Ayudantía: APIs y Webscraping

IMT2200 2025-2

Contenido

1 Introducción

En esta ayudantía veremos de manera introductoria:

- Qué es HTML y porqué nos sirve saber de él.
- Qué son los protocolos y cómo se relacionan con la web.
- Ejemplos prácticos de Webscraping en Python.
- Ejemplo de uso de una API.

---

2 ¿Qué es HTML?

HTML es el lenguaje de marcado que estructura el contenido de la web, de ahí vienen los archivos *nombre.html* . Estos archivos son los que visualizamos a la hora de entrar a cualquier página web, es lo que los navegadores procesan y muestran en nuestras pantallas.

Este lenguaje tiene una estructura y una lógica. No es necesario que conozcamos todas sus características, pero sí entendamos cómo se comporta.

Los bloques que construyen un *archivo.html* son las **etiquetas/tags** (eg: `<h1>`,`<p>`,`<div>`, etc), cada una de estas es interpretada como un distinto tipo de información o bloque y contienen características intrínsecas (que pueden ser modificadas).

Por ejemplo :

```
<h1>Hola mundo</h1>
<p>Este es un párrafo con un <a href="https://www.wikipedia.org">link</a>.</p>
```

Se renderizaría en una página web así :

Hola mundo

Este es un párrafo con un link.

Podemos hacernos una idea visual del comportamiento de los elementos como sigue:

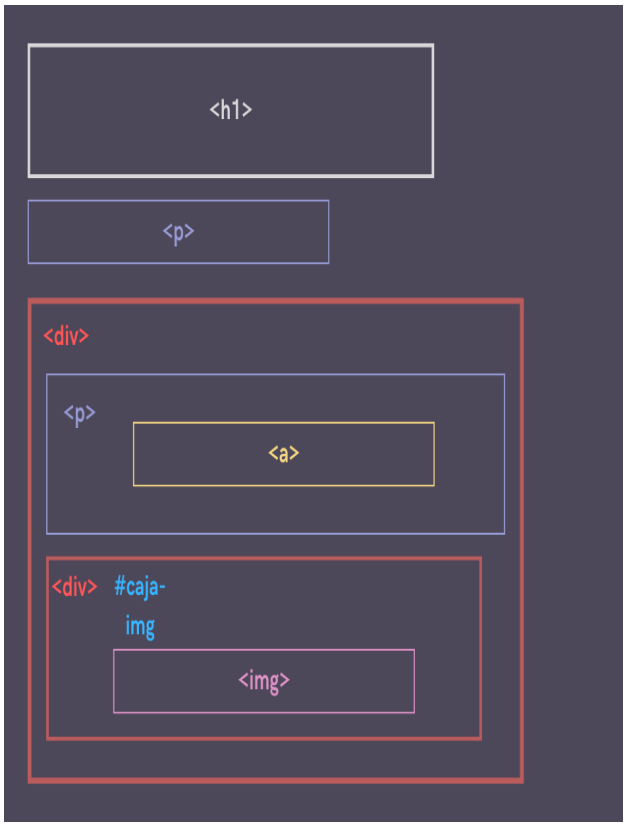

Figure 1: Esquema HTML

Es bueno pensar que cada etiqueta **html** es como un objeto de **python**, y por lo tanto tiene atributos (que son las cosas que podemos editar), los que nos sirve saber para este curso :

- `id`: Identificador único de algún elemento.
- `class`: Clase o categoría a la que pertenece el elemento, por ejemplo: `class="img-pequenas"`.
- `style`: Estilos CSS aplicados al elemento. Aquí podemos modificar márgenes, anchos, alturas, relleno, colores, etc. Es todo otro lenguaje, que se escapa del alcance del curso.

Ahora, que conocemos el material con el que se trabaja en la web, veamos cómo se comunican los diferentes agentes de este proceso, y que hacen posible la existencia del internet.

3 Protocolos: HTTP y HTTPS

Para que sea posible acceder a una página web, dependiendo de su contenido, pasan una serie de procesos y protocolos. Tenemos **HTTP** (HyperText Transfer Protocol) y su versión segura **HTTPS** (HTTP Secure).

Podemos decir que este protocolo se compone de los siguientes pasos:

- Tu navegador (cliente) envía una solicitud (request).(¿A quién?)
- Digamos que a otro computador, el servidor le llamaremos, quien recibe esa señal, dependiendo del mensaje, da una respuesta.
- la respuesta llega, y el navegador la procesa para mostrarla al usuario.

(esta ejemplificación está muy simplificada, pero es la idea general)

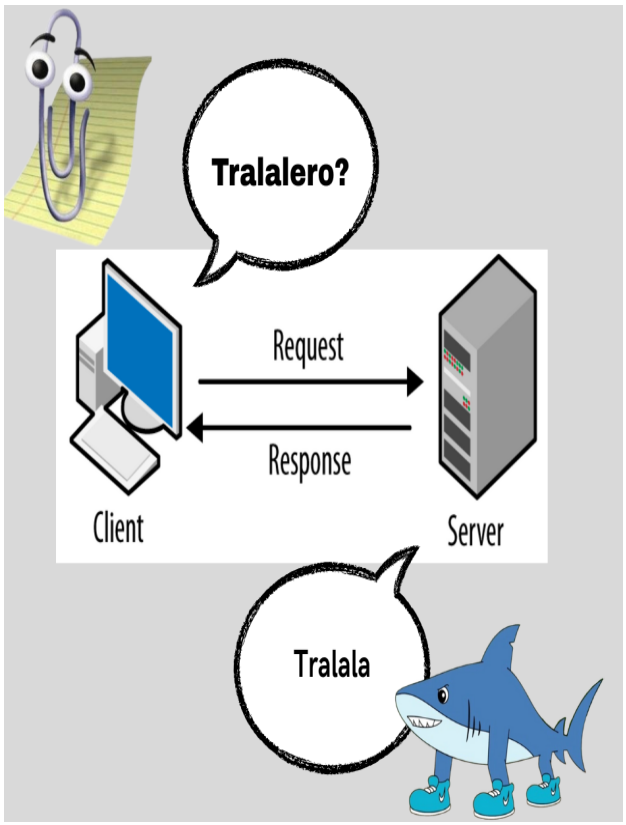

Figure 2: Arquitectura Cliente-Servidor

3.0.1 Las solicitudes (HTTP Requests)

Las solicitudes HTTP son la forma en que los navegadores y otros clientes se comunican con los servidores. Estas anticipan la naturaleza de la interacción que se desea tener con el servidor. Las más relevantes son:

- **GET**: Solicita datos del servidor. Es la más común y se utiliza para obtener información.
- **POST**: Envía datos al servidor. Se utiliza para enviar información, como formularios.
- **PUT**: Actualiza datos en el servidor.
- **DELETE**: Elimina datos del servidor.

Ejemplo:

Cuando escribes `https://www.google.com`, tu navegador hace un GET al servidor de Google. Google recibe la solicitud, la revisa y responde con su página de inicio. Esto finalmente se muestra en tu navegador.

---

4 Webscraping

¿De que nos sirve saber todo esto? En ciencia de datos, muchas veces queremos extraer información de páginas web. Entonces, ahora que ya conocemos, a grandes rasgos, el comportamiento y estructura de la web, podemos entender mejor los procesos que son necesarios para el Webscraping.

Pero ¿qué es el Webscraping? Es la técnica utilizada para extraer información de sitios web de manera automatizada (cuando no podemos obtener la información de otra forma). En esta ocasión ocuparemos algunas librerias de python para hacer el scrapping HTML de las páginas web.

Veamos un pequeño ejemplo :

```
import requests
from bs4 import BeautifulSoup

url = "https://quotes.toscrape.com/"
response = requests.get(url) # Hacemos la solicitud GET
print(response.status_code) # 200 es que todo salió bien
print(response)
```

```
200
<Response [200]>
```

¿Qué esta pasando?

`response.text` nos da el HTML (de la página que solicitamos) como un string. Luego usaremos *BeautifulSoup* para transformar ese string de HTML en una estructura más manejable.

```
bs = BeautifulSoup(response.text, 'html.parser') # Parseamos el HTML
quotes = [q.get_text() for q in bs.select(".quote span.text")]
quotes[:5]
```

```
['"The world as we have created it is a process of our thinking. It cannot be changed without cha
 '"It is our choices, Harry, that show what we truly are, far more than our abilities."',
 '"There are only two ways to live your life. One is as though nothing is a miracle. The other is
 '"The person, be it gentleman or lady, who has not pleasure in a good novel, must be intolerably
 ""Imperfection is beauty, madness is genius and it's better to be absolutely ridiculous than abs
```

Pero lo que buscamos es información específica.Con `.select()` buscamos dentro del HTML todos los elementos que cumplan con lo especificado en los argumentos.

En este caso:

`.quote` significa "cualquier elemento con la clase quote" (las clases se designan con un "`.`").

`span.text` significa "dentro de eso, los `<span>` que tengan la clase text".

Esto nos devuelve una lista de esos elementos solicitados. Luego simplemente imprimimos los primeros 5 para verificar que se ejecutó correctamente nuestro código.

5 APIs

Como ya vieron en clases, también existen las APIs (Application Programming Interfaces) que nos permiten obtener datos de forma estructurada, sin necesidad de webscraping. Las APIs también requieren cierto protocolo y estructura, dependen sobre todo del proveedor. Algunas son de libre acceso, otras pagadas, otras requieren autenticación, etc.

Ejemplo manejo de una API pública (Pokémon):

```
import requests

url = "https://pokeapi.co/api/v2/pokemon/pikachu"
data = requests.get(url).json() # pasamos la respuesta a JSON
data["abilities"] # JSON podemos acceder a los atributos como si fuera un diccionario (tiene estr
```

```
[{'ability': {'name': 'static', 'url': 'https://pokeapi.co/api/v2/ability/9/'},
  'is_hidden': False,
  'slot': 1},
 {'ability': {'name': 'lightning-rod',
   'url': 'https://pokeapi.co/api/v2/ability/31/'},
  'is_hidden': True,
  'slot': 3}]
```