

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, usará 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**

The screenshot shows the API Explorer interface for the School Library API. The 'Parameters' section shows query parameters: 'includeISBN' (boolean, default false), 'sortBy' (string, default id), 'author' (string, optional), and 'page' (integer, optional). The 'Responses' section shows the response content type as 'application/json'. Below the interface, a terminal window shows the command 'curl -X GET 'http://library.demo.local/api/v1/books' -H 'accept: application/json'' and the resulting JSON response.

```
curl -X GET 'http://library.demo.local/api/v1/books' -H 'accept: application/json'
```

```
{
  "id": 0,
  "title": "IP Routing Fundamentals",
  "author": "Mark A. Sportack"
},
{
  "id": 1,
  "title": "Python for Dummies",
  "author": "Stef Maruch Aahz Maruch"
},
{
  "id": 2,
  "title": "Linux for Networkers",
  "author": "Cisco Systems Inc."
},
{
  "id": 3,
  "title": "NetAcad: 20 Years Of Online-Learning",
  "author": "Cisco Systems Inc."
}
}
```

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

```
devasc@labvm: ~
File Edit View Search Terminal Help
devasc@labvm:~$ curl -X GET 'http://library.demo.local/api/v1/books' -H 'accept: application/json'
[
  {
    "id": 0,
    "title": "IP Routing Fundamentals",
    "author": "Mark A. Sportack"
  },
  {
    "id": 1,
    "title": "Python for Dummies",
    "author": "Stef Maruch Aahz Maruch"
  },
  {
    "id": 2,
    "title": "Linux for Networkers",
    "author": "Cisco Systems Inc."
  },
  {
    "id": 3,
    "title": "NetAcad: 20 Years Of Online-Learning",
    "author": "Cisco Systems Inc."
  }
]
devasc@labvm:~$
```

de libros en formato JSON.

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

GET

/books

Parameters

Cancel

Name	Description
includeISBN boolean (query)	Include in the results the ISBN numbers. Default=false
sortBy string (query)	Sort results using the specified parameter. Default=id
author string (query)	Return only books by the given Author.
page integer (query)	To save resources and bandwidth, larger replies might be broken down into smaller pages with 10 records per page. The RFC5988 (Web Linking) standard in HTTP Reply Headers is used for pagination. The "page" parameter is then used to specify the page number.

ExecuteClear

Responses

Response content type: application/json

Curl

curl -X GET "http://library.demo.local/api/v1/books?includeISBN=true" -H "accept: application/json"

Request URL

http://library.demo.local/api/v1/books?includeISBN=true

Server response

CodeDetails

200

Response body

```
{
  "id": 0,
  "title": "IP Routing Fundamentals",
  "author": "Mark A. Smeetsack",
  "isbn": "978-1578780714"
},
{
  "id": 1,
  "title": "Python for Dummies",
  "author": "Pyter Maruch Ashz Maruch",
  "isbn": "978-0471778646"
},
{
  "id": 2,
  "title": "Linux for Networkers",
  "author": "Cisco Systems Inc.",
  "isbn": "000-000000123"
},
{
  "id": 3,
  "title": "NetAcad: 20 Years Of Online-Learning",
  "author": "Cisco Systems Inc.",
  "isbn": "000-000001123"
}

```

Download

Response headers

```
access-control-allow-origin: *
content-length: 509
content-type: application/json
date: Tue, 18 Apr 2023 21:58:29 GMT
server: Werkzeug/0.14.1 Python/3.8.2

```

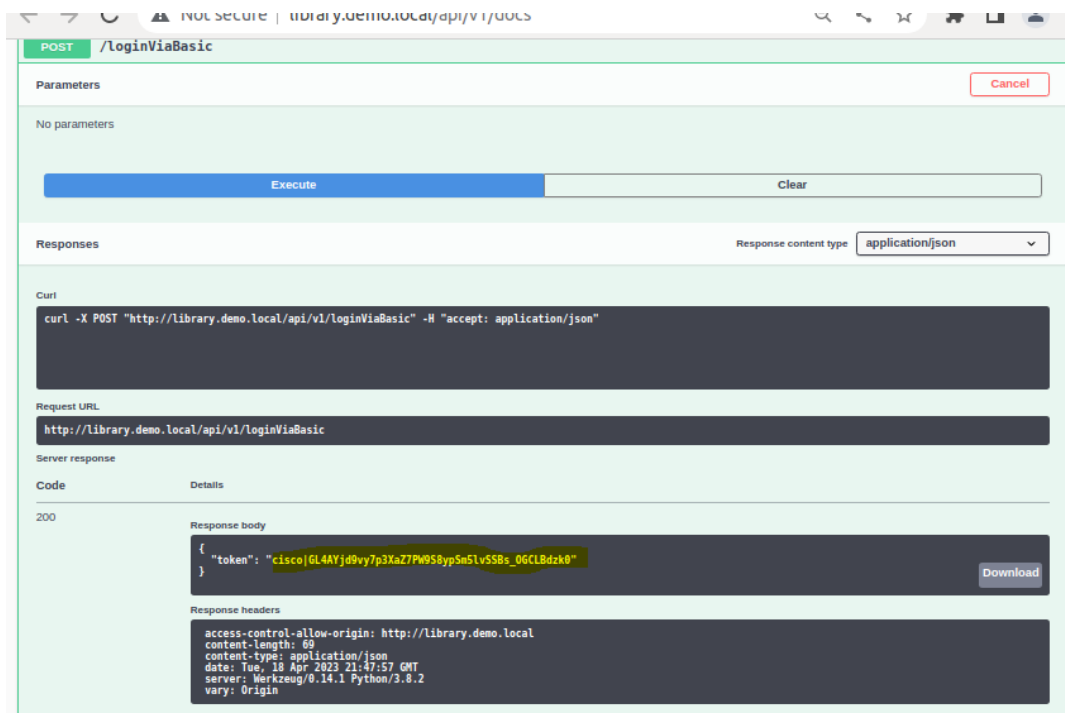
Responses

CodeDescription

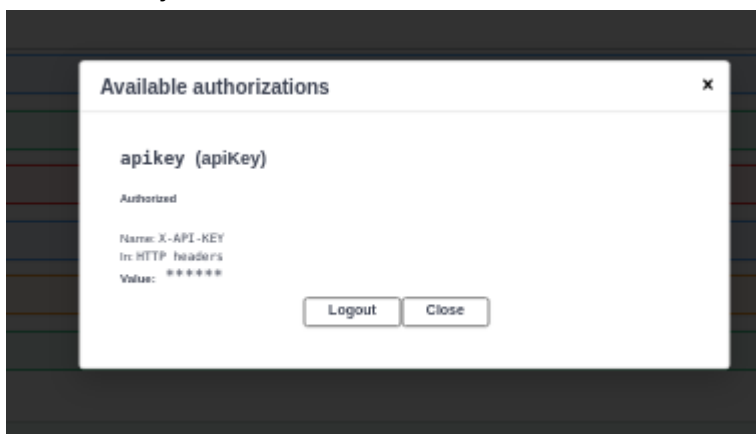
200

Success

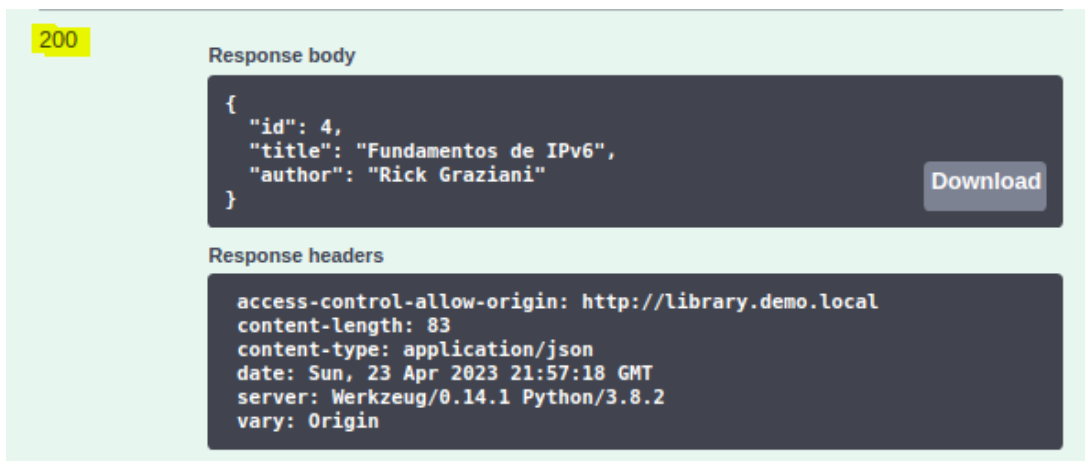
Ahora obtendremos un token usando la **API POST/LoginViaBasic**



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



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



libro añadido, código "200", ahora verificamos que se hallan añadidos 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 días 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 días 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”

GET
/books/{id}

Parameters
Cancel

Name	Description
id <small>★ required</small>	
integer	4
(path)	

Execute
Clear

Curl

curl -X GET "http://library.demo.local/api/v1/books/4" -H "accept: application/json"

Request URL

http://library.demo.local/api/v1/books/4

200

Response body

{
 "id": 4,
 "title": "Fundamentos de IPv6",
 "author": "Rick Graziani"
}
Download

Response headers

access-control-allow-origin: *
content-length: 83
content-type: application/json
date: Sun, 23 Apr 2023 22:12:23 GMT
server: Werkzeug/0.14.1 Python/3.8.2

API DELETE /books {id}

Eliminaremos un libro en específico usando el “id”.

The screenshot shows a REST client interface with a DELETE request to `/books/{id}`. The parameter `id` is set to 4. The response is a JSON object for the book with id 4.

Parameters

Name	Description
id * required	
integer (path)	4

Responses

Response content type: `application/json`

Curl

```
curl -X DELETE "http://library.demo.local/api/v1/books/4" -H "accept: application/json" -H "X-API-KEY: ciscoIgsTNA_38QIav8aSI4FRfsQZUS-gppVar0LPf0XHW1C0"
```

Request URL

```
http://library.demo.local/api/v1/books/4
```

Server response

Code	Details
200	<pre>{ "id": 4, "title": "Fundamentos de IPv6", "author": "Rick Graziani" }</pre>

Response body

```
{
  "id": 4,
  "title": "Fundamentos de IPv6",
  "author": "Rick Graziani"
}
```

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

The screenshot shows the response body of a GET request to `/books`. The response is a JSON array of book objects. The book with id 4 is missing.

Response body

```
{
  "id": 0,
  "title": "IP Routing Fundamentals",
  "author": "Mark A. Sportack",
  "isbn": "978-1578700714"
},
{
  "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": 5,
  "title": "31 dias antes de su examen",
  "author": "Allan Johson"
}
]
```

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.

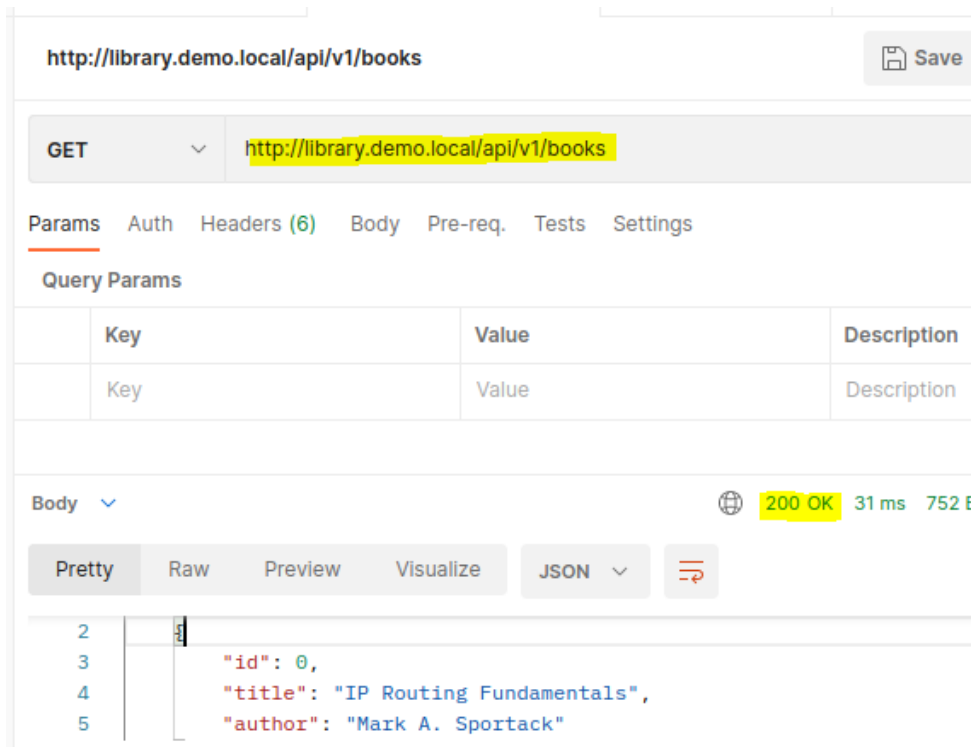
Administrar documentación

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**.

The screenshot shows a REST client interface with a single request selected. The request is a POST to `http://library.demo.local/api/v1/loginViaBasic`. The 'Auth' tab is active, showing 'Username' as 'cisco' and 'Password' as 'Cisco123!'. The response status is '200 OK' with a response time of '9 ms' and a size of '247 B'. The 'Raw' tab is selected, displaying the JSON response: `{"token": "cisco|6dszYDk4h7Cki2IMuc8PsmBZMNGVL__sK5pLGqBdv1g"}`.

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

The screenshot shows a REST client interface with a single request selected. The request is a POST to `http://library.demo.local/api/v1/books`. The 'Body' tab is active, showing the JSON payload: `{ "id": 4, "title": "IPv6 Fundamentals", "author": "Rick Graziani" }`. The response status is '200 OK' with a response time of '10 ms' and a size of '259 B'.

- verificamos el libro adicional con API GET/book

Overview GET http://libi POST http://lib POST http://lib + ... No Environment

http://library.demo.local/api/v1/books Save

GET http://library.demo.local/api/v1/books Send

Params Auth Headers (6) Body Pre-req. Tests Settings

Body 200 OK 5 ms 854 B Save Response

Pretty Raw Preview Visualize JSON

```

20  "author": "Cisco Systems Inc.",
21  },
22  {
23    "id": 4,
24    "title": "IPv6 Fundamentals",
25    "author": "Rick Graziani"
26  },
27  {
28    "id": 5,
29    "title": "31 días antes de su examen",
30    "author": "Allan Johnson"
31  }

```

- Usamos los demás “parámetros” del API GET/book y lo ordenamos mediante los autores

200 Response body

```

{
  "id": 5,
  "title": "31 días antes de su examen",
  "author": "Allan Johnson"
},
{
  "id": 2,
  "title": "Linux for Networkers",
  "author": "Cisco Systems Inc.",
  "isbn": "000-000000123"
},
{
  "id": 3,
  "title": "NetAcad: 20 Years Of Online-Learning",
  "author": "Cisco Systems Inc.",
  "isbn": "000-000000123"
},
{
  "id": 0,
  "title": "IP Routing Fundamentals",
  "author": "Mark A. Sportack",
  "isbn": "978-1578700714"
},
{
  "id": 4,
  "title": "IPv6 Fundamentals",
  "author": "Rick Graziani",
  "isbn": "978 158144778"
},
{
  "id": 1,
  "title": "...",
  "author": "..."
}

```

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

Overview GET http://libi POST http://lib POST http://lib + ... No Environment

http://library.demo.local/api/v1/books?includeISBN=true&sortBy=author Save

GET http://library.demo.local/api/v1/books?includeISBN=true&sortBy=author Send

Params Auth Headers (6) Body Pre-req. Tests Settings

Query Params

Key	Value	Description	...	B
<input checked="" type="checkbox"/> includeISBN	true			
<input checked="" type="checkbox"/> sortBy	author			
Key	Value	Description		

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(), fake
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': 'Timothy 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 Webster', '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)
```

```
Book {'id': 197, 'title': 'Devolved disintermediate array', 'author': 'Sara Jones DDS', 'isbn': '978-0-9581425-4-0'} added.
Book {'id': 198, 'title': 'Pre-emptive modular middleware', 'author': 'Joshua Gonzalez', 'isbn': '978-0-14-883652-5'} added.
Book {'id': 199, 'title': 'Assimilated systematic firmware', 'author': 'Elizabeth Moore', 'isbn': '978-1-57100-696-7'} added.
Book {'id': 200, 'title': 'Focused leadingedge contingency', 'author': 'Jose Lopez', 'isbn': '978-0-00-158023-7'} added.
Book {'id': 201, 'title': 'Synergistic bi-directional model', 'author': 'Debra Turner', 'isbn': '978-0-7244-8657-1'} added.
Book {'id': 202, 'title': 'Enterprise-wide 5thgeneration help-desk', 'author': 'Carla Ramos', 'isbn': '978-1-60743-274-6'} added.
Book {'id': 203, 'title': 'Polarized cohesive encryption', 'author': 'Cheryl 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 Library muestra los datos introducidos en API .

Usamos Postman ya que puede enviar solicitudes HTTP a las APIs 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.