Módulo 2: Captura de datos Python y API Rest

Interactuando con Web Services

Germán Alonso Lascurain – german@campus2b.com

Data BootCamp – Cámara de Comercio Bilbao

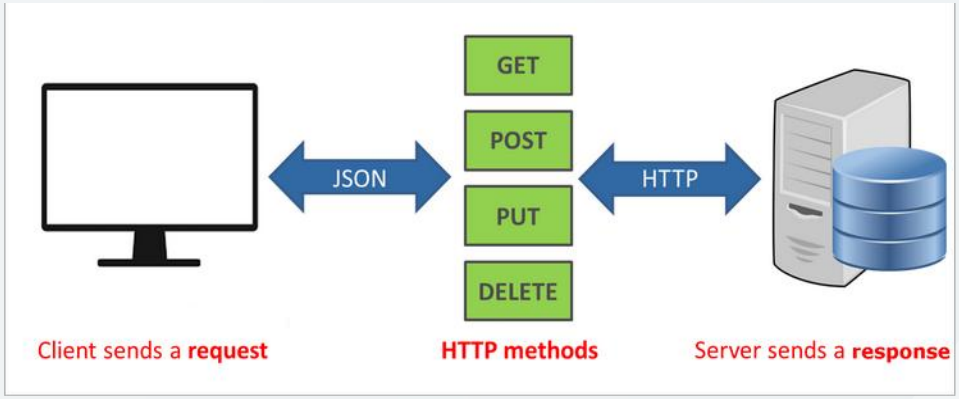
1. Arquitectura REST
2. API Rest y Servicios Web
   i. Métodos HTTP
   ii. Status Codes
   iii. API Endpoints
3. Consumiendo APIs
   i. GET
   ii. POST
   iii. PUT
   iv. PATCH
   v. DELETE

Índice de contenidos

REST = Representational State Transfer

Es una arquitectura de software (**interfaz de programación de aplicaciones**) que define un patrón de comunicaciones CLIENTE-SERVIDOR.

REST facilita el rendimiento, escalabilidad, simplificación de los procesos de acceso a los datos.



Hoy en día, la mayoría de las empresas usan API Rest para crear servicios.

Las API Rest nos permiten creación de servicios web a través de un estándar lógico y eficiente.

Una API es básicamente una forma de hacer una **solicitud a otra aplicación**.

REST nos permite crear servicios y aplicaciones que puedan ser usadas por cualquier dispositivo o cliente que entienda HTTP

Algunos ejemplos:

- Los sistemas de identificación de Facebook,
- La autenticación en los servicios de Google (hojas de cálculo, Google Analytics, ...

Las restricciones que definen un sistema RESTfull son:

- **Cliente-servidor:** esta restricción mantiene al cliente y al servidor débilmente acoplados. Esto quiere decir que el cliente no necesita conocer los detalles de implementación del servidor y el servidor se "despreocupa" de cómo son usados los datos que envía al cliente.

- **Sin estado:** aquí decimos que cada petición que recibe el servidor debería ser independiente, es decir, **no es necesario mantener sesiones.**

- **Cacheable:** debe admitir un sistema de almacenamiento en caché. La infraestructura de red debe soportar una caché de varios niveles. **Este almacenamiento evitará repetir varias conexiones entre el servidor y el cliente para recuperar un mismo recurso.**

Las restricciones que definen un sistema RESTfull son:

- **Interfaz uniforme:** define una **interfaz genérica para administrar cada interacción que se produzca entre el cliente y el servidor** de manera uniforme, lo cual simplifica y separa la arquitectura. Esta restricción indica que cada recurso del servicio REST debe tener una única dirección, “URI”.

- **Sistema de capas:** el servidor puede disponer de varias capas para su implementación. Esto ayuda a mejorar la escalabilidad, el rendimiento y la seguridad.

Un servicio Web REST es un servicio que se adhiere a las restricciones previas.

Los servicios Web Rest, exponen los datos la mundo, a través de una API.

Por ejemplo, al API Rest de Github:

```
https://api.github.com/users/<username>
```

El ejemplo anterior, nos permite consultar vía HTTP información sobre los diferentes usuarios de Github

Existen 5 métodos principales HTTP:

1) **GET**. Recupera información de una fuente existente (descarga de dato).

2) **POST**. Crea un nuevo recurso (carga de dato).

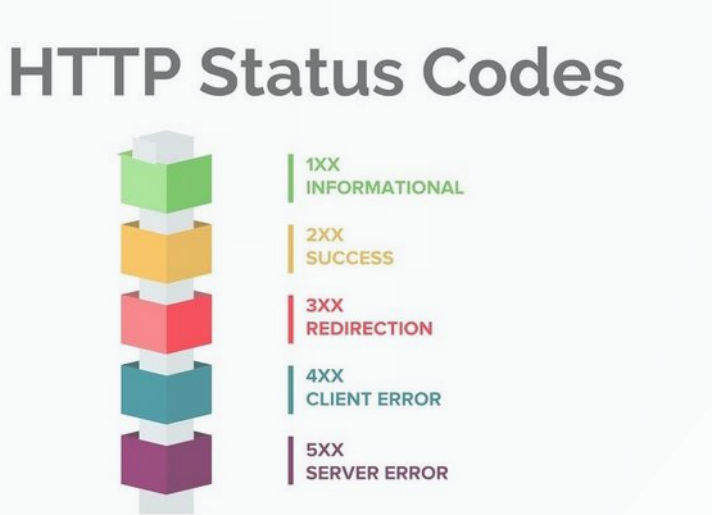
3) **PUT**. Actualiza la información existente.

4) **PATCH**. Actualización parcial de una fuente existente.

5) **DELETE**. Borrar un recurso.

Una vez que la API Rest recibe y procesa una petición HTTP, nos devolverá una **respuesta HTTP.** Incluida en la respuesta va el **Status Code** que muestra el resultado de la petición.

Obtendremos diferentes Status Codes en función de si la petición se ha procesado correctamente o ha habido errores.



El listado de errores por rango es el siguiente:

Rango	Categoría
2xx	Respuestas satisfactorias
3xx	Redirección
4xx	Errores de cliente
5xx	Errores de los servidores

El listado con algunos de los errores es el siguiente:

Código	Significado	Descripción
200	OK	La solicitud ha tenido éxito.
201	Creado	La solicitud ha tenido éxito y se ha creado un nuevo recurso como resultado de ello.
202	Aceptado	La solicitud se ha recibido, pero aún no se ha actuado. Está pensado para los casos en que otro proceso o servidor maneja la solicitud, o para el procesamiento por lotes.
204	Sin contenido	La petición se ha completado con éxito pero su respuesta no tiene ningún contenido.

El listado con algunos de los errores es el siguiente:

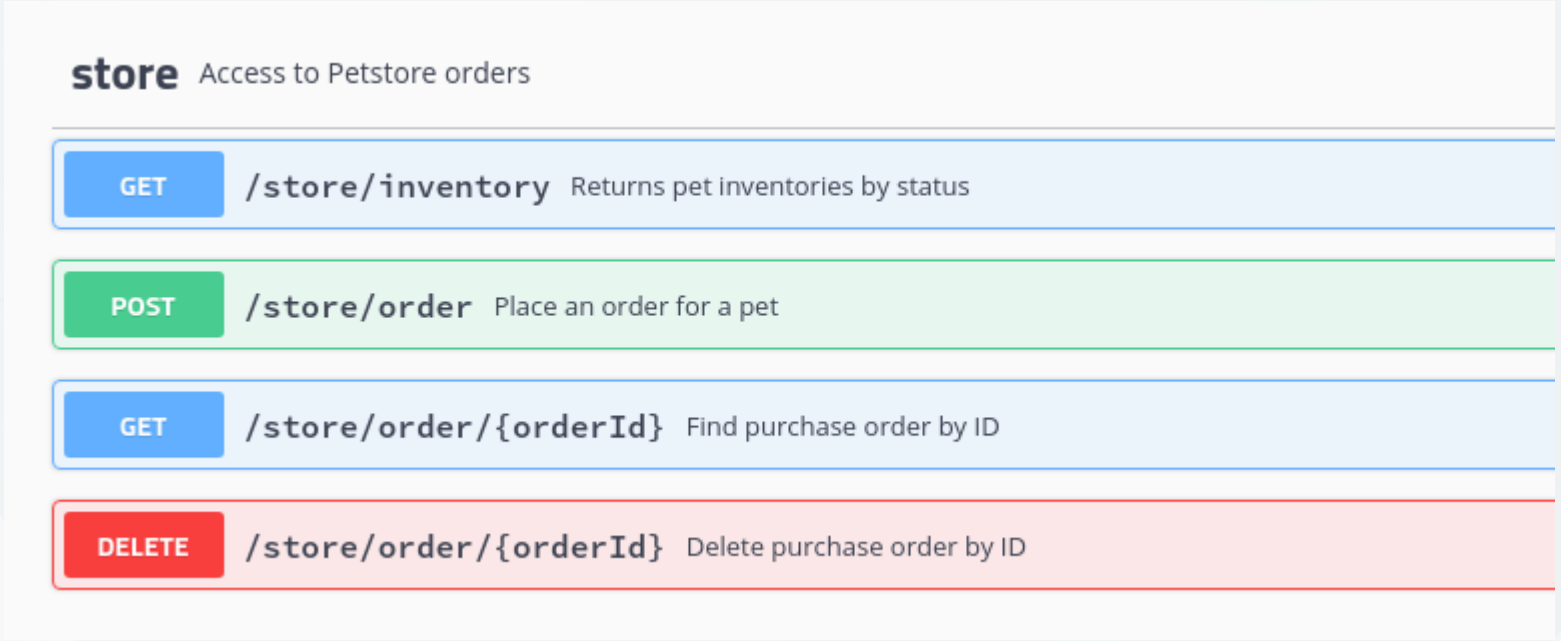
Código	Significado	Descripción
400	Bad Request	La solicitud no válida.
401	No autorizado	La petición no se ha procesado por carecer de credenciales validas de autenticación.
403	Prohibido	El cliente no posee los permisos necesarios para cierto contenido
404	No encontrado	El servidor no pudo encontrar el contenido solicitado.
415	Tipo no soportado	El recurso de destino no admite el formato de datos del cuerpo de la solicitud especificado en la cabecera de Content-Type.

El listado con algunos de los errores es el siguiente:

Código	Significado	Descripción
500	Error interno	El servidor ha encontrado una situación que no sabe cómo manejarla.
503	Servicio no disponible	El servidor no está listo para manejar la petición. Causas comunes puede ser que el servidor está caído por mantenimiento o está sobrecargado.

La API Rest, permite exponer un conjunto de URLs públicas, de manera que las aplicaciones cliente pueden acceder a sus recursos vía servicios web.

Es lo que llamamos ENDPOINTS



Acceder a:

https://petstore.swagger.io/

https://api.example.com/clientes

Método HTTP	API Endpoint	Descripción
GET	/clientes	Obtener la lista de clientes
GET	/clientes/<clientes_id>	Obtener un clientes
POST	/clientes/<clientes_id>	Crear un nuevo cliente
PUT	/clientes/<clientes_id>	Actualizar un cliente
PATCH	/clientes/<clientes_id>	Actualización parcial de un cliente
DELETE	/clientes/<clientes_id>	Borrar cliente

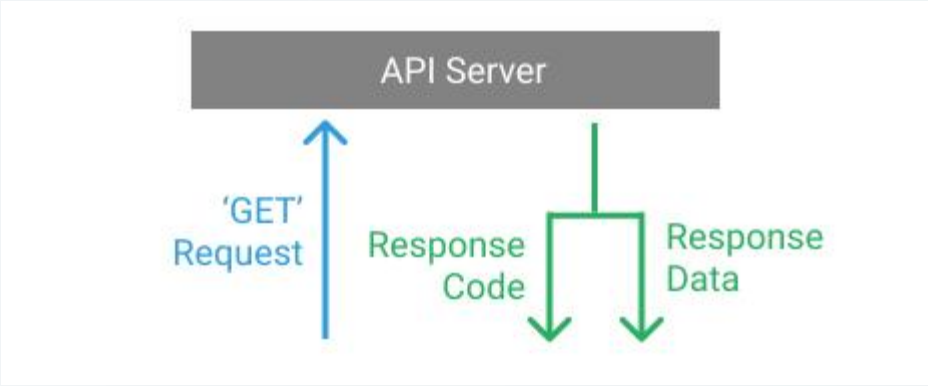
Interacción con las APIs a través de la librería **requests**

```
pip install requests
import requests
```

```python
def execute_dax(access_token, dataset_id, dax_query):
    '''Execute a DAX query using the Power BI REST API'''

    api_url = f'https://api.powerbi.com/v1.0/myorg/datasets/{dataset_id}/executeQueries'

    headers = {
        'Content-Type': 'application/json',
        'Authorization': 'Bearer ' + access_token
    }

    r = requests.post(
        api_url,
        headers = headers,
        json = {
            'queries': [
                {
                    'query': f'{dax_query}'
                }
            ]
        }
    )

    r.raise_for_status()
    r.encoding = 'utf-8-sig'
    return r.json()
```

GET

- Es el método más habitual.
- Permite recibir información desde una API.
- Es una operación de SÓLO lectura.



POST

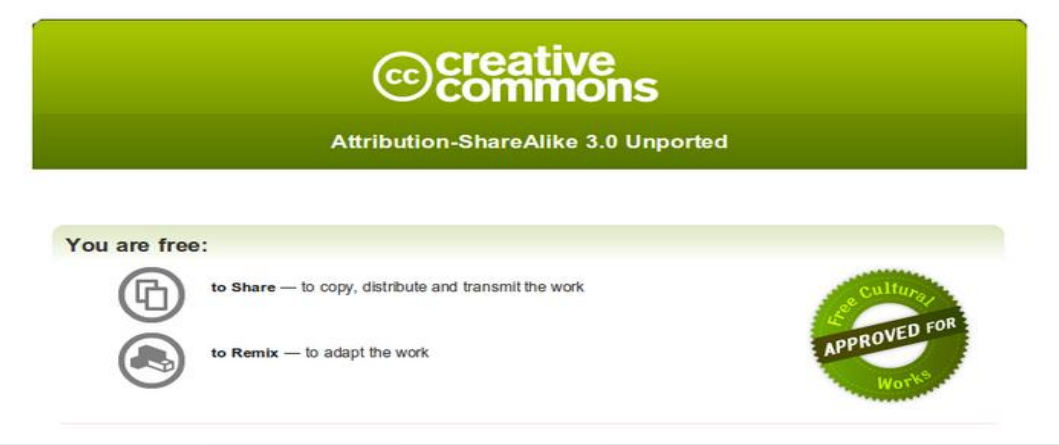
- Permite enviar información a una API.
- Es una operación de escritura.
- Se genera un registro (o varios) completamente.

PATCH

- Permite enviar información a una API.
- Es una operación de escritura.
- Permite actualizar parcialmente, un registro (o varios).

DELETE

- Permite eliminar información vía API.
- Es una operación de escritura.
- Permite eliminar un registro (o varios).



Germán Alonso Lascurain
german@campus2b.com
germanalonso@opendeusto.es