Práctica REST

1. Objetivo

- Construir un API Rest en Python Flask
- Simular el envío de datos a la plataforma de Internet de las Cosas (IoT) Ubidots, usando el API Rest que provee esta plataforma.

2. Herramientas a Utilizar

- Maquina virtual de Centos en Vagrant
- Python Flask
- Ubidots

3. Instalación de Python Flask

Siga los siguientes pasos para instalar Python Flask:

En Ubuntu

```
$ sudo apt-get install python3-pip
$ pip3 install Flask
$ pip3 freeze | grep Flask
```

4. Desarrollo de la Practica

PARTE 1. API REST EN PYTHON

Clonar el repositorio https://github.com/omondragon/APIRestFlask el cual contiene la implementación de un webService RestFul en Python Flask, como se muestra a continuación

```
!flask/bin/python
from flask import Flask, jsonify
from flask import abort
from flask import request

app = Flask(__name__)

books = [
```

```
    {
        'id': 1,
        'title': 'La hojarasca',
        'description': 'Good one',
        'author': 'Gabo'
    },
    {
        'id': 2,
        'title': 'El coronel no tiene quien le escriba',
        'description': 'Interesting',
        'author': 'Gabo'
    }
]

# Get all books
# For testing: curl -i http://localhost:5000/books
@app.route('/books', methods=['GET'])
def get_books():
    return jsonify({'books': books})

# Get one book by id
# For testing: curl -i http://localhost:5000/books/2
@app.route('/books/<int:book_id>', methods=['GET'])
def get_book(book_id):
    book = [book for book in books if book['id'] == book_id]
    if len(book) == 0:
        abort(404)
    return jsonify({'book': book[0]})

# Add new book
# For testing: curl -i -H "Content-Type: application/json" -X POST -d '{"title":"El
libro"}' http://localhost:5000/books
@app.route('/books', methods=['POST'])
def create_book():
    if not request.json or not 'title' in request.json:
        abort(400)
    book = {
        'id': books[-1]['id'] + 1,
        'title': request.json['title'],
        'description': request.json.get('description', ""),
        'author': request.json.get('author', ""),
    }
    books.append(book)
    return jsonify({'book': book}), 201

# Edit a Book
# For testing: curl -i -H "Content-Type: application/json" -X PUT -d
'{"author":"Jorgito"}' http://localhost:5000/books/2
@app.route('/books/<int:book_id>', methods=['PUT'])
def update_book(book_id):
    book = [book for book in books if book['id'] == book_id]
    if len(book) == 0 or not request.json:
        abort(404)
    book[0]['title'] = request.json.get('title', book[0]['title'])
    book[0]['description'] = request.json.get('description', book[0]['description'])
    book[0]['author'] = request.json.get('author', book[0]['author'])
    return jsonify({'book': book[0]})

# Delete a Book
# For testing: curl -i -H "Content-Type: application/json" -X DELETE
http://localhost:5000/books/1
@app.route('/books/<int:book_id>', methods=['DELETE'])
def delete_book(book_id):
    book = [book for book in books if book['id'] == book_id]
    if len(book) == 0:
        abort(404)
    books.remove(book[0])
    return jsonify({'result': True})
```

```
if __name__ == '__main__':
    app.run(host='0.0.0.0', debug=True)
```

Ejecutar el Código

```
export FLASK_APP=apirest.py
export FLASK_ENV=development
python3 -m flask run --host=0.0.0.0
```

Pruebe los métodos usando curl o postman con las peticiones de ejemplo que están en los comentarios del código.

PARTE 2. Usar API REST de Plataforma Ubidots

Para esta parte usaremos el siguiente código para envar solicitudes simples de Python desde la terminal de su computadora para probar e interactuar con la API REST de Ubidots.

Instalar Request

Requests es una biblioteca de Python que simplifica la realización de solicitudes HTTP desde cualquier script de Python que se pueda ejecutar en la terminal de su computadora o en cualquier dispositivo Linux integrado. Instale request:

```
$ pip3 install requests
```

API Rest de Ubidots

En la solicitud http que se envía a la API de Ubidots se esperan los siguientes datos:

Método: HTTP permite varios métodos (GET, POST, PUT, DELETE, etc.). Para crear un valor en Ubidots usamos una solicitud POST.

URL: la URL del recurso al que desea acceder (things.ubidots.com/api/v1.6/your-device-label/?token=your-token)

Encabezados HTTP: la API de Ubidots usa JSON como tipo de datos, por lo que el encabezado sería "Content-Type: application / json"

Cuerpo: una cadena JSON que contiene la etiqueta de la (s) variable (s) y sus valores. Por ejemplo: {"temperatura": 23, "humedad": 98}

Crear Script para envió de datos

Clone el repositorio

git clone https://github.com/omondragon/UbidotsClient

o en su defecto copie y cree un archivo llamado testUbidots.py en una dentro de una carpera UbidotsClient

Asigne su Ubidots TOKEN donde se indique en el código.

```
import time
import requests
import math
import random

TOKEN = "..."  # Put your TOKEN here
DEVICE_LABEL = "machine"  # Put your device label here
VARIABLE_LABEL_1 = "temperature"  # Put your first variable label here
VARIABLE_LABEL_2 = "humidity"  # Put your second variable label here
VARIABLE_LABEL_3 = "position"  # Put your second variable label here


def build_payload(variable_1, variable_2, variable_3):
    # Creates two random values for sending data
    value_1 = random.randint(-10, 50)
    value_2 = random.randint(0, 85)

    # Creates a random gps coordinates
    lat = random.randrange(34, 36, 1) + \
        random.randrange(1, 1000, 1) / 1000.0
    lng = random.randrange(-83, -87, -1) + \
        random.randrange(1, 1000, 1) / 1000.0
    payload = {variable_1: value_1,
               variable_2: value_2,
               variable_3: {"value": 1, "context": {"lat": lat, "lng":
lng}}}

    return payload


def post_request(payload):
    # Creates the headers for the HTTP requests
    url = "http://industrial.api.ubidots.com"
    url = "{}/api/v1.6/devices/{}".format(url, DEVICE_LABEL)
    headers = {"X-Auth-Token": TOKEN, "Content-Type":
"application/json"}

    # Makes the HTTP requests
    status = 400
    attempts = 0
    while status >= 400 and attempts <= 5:
        req = requests.post(url=url, headers=headers, json=payload)
        status = req.status_code
        attempts += 1
```

```
        time.sleep(1)

    # Processes results
    if status >= 400:
        print("[ERROR] Could not send data after 5 attempts, please
check \
            your token credentials and internet connection")
        return False

    print("[INFO] request made properly, your device is updated")
    return True


def main():
    payload = build_payload(
        VARIABLE_LABEL_1, VARIABLE_LABEL_2, VARIABLE_LABEL_3)

    print("[INFO] Attemping to send data")
    post_request(payload)
    print("[INFO] finished")


if __name__ == '__main__':
    while (True):
        main()
        time.sleep(1)
```

Ejecute su script

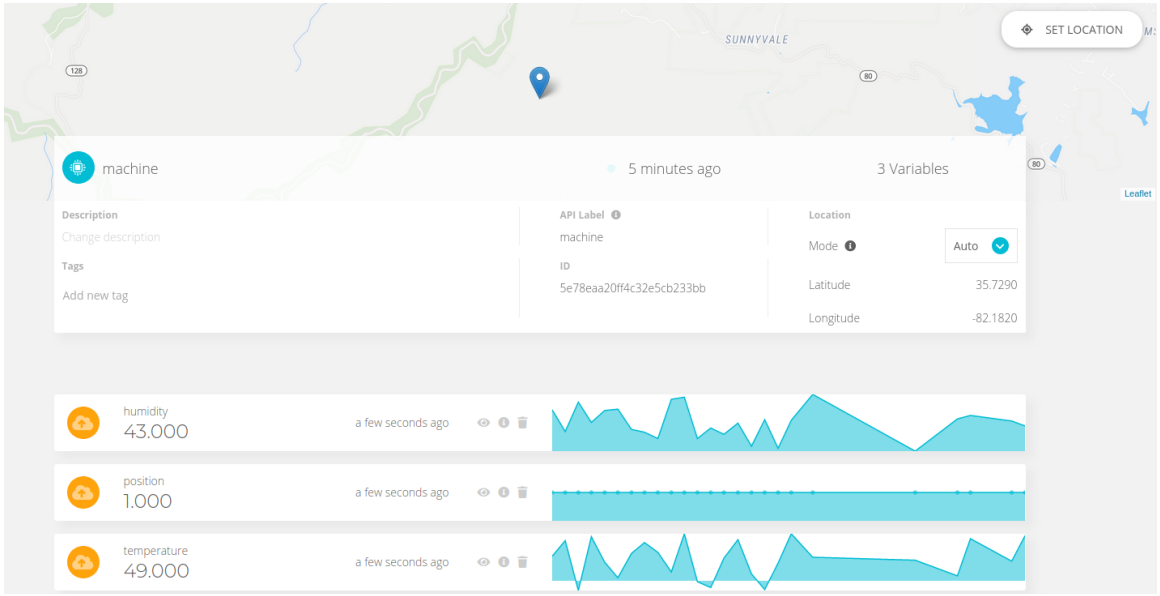
Desde su terminal, ejecute el script con el siguiente comando.

```
$ cd UbidotsClient

$ python3 testUbidots.py
```

Visualiza tus resultados

Ejecute el código en su computadora o terminal de dispositivo para comenzar a simular datos y visualizar su trabajo iniciando sesión en su cuenta de Ubidots.



1. Ejercicio

PARTE 1

1. Pruebe el API Rest de la parte 1
    a. Pruebe el API Rest usando curl
    b. Pruebe el API Rest usando Postman

PARTE 2

2. Demuestre la recepción de datos enviados desde el código Python en Ubidots

3. Realice pruebas de envío de datos para actualizar las variables creadas **usando curl**

   Por ejemplo:

   curl --header "Content-Type: application/json" -H 'X-Auth-Token: <su-token>' --request POST --data '{"temperature":13}' http://things.ubidots.com/api/v1.6/devices/machine

   o

   curl -i -H "Content-Type: application/json" -X POST -d '{"temperature":"50"}' http://things.ubidots.com/api/v1.6/devices/machine/?token=<su-token>

4. Realice pruebas de envío de datos para actualizar las variables creadas **usando Postman** u otra herramienta online para prueba de API Rest.

5. Modifique el código para agregar dos variables más de su elección, las cuales deben ser reportadas de la misma manera a ubidots.

6. En Ubidots diseñe un dashboard con al menos la siguiente información:

    a. Humidity representado como un widget doble axis
    b. Position representado como un mapa
    c. Temperature representado como un diagrama de barras
    d. Las dos nuevas variables representadas de la forma que usted elija

    Visualice las actualizaciones del dashboard en tiempo real.

7. En Ubidots cree un evento a partir de una de las variables monitoreadas, el cual debe ser reportado a su correo electrónico una vez suceda.

2. Bibliografía

- Postman. https://www.postman.com/
- Simulate data in Ubidots using Python. https://help.ubidots.com/en/articles/569964-simulate-data-in-ubidots-using-python
- Sending data without hardware - Ubidots for Education. https://www.youtube.com/watch?v=RsyJJbCinIY

3. Desafio [Vale por 0.5 puntos en las notas de practicas]

Implemente el ejercicio de la parte 1 usando otro lenguaje de programación

4. Bibliografía

- Postman. https://www.postman.com/