

Paso a Paso punto 1:

Instalación y configuración de Prometheus:

1. Realizamos lo descrito en la guía del parcial y lo comprobamos:

```
vagrant@servidorRest:~$ curl -i http://localhost:5000/books
HTTP/1.1 200 OK
Server: Werkzeug/3.0.1 Python/3.10.12
Date: Wed, 15 Nov 2023 01:28:21 GMT
Content-Type: application/json
Content-Length: 320
Connection: close

{"books":[{"author":"Telematicos","description":"Brillante","id":2,"title":"El principito"}, {"author":"Autor del nuevo libro","description":"Descripci\u00f3n del nuevo libro","id":4,"title":"Nuevo Libro"}, {"author":"Autor del nuevo libro","description":"Descripci\u00f3n del nuevo libro","id":5,"title":"Nuevo Libro"}]}

vagrant@servidorRest:~$ curl -i -H "Content-Type: application/json" -X POST -d '{"title":"El libro"}' http://localhost:5000/books
HTTP/1.1 201 CREATED
Server: Werkzeug/3.0.1 Python/3.10.12
Date: Wed, 15 Nov 2023 01:28:35 GMT
Content-Type: application/json
Content-Length: 30
Connection: close

{"book":{"title":"El libro"}}

vagrant@servidorRest:~$ curl -i -H "Content-Type: application/json" -X PUT -d '{"author":"Jorgito"}' http://localhost:5000/books/2
HTTP/1.1 200 OK
Server: Werkzeug/3.0.1 Python/3.10.12
Date: Wed, 15 Nov 2023 01:28:48 GMT
Content-Type: application/json
Content-Length: 91
Connection: close

{"book":{"author":"Telematicos","description":"Brillante","id":2,"title":"El principito"}}

vagrant@servidorRest:~$ curl -i -H "Content-Type: application/json" -X DELETE http://localhost:5000/books/5
HTTP/1.1 200 OK
Server: Werkzeug/3.0.1 Python/3.10.12
Date: Wed, 15 Nov 2023 01:29:11 GMT
Content-Type: application/json
Content-Length: 16
Connection: close

{"result":true}

vagrant@servidorRest:~$ curl -i http://localhost:5000/books
HTTP/1.1 200 OK
Server: Werkzeug/3.0.1 Python/3.10.12
Date: Wed, 15 Nov 2023 01:29:17 GMT
Content-Type: application/json
Content-Length: 260
Connection: close

{"books":[{"author":"Jorgito","description":"Brillante","id":2,"title":"El principito"}, {"author":"Autor del nuevo libro","description":"Descripci\u00f3n del nuevo libro","id":4,"title":"Nuevo Libro"}, {"author":"","description":"","id":6,"title":"El libro"}]}

vagrant@servidorRest:~$ |
```

2. Codificamos el código que funciona como cliente para la API REST probada en el anterior punto con todas las funciones:

```
import requests
import json

base_url = 'http://localhost:5000/books'

# Función para realizar una solicitud GET
def get_books():
    response = requests.get(base_url)
    print(response.json())

# Función para realizar una solicitud GET por ID
def get_book(book_id):
    url = f'{base_url}/{book_id}'
    response = requests.get(url)
    print(response.json())
```

```

# Función para realizar una solicitud POST
def create_book(title, description, author):
    headers = {'Content-Type': 'application/json'}
    data = {'title': title, 'description': description, 'author':
author}
    response = requests.post(base_url, headers=headers,
data=json.dumps(data))
    print(response.json())

# Función para realizar una solicitud PUT
def update_book(book_id, author):
    url = f'{base_url}/{book_id}'
    headers = {'Content-Type': 'application/json'}
    data = {'author': author}
    response = requests.put(url, headers=headers,
data=json.dumps(data))
    print(response.json())

# Función para realizar una solicitud DELETE
def delete_book(book_id):
    url = f'{base_url}/{book_id}'
    response = requests.delete(url)
    print(response.json())

# Pruebas- Se descomenta la funcion que se desea usar

# Obtener todos los libros
get_books()

# Obtener un libro por ID
#get_book(1)

# Agregar un nuevo libro
#create_book('Nuevo Libro', 'Descripción del nuevo libro', 'Autor
del nuevo libro')

# Actualizar un libro por ID
#update_book(2, 'Telematicos')

```

- Para comprobar su funcionamiento, usaremos todas las solicitudes, usamos el código:

```
python3 cliente.py
```