

Práctica REST

1. Objetivo

Construir un API Rest en Python Flask

2. Herramientas a Utilizar

- Maquina virtual de Centos en Vagrant
- Python Flask

3. Instalación de Python Flask

Siga los siguientes pasos para instalar Python Flask:

En Centos

Abra un terminal en su maquina Centos y verifique que tenga instalado Python.

```
$ python3
Python 3.6.8 (default, Nov 16 2020, 16:55:22)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-44)] on linux
Type "help", "copyright", "credits" or "license" for more information.
```

Instale pip:

```
sudo yum install epel-release
sudo yum install python3-pip
pip3 --version
```

Instale Flask:

```
$ sudo pip3 install Flask
$ pip freeze | grep Flask
```

En Ubuntu

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

4. Desarrollo de la Practica

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

```

1. Ejercicio

1. Pruebe el API Rest
 - a. Pruebe el API Rest usando curl
 - b. Pruebe el API Rest usando Postman
2. Implemente el mismo ejercicio usando otro lenguaje de programación

2. Bibliografía

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