UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE CIENCIAS

ÁREA DE CIENCIA DE LA COMPUTACIÓN

3° LABORATORIO - CC312

1 PARTE

- **TÍTULO:** Explore las API ResT con el Simulador API y Postman

- **ALUMNO:**

JHONATAN POMA MARTINEZ 20182729F

- **PROFESORES:** YURI JAVIER.,CCOICCA PACASI

2023

Parte 1: Inicie el DevNet VM
Parte 2: Explore la documentación de API usando el simulador de API
Parte 3: Use el Postman para realizar llamadas API al simulador de API
Parte 4: Use Python para agregar 100 libros para el simulador de API

Aspectos básicos:

En este laboratorio, usted aprenderá a usar el simulador de API de la biblioteca escolar para realizar llamadas a la API para enumerar, agregar y eliminar libros. Más tarde, usarás Postman para realizar estas mismas llamadas a la API.

DEFINICIONES previas:

API :Las API son mecanismos que permiten a dos componentes de software comunicarse entre sí mediante un conjunto de definiciones y protocolos. Por ejemplo, el sistema de software del instituto de meteorología contiene datos meteorológicos diarios. La aplicación meteorológica de su teléfono “habla” con este sistema a través de las API y le muestra las actualizaciones meteorológicas diarias en su teléfono.

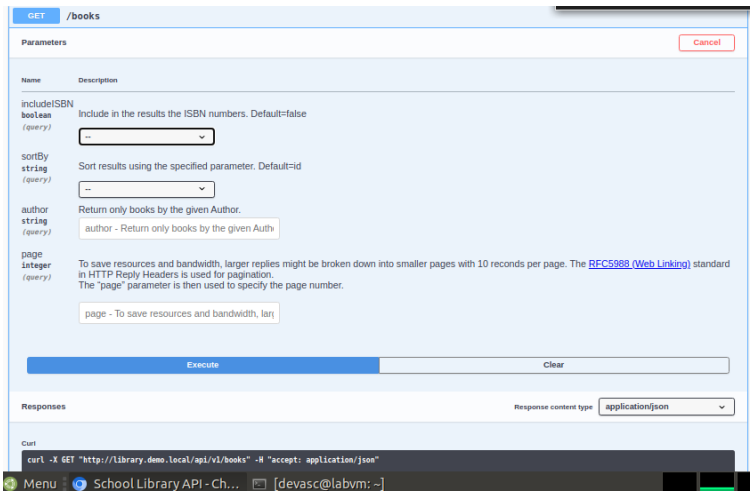
API REST significa transferencia de estado representacional. REST define un conjunto de funciones como GET, PUT, DELETE, etc. que los clientes pueden utilizar para acceder a los datos del servidor. Los clientes y los servidores intercambian datos mediante HTTP.

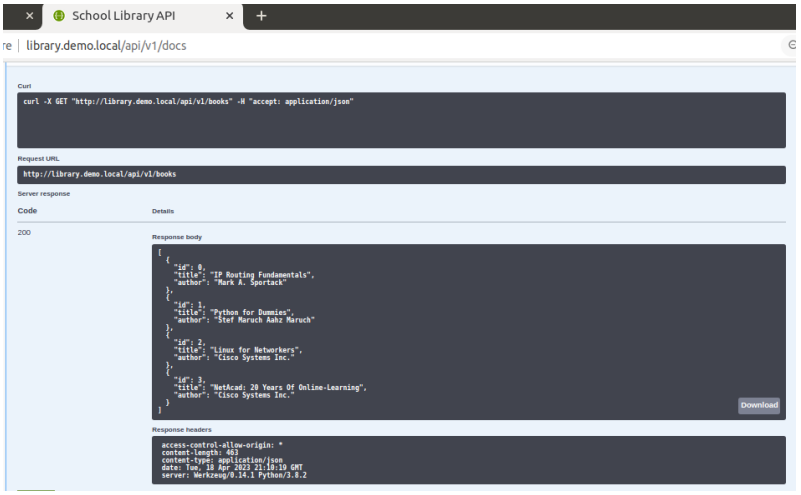
Parte 1: Inicie el DevNet VM

// Maquina virtual iniciada.

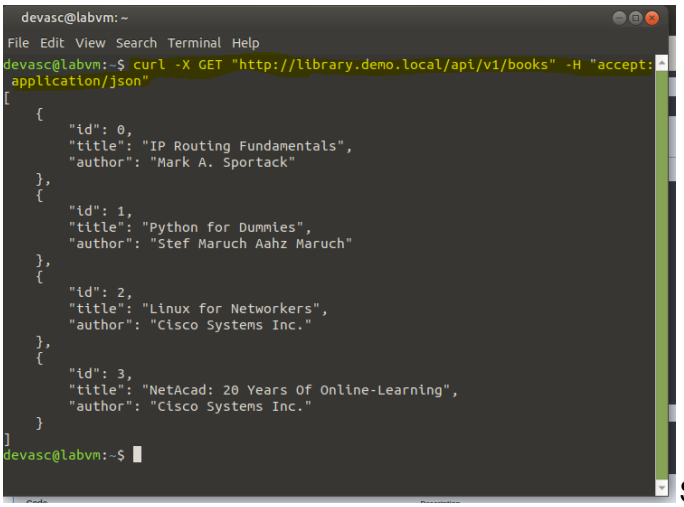
Parte 2: Explore la documentación de API usando el simulador de API

Exploramos el **API GET/books**



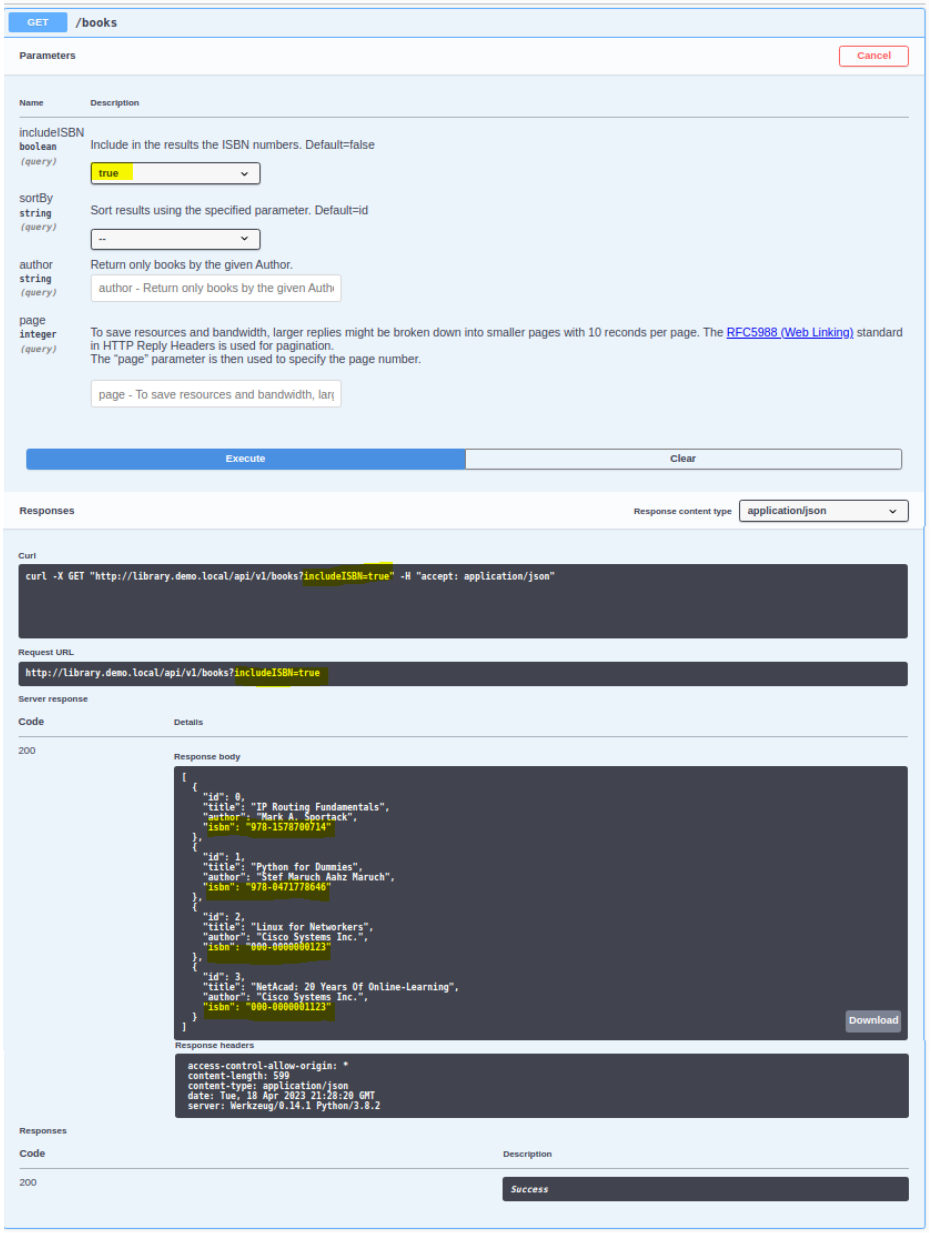


Si usamos la terminal y pegamos el **CURL** notaremos que de igual modo me muestra la lista de libros en formato J

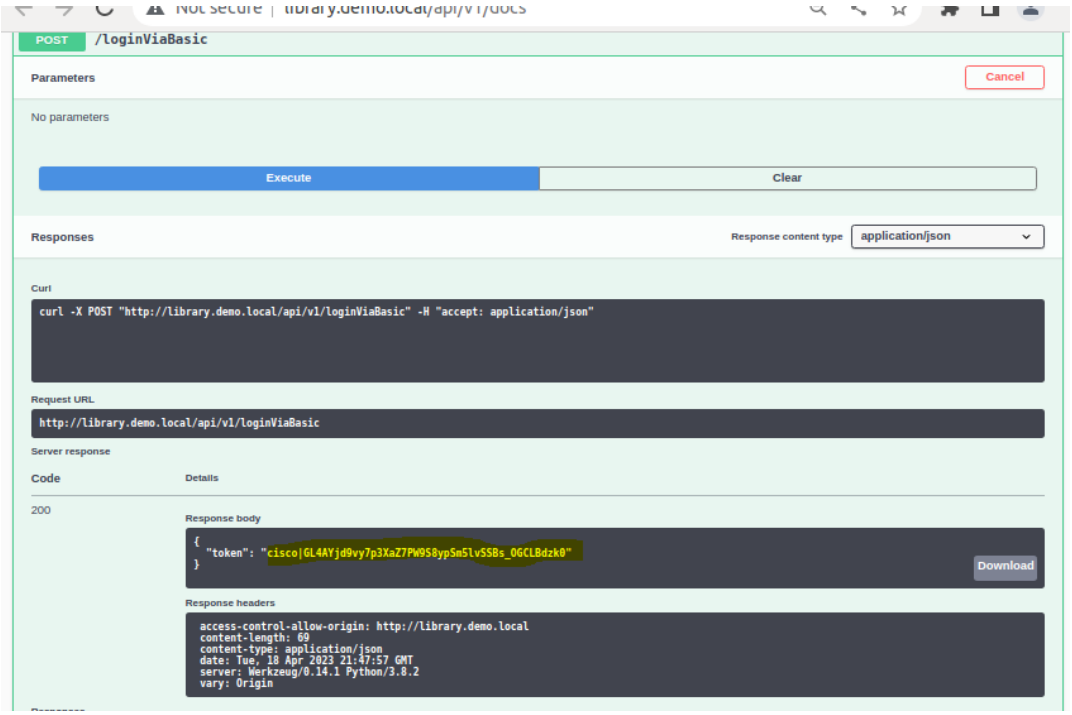


SON.

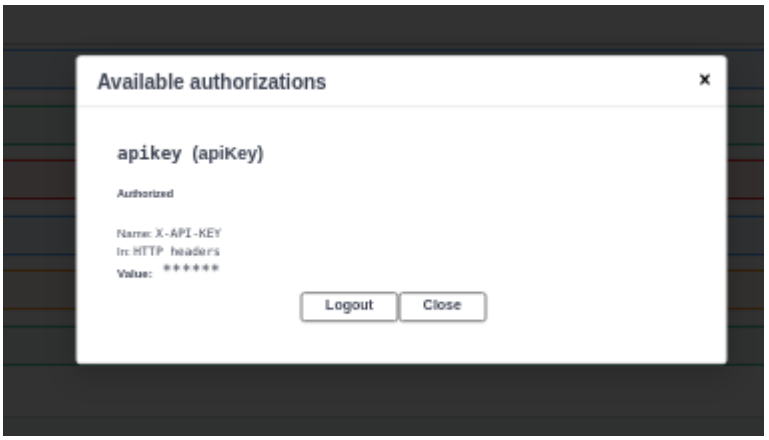
Ahora incluiremos el ISBN (true), notaremos que ahora en el **CURL, URL, cuerpo de Respuesta** incluyen el **ISBN=true.**



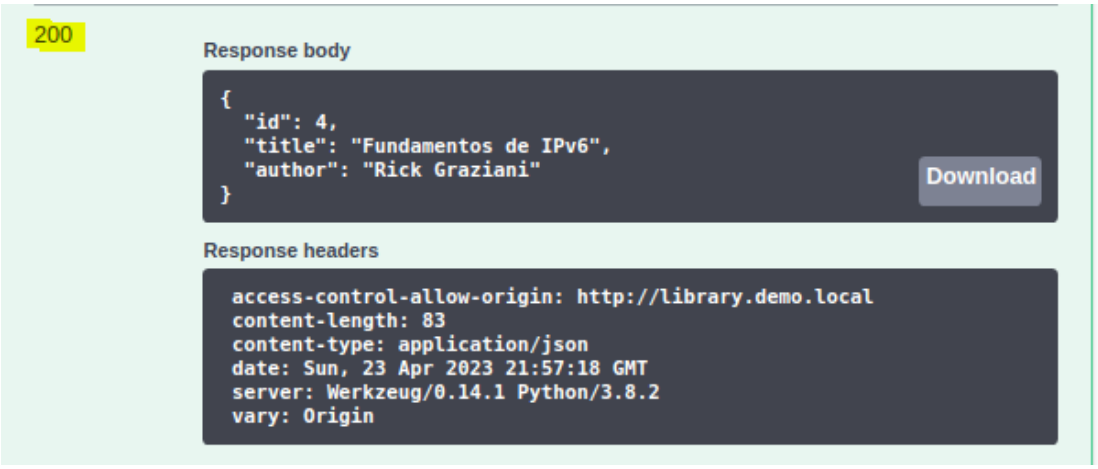
Ahora obtendremos un token usando la **API POST/LoginViaBasic**



Autorizaremos y pegaremos el TOKEN que tenemos.El nombre es X-API-KEY, esta información junto con el token se usara mas adelante en POSTMAN.



Se nos desbloqueará algunas APIS. Entonces ahora podremos añadir mas libros.
Paso 9: (añadiendo mas libros)



libro añadido, codigo “200”, ahora verificamos que se hallan añadido en “nuestros libros”

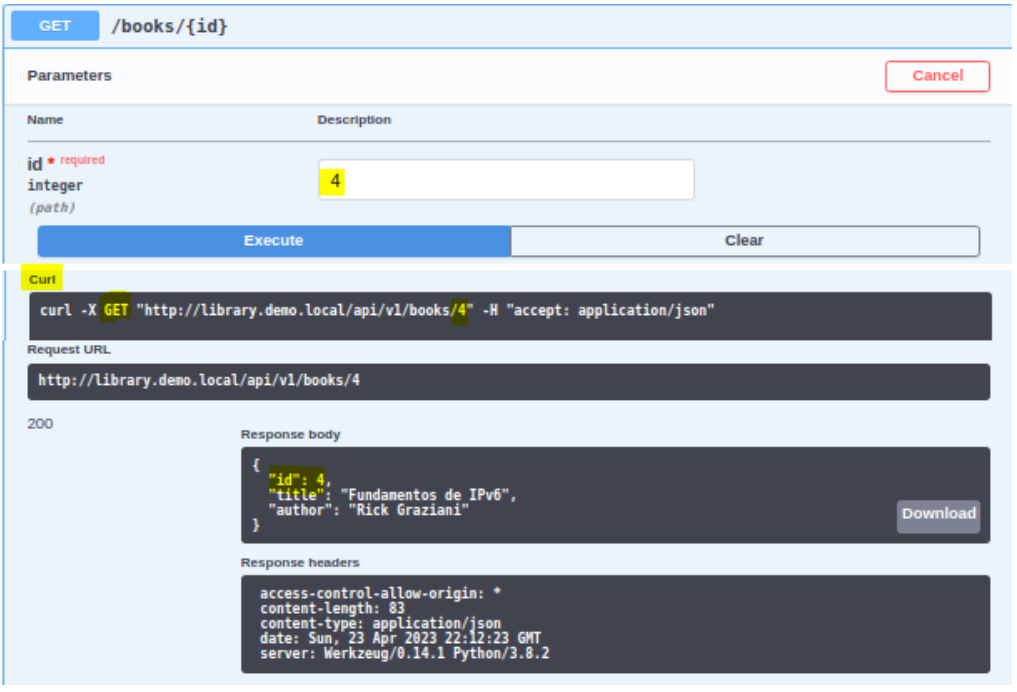
Book ID	Title	Author
0	IP Routing Fundamentals	Mark A. Sportack
1	Python for Dummies	Stef Maruch Aahz Maruch
2	Linux for Networkers	Cisco Systems Inc.
3	NetAcad: 20 Years Of Online-Learning	Cisco Systems Inc.
4	Fundamentos de IPv6	Rick Graziani
5	31 dias antes de su examen	Allan Johson

En la API GET/Books , ejecutamos y veremos los 2 libros añadidos.


```
{
  "id": 1,
  "title": "Python for Dummies",
  "author": "Stef Maruch Aahz Maruch",
  "isbn": "978-0471778646"
},
{
  "id": 2,
  "title": "Linux for Networkers",
  "author": "Cisco Systems Inc.",
  "isbn": "000-0000000123"
},
{
  "id": 3,
  "title": "NetAcad: 20 Years Of Online-Learning",
  "author": "Cisco Systems Inc.",
  "isbn": "000-0000001123"
},
{
  "id": 4,
  "title": "Fundamentos de IPv6",
  "author": "Rick Graziani"
},
{
  "id": 5,
  "title": "31 dias antes de su examen",
  "author": "Allan Johson"
}
```

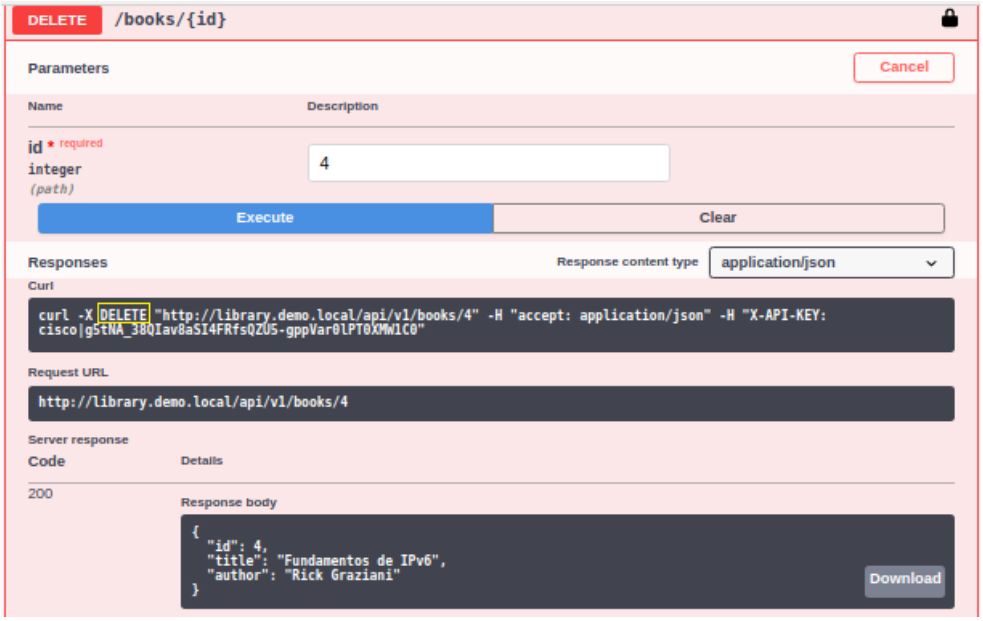

API GET/books/{id}

Mostraremos un libro usando la **API GET/books/{id}** , para ello ingresaremos el “id”

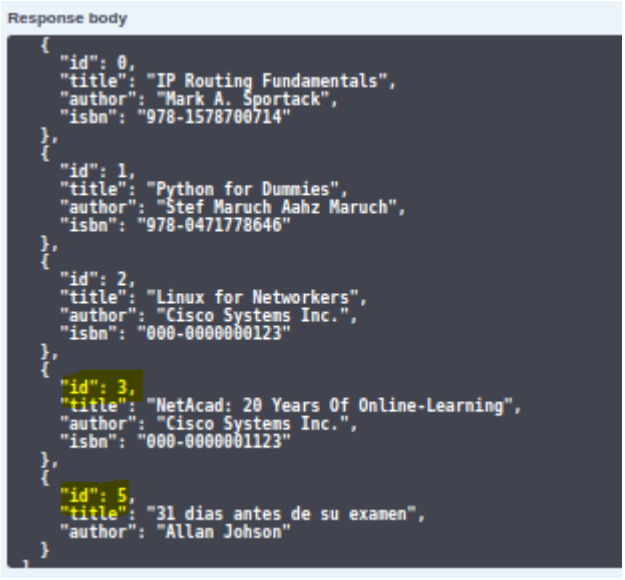


API DELETE /books {id}

Eliminaremos un libro en especifico usando el “**id**”.



Ahora mostramos de nuevo el **API GET/books** y ejecutamos:
veremos que ya no aparece el libro con ID=4 porque anteriormente lo eliminamos.



Parte 3: Use el Postman para realizar llamadas API al simulador de API

POSTMAN

Postman es una plataforma que permite y hace más sencilla la creación y el uso de APIs. Esta herramienta es muy útil para programar porque da la posibilidad hacer pruebas y comprobar el correcto funcionamiento de los proyectos que realizan los desarrolladores web. ¡Todo en base a una extensión en Google Chrome!

Para qué sirve Postman

De manera general, Postman sirve para:

- Probar colecciones o catálogos APIs, ya sea para Frontend o Backend.
- Clasificar y organizar en carpetas funciones y módulos de los servicios web.
- Ofrece la posibilidad de gestionar el ciclo de vida de la API.
- Generar documentos y monitorizar las APIs.

Postman es una plataforma con diversas funciones muy útiles para los desarrolladores web. Algunas de estas son las que detallamos a continuación.

Desarrollo de peticiones

Postman ofrece la posibilidad de desarrollar y crear peticiones HTTP a cualquier API a través de una interfaz gráfica. De esta manera, se convierte en una herramienta muy práctica para programar y realizar pruebas para comprobar que los desarrollos que están llevando a cabo los programadores se están ejecutando correctamente.

Crear y probar colecciones de APIs

Como hemos comentado, con Postman podemos probar colecciones de APIs, ya sea para Frontend como Backend. Además, las colecciones de Postman ayudan a los desarrolladores a organizar las solicitudes de API que están relacionadas.

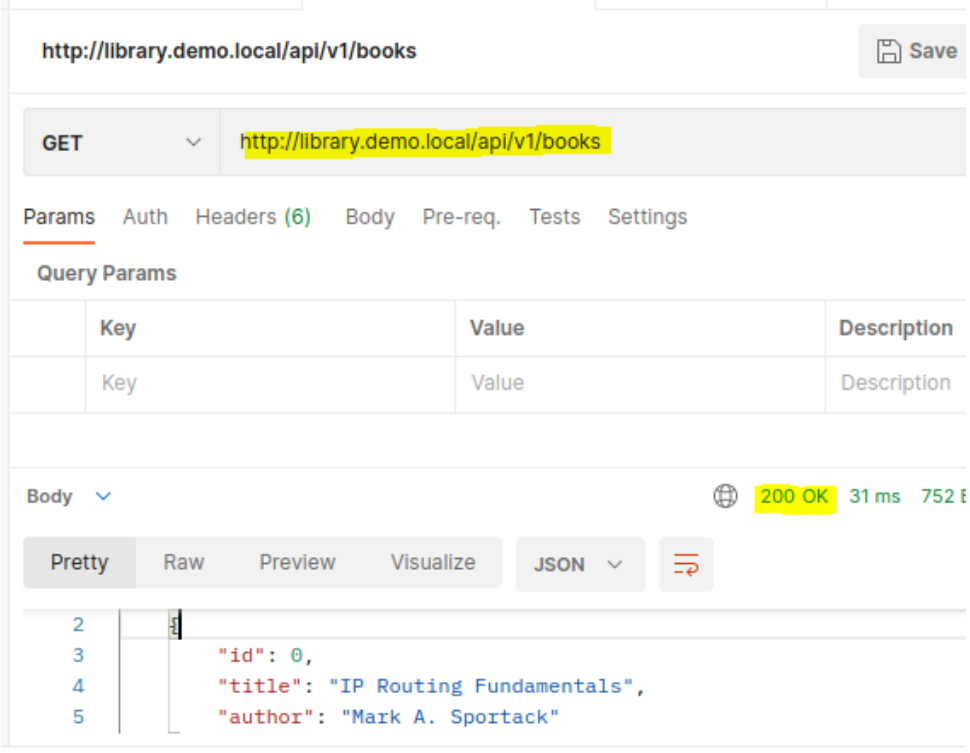
Administrar documentación

Con Postman parece todo más sencillo, ya que da la posibilidad de crear documentación fundamentada en las API y en las colecciones que se hayan desarrollado y, además, hacerla pública.

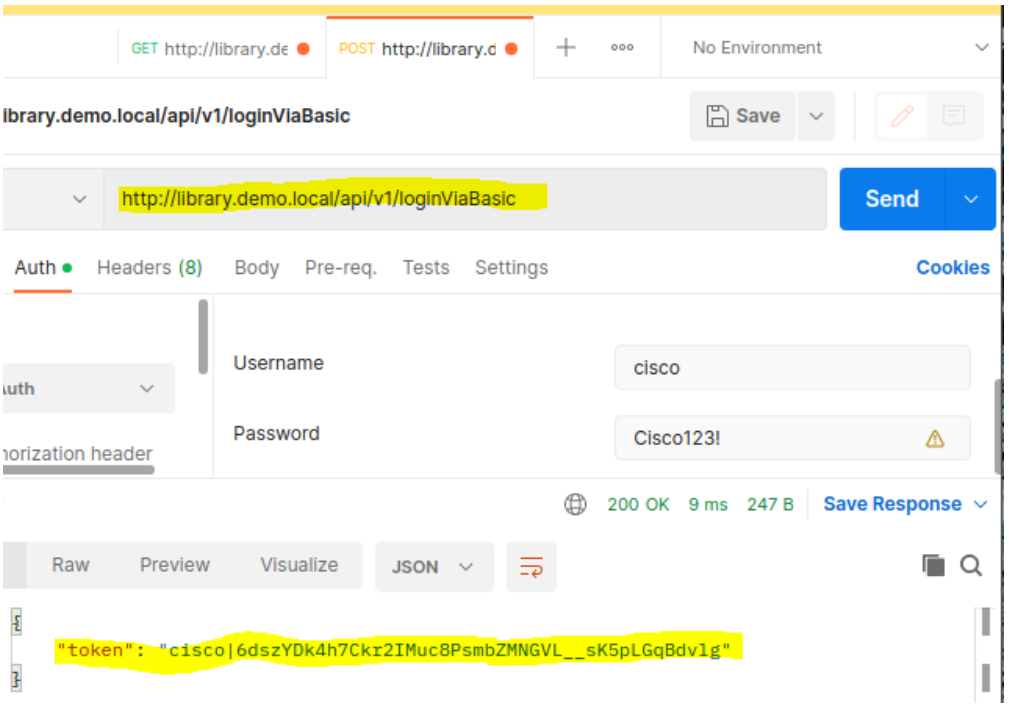
Trabajar con entornos de colaboración

Esta plataforma también permite trabajar con entornos para después poder compartir la información con el resto de miembros del equipo que forman parte del desarrollo de la API.

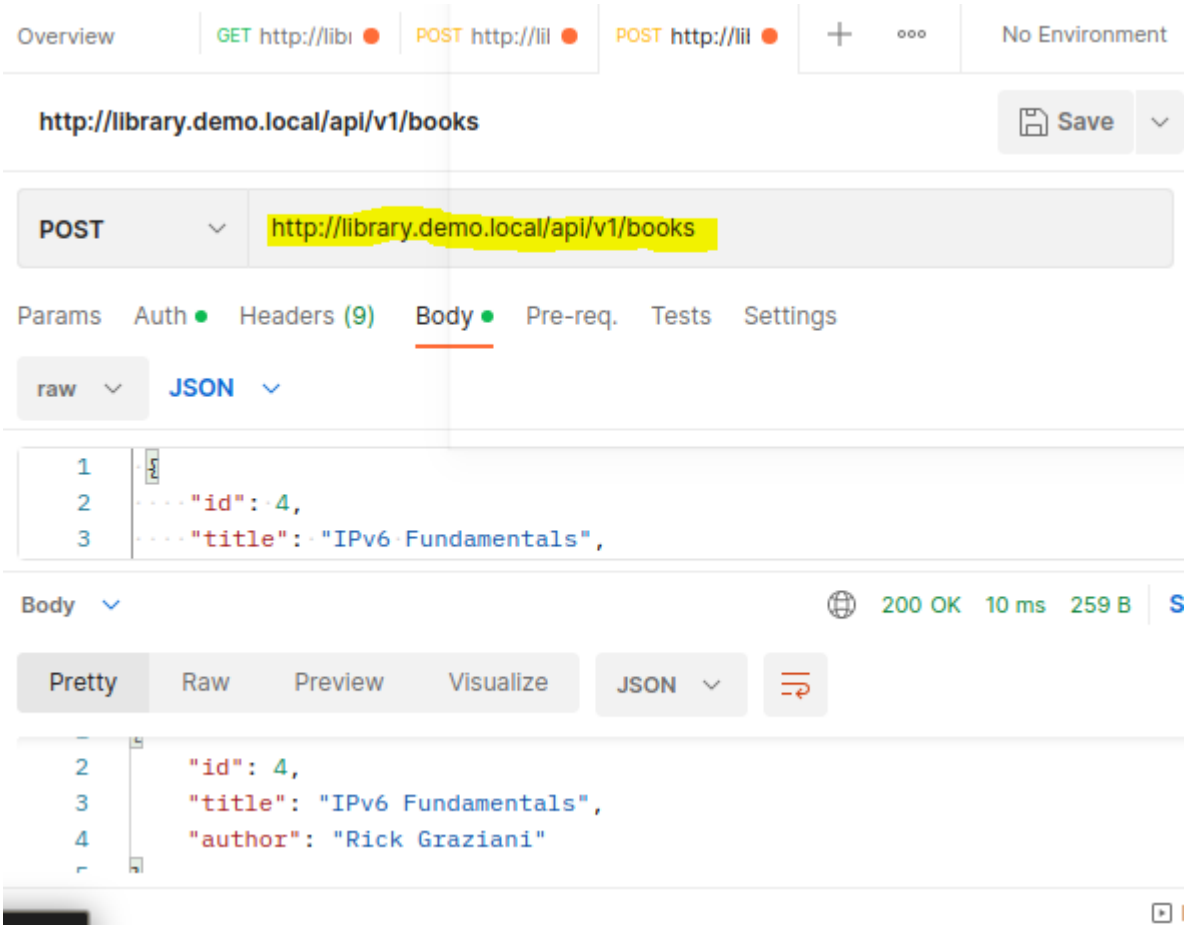
Usando postman abrimos el **API GET/books**



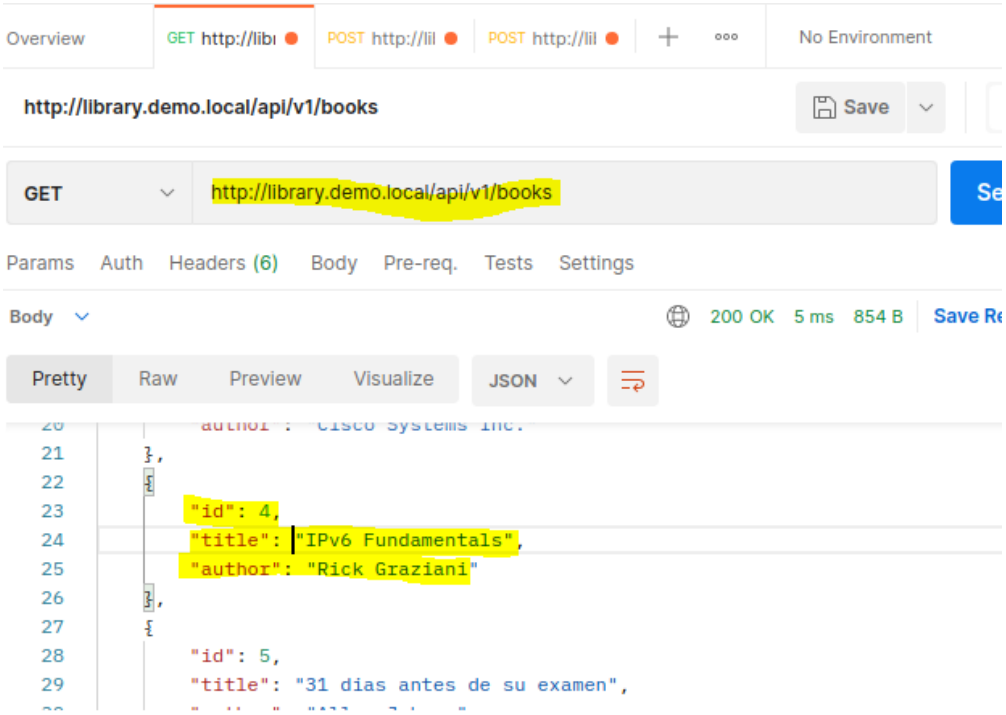
obtener un token usando la **API POST /LoginViaBasic.**



- **Obtenido el Token podemos añadir un libro mediante API POST/books**

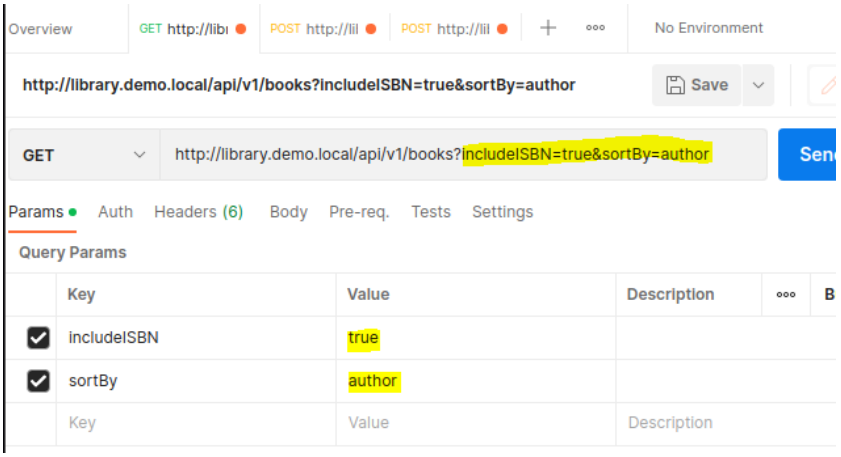


- **verificamos el libro adicional con API GET/book**



- **Usamos los demás "parámetros" del API GET/book y lo ordenamos mediante los autores**

- **Ahora editamos en el postman , le agregamos include ISBN, sortBy**



Parte 4: Use Python para agregar 100 libros para el simulador de API

- **Investigue las bibliotecas utilizadas por el programa add100RandomBooks.py.**

```
devasc@labvm:~$ python3
Python 3.8.2 (default, Apr 27 2020, 15:53:34)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more informati
>>> from faker import Faker
>>> fake = Faker()
>>> fake.
Display all 252 possibilities? (y or n)
fake.add_provider(                    fake.longitude(
fake.address(                         fake.mac_address(
fake.am_pm(                           fake.mac_platform_token(
fake.android_platform_token(          fake.mac_processor(
fake.ascii_company_email(             fake.md5(
fake.ascii_email(                     fake.military_apo(
fake.ascii_free_email(                fake.military_dpo(
fake.ascii_safe_email(                fake.military_ship(
fake.bank_country(                    fake.military_state(
fake.bban(                            fake.mime_type(
fake.binary(                          fake.month(
fake.boolean(                         fake.month_name(
```

- **Practique generar datos aleatorios usando la biblioteca de falsificadores.**

```
>>> print('My name is {}'.format(fake.name()))
My name is Matthew Clark
>>> print('My name is {}'.format(fake.name()))
My name is Luis Riggs
>>> print('My name is {} and I wrote "{}" (ISBN {})'.format(fake.name(),fak
e.catch_phrase(),fake.isbn13()))
My name is Maria Taylor and I wrote "Future-proofed directional help-desk"
(ISBN 978-1-311-63033-9)
>>>
>>> for i in range(10):
...     print(fake.name())
...
Robin Tucker
Laura Roman
Thomas Campbell
Scott Carter
Tiffany Martin
Samuel Moore
Shannon Peters
Brent Morgan
Susan Robinson
Richard Harvey
```

- **Ejecute y verifique el programa add100RandomBooks.py.**

```
# Using the faker module, generate random "fake" books
fake = Faker()
for i in range(4, 105):
    fakeTitle = fake.catch_phrase()
    fakeAuthor = fake.name()
    fakeISBN = fake.isbn13()
    book = {"id":i, "title": fakeTitle, "author": fakeAuthor,
    # add the new random "fake" book using the API
    addBook(book, apiKey)
```

```
devasc@labvm:~/labs/devnet-src/school-library$ python3 add100RandomBooks.py
Book {'id': 4, 'title': 'Optional intermediate database', 'author': 'Timoth
y Goodwin', 'isbn': '978-0-8281-9869-1'} added.
Book {'id': 5, 'title': 'Digitized maximized hierarchy', 'author': 'Taylor
Davis', 'isbn': '978-1-58306-304-0'} added.
Book {'id': 6, 'title': 'Reverse-engineered modular task-force', 'author':
'Rebecca Coffey', 'isbn': '978-0-517-63255-0'} added.
```

```
Book {'id': 101, 'title': 'Monitored uniform synergy', 'author': 'John Webs
ter', 'isbn': '978-1-310-60077-7'} added.
Book {'id': 102, 'title': 'Progressive upward-trending function', 'author':
 'Carl Gomez', 'isbn': '978-1-109-74780-5'} added.
Book {'id': 103, 'title': 'Optional intangible data-warehouse', 'author': '
Tammy Moore', 'isbn': '978-0-945067-39-9'} added.
Book {'id': 104, 'title': 'Exclusive uniform budgetary management', 'author
': 'Scott Norris', 'isbn': '978-1-55783-548-2'} added.
devasc@labvm:~/labs/devnet-src/school-library$
```

- **Actualizando el Schools Library**

Here is a list of our books currently in the library.

95	Up-sized multi-tasking neural-net	Lee Trevino
96	Realigned scalable budgetary management	Seth Morgan
97	Intuitive optimal encryption	Jessica Brown
98	User-friendly multi-tasking open system	Anthony Leonard
99	Stand-alone global hardware	Patrick Griffin
100	Public-key value-added open architecture	Mrs. Tonya Trujillo
101	Monitored uniform synergy	John Webster
102	Progressive upward-trending function	Carl Gomez
103	Optional intangible data-warehouse	Tammy Moore
104	Exclusive uniform budgetary management	Scott Norris

- **Cambie el bucle for para tener un rango de 104 a 204.**

```
# Using the faker module, generate random "fake" books
fake = Faker()
for i in range(104   , 205):
    fakeTitle = fake.catch_phrase()
    fakeAuthor = fake.name()
    fakeISBN = fake.isbn13()
    book = {"id":i, "title": fakeTitle, "author": fakeAuthor,
    # add the new random "fake" book using the API
    addBook(book, apiKey)
```

```
 Jones DDS', 'isbn': '978-0-9581425-4-0'} added.
Book {'id': 198, 'title': 'Pre-emptive modular middleware', 'author': 'Josh
ua Gonzalez', 'isbn': '978-0-14-883652-5'} added.
Book {'id': 199, 'title': 'Assimilated systematic firmware', 'author': 'Eli
zabeth Moore', 'isbn': '978-1-57100-696-7'} added.
Book {'id': 200, 'title': 'Focused leadingedge contingency', 'author': 'Jos
e Lopez', 'isbn': '978-0-00-158023-7'} added.
Book {'id': 201, 'title': 'Synergistic bi-directional model', 'author': 'De
bra Turner', 'isbn': '978-0-7244-8657-1'} added.
Book {'id': 202, 'title': 'Enterprise-wide 5thgeneration help-desk', 'autho
r': 'Carla Ramos', 'isbn': '978-1-60743-274-6'} added.
Book {'id': 203, 'title': 'Polarized cohesive encryption', 'author': 'Chery
l Williams', 'isbn': '978-0-8038-9173-9'} added.
Book {'id': 204, 'title': 'Pre-emptive directional system engine', 'author'
: 'Janet Bruce', 'isbn': '978-0-05-726012-1'} added.
devasc@labvm:~/labs/devnet-src/school-library$
```

Resumen :

La pagina School Lirary muestra los datos introducidos en API .

Usamos Postman ya que puede enviar solicitudes HTTP a las Api's y ver las respuestas en formato JSON, XML, HTML o cualquier formato disponible, también vi que tiene la posibilidad de realizar solicitudes con diferentes métodos HTTP como GET, POST, PUT, DELETE, gracias a esta herramienta también se pudo realizar cambios de manera más simplificada.