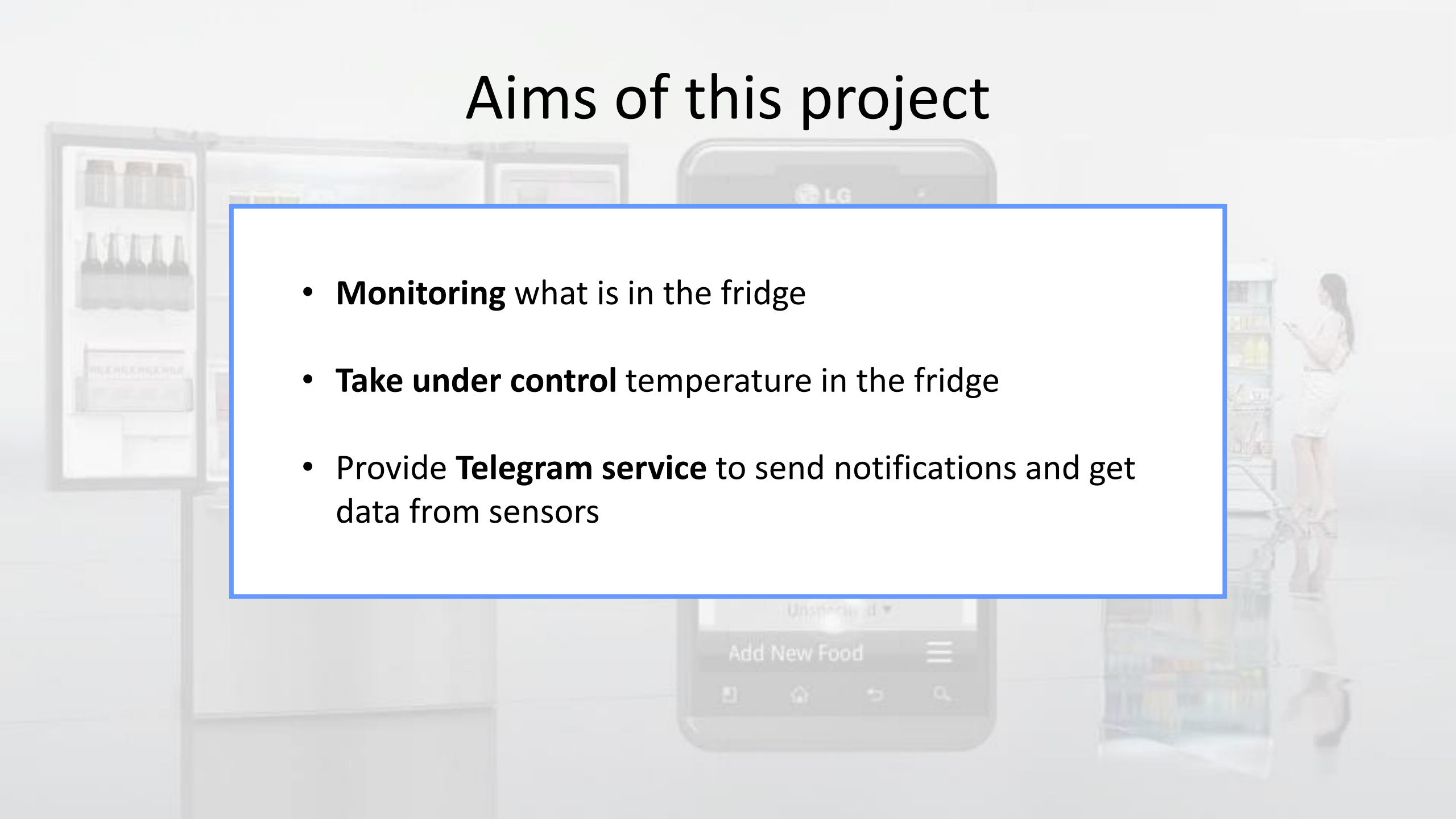
# Smart Refrigerator

## Programming for IoT project

Boretto Luca s244218
Loris Carta s239932
Jarvand Aysan s246437
Toscano Alessia s244143

# Aims of this project

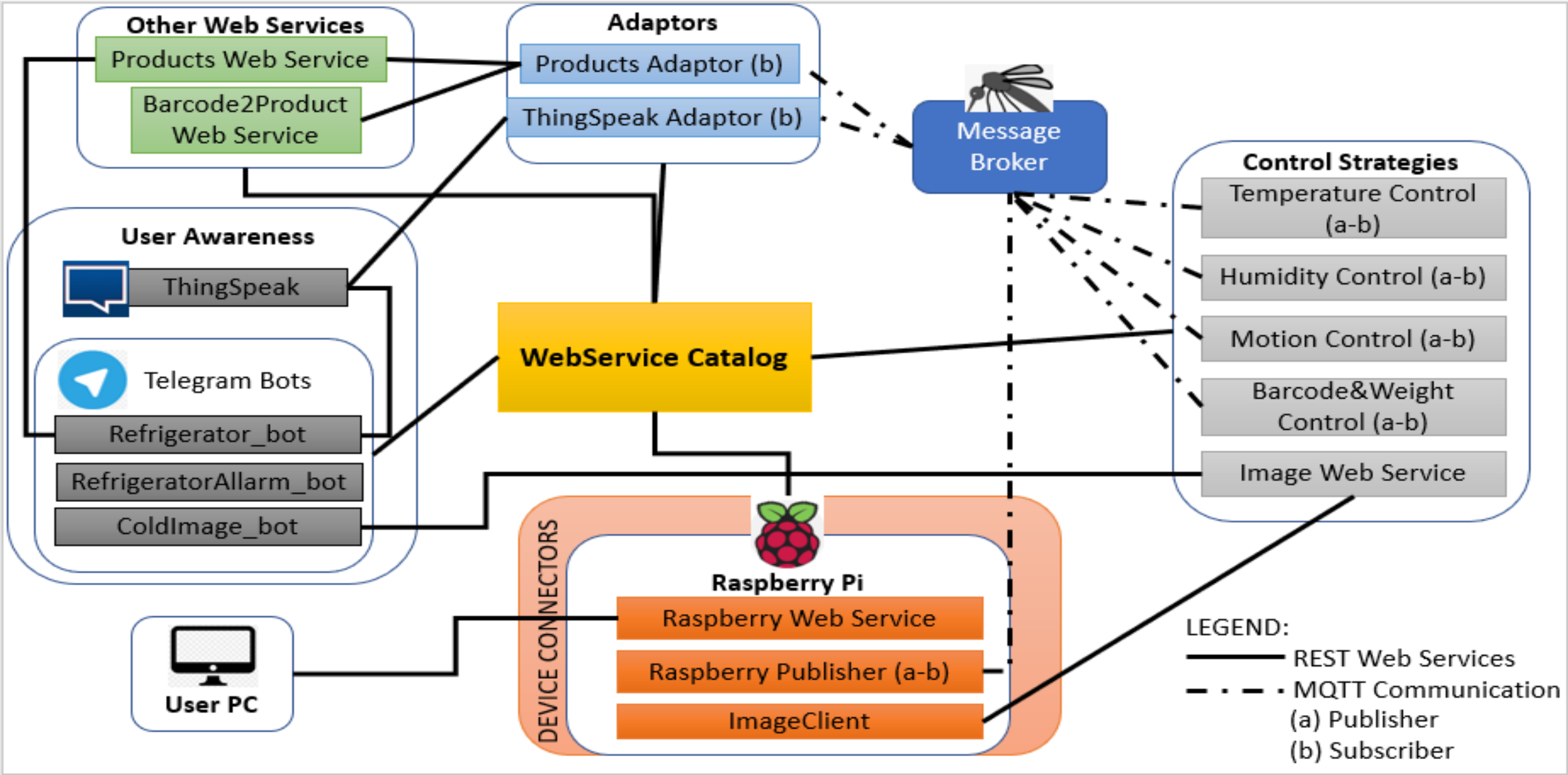



- **Monitoring** what is in the fridge
- **Take under control** temperature in the fridge
- Provide **Telegram service** to send notifications and get data from sensors


# Equipment:

- Raspberry Pi 3
- Temperature-humidity sensor
- Motion sensor
- Load-cell
- 2 WebCams

# How it works


Other Web Services
Products Web Service
Barcode2Product Web Service
Adaptors
Products Adaptor (b)
ThingSpeak Adaptor (b)
Message Broker
Control Strategies
Temperature Control (a-b)
Humidity Control (a-b)
Motion Control (a-b)
Barcode&Weight Control (a-b)
Image Web Service
User Awareness
ThingSpeak
Telegram Bots
Refrigerator_bot
RefrigeratorAllarm_bot
ColdImage_bot
WebService Catalog
DEVICE CONNECTORS
Raspberry Pi
Raspberry Web Service
Raspberry Publisher (a-b)
ImageClient
User PC
LEGEND:
REST Web Services
MQTT Communication
(a) Publisher
(b) Subscriber

# Web Service Catalog

- Receive information from all the other actors
- Provide information of all the other actors
- It is the only actor that can directly access to JSON catalog

Initial structure of the JSON catalog →


```
{
    "microservices": {
    },
    "number_of_devices": 0,
    "users_list": {
    },
    "devices": {
    },
    "number_of_users": 0,
    "msgbroker": {
        "IP": "iot.eclipse.org",
        "PORT": 1883
    },
    "last_edit": 0
}
```

# Web Service Catalog - APIs

POST /login data=json.dumps({"user":user,"password":psw})

POST /register
data=json.dumps({"user":user,"password":psw})

GET /devices
GET /devices?deviceID=deviceID
GET /devices?resources=resources
GET /microservices
GET /microservices?microserviceID=microserviceID
GET /microservices?category=category
GET /contacts
GET /msgbroker

PUT /newdevice
data=json.dumps({"deviceID":deviceID,"endpoints":endpoints,"protocol":protocol,"resources":resources})

PUT /newcontact
data=json.dumps({"telegramID":telegramID,"thingspeak_chID":channelID,"thingspeak_rkey":readkey,"thingspeak_wkey":writekey})

PUT /newmicroservice
data=json.dumps({"category":category,"endpoints":endpoints,"microserviceID":microserviceID,"protocol":protocol})

DELETE /delete?deviceID=deviceID
DELETE /clear?age=age

# Raspberry Web Service

- Lets your personal Smart Fridge to start working
- REST Connections to the Web Service Catalog:
  - Register a new account → POST method
  - Join device to Web Service Catalog account → POST method
  - Set ThingSpeak keys → PUT method
- Manage sensors settings → Set Temperature threshold

# Raspberry Publisher

**It is a thread of sensors objects (temperature, humidity, motion, scale, webcam devices) working as MQTT publisher publishing its data**

1. **Loads sensors settings**
   - deviceID
   - Pin
   - Resource type
   - Threshold
   - …

**3. Data are published in SenML format**

message = {"bn":endpoints,"e":[{ "n": resource, "u":unit, "t": current_time, "v":value }]}

Temperature publisher adds the key «threshold»

**2. What does it do?**

- **Login to Web Service Catalog** using the coupled account
- **Retreive data from sensor**
- **Get Message Broker** information from Web Service Catalog
- **Publish on a topic**:
  - Topics format for MQTT comunication:
    /Raspberry/user/resource/deviceID
- **Update user's devices list on Web Service Catalog** → PUT method

# How does Raspberry Publisher start the webcam for barcode detection?

- Barcode Webcam subprocess subscribes to motion sensor topic
- Every times that a motion sample is published from motion sensor and received from barcode subscriber, it is added to a virtual buffer
- When buffer is full of 1 webcam for the detection of a barcode is started

# Raspberry Image Client

REST communication with Web Service Catalog to:

- Put its information
- Get Image Web Service information

REST communication with Image Web Service to:

- Register the user obtaining a key
- Upload of a new photo from the internal of the fridge every 20 seconds

Image Web Service

# APIs:

- **Register obtaining a key and upload an image**

POST /register data=json.dumps({'user':user,'image':image})

- **Upload a new image**

PUT /upload?key=key data=json.dumps({'user':user,'image':image})

- **Get current image**

**GET /?key=key**

Temperature control

- **Gets Message Broker information** → from Web Service Catalog (REST)
- **Obtains** available **temperature sensors endpoints** → from Web Service Catalog in a **loop**
- **Subscribes** to them (Raspberry Publisher topics)

- **Checks** if temperature is **overthreshold** → if it is, **RefrigeratorAlarm_bot sends a message to proper user**

- **Publishes new topics** via MQTT **with a lower frequency** :

  /MicroServices/temperature/TemperatureService/user/temperature/deviceID

- Format of the **published message**:

  the **key «user» is added** to the SenML received message

Motion Control & Humidity control

- **Gets Message Broker Information** -> from Web Service Catalog (REST)
- **Obtains** available **humidity/motion sensor endopints** -> from the Web Service Catalog in a **loop**
- **Subscribes** to them (Raspberry Publisher topics)
- **Publishes new topics via MQTT with a lower frequency**:

  /MicroServices/humidity/HumidityService/user/humidity/deviceID

  /MicroServices/motion/MotionService/user/motion/deviceID

- Format of the **pubished message**:

  The **key <<user>> is** added to the SenML received message

**Weight & Barcode control**

- **Get Message Broker information** → from Web Service Catalog (REST)
- **Obtains** available **weight sensors and cams endpoints** → from Web Service Catalog in a **loop**
- **Subscribes** to them (Raspberry Publisher topics)
- **Publishes new weight topics via MQTT with a lower frequency**:

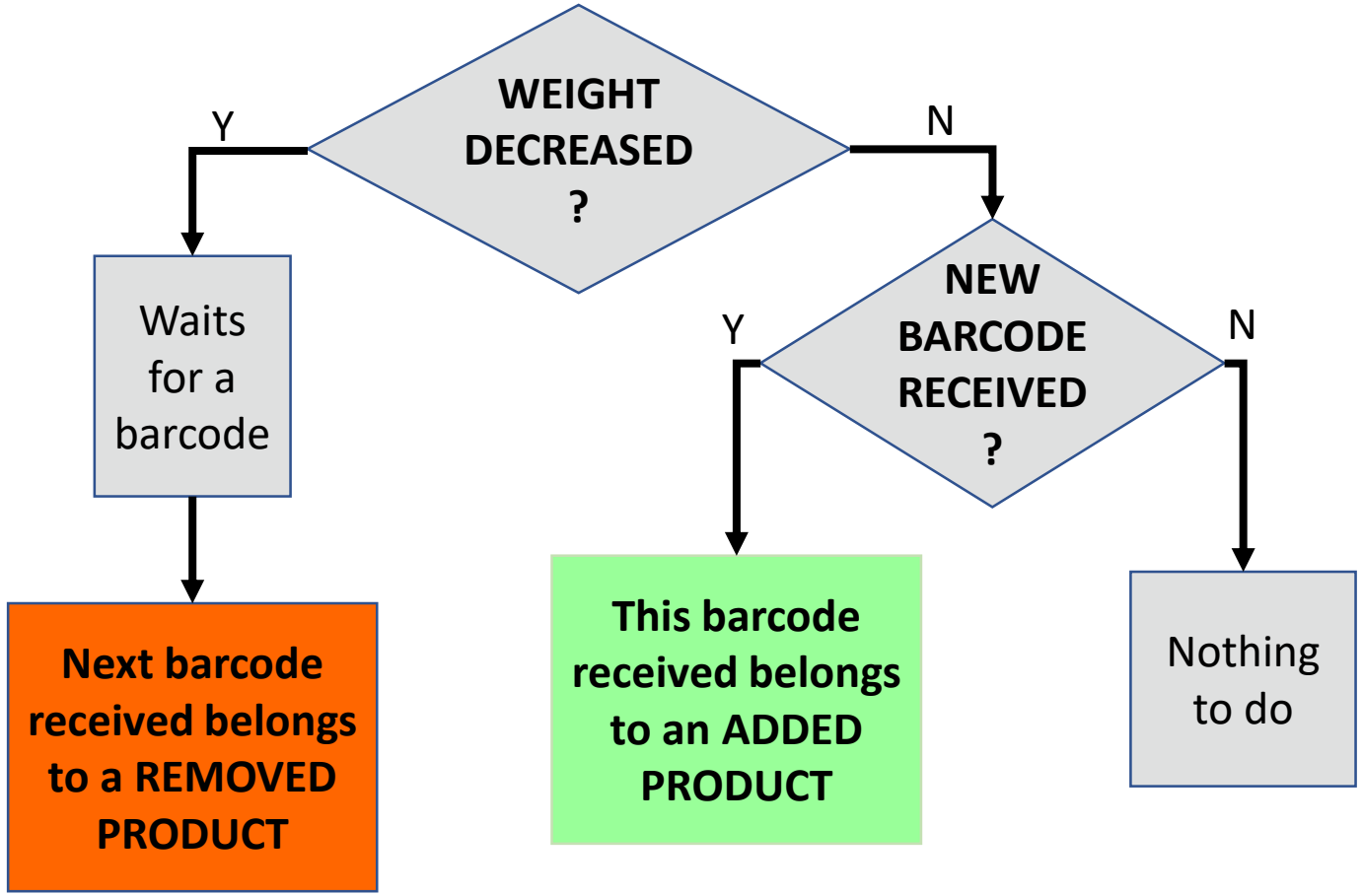
  /MicroServices/weight/WeightService/user/weight/deviceID

- Format of the **pubished message**:

  The **key <<user>> is** added to the SenML received message

# Checks received weight values and waits for barcodes

Weight & Barcode control



**Weight & Barcode control**

- **If a product is added or removed** into/from the fridge:
- **A message in SenML format with the addition of «user» and «action» keys is published** on the topic /MicroServices/barcode/BarcodeService/user/barcode/deviceID

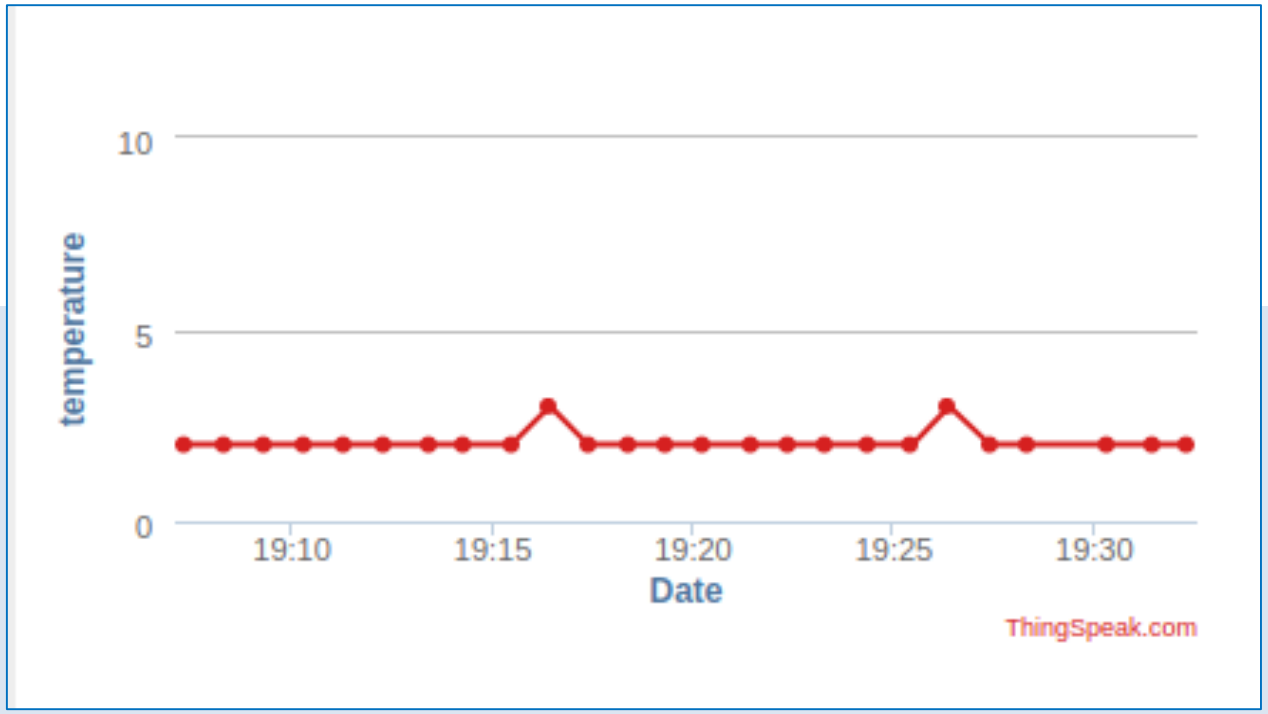
- If a product is added ‘action’ key as value **‘add’**

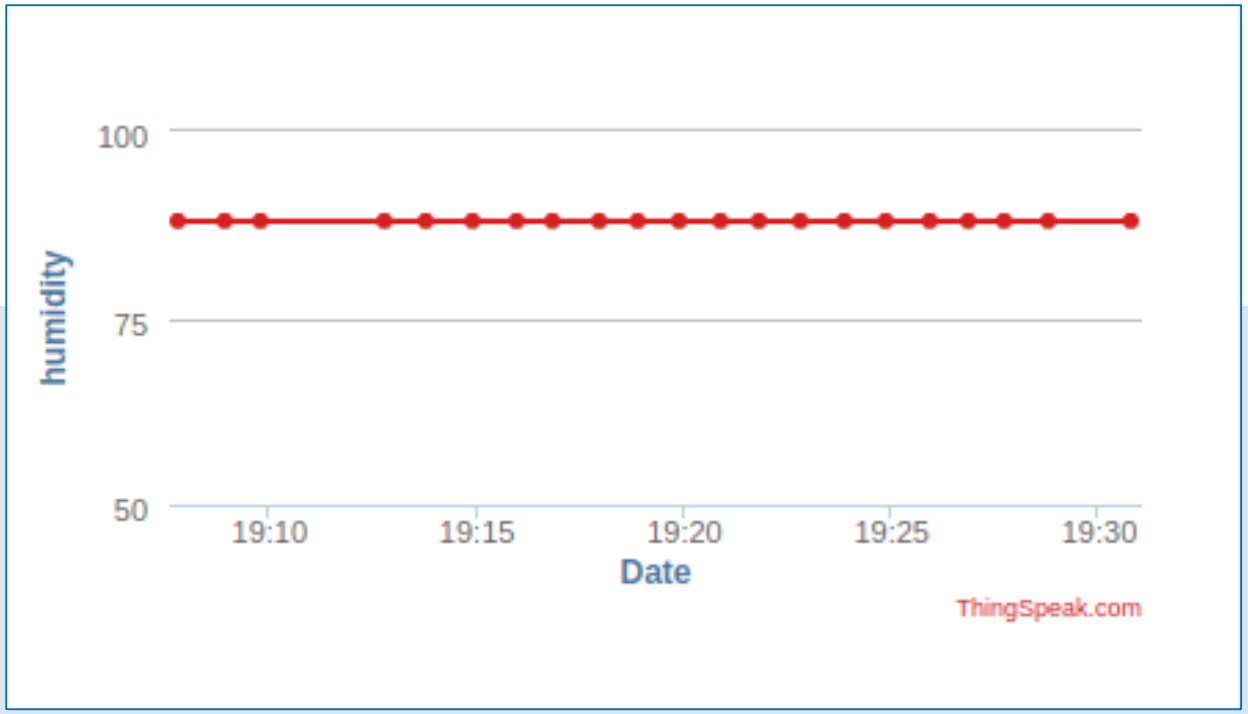
- If a product is removed ‘action’ key as value **‘remove’**

# ThingSpeack Adaptor

- Obtains available **Control Strategies endpoints** → from Web Service Catalog in a **loop**
- Works as **MQTT subscriber**:
  - Receiving Temperature, Humidity, Motion, Weight data from Control Strategies topics
- Works as **REST client**:
  - Obtaining from Web Service Catalog the ThingSpeak contacts of the user found in the current message
  - Uses data received via MQTT and the provided ThingSpeak write key to make GET requests to ThingSpeak
  - Update current user's proper channel field

# ThingSpeack Adaptor – examples of uploading



# Product Adaptor

- Obtains available **Barcode Control endpoints** → from Web Service Catalog in a **loop**
- Works as **MQTT subscriber**:
  - Receiving Barcode data from Weight&Barcode Control topics (only barcode)
- Works as **REST client**:
  - Registering to Product Web Service obtaining a write key (only the first time)
  - Using barcode data received via MQTT to make a GET request to Barcode2Product Web Service obtaining a product and its information
  - Using the obtained product, its information and the provided Products Web Service write key to make PUT requests to Products Web Service

Update the proper user product list:

if «action» = «add» → PUT /add? ...
Else if «action» = «remove» → PUT /remove?...

# Products Web Service

## APIs

- **Register obtaining a key**

POST /register

- **Get a specific product list**

GET /products?user=user

- **Add a new product to a specific list**

PUT /add?api_key=key&user=user&name=name&weight=weight

- **Remove a product from a specific list**

PUT /remove?api_key=key&user=user&name=name&weight=weight

# Barcode2Product Web Service

API

- **Get a specific product from its barcode**

GET /product?barcode=barcode

# Telegram BOTs

## REFRIGERATOR BOT

- Sends to the user his ThingSpeak data
- REST comunication with WebService Catalog to obtain user's channel and key of Thingspeak
- REST communication with Product Web Service to receive product list

## REFRIGERATOR ALARM BOT

It's used by Temperature Control to alarm the user if the temperature is over threshold

## COLD IMAGE BOT

- REST comunication with WebService Catalog to obtain proper user's Image Web Service endpoints
- REST communication with Image Web Service to obtain images of the internal part of the fridge to send to the user

# Further development

- Recipes provider
- Neural Network to analyze data
- Products expiration date control
- Automatic online shop


# Thank you for listening!