



Universidad Nacional Autónoma de México

Facultad de ciencias

Semestre 2023 - 1

Manejo de datos

Web Scraper

Basaldud Ortega Rodrigo

Daniel Pérez Isaac

López Canche Luis Brayan

- **Marco Teórico**

Existe una inmensa cantidad de información en la Web, mucha de esta información es muy útil para una persona en particular o alguna empresa, la forma de obtener esta información es una parte muy importante.

Ahora en Internet hay diferentes tipos de páginas web, con diferentes funciones y objetivos, es importante las características de estas porque con estos datos se puede tomar la decisión de qué tipos de páginas web se puede extraer la información.

Las tecnologías nos permiten automatizar la forma de obtener datos de Internet, Web Scraping es la técnica de software para automatizar la obtención de estos datos, en la actualidad existen varias herramientas de Web Scraping, tanto software en diferentes lenguajes y también empresas que dan este servicio.

Web Scraping (raspado web o extracción de datos web) es el proceso de extracción de datos de sitios web, preferiblemente usando un programa que simula la exploración humana mediante el envío de peticiones HTTP simples o emulando un navegador web completo. Web Scraping, Content Scraping, Screen Scraping, Web Harvesting o Web Data Extraction son todos términos análogos. En general, cualquier cosa que se puede ver en Internet puede ser extraída y este proceso puede ser automatizado. Web Scraping se utiliza normalmente por diferentes motivos, como la detección de cambio, la investigación de mercado, seguimiento de datos y en algunos casos hasta el robo de datos.

El funcionamiento de Web Scraping consiste en una simulación de la navegación, que una persona realizaría, ya sea implementado a bajo nivel, como en HTTP (Hypertext Transfer Protocol), o incluido en ciertos navegadores web. Ésta simulación se realiza mediante programas denominados “bots”, que crean una clonación del cliqueo, la lectura y el “copypaste”, automatizando la tarea de búsqueda y de recolección de datos. De modo que, Web Scraping se centra en la transformación del contenido no estructurado, por lo general en formato HTML, en datos estructurados que pueden ser almacenados y analizados.

## ● Pasos para realizar un web scraper

De manera general el proceso para el web scraper es:

1. Recopilar las URL de las páginas de las que se desea extraer datos.
2. Realizar una solicitud a estas URL para obtener el HTML de la página.
3. Utilizar localizadores para encontrar los datos HTML.
4. Guardar los datos en formato estructurado.

A continuación presentamos los pasos un poco más detallados para la realización del proyecto.

Lo primero es importar librerías en Python, que nos ayudarán a lo largo del código.

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
import numpy as np
import pandas as pd
from datetime import datetime
from IPython.display import display,HTML
```

La librería **Selenium**, es de vital importancia, pues es un recurso que se utiliza para automatizar pruebas de sistemas. Esta herramienta permite al usuario reproducir el ambiente real de la aplicación. La librería de **Pandas** nos ofrece unas estructuras muy poderosas y flexibles que facilitan la manipulación y tratamiento de datos.

Ya que tenemos listo esto, pasamos a crear algunas funciones para poder replicarlo con distintas páginas web. La primera la llamaremos “*List maker attribute*”, donde se centrará en crear listas o arrays de las url´s de la página. Como parámetros le pasaremos un *Tagname*, que se refiere al nombre del elemento en el código HTML, el valor del atributo especificado en el elemento que se le conoce como “*Getattribute*” y las repeticiones, la cual es el número de veces que se va a repetir el proceso, en este caso queremos que se repita tantas veces como elementos tenga la lista.

```
# crea lista de urls (tag name y get attribute)
def list_maker_attribute(repeticion,lista,tag_name,attribute) -> list: #para todos
    lista_final=[]
    for i in range(repeticion):
        try:
            lista_final.append(lista[i].find_element(By.TAG_NAME,tag_name).get_attribute(attribute))
        except Exception:
            lista_final.append(np.nan)
    return lista_final
```

Una segunda función es “*List maker class*”, que es la encargada de crear una nueva lista, pero a diferencia de la anterior, sus elementos son el HTML específico indicado por la **classname**; sus parámetros son nuevamente la repetición, que es la misma idea que la función anterior, y el classname.

```
def list_maker_class(repeticion,lista,class_name) -> list: #para todos
    lista_final=[]
    for i in range(repeticion):
        try:
            lista_final.append(lista[i].find_element(By.CLASS_NAME,class_name).text)
        except Exception:
            lista_final.append(np.nan)
    return lista_final
```

Finalmente agregamos dos funciones más, “*df to excel*” e “*imagen*”, la primera nos pasa todos los datos recolectados a un archivo de excel para un mejor manejo de información y la última nos va a ayudar a visualizar una imagen del producto cuando se ejecute nuestro código.

```
def df_to_excel(dataframe,tienda,producto): #PARA TODOS
    dataframe.to_excel(f"scraper_{tienda}_{producto}.xlsx")

def imagen(path): #PARA TODOS
    return f''
```

Con estas funciones cumplimos el primer objetivo, *recopilar las URL de las páginas de las que se desea extraer datos*.

A continuación, procederemos a crear la instancia de la clase WebDriver, esperamos hasta que se haya cargado por completo antes de devolver el control a su prueba o secuencia de comandos. Un método que vamos a utilizar es get navegará a una página proporcionada por la URL. Por ejemplo, si nosotros quisiéramos buscar un producto en la tienda online de Walmart, tendríamos que ingresar la URL del sitio. Presentamos un ejemplo de cómo se vería en código.

```
driver = webdriver.Chrome(service=Service("C://Users//basal//Downloads//chromedriver_win32//chromedriver.exe"))
driver.implicitly_wait(5)
producto="the last of us"
producto=producto.replace(" ", "%20")
driver.get(f"https://www.walmart.com.mx/productos?Ntt={producto}")
driver.implicitly_wait(5)
productos=driver.find_elements(By.CLASS_NAME,"product_container_3DSm4")
```





Notemos que el producto es un juego “*The last of us*”, el cual estamos buscando en Walmart Online.

Ya que tenemos todos los datos, lo que hacemos ahora es clasificar los datos de los productos en listas, es decir una lista corresponde al nombre del producto, otra a la marca, otra a su URL y otra a la imagen del producto. Es importante aquí especificar cómo se encuentra el dato en código HTML; note el ejemplo, la lista “*Lista nombres*” aplicamos la función “*List maker class*”, introducimos el tamaño de la lista (la que corresponde al parámetro repeticiones), el producto (para extraer su nombre) y su identificación HTML.

```
lista_nombres=list_maker_class(len(productos),productos,"nav-link_navLink_2oJ29.product_name_1YFfy")
lista_marcas=list_maker_class(len(productos),productos,"product_brand_1CsRr")
lista_url_imagenes=list_maker_attribute(len(productos),productos,"img","src")
lista_urls_productos=list_maker_attribute(len(productos),productos,"a","href")
```

Finalmente creamos un diccionario que va a tener como elementos los datos del producto (marca, producto, precio, imagen y url) y como clave las listas anteriores. Con este diccionario creamos un dataframe en pandas, quitamos algunos caracteres especiales con un replace, para que al momento de mostrar la tabla que nos genera el código sea mucho mejor. Mostramos como debe de quedar el resultado final.

```
data={"MARCA":lista_marcas,"PRODUCTO":lista_nombres,"PRECIO":lista_precios,"IMAGEN":lista_url_imagenes,"LINK":lista_urls_productos}
df_walmart=pd.DataFrame(data)
df_walmart.insert(0,"FECHA",datetime.now().strftime("%d/%m/%Y"))
df_walmart.insert(1,"TIENDA","WALMART")
df_walmart["PRECIO"]=df_walmart["PRECIO"].str.replace("$"," ")
```

	FECHA	TIENDA	MARCA	PRODUCTO	PRECIO	IMAGEN	LINK
0	16/12/2022	WALMART	playstation 4	The Last of Us Part II PlayStation 4 Edición ...	971.03		<a href="https://www.walmart.com.mx/videojuegos/playstation-4/juegos-ps4/the-last-of-us-part-ii-playstation-4-edicion-estandar_00071171951911">https://www.walmart.com.mx/videojuegos/playstation-4/juegos-ps4/the-last-of-us-part-ii-playstation-4-edicion-estandar_00071171951911</a>
1	16/12/2022	WALMART	playstation 4	The Last of Us Remasterizado PlayStation 4 Fi...	590.00		<a href="https://www.walmart.com.mx/videojuegos/playstation-4/juegos-ps4/the-last-of-us-remasterizado-playstation-4-fisico_00071171952635">https://www.walmart.com.mx/videojuegos/playstation-4/juegos-ps4/the-last-of-us-remasterizado-playstation-4-fisico_00071171952635</a>
2	16/12/2022	WALMART	tubbz	Figura de pato coleccionable de Ellie de TUBB...	817.00		<a href="https://www.walmart.com.mx/juguetes/manualidades-y-juguetes-didacticos/otros-juguetes/figura-de-pato-coleccionable-de-ellie-de-tubbz-the-last-of-us-mercancia-oficial-de-the-last-of-us-tubbz-tubbz_00505628041519">https://www.walmart.com.mx/juguetes/manualidades-y-juguetes-didacticos/otros-juguetes/figura-de-pato-coleccionable-de-ellie-de-tubbz-the-last-of-us-mercancia-oficial-de-the-last-of-us-tubbz-tubbz_00505628041519</a>
3	16/12/2022	WALMART	editorial kamite	THE LAST OF US TOMO EDICION ESPECIAL Editoria...	319.00		<a href="https://www.walmart.com.mx/libros-y-revistas/literatura-y-novelas/clasicos/the-last-of-us-tomo-edicion-especial-editorial-kamite-7500588004658_00750058800465">https://www.walmart.com.mx/libros-y-revistas/literatura-y-novelas/clasicos/the-last-of-us-tomo-edicion-especial-editorial-kamite-7500588004658_00750058800465</a>

## **Bibliografia**

Mitchell Ryan C.(15/12/22) *Web scraping with Python*.  
chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://coddyschool.com/upload/Web-Scraping-with-Python\_proglib.pdf.