T. Cicchini y S. Pinto

Clase Nº 18

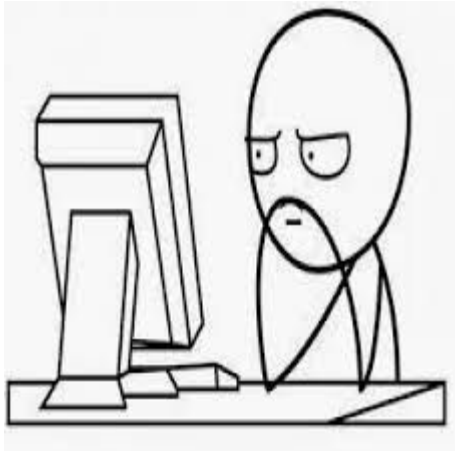
Interacción con páginas webs mediante APIs

Motivación

Imaginemos que:

- Queremos armar una aplicación de alertas o recomendaciones

- Queremos empezar una investigación y necesitamos datos



Motivación

Imaginemos que, además:

- La aplicación con la que queremos trabajar nos facilita la comunicación mediante determinadas reglas

Motivación

Entonces, podremos usar este mecanismo de interacción, para, entre otras cosas, atenernos al marco que establece la aplicación para hacerlo.

Motivación

Entonces, podremos usar este mecanismo de interacción, para, entre otras cosas, atenernos al marco que establece la aplicación para hacerlo.

Por lo menos hasta que nos cansemos, no entendamos la documentación (o simplemente, no tenga API), y scrapiemos...

Me dijiste que la documentación estaba clara

En chino

Motivación

Entonces, podremos usar este mecanismo de interacción, para, entre otras cosas, atenernos al marco que establece la aplicación para hacerlo.

Por lo menos hasta que nos cansemos, no entendamos la documentación (o simplemente, no tenga API), y scrapiemos...

...pero para eso, esperamos al martes que viene...

Índice

- ¿Qué son las APIs y para qué se usan?
- ¿Cómo trabajamos con ellas? HTTP: protocolo comunicación con los servidores.
- Tipo de datos que devuelven las APIs.
- Algunos ejemplos (algunos desarrolladas en los colabs).

¿Qué son?

API - Application Programming Interface- es un conjunto de reglas, funciones, etc, que establecen las formas para comunicarnos con determinada aplicación.

Es decir, es una interfaz, distinta a la app, pero que establece cómo interactuar con la misma.

¿Para qué se usan?

Justamente, la utilizamos para establecer una conexión con la aplicación en el marco de las reglas establecidas por la misma.

Muchas veces, utilizar la API resulta obligatorio para poder trabajar con la aplicación.

Otras, hacer uso de la misma nos permite ahorrarnos el engorro de scrapear las páginas y meterse con el tratamiento de los html.

¿Cómo se trabaja con ellas?

Va a depender mucho de la particularidad del caso, pero, en general:

- Mediante un request (con el protocolo HTTP), hacemos un pedido a la aplicación mediante un url y ciertos atributos

- Utilizamos un paquete de python que tenga integrada la API

HTTP

- El Hypertext Transfer Protocol (HTTP) está designado para habilitar comunicaciones entre clientes y servidores.
- HTTP funciona como un protocolo de *request-response* (pedido-respuesta) entre cliente y servidor.
- Ejemplo: un cliente (un navegador web) manda un HTTP request a un servidor, que devuelve una respuesta al cliente. La respuesta contiene no solo información sobre la respuesta en si misma (si fue exitosa por ejemplo) sino también el contenido solicitado.

https://www.w3schools.com/tags/ref_httpmessages.asp

HTTP Métodos

El request se hace a través de diferentes métodos. Los más comunes son:

- GET: es el utilizado para pedirle datos a un servidor (un pedido de información típico).

- POST: es el utilizado para enviar datos a un servidor para crear o modificar un recurso (por ejemplo, para transmitirle claves de acceso).

https://www.w3schools.com/tags/ref_httpmessages.asp

HTTP status code

Cuando hacemos un request siempre obtenemos un código de respuesta que nos da información sobre la respuesta en si misma. Algunos típicos son:

200 OK		The request is OK (this is the standard response for successful HTTP requests)
404 Not Found		The requested page could not be found but may be available again in the future
400 Bad Request		The request cannot be fulfilled due to bad syntax
401 Unauthorized		The request was a legal request, but the server is refusing to respond to it. For use when authentication is possible but has failed or not yet been provided

https://www.w3schools.com/tags/ref_httpmessages.asp

HTTP status code

429 Too Many Requests

- Requests limits: en la documentación de las APIs vamos a encontrar especificado el límite en la cantidad de requests que podemos hacerle en un determinado periodo de tiempo.
- Si nos pasamos vamos a tener que esperar un tiempo antes de volver a usarlas (hasta que se renueve la cuota de requests).
- Es importante conocer este límite si se va a usar la API sistemáticamente.

https://www.w3schools.com/tags/ref_httpmessages.asp

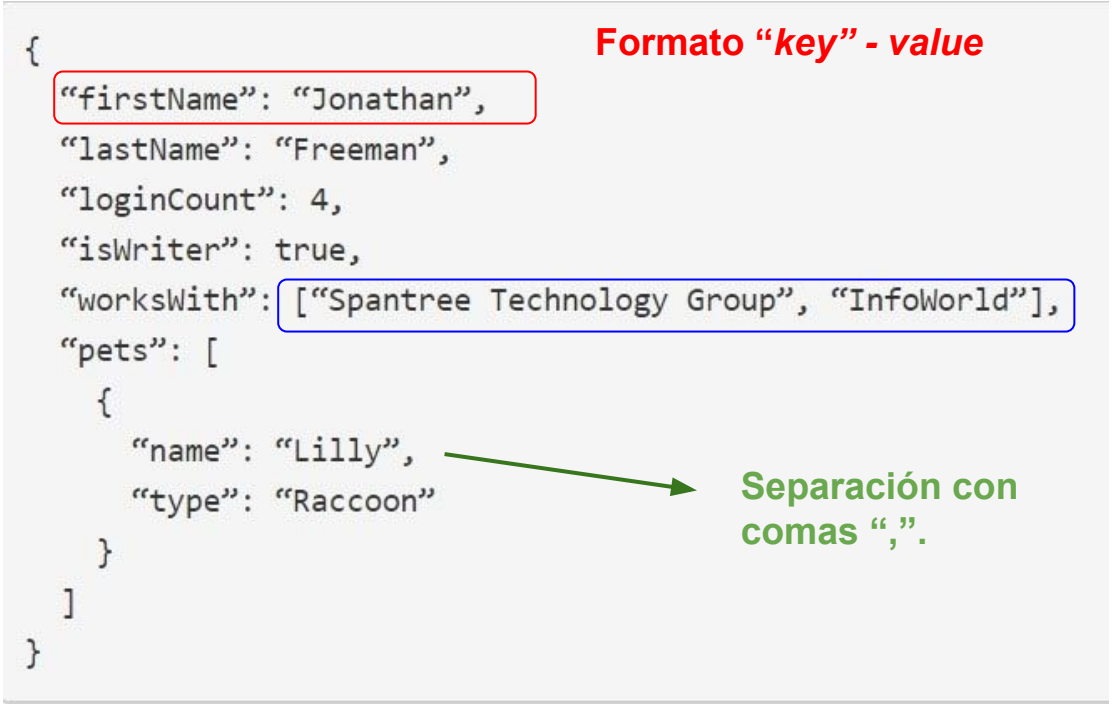
Tipos de Datos

En general, cuando obtenemos una respuesta de la aplicación mediante la API, se nos devuelve información solicitada en determinados formatos.

Lo más usuales resultan:

- json
- xml
- html (en menor medida)

JSON (JavaScript Object Notation)



Pueden contener listas (mismo formato de python "[...]")

Estructura tipo diccionario de python

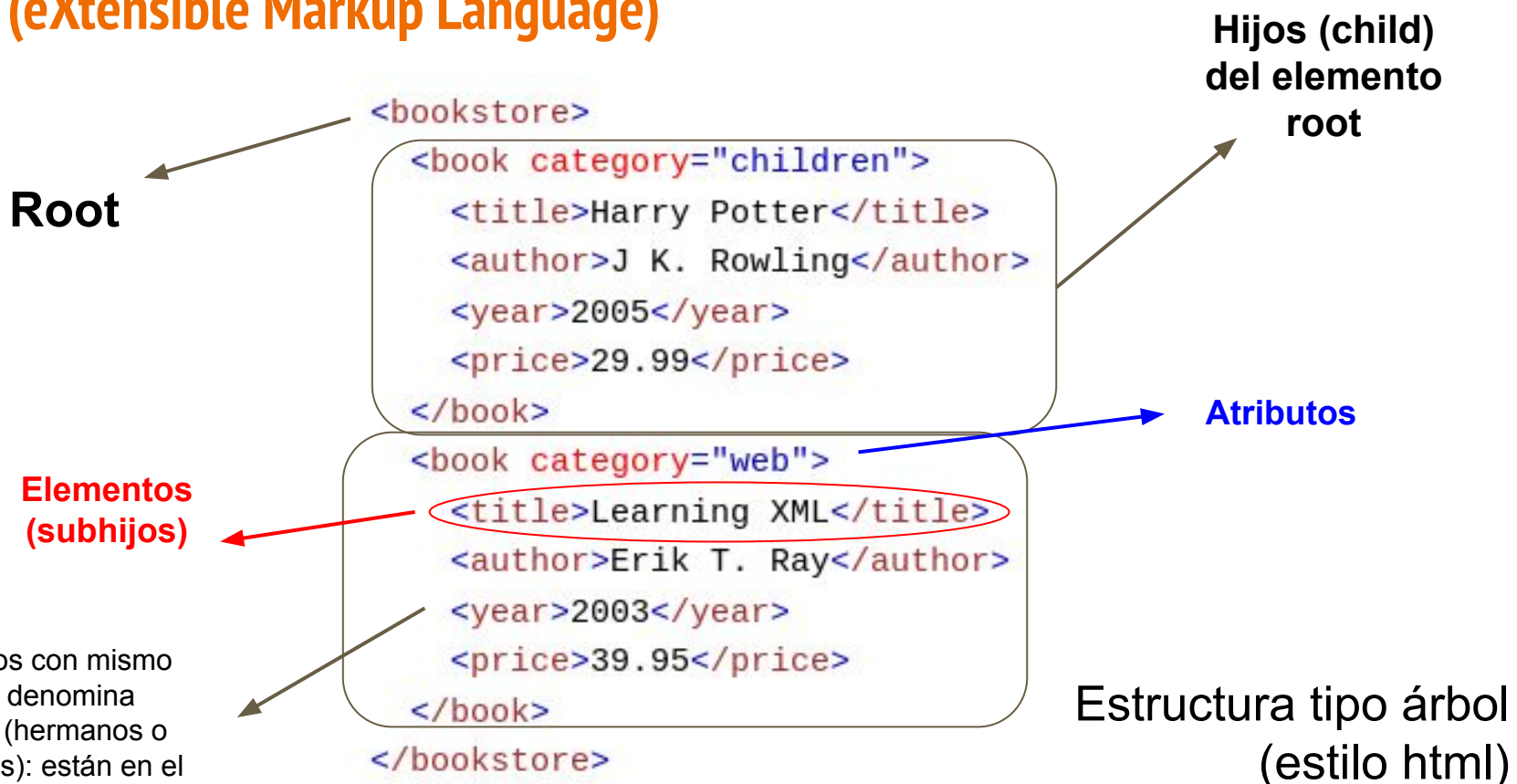
JSON (JavaScript Object Notation)

- El formato es prácticamente igual a los diccionarios y listas de python.

- Los datos están encerrados en llaves “{...}” (diccionario) o corchetes “[...]” (listas).

- Todos los elementos están separados por comas “,”.

XML (eXtensible Markup Language)



Elementos con mismo padre se denomina ***siblings*** (hermanos o hermanas): están en el mismo nivel.

Estructura tipo árbol (estilo html)

XML (eXtensible Markup Language)

- Todos los elementos se denotan entre tags (tag de inicio y tags de cierre). Los saltos de línea y espacios no cumplen ninguna función:

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```

< tag > …. </tag>

- Los nombres son autoexplicativos (no son palabras clave reservadas).

Comparación JSON - XML

```json
{"employees":[
  { "firstName":"John", "lastName":"Doe" },
  { "firstName":"Anna", "lastName":"Smith" },
  { "firstName":"Peter", "lastName":"Jones" }
]}
```

JSON

```xml
<employees>
  <employee>
    <firstName>John</firstName> <lastName>Doe</lastName>
  </employee>
  <employee>
    <firstName>Anna</firstName> <lastName>Smith</lastName>
  </employee>
  <employee>
    <firstName>Peter</firstName> <lastName>Jones</lastName>
  </employee>
</employees>
```

XML

Comparación JSON - XML

- Ambos tipo de datos pueden utilizarse indistintamente.

- En general, JSON es mejor que XML porque suele ser más corto, más fácil de *parsear* (transformarlo en un objeto manipulable) y permite el manejo de listas.

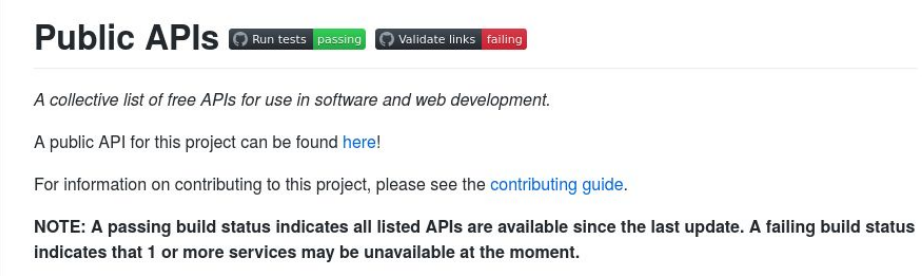
Algunos Ejemplos

Dentro del universo de las APIs, algunas son:

- Redes Sociales:
  - Twitter
  - Spotify
- Organismos gubernamentales:
  - API Transporte
  - Diputados Nación
- Wikipedia
- Google Search (y todo el entorno google, poniendo una tarjeta de crédito...)

Algunos Ejemplos

Solo googlear “lista de APIs” y cosas por el estilo ya nos da varias páginas con muchos ejemplos (no todos necesariamente gratuitos):

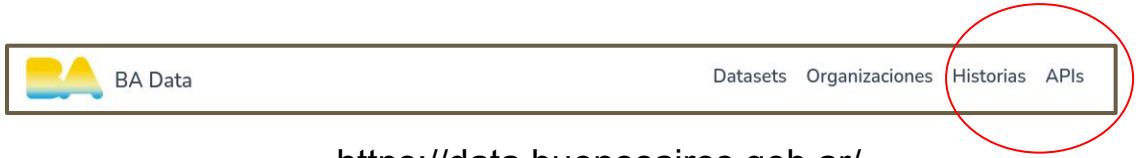


https://github.com/public-apis/public-apis

https://betterprogramming.pub/a-curated-list-of-100-cool-and-fun-public-apis-to-inspire-your-next-project-7600ce3e9b3

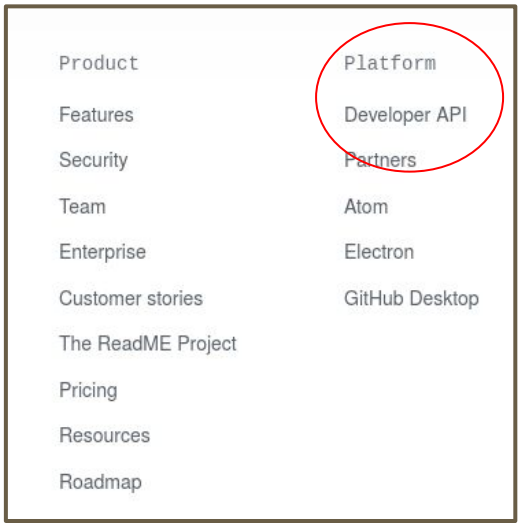
Algunos Ejemplos

Siempre buscar una sección “desarrolladores” (“developers”) o “API” en las plataformas de interés (generalmente al final de la página):



https://data.buenosaires.gob.ar/

https://www.mercadolibre.com.ar/



https://github.com/

Paquetes de Python

Sí o sí, van a resultar necesarios paquetes que permitan generar pedidos a un url determinado, por ejemplo:

- requests

Para el caso de necesitar interactuar con los objetos retornados, pueden ser necesarios:

- json
- bs4 (BeautifulSoup)

Algunas aplicaciones, tienen sus propios paquetes, por ejemplo:

- Tweepy
- Spotipy