE DIAGRAM
    - Corresponding REST and MQTT Connections
  - CATALOG
    - Can have one unique catalog, or a SERVICE catalog and a RESOURCE catalog
    - ⚠️ ⚠️ ALL MICROSERVICES / RESOURCES MUST REGISTER TO THE CATALOG ⚠️ ⚠️



Projects Proposals

- Example of Proposal as guideline
  - Data Analytics MUST BE COMPLEX ENOUGH!!
    - NOT just if/else (i.e. thresholds)
    - Make use of historical data



Projects Proposals – Final Remarks

- Project MUST BE SCALABLE
  - Example of Smart Home
    - Multiple Homes
    - Multiple Users
    - Multiple Devices

Projects Proposals – Final Remarks

- Project MUST BE SCALABLE
  - Example of Smart Home
    - Multiple Homes
    - Multiple Users
    - Multiple Devices

- Proposal is LIKE A CONTRACT !!!
  - Once approved, the final delivery MUST BE at least as complex as indicated in the proposal

Projects Proposals – Final Remarks

- Project MUST BE SCALABLE
  - Example of Smart Home
    - Multiple Homes
    - Multiple Users
    - Multiple Devices

- Proposal is LIKE A CONTRACT !!!
  - Once approved, the final delivery MUST BE at least as complex as indicated in the proposal

- **PLEASE CHECK GRAMMAR AND SPELLING AND READ THE PROPOSAL BEFORE SENDING IT !!!!!!!!**

Projects Proposals – Final Remarks

- Project MUST BE SCALABLE
  - Example of Smart Home
    - Multiple Homes
    - Multiple Users
    - Multiple Devices

- Proposal is LIKE A CONTRACT !!!
  - Once approved, the final delivery MUST BE at least as complex as indicated in the proposal

- **PLEASE CHECK GRAMMAR AND SPELLING AND READ THE PROPOSAL BEFORE SENDING IT !!!!!!!!**

Project Development

Tips and Common mistakes

Projects Development

- Catalog
  - All 4 REST methods (GET, POST, PUT, DELETE)
- Correlation among entities (e.g. Users / Devices / Houses)

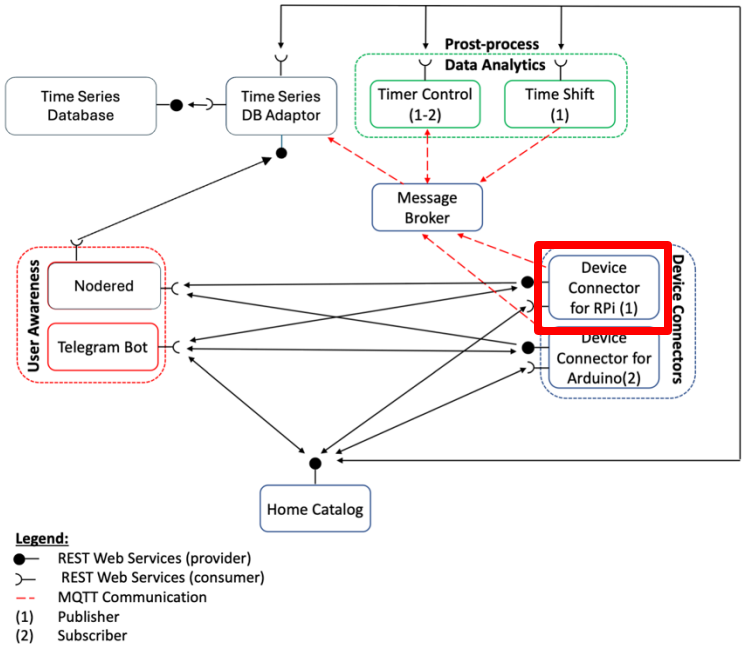

```
"usersList": [
  {
    "userName": "Jake Blues",
    "userID": 1,
    "chatID": 123456,
    "houses": [
      {
        "houseID": 1
      }
    ]
  },
```



```
"housesList":
  {
    "userID": 1,
    "houseID": 1,
    "devicesList": [
      {
        "deviceID": 1,
        "deviceName": "DHT11",
        "measureType": [
          "Temperature",
          "Humidity"
        ],
        "availableServices": [
          "MQTT",
          "REST"
        ],
        "servicesDetails": [
          {
            "serviceType": "MQTT",
            "topic": [
              "MySmartThingy/1/temp",
              "MySmartThingy/1/hum"
            ]
          },
          {
            "serviceType": "REST",
            "serviceIP": "192.1.1.1:8080"
          }
        ],
        "lastUpdate": "2020-03-30"
      }
    ]
  },
```

Projects Development

- Catalog
  - All 4 REST methods (GET, POST, PUT, DELETE)
- Correlation among entities (e.g. Users / Devices / Houses)
- SenML format
- Raspberry Pi is a DEVICE CONNECTOR
  - **IT IS NOT FOR RUNNING ALL THE MICROSERVICES INSIDE**



Projects Development

- Catalog
  - All 4 REST methods (GET, POST, PUT, DELETE)
- Correlation among entities (e.g. Users / Devices / Houses)
- SenML format
- Raspberry Pi is a DEVICE CONNECTOR
- Project SCALABLE
  - MULTIPLE users, devices, houses
  - NOT restricted by having only ONE Raspberry
    - CAN SIMULATE OTHERS for example

Projects Development

- Catalog
  - All 4 REST methods (GET, POST, PUT, DELETE)
- Correlation among entities (e.g. Users / Devices / Houses)
- SenML format
- Raspberry Pi is a DEVICE CONNECTOR
- Project SCALABLE
  - MULTIPLE users, devices, houses
  - NOT restricted by having only ONE Raspberry
    - CAN SIMULATE OTHERS for example
- MQTT Publisher and Subscriber can be the same microservice!!

```python
class LightActuator:
    def __init__(self,clientID,broker,port,topic_subscribe,topic_publish):
        self.lightStatus = 0
        self.lightClientSub = MyMQTT(clientID,broker,port,self)
        self.__message={'client':clientID,'message':'','timestamp':''}
        self.topic_subscribe = topic_subscribe
        self.topic_publish = topic_publish

    def notify(self,topic,payload):
        message_json = json.loads(payload)
        print(message_json)

    def startSim(self):
        self.lightClientSub.start()
        self.lightClientSub.mySubscribe(self.topic_subscribe)

    def stopSim(self):
        self.lightClientSub.unsubscribe()
        self.lightClientSub.stop()

    def publish(self,message_to_sent):
        message=self.__message
        message['message']=message_to_sent
        message['timestamp']=time.time()
        self.lightClientSub.myPublish(self.topic_publish,message)
        print("Alert Sent!")

if __name__ == "__main__":
    clientID = "rafafontana12131997"
```

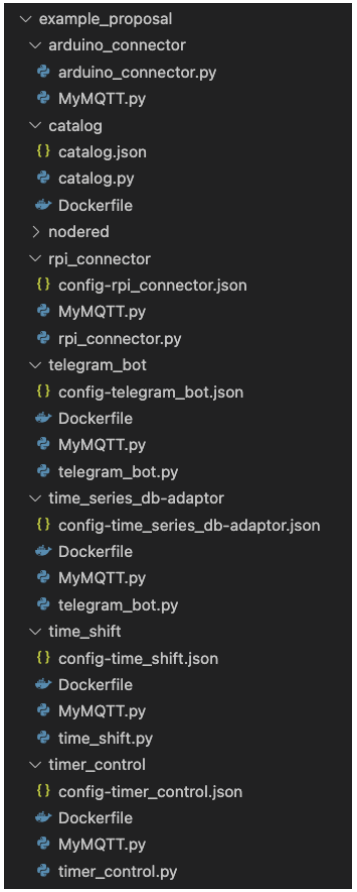
Projects Development

- Catalog
  - All 4 REST methods (GET, POST, PUT, DELETE)
- Correlation among entities (e.g. Users / Devices / Houses)
- SenML format
- Raspberry Pi is a DEVICE CONNECTOR
- Project SCALABLE
  - MULTIPLE users, devices, houses
  - NOT restricted by having only ONE Raspberry
    - CAN SIMULATE OTHERS for example
- MQTT Publisher and Subscriber can be the same microservice!!
  - **DO NOT USE SAME CLIENT ID !!!**
  - **MUST BE UNIQUE FOR EACH MICROSERVICE/DEVICE !!**

```python
class LightActuator:
    def __init__(self,clientID,broker,port,topic_subscribe,topic_publish):
        self.lightStatus = 0
        self.lightClientSub = MyMQTT(clientID,broker,port,self)
        self.__message={'client':clientID,'message':'','timestamp':''}
        self.topic_subscribe = topic_subscribe
        self.topic_publish = topic_publish

    def notify(self,topic,payload):
        message_json = json.loads(payload)
        print(message_json)

    def startSim(self):
        self.lightClientSub.start()
        self.lightClientSub.mySubscribe(self.topic_subscribe)

    def stopSim(self):
        self.lightClientSub.unsubscribe()
        self.lightClientSub.stop()

    def publish(self,message_to_sent):
        message=self.__message
        message['message']=message_to_sent
        message['timestamp']=time.time()
        self.lightClientSub.myPublish(self.topic_publish,message)
        print("Alert Sent!")

if __name__ == "__main__":
    clientID = "rafafontana12131997"
```

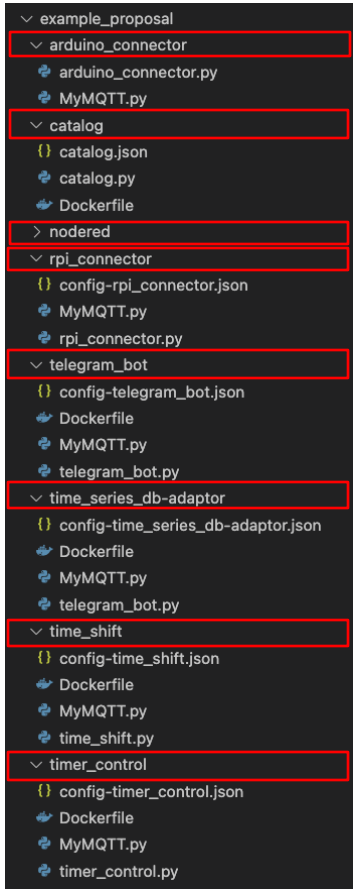
Projects Development

- Catalog
  - All 4 REST methods (GET, POST, PUT, DELETE)
- Correlation among entities (e.g. Users / Devices / Houses)
- SenML format
- Raspberry Pi is a DEVICE CONNECTOR
- Project SCALABLE
  - MULTIPLE users, devices, houses
  - NOT restricted by having only ONE Raspberry
    - CAN SIMULATE OTHERS for example
- MQTT Publisher and Subscriber can be the same microservice!!
  - **DO NOT USE SAME CLIENT ID !!!**
- DOCKER IS MANDATORY !!!
  - One folder and one container for each microservice



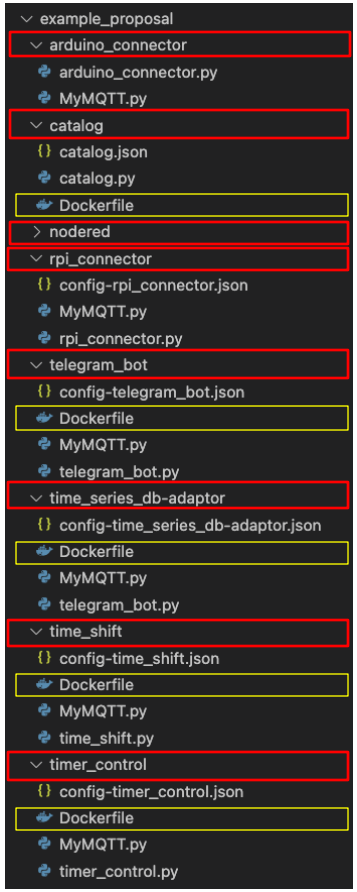
Projects Development

- Catalog
  - All 4 REST methods (GET, POST, PUT, DELETE)
- Correlation among entities (e.g. Users / Devices / Houses)
- SenML format
- Raspberry Pi is a DEVICE CONNECTOR
- Project SCALABLE
  - MULTIPLE users, devices, houses
  - NOT restricted by having only ONE Raspberry
    - CAN SIMULATE OTHERS for example
- MQTT Publisher and Subscriber can be the same microservice!!
  - **DO NOT USE SAME CLIENT ID !!!**
- DOCKER IS MANDATORY !!!
  - One folder and one container for each microservice

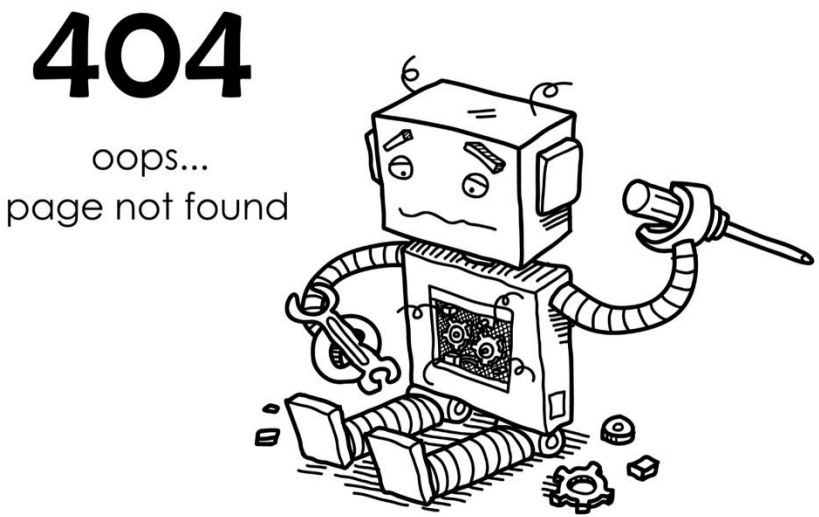


Projects Development

- Catalog
  - All 4 REST methods (GET, POST, PUT, DELETE)
- Correlation among entities (e.g. Users / Devices / Houses)
- SenML format
- Raspberry Pi is a DEVICE CONNECTOR
- Project SCALABLE
  - MULTIPLE users, devices, houses
  - NOT restricted by having only ONE Raspberry
    - CAN SIMULATE OTHERS for example
- MQTT Publisher and Subscriber can be the same microservice!!
  - **DO NOT USE SAME CLIENT ID !!!**
- DOCKER IS MANDATORY !!!
  - One folder and one container for each microservice



Projects Development – Common Errors



Projects Development – Common Errors

- Hardcoded Information
  - **USE CONFIGURATION FILES**

```
timer_control
{} config-timer_control.json
Dockerfile
MyMQTT.py
timer_control.py
```

```python
class TimerControl():
    def __init__(self,clientid,broker,port,topic,catalog_url):
        self.catalog_url=catalog_url
        self.client=MyMQTT(clientid,broker,port,self)
        self.topic=topic
```

```python
if __name__=="__main__":
    catalog_website="http://catalog.com:8080/"
    client_id= "clientIDrandom0010204"
    broker = "mqtt.eclipseprojects.io"
    port = 1883
    topic = "IoT/sensor_id/measurements"
    timer_control_instance=TimerControl(client_id,broker,port,topic,catalog_website)
    timer_control_instance.start()
    while True:
        ....
```

Projects Development – Common Errors

- Hardcoded Information
  - **USE CONFIGURATION FILES**

```
timer_control
{} config-timer_control.json
Dockerfile
MyMQTT.py
timer_control.py
```

```python
class TimerControl():
    def __init__(self,clientid,broker,port,topic,catalog_url):
        self.catalog_url=catalog_url
        self.client=MyMQTT(clientid,broker,port,self)
        self.topic=topic
```

```python
if __name__=="__main__":
    catalog_website="http://catalog.com:8080/"
    client_id= "clientIDrandom0010204"
    broker = "mqtt.eclipseprojects.io"
    port = 1883
    topic = "IoT/sensor_id/measurements"
    timer_control_instance=TimerControl(client_id,broker,port,topic,catalog_website)
    timer_control_instance.start()
    while True:
        ....
```

Projects Development – Common Errors

- Hardcoded Information
  - **USE CONFIGURATION FILES**

```
timer_control
{} config-timer_control.json
Dockerfile
MyMQTT.py
timer_control.py
```

```
class TimerControl():
    def __init__(self,clientid,broker,port,topic,catalog_url):
        self.catalog_url=catalog_url
        self.client=MyMQTT(clientid,broker,port,self)
        self.topic=topic
```

```
if __name__=="__ma
    catalog_webs
    client_id= "c
    broker = "mqtt.ec
    port = 1883
    topic = "IoT/sensor
    timer_control_in
    timer_control
    while True:
        ....
```

Projects Development – Common Errors

- Hardcoded Information
  - **USE CONFIGURATION FILES**

```
timer_control
{} config-timer_control.json
Dockerfile
MyMQTT.py
timer_control.py
```

```
{
    "catalog_website":"http://catalog.com:8080/",
    "mqtt_info":{
        "clientID":"clientIDrandom0010204",
        "broker":"mqtt.eclipseprojects.io",
        "port":1883,
        "topic_subscribe":"IoT/sensor_id/measurements"
    }
}
```

```
class TimerControl():
    def __init__(self,clientid,broker,port,topic,catalog_url):
        self.catalog_url=catalog_url
        self.client=MyMQTT(clientid,broker,port,self)
        self.topic=topic
```

```
if __name__=="__ma
    catalog_webs
    client_id= "c
    broker = "mqtt.ec
    port = 1883
    topic = "IoT/sensor
    timer_control_in
    timer_control
    while True:
        ....
```

Projects Development – Common Errors

- Hardcoded Information
  - **USE CONFIGURATION FILES**

```
timer_control
{} config-timer_control.json
Dockerfile
MyMQTT.py
timer_control.py
```

```
{
    "catalog_website":"http://catalog.com:8080/",
    "mqtt_info":{
        "clientID":"clientIDrandom0010204",
        "broker":"mqtt.eclipseprojects.io",
        "port":1883,
        "topic_subscribe":"IoT/sensor_id/measurements"
    }
}
```

```
class TimerControl():
    def __init__(self,clientid,broker,port,topic,catalog_url):
        self.catalog_url=catalog_url
        self.client=MyMQTT(clientid,broker,port,self)
        self.topic=topic
```

```
if __name__=="__ma
    catalog_webs
    client_id= "c
    broker = "mqtt.ec
    port = 1883
    topic = "IoT/sensor
    timer_control_in
    timer_control
    while True:
        ...
```

```
if __name__=="__main__":
    settings=json.load(open("config-timer_control.json"))
    catalog_website=settings["catalog_website"]
    client_id= settings["mqtt_info"]["clientID"]
    broker = settings["mqtt_info"]["broker"]
    port = settings["mqtt_info"]["port"]
    topic = settings["mqtt_info"]["topic_subscribe"]
    timer_control_instance=TimerControl(client_id,broker,port,topic,catalog_website)
    timer_control_instance.start()
    while True:
        ...
```

Projects Development – Common Errors

- Hardcoded Information
  - **USE CONFIGURATION FILES**
- Wrong Key-Value Pairs in catalog

```
"servicesList": {
  "time_shift":{
    "REST_endpoint":"",
    "MQTT_topic": "",
    "timestamp": ""
  },
  "telegram_bot":{
    "REST_endpoint":"",
    "MQTT_topic": "",
    "timestamp": "",
    "token": ""
  }
},
"usersList":{
  "user1":{
    "name":"Jake Blues",
    "userID": 1,
    "chatID": 123456,
    "houses": []
  },
  "user2":{
    "name":"Elwood Blues",
    "userID": 2,
    "chatID": 678901,
    "houses": []
  }
}
```

Projects Development – Common Errors

- Hardcoded Information
  - **USE CONFIGURATION FILES**
- Wrong Key-Value Pairs in catalog

```
"servicesList": {
  "time_shift":{
    "REST_endpoint":"",
    "MQTT_topic": "",
    "timestamp": ""
  },
  "telegram_bot":{
    "REST_endpoint":"",
    "MQTT_topic": "",
    "timestamp": "",
    "token": ""
  }
},
"usersList":{
  "user1":{
    "name":"Jake Blues",
    "userID": 1,
    "chatID": 123456,
    "houses": []
  },
  "user2":{
    "name":"Elwood Blues",
    "userID": 2,
    "chatID": 678901,
    "houses": []
  }
}
```

Projects Development – Common Errors

- Hardcoded Information
  - **USE CONFIGURATION FILES**
- Wrong Key-Value Pairs in catalog

```
"servicesList": {
  "time_shift":{
    "REST_endpoint":"",
    "MQTT_topic": "",
    "timestamp": ""
  },
  "telegram_bot":{
    "REST_endpoint":"",
    "MQTT_topic": "",
    "timestamp": "",
    "token": ""
  }
},
"usersList":{
  "user1":{
    "name":"Jake Blues",
    "userID": 1,
    "chatID": 123456,
    "houses": []
  },
  "user2":{
    "name":"Elwood Blues",
    "userID": 2,
    "chatID": 678901,
    "houses": []
  }
}
```

Projects Development – Common Errors

- Hardcoded Information
  - **USE CONFIGURATION FILES**
- Wrong Key-Value Pairs in catalog

```
"servicesList": {
  "time_shift":{
    "REST_endpoint":"",
    "MQTT_topic": "",
    "timestamp": ""
  },
  "telegram_bot":{
    "REST_endpoint":"",
    "MQTT_topic": "",
    "timestamp": "",
    "token": ""
  }
},
"usersList":{
  "user1":{
    "name":"Jake Blues",
    "userID": 1,
    "chatID": 123456,
    "houses": []
  },
  "user2":{
    "name":"Elwood Blues",
    "userID": 2,
    "chatID": 678901,
    "houses": []
  }
}
```

Projects Development – Common Errors

- Hardcoded Information
  - **USE CONFIGURATION FILES**
- Wrong Key-Value Pairs in catalog

```
"servicesList": {
  "time_shift":{
    "REST_endpoint":"",
    "MQTT_topic": "",
    "timestamp": ""
  }
    legram_bot":{
      T_endpoint"
      topic":
      p"

"u
  "

      :{
      me":"Elwood B
      "userID": 2,
      "chatID": 678901,
      "houses": []
    }
}
```

```
"servicesList":[
  {
    "serviceID":"Time Shift 001",
    "REST_endpoint":"",
    "MQTT_topic": "",
    "timestamp": ""
  },
  {
    "serviceID":"Telegram Bot",
    "REST_endpoint":"",
    "MQTT_topic": "",
    "timestamp": "",
    "token": ""
  }
],
"usersList": [
  {
    "userName": "Jake Blues",
    "userID": 1,
    "chatID": 123456,
    "houses": []
  },
  {
    "userName": "Elwood Blues",
    "userID": 2,
    "chatID": 678901,
    "houses": []
  }
],
```

Projects Development – Common Errors

- Hardcoded Information
  - **USE CONFIGURATION FILES**
- Wrong Key-Value Pairs in catalog

```
"servicesList": {
  "time_shift":{
    "REST_endpoint":"",
    "MQTT_topic": "",
    "timestamp": ""
  }
    legram_bot":{
      T_endpoint"
      topic"
      p"

"u
  "
:{
me":"Elwood B
"userID": 2,
"chatID": 678901,
"houses": []
}
}
```

```
"servicesList":[
  {
    "serviceID":"Time Shift 001",
    "REST_endpoint":"",
    "MQTT_topic": "",
    "timestamp": ""
  },
  {
    "serviceID":"Telegram Bot",
    "REST_endpoint":"",
    "MQTT_topic": "",
    "timestamp": "",
    "token": ""
  }
],
"usersList": [
  {
    "userName": "Jake Blues",
    "userID": 1,
    "chatID": 123456,
    "houses": []
  },
  {
    "userName": "Elwood Blues",
    "userID": 2,
    "chatID": 678901,
    "houses": []
  }
],
```

Projects Development – Common Errors

- Hardcoded Information
  - **USE CONFIGURATION FILES**
- Wrong Key-Value Pairs in catalog

```
"servicesList": {
  "time_shift":{
    "REST_endpoint":"",
    "MQTT_topic": "",
    "timestamp": ""
  }
    legram_bot":{
      T_endpoint"
      topic":
      p"

"u
  "
        :{
      me":"Elwood B
      "userID": 2,
      "chatID": 678901,
      "houses": []
    }
}
```

```
"servicesList":[
  {
    "serviceID": "Time Shift 001",
    "REST_endpoint":"",
    "MQTT_topic": "",
    "timestamp": ""
  },
  {
    "serviceID": "Telegram Bot",
    "REST_endpoint":"",
    "MQTT_topic": "",
    "timestamp": "",
    "token": ""
  }
],
"usersList": [
  {
    "userName": "Jake Blues",
    "userID": 1,
    "chatID": 123456,
    "houses": []
  },
  {
    "userName": "Elwood Blues",
    "userID": 2,
    "chatID": 678901,
    "houses": []
  }
],
```

Projects Development – Common Errors

- Hardcoded Information
  - USE CONFIGURATION FILES
- Wrong Key-Value Pairs in catalog
- NO EXCHANGE OF INFORMATION USING LOCAL FILES !!!!
- NO MONOLITHIC SOFTWARE
- Use Cherrypy!! (in the correct way)