

21 DE ABRIL DE 2021

REPORTE DE INVESTIGACION

-PROYECTO-WEB SCRAPING CON FINES DE COMERCIO Y VENTAS-

- **Integrantes:**
 - Lorenzo Pérez Edna Yuritzy
 - Rosas Garcia Eric Jonathan
- **Motivo:** Primera Estancia
- **Periodo:** Marzo-Abril 2021
- **Institución:** Universidad Politécnica Metropolitana de Hidalgo

-Web Scraping - Conceptos y teoría-

❖ Descripción general de Web Scraping

Scrapear, o web scraping, es un método que utiliza líneas de código (normalmente en python) con el fin de conseguir información de páginas web. Estos programas normalmente imitan la forma que tiene un usuario de navegar en la red y recopilan la información indicada en el algoritmo desarrollado. Además, con el web scraping se pueden almacenar los datos de otras webs y después trabajar con esos datos para crear nuevos proyectos.

Usos: Algunos de los usos más comunes del web scraping es comparar precios, monitorizar datos del clima, detectar modificaciones en espacios web o incorporar información determinada.

Gracias a todas estas aplicaciones, el web scraping ha adquirido una gran popularidad en los últimos años, ya que no se necesita de una API para poder extraer la información que deseamos.

(El término API es una abreviatura de Application Programming Interfaces, que en español significa interfaz de programación de aplicaciones. Se trata de un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones, permitiendo la comunicación entre dos aplicaciones de software a través de un conjunto de reglas. Así pues, podemos hablar de una API como una especificación formal que establece cómo un módulo de un software se comunica o interactúa con otro para cumplir una o muchas funciones.)

❖ Frameworks y librerías más conocidos con los que se puede realizar web scraping (en este caso tomando el lenguaje de programación Python)

	SCRAPY	BEAUTIFULSOUP	SELENIUM
¿Qué es?	Es un framework de scraping y crawling, es en código abierto, escrito en Python.	Es una librería de Python para analizar documentos HTML o XML y poder extraer información de estos.	Es una librería de automatización de pruebas.

Ventajas	Tiene módulos para enviar solicitudes y analizar respuestas. Es eficiente. Consume poca memoria y utiliza los recursos de la CPU al mínimo. Es portable.	Fácil de usar Fácil de aprender Compatible con XML Analiza páginas como lo hace un navegador web Crea un HTML5 válido.	Es Versátil Trabaja bien con JavaScript Automatiza los navegadores web
Desventajas	No es fácil de usar	Es ineficiente Requiere dependencias	Es ineficiente No está destinado para hacer web scraping

❖ *Descripción de soluciones de web scraping*

-Selenium

Es un driver que entre otras cosas, permite automatizar las acciones que hace un sistema como si de un humano se tratara, siendo usado ampliamente para pruebas en entornos reales. Algunas de las acciones que puede realizar, si se usa en un browser, son:

- Clickear
- Hacer Scroll
- Hacer paginación
- Ejecutar scripts de JavaScript
- Extraer información (web scraping)

Las herramientas que puede usar selenium son el Selenium Remote Control y Selenium WebDriver.

Selenium Remote Control: Remote Control (RC) es un sistema cliente/servidor que permite utilizar el navegador web de forma local o en otro ordenador. Lo mejor de todo es que se puede utilizar casi cualquier lenguaje de programación y formato de pruebas, pero con la liberalización de Selenium 2 se ha descartado en gran medida para favorecer a WebDriver.

Selenium WebDriver: Siendo el sucesor de Remote Control permite utilizar un navegador de forma local o en remoto. No requiere un servidor especial ya que se inicia en una instancia del navegador y se controla de esta manera. Algunos lenguajes con los que es compatible son: PHP, Java, .Net, Perl, Ruby y Python.

La arquitectura de WebDriver es más simple que la de Remote Control pero eso no significa que sea peor. Por ejemplo, WebDriver es más rápido que RC, ya que habla directamente al navegador y, lo mejor de todo, utiliza el motor del propio navegador que se está utilizando en ese momento.

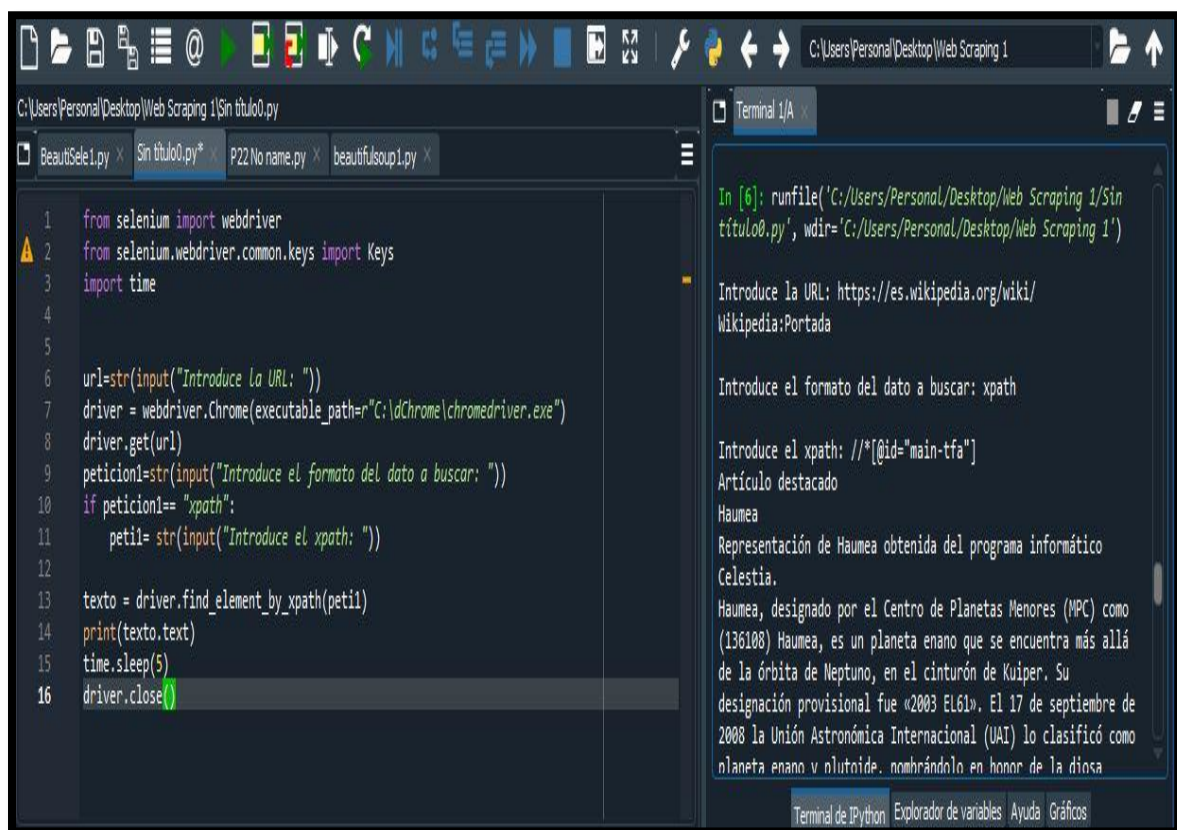
También interactúa con los elementos de la página de manera más realista, simulando las respuestas de un ser humano.

Características en cuanto a funcionalidad para Web Scraping:

- Es versátil
- Trabaja bien con scripts de JavaScript
- Es algo lento
- No está destinado al web scraping, sin embargo es posible realizarlo.
- Ineficiente por sí solo en varios sitios web

Al haber sido creado enfocado al testing, no es muy eficiente para hacer web scraping, aunque es posible combinarlo con otras librerías o frameworks para poder hacer scraping de sitios web donde se necesita interactuar con la página para obtener toda la información de esta.

Ejemplo:



The screenshot shows a Jupyter Notebook environment. The left pane displays a Python script using Selenium WebDriver to interact with a web page. The script prompts the user for a URL and an XPath selector, then finds and prints the element text. The right pane shows the terminal output of the script, which includes the URL 'https://es.wikipedia.org/wiki/Wikipedia:Portada' and the XPath '//*[@id="main-tfa"]', resulting in the text 'Artículo destacado Haumea'.

```
1 from selenium import webdriver
2 from selenium.webdriver.common.keys import Keys
3 import time
4
5
6 url=str(input("Introduce la URL: "))
7 driver = webdriver.Chrome(executable_path=r"C:\dChrome\chromedriver.exe")
8 driver.get(url)
9 petición1=str(input("Introduce el formato del dato a buscar: "))
10 if petición1== "xpath":
11     petil= str(input("Introduce el xpath: "))
12
13 texto = driver.find_element_by_xpath(petil)
14 print(texto.text)
15 time.sleep(5)
16 driver.close()
```

```
In [6]: runfile('C:/Users/Personal/Desktop/Web Scraping 1/Sin título0.py', wdir='C:/Users/Personal/Desktop/Web Scraping 1')

Introduce la URL: https://es.wikipedia.org/wiki/Wikipedia:Portada

Introduce el formato del dato a buscar: xpath

Introduce el xpath: //*[@id="main-tfa"]
Artículo destacado
Haumea
Representación de Haumea obtenida del programa informático Celestia.
Haumea, designado por el Centro de Planetas Menores (MPC) como (136108) Haumea, es un planeta enano que se encuentra más allá de la órbita de Neptuno, en el cinturón de Kuiper. Su designación provisional fue «2003 EL61». El 17 de septiembre de 2008 la Unión Astronómica Internacional (UAI) lo clasificó como planeta enano y plutóide, nombrándolo en honor de la diosa
```

-BeautifulSoup

BeautifulSoup es un paquete (librería) de Python para analizar documentos HTML y XML (incluido el marcado con formato incorrecto, es decir, etiquetas no cerradas, llamadas así por la sopa de etiquetas). Crea un árbol de análisis para las páginas analizadas que se puede utilizar para extraer datos de HTML, lo que es útil para el web scraping.

Características en cuanto a funcionalidad para Web Scraping:

- Es muy fácil de aprender y dominar
- Requiere dependencias
- Poderoso para su simpleza
- Tiene muchas características y funciones
- Ideal para proyectos rápidos
- Ineficiente por sí sola en varios sitios web

Es perfecta para principiantes en el web scraping, además que puede combinarse con selenium y otras librerías para realizar este proceso en diferentes tipos de sitios web independientemente de su estructura.

Esta librería tiene 3 pasos principales al momento de extraer información de páginas web:

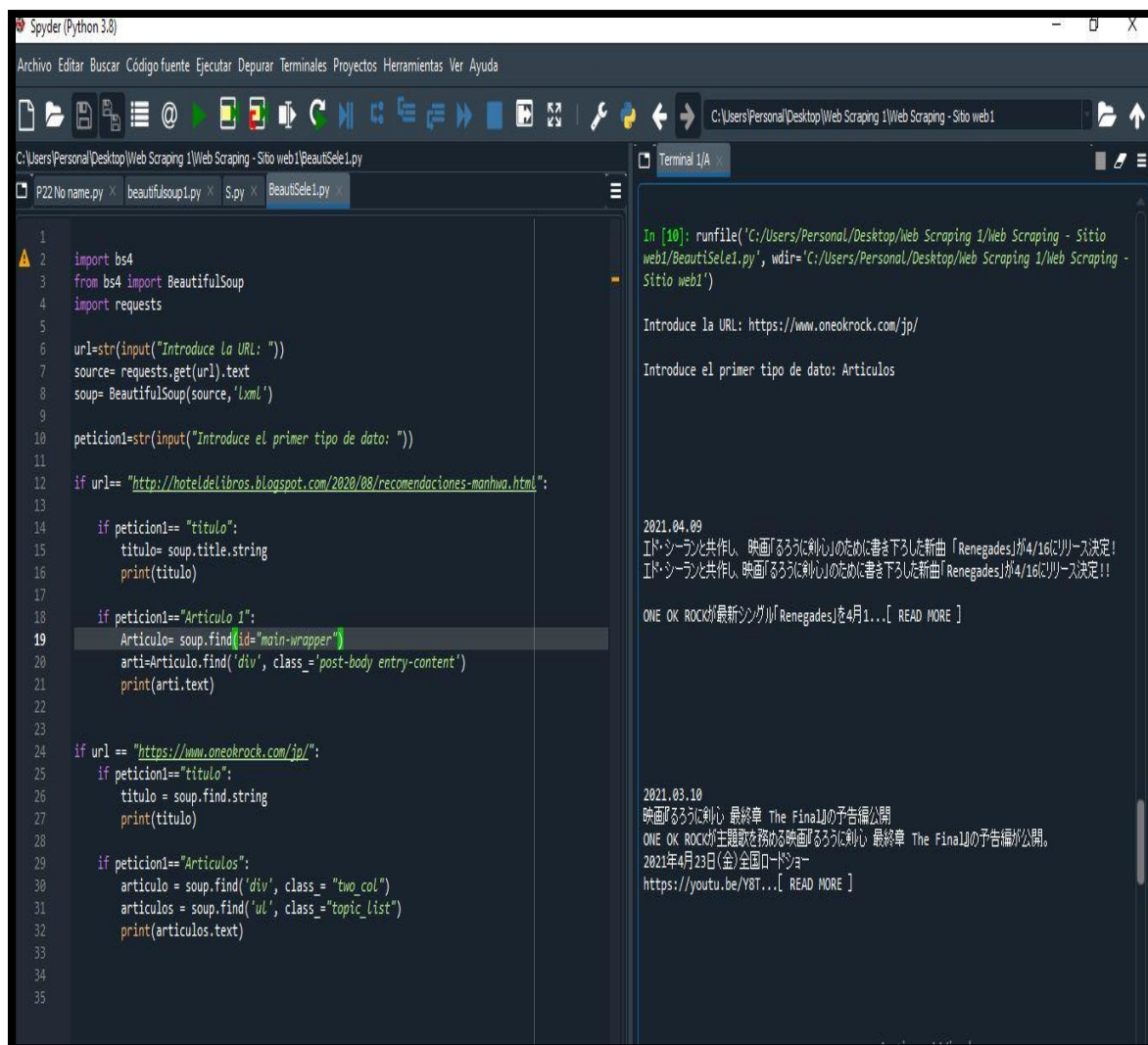
- Obtener la información
- Parsear la información
- Guardar o salida de la información

Obtener la información: Aquí se utilizan los métodos de la librería para obtener todas las etiquetas que se deseen a partir de una URL semilla y reglas de paginación, en el caso de que exista scraping vertical y horizontal.

Parsear la información: Se usan métodos para convertir las etiquetas en formato html, al formato que se desee y se almacena en el mismo, ya sea en una base de datos o algún tipo de archivo.

Guardar la información: Una vez que se parsea la data obtenida, se almacena y se organiza usando librerías como Pandas de python, la cual ayuda a organizar y administrar eficazmente grandes cantidades de información en poco tiempo. Todo esto para usarse posteriormente.

Ejemplo:



-Scrapy

Es el framework de raspado web más popular en Python. Un marco de código abierto y colaborativo para extraer los datos que necesita de los sitios web. De una forma rápida, sencilla pero ampliable.

Scrapy tiene las siguientes características:

- Rápida y poderosa: Escribes las reglas para extraer los datos y scrapy hace el resto.
- Fácilmente extensible: Dada su configuración, puede generar nueva funcionalidad sin tener que modificar el código fuente.
- Portable y Pythonico: Está escrito en Python y puede correr en Linux, Windows, Mac y BSD.

Dado que es un framework, Scrapy tiene una serie de herramientas poderosas para hacer el "scraping" o extraer información de webs de manera fácil y eficiente. Estas herramientas comprenden:

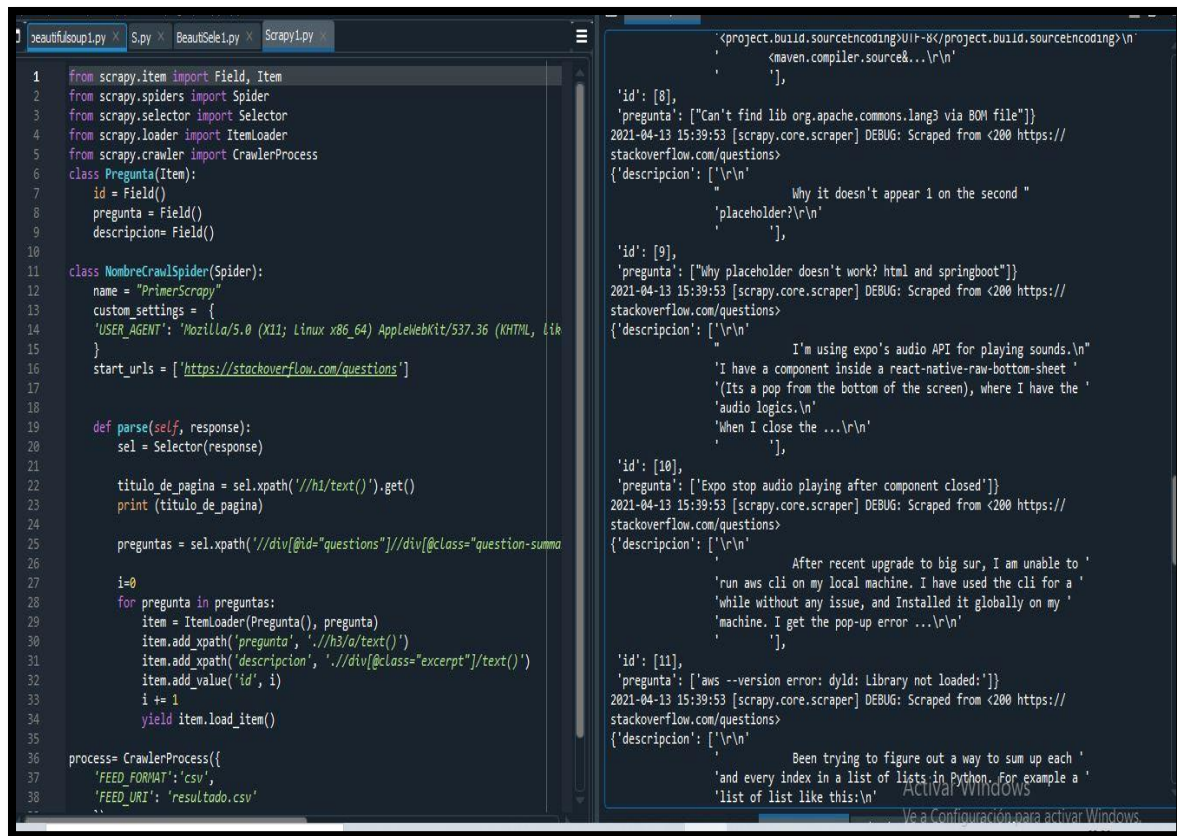
- Soporte para extraer y seleccionar datos de fuentes HTML/XML usando selectores CSS y expresiones XPath, con métodos de ayuda para extraer usando expresiones regulares.
- Una consola interactiva en IPython para ensayar los CSS y expresiones XPath para extraer datos, muy útil cuando se construyen métodos propios.
- Soporte para exportar los registros en formatos múltiples como JSON, CSV y XML.
- Soporte para manejar declaraciones foráneas, no estándares y códigos rotos.
- Fuerte extensibilidad, ya que te permite conectar tu propia funcionalidad usando señales, extensiones y pipelines.

Características en cuanto a funcionalidad para Web Scraping:

- Alta eficiencia
- Es portable
- Trabaja con ítems específicos
- No es muy fácil de aprender, es para usuarios avanzados
- Ideal para proyectos completos
- Poderosa y customisable

Es el framework más popular para realizar web scraping con python, además de ser el más completo, lo cual lo hace perfecto para proyectos complejos y muy estructurados. Se puede combinar con otras librerías para poder acceder a diferentes sitios web sin importar su estructura, lo cual es un limitante para la extracción de datos.

Ejemplo:



The image shows a code editor with two panes. The left pane displays a Python script for a Scrapy spider named 'NombreCrawlSpider'. The script imports necessary modules, defines an item class 'Pregunta', and implements the spider's logic to parse Stack Overflow questions. The right pane shows the output of the spider, which is a JSON-like structure containing the scraped data for several questions, including their IDs, titles, and descriptions.

```
1 from scrapy.item import Field, Item
2 from scrapy.spiders import Spider
3 from scrapy.selector import Selector
4 from scrapy.loader import ItemLoader
5 from scrapy.crawler import CrawlerProcess
6 class Pregunta(Item):
7     id = Field()
8     pregunta = Field()
9     descripcion = Field()
10
11 class NombreCrawlSpider(Spider):
12     name = "PrimerScrapy"
13     custom_settings = {
14         'USER_AGENT': 'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4399.90 Safari/537.36'
15     }
16     start_urls = ['https://stackoverflow.com/questions']
17
18     def parse(self, response):
19         sel = Selector(response)
20
21         titulo_de_pagina = sel.xpath('//h1/text()').get()
22         print(titulo_de_pagina)
23
24         preguntas = sel.xpath('//div[@id="questions"]//div[@class="question-summary"]')
25
26         i=0
27         for pregunta in preguntas:
28             item = ItemLoader(Pregunta(), pregunta)
29             item.add_xpath('pregunta', './h3/a/text()')
30             item.add_xpath('descripcion', './div[@class="excerpt"]/text()')
31             item.add_value('id', i)
32             i += 1
33             yield item.load_item()
34
35 process = CrawlerProcess({
36     'FEED_FORMAT': 'csv',
37     'FEED_URI': 'resultado.csv'
38 })
39 process.crawl(NombreCrawlSpider)
40 process.start()
```

```
{
  'id': [8],
  'pregunta': ["Can't find lib org.apache.commons.lang3 via BOM file"],
  'descripcion': ["\n\nWhy it doesn't appear 1 on the second '\n\nplaceholder?'\n\n"],
  'id': [9],
  'pregunta': ["Why placeholder doesn't work? html and springboot"],
  'descripcion': ["\n\nI'm using expo's audio API for playing sounds.\n\nI have a component inside a react-native-rnw-bottom-sheet '\n\n(It's a pop from the bottom of the screen), where I have the '\n\naudio logs.\n\n'\n\nWhen I close the ...'\n\n"],
  'id': [10],
  'pregunta': ["Expo stop audio playing after component closed"],
  'descripcion': ["\n\nAfter recent upgrade to big sur, I am unable to '\n\nrun aws cli on my local machine. I have used the cli for a '\n\nwhile without any issue, and installed it globally on my '\n\nmachine. I get the pop-up error ...'\n\n"],
  'id': [11],
  'pregunta': ["aws --version error: dyld: Library not loaded:"],
  'descripcion': ["\n\nBeen trying to figure out a way to sum up each '\n\nand every index in a list of lists in Python. For example a '\n\nlist of list like this:\n\n"]
}
```

-Import.io

Import.io es una plataforma web gratuita que pone en sus manos el poder de la web legible por máquina. Con sus herramientas, puede crear una API o rastrear un sitio web completo en una fracción del tiempo que los métodos tradicionales, sin necesidad de codificación. Está pensada para no programadores que necesitan datos (y programadores que no quieren complicarse la vida). Algunas de las **herramientas** que ofrece esta plataforma son las siguientes:

- API's
- Alertas y notificaciones
- Análisis de tendencias
- Creación de informes y estadísticas
- Creación de paneles de comunicaciones
- Extracción de datos (web scraping)
- Importación y exportación de datos
- Informes personalizables
- Supervisión
- Transformación de datos
- Visualización de datos

Características en cuanto a funcionalidad para Web Scraping:

- Es fácil de usar, incluso para personas sin conocimientos de programación
- Tiene limitantes de uso por usuario
- De paga
- Tiene ejecución paralela rápida
- Está basado en web y funciona en browsers como Chrome, Safari y Firefox
- Puede interactuar con los sitios web y extraer una gran variedad de datos
- Es eficaz si se paga por usarlo

Import.io es una solución rápida y fiable para personas que no saben de programación y principiantes que solo buscan extraer data, aunque se tiene que pagar para obtener sus funciones completas, y como toda herramienta de web scraping, tiene sus limitantes en cuanto a la estructura de las páginas web y los datos que puede extraer, teniendo principales problemas con pop ups, captchas, páginas de desplazamiento infinito y cuadros de búsqueda.

-ParseHub

Puede extraer datos desde cualquier lugar. ParseHub funciona con aplicaciones de una sola página, aplicaciones de varias páginas y casi cualquier otra tecnología web moderna. ParseHub puede manejar JavaScript, AJAX, cookies, sesiones y redireccionamientos. Puede completar formularios fácilmente, recorrer los menús desplegables, iniciar sesión en sitios web, hacer clic en mapas interactivos e incluso lidiar con el desplazamiento infinito.

ParseHub ofrece un plan gratuito para todos y planes empresariales personalizados para la extracción masiva de datos.

Algunas de las opciones de extracción y herramientas que tiene son las siguientes:

- Extracción de datos web
- Extracción de direcciones IP
- Extracción de direcciones de correo electrónico
- Extracción de precios
- Extracción de números de teléfonos móviles
- Extracción de imágenes
- Resumen y publicación de datos
- API's
- Almacenamiento de documentos
- Almacenamiento seguro de datos
- Análisis en tiempo real
- Análisis predictivo
- Autorizaciones basadas en roles
- Captura y transferencia de datos
- Cartografiado de datos

Características en cuanto a funcionalidad para Web Scraping:

- De fácil uso, aunque requiere de unos cuantos conocimientos previos
- Ideal para proyectos pequeños y medianos

- Tiene plan gratuito aunque con sus limitaciones en cuanto a la cantidad de data que se puede extraer
- Eficaz por sí solo aunque no supera a las librerías de python ni al framework de scrapy
- Es una aplicación de escritorio, por lo tanto debe de instalarse para poder usarse
- Al igual que los otros, organiza la información obtenida en tablas

La aplicación de escritorio de ParseHub admite sistemas como Windows, Mac OS X y Linux, o puede usar la extensión del navegador para lograr un scraping instantáneo. No es totalmente gratuito, pero aún puede configurar hasta cinco tareas de scraping de forma gratuita. El plan de suscripción de pago le permite configurar al menos 20 proyectos privados. Puede llegar a ser un poco confuso para principiantes, aunque tampoco es tan difícil como el framework de scrapy, por lo tanto es una opción viable para proyectos pequeños y medianos.

-Scrapinghub

Scrapinghub es un proveedor de soluciones de extracción web de pila completa que ofrece productos de plataforma para transformar sitios web en datos a escala. Facilita y automatiza el proceso de extracción de datos. Sus raspadores web visual funcionan directamente desde su navegador, por lo que no necesita descargar ni instalar nada.

Algunas de las herramientas y opciones que ofrece son las siguientes:

- API's
- Extracción de datos
- Extracción de datos web
- Extracción de direcciones IP
- Extracción de direcciones de correo electrónico
- Extracción de documentos
- Extracción de precios
- Extracción de números de teléfonos móviles
- Extracción de imágenes
- Generación de clientes potenciales
- Integraciones de terceros
- Panel de actividades
- Resumen y publicación de datos
- Supervisión
- Visualización de datos
- Supervisión en tiempo real
- Varias cuentas de usuario

Características en cuanto a funcionalidad para Web Scraping:

- Evita el tener que adquirir proxies
- No tiene suficiente documentación, lo cual limita el uso que se le puede dar
- Es relativamente fácil de usar
- Es rápido

- Ofrece otras herramientas para organización y administración de datos
- Tiene un plan gratuito y un plan de paga, cada uno con sus limitaciones
- Tiene deficiencias en sus soporte
- Es escalable en los planes de paga, lo cual brinda más características y funcionalidades
- Fácil de integrar en conjunto con scrapy
- Interfaz intuitiva

Combinando tecnología y servicios para satisfacer las necesidades de los usuarios y desarrolladores empresariales de todos los niveles, los planes personalizados de extracción de datos bajo demanda de Scrapinghub están respaldados por ingenieros de extracción web cualificados. Además, su producto Crawlera ofrece una gestión inteligente de proxy de IP que funciona enviando una solicitud HTTP a su API para proporcionar rotación automática de proxy y administración de prohibiciones. Crawlera está disponible con planes de precios escalables para adaptarse a la extracción de todos los niveles en función de las solicitudes mensuales y concurrentes. Ofrece detección automática de prohibición, simulación de comportamiento del usuario y ancho de banda ilimitado. El navegador *Splash*, de Scrapinghub, también es compatible con navegadores sin encabezado. Esta solución especialmente diseñada está disponible como proyecto de código abierto o producto SaaS alojado. Este navegador ligero y sin encabezado extrae datos de sitios web a escala, incluido el contenido generado por JavaScript, al tiempo que simula el comportamiento del usuario con secuencias de comandos personalizadas. *Scrapy Cloud* es un proveedor de plataforma basada en la nube que ayuda a administrar y automatizar la implementación de rastreadores web o arañas a través de una interfaz de panel en tiempo real. Las herramientas de control de calidad incorporada y la integración del marco propio Spidermon de Scrapinghub se unen a opciones de programación inteligente y contenedores personalizables. Por último, la API de extracción automática de datos (beta) de Scrapinghub proporciona extracción de datos de comercio electrónico y artículos a escala. La API de Select Product devuelve datos estructurados, como información del producto, de las URL de comercio electrónico. La API de Select Article presenta artículos de noticias y publicaciones de blogs.

➤ *Además de estas, existen numerosas soluciones que cumplen con la tarea de la extracción de datos de la web (web scraping), todas con sus pros y contras, sin embargo, las antes mostradas logran destacar de entre el resto por su popularidad y funcionalidad.*

>Algunas de las otras opciones para realizar web scraping son las siguientes:

→ **Octoparse:** Octoparse es una herramienta robusta de web scraping que también proporciona un servicio de web scraping para propietarios de empresas y empresas. La extracción de datos incluye, entre otros, redes sociales, comercio electrónico, marketing, listado de bienes raíces y muchos otros. A diferencia de otros web scraper que solo scrape contenido con una estructura HTML simple, Octoparse puede manejar sitios web estáticos y dinámicos con AJAX, JavaScript, cookies, etc. Octoparse también proporciona servicios de extracción que pueden ayudarlo a personalizar la tarea de scraping o scrapee los datos por usted.

- **Visual Scraper:** Visual Scraper es otro gran web scraper gratuito. Con su interfaz de apuntar y hacer clic, los usuarios con poca o ninguna habilidad de programación pueden configurar la extracción de datos con sus propias preferencias. Su función en tiempo real le permite probar y ver el resultado de los datos de inmediato. Este programa gratuito está disponible para Windows, puede scrapear datos de hasta 50,000 páginas web. Puede scrapear más de 100,000 páginas web con su Plan Premium.

- **Outwit Hub:** Outwit Hub es una extensión de Firefox, y se puede descargar fácilmente desde la tienda de complementos de Firefox. Una vez instalado y activado, puede scrapear el contenido de los sitios web al instante. Tiene características sobresalientes de "Fast Scrape", que extrae rápidamente datos de una lista de URL que usted ingresa. La extracción de datos de sitios que usan Outwit Hub no exige habilidades de programación. El proceso de scraping es bastante fácil de aprender.

- **Data Scraper (Chrome):** Data Scraper puede extraer datos de tablas y datos de tipo de listado de una sola página web. Su plan gratuito debería satisfacer el scraping más simple con una pequeña cantidad de datos. El plan pagado tiene más funciones, como API y muchos servidores proxy IP anónimos. Puede recuperar un gran volumen de datos en tiempo real más rápido. Puede scrapear hasta 500 páginas por mes, si necesita scrapear más páginas, necesita actualizar a un plan pago.

- **Web Scraper:** Web scraper tiene una extensión de Chrome y una extensión de nube. Para la extensión de Chrome, puede crear un mapa del sitio (plan) sobre cómo se debe navegar un sitio web y qué datos se deben scraping. La extensión de la nube puede scrapear un gran volumen de datos y ejecutar múltiples tareas de scrapear simultáneamente. Puede exportar los datos en CSV o almacenar los datos en Couch DB.

- **Scraper (Chrome):** El Scraper es otro raspador web de pantalla fácil de usar que puede extraer fácilmente datos de una tabla en línea y subir el resultado a Google Docs. Simplemente seleccione un texto en una tabla o lista, haga clic con el botón derecho en el texto seleccionado y elija "Scrape similar" en el menú del navegador. Luego obtendrá los datos y extraerá otro contenido agregando nuevas columnas usando XPath o JQuery. Esta herramienta está destinada a usuarios de nivel intermedio a avanzado que saben cómo escribir XPath.

- **Dexi.io (También conocido como Cloud Scrape):** Dexi.io está destinado a usuarios avanzados que tienen habilidades de programación competentes. Tiene tres tipos de robots para que pueda crear una tarea de scraping: extractor, rastreador y tuberías. Proporciona varias herramientas que le permiten extraer los datos con mayor precisión. Con su característica moderna, podrá abordar los detalles en cualquier sitio web. Para las personas sin habilidades de programación, es posible que deba tomarse un tiempo para acostumbrarse antes de crear un robot de web scraping. El software gratuito proporciona servidores proxy web anónimos para el web scraping. Los datos extraídos se alojarán en los servidores de Dexi.io durante dos semanas antes de archivarlos, o puede exportar directamente los datos extraídos a archivos JSON o CSV. Ofrece servicios pagos para satisfacer sus necesidades de obtener datos en tiempo real.

- **Webhose.io:** Webhose.io le permite obtener datos en tiempo real al extraer fuentes en línea de todo el mundo en varios formatos limpios. Incluso puede scrape información en la web oscura. Este web scraping permite scrapear datos en muchos idiomas diferentes utilizando múltiples filtros y exportar datos raspados en formatos XML, JSON y RSS. El programa gratuito ofrece un plan de suscripción gratuito para que pueda realizar 1000 solicitudes HTTP por mes y planes de suscripción pagados para realizar más solicitudes HTTP por mes para satisfacer sus necesidades de web scraping.

❖ *Lenguaje XPATH*

Xpath (*XML Path Language*) es un lenguaje que permite construir expresiones para recorrer y extraer secciones e información de documentos “XML” como lo es el DOM HTML de los sitios web. Con este lenguaje se pueden ubicar fácilmente etiquetas y elementos del código HTML de la página lo cual es muy útil cuando se realiza web scraping, además de que algunos browsers ya entregan la expresión Xpath de los elementos que se ubican inspeccionando el código. XPath permite buscar y seleccionar teniendo en cuenta la estructura jerárquica del XML. XPath fue creado para su uso en el estándar XSLT, en el que se usa para seleccionar y examinar la estructura del documento de entrada de la transformación. XPath fue definido por el consorcio W3C.

-XML

Todo el procesamiento realizado con un fichero XML está basado en la posibilidad de direccionar o acceder a cada una de las partes que lo componen, de modo que podamos tratar cada uno de los elementos de forma diferenciada.

El tratamiento del fichero XML comienza por la localización del mismo a lo largo del conjunto de documentos existentes en el mundo. Para llevar a cabo esta localización de forma unívoca, se utilizan los URI (Uniform Resource Identifiers), de los cuales los URL (Uniform Resource Locators) son sin duda los más conocidos.

Una vez localizado el documento XML, la forma de seleccionar información dentro de él es mediante el uso de XPath, que es la abreviación de lo que se conoce como XML Path Language. Con XPath podremos seleccionar y hacer referencia a texto, elementos, atributos y cualquier otra información contenida dentro de un fichero XML.

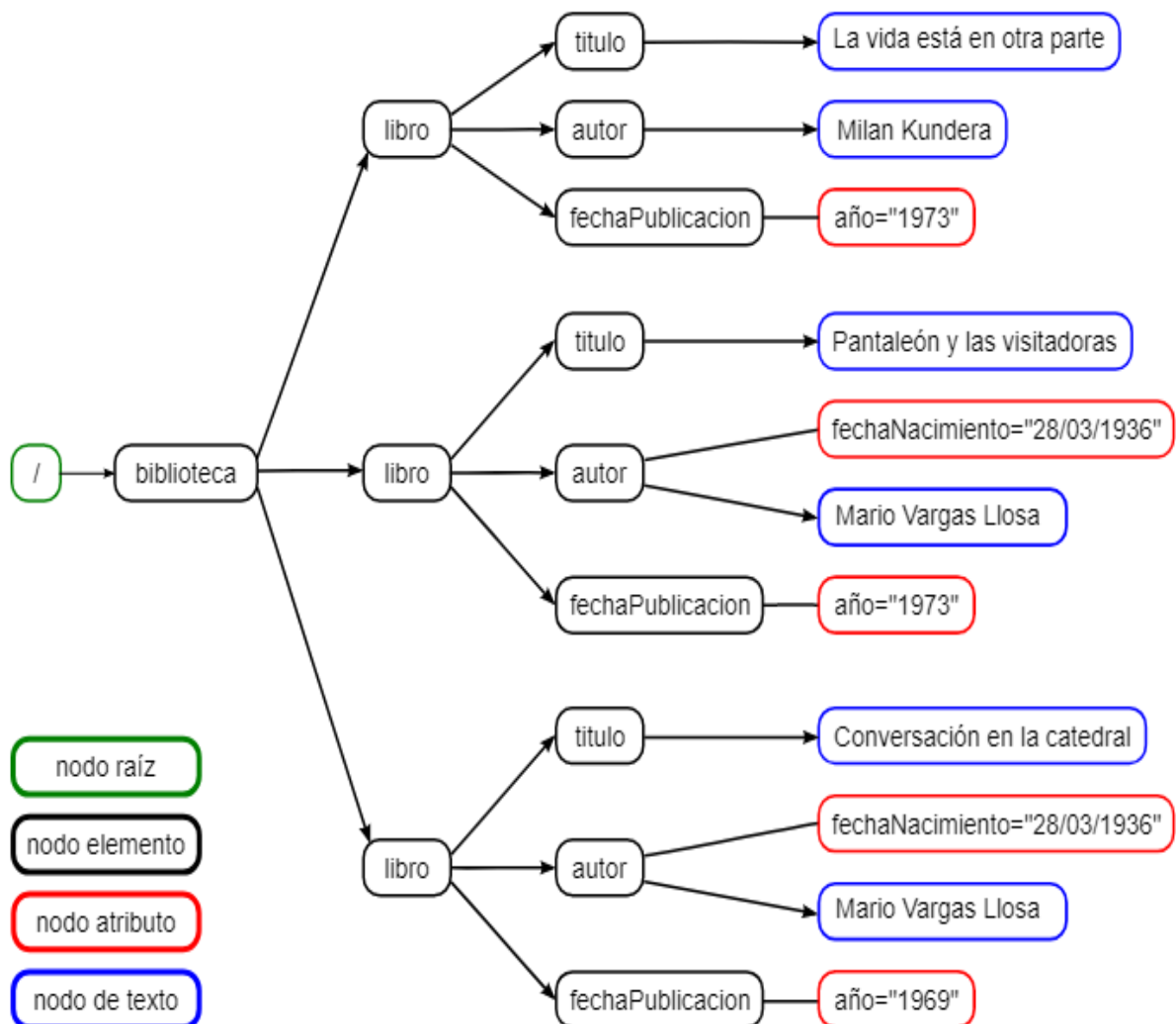
-Modelo de datos de XPATH

Un documento XML es procesado por un analizador (o parser) construyendo un árbol de nodos. Este árbol comienza con un elemento raíz, que se diversifica a lo largo de los elementos que cuelgan de él y acaba en nodos hoja, que contienen solo texto, comentarios, instrucciones de proceso o incluso que están vacíos y sólo tienen atributos.

La forma en que XPath selecciona partes del documento XML se basa precisamente en la representación arbórea que se genera del documento.

Un caso especial de nodo son los nodos atributo. Un nodo puede tener tantos atributos como desee, y para cada uno se le creará un nodo atributo. No obstante, dichos nodos atributo no se consideran como hijos suyos, sino más bien como etiquetas añadidas al nodo elemento.

-Ejemplo de árbol de un documento XML



❖ *Tabla de características de soluciones para Web Scraping*

BeautifulSoup:

ANALIZADOR	CÓDIGO	VENTAJA	DESVENTAJA
Biblioteca estándar	BeautifulSoup(markup, "html.parser")	1.- Biblioteca estándar incorporada de Python 2.- Es rápida la ejecución 3.- Es tolerante a fallas	1.- Sólo está en uso para Python 2.7.3 o 3.2.2 o anteriores.
lxml	BeautifulSoup(markup, "lxml")	1.- Es rápido 2.- Tolerante a fallas	1.- Se necesita instalar la biblioteca de lenguaje C
XML lxml	BeautifulSoup(markup, ["lxml",xml]) BeautifulSoup(markup, "xml")	1.- Es rápido 2.- Es el único analizador que acepta XML	1.- Se necesita instalar la biblioteca de lenguaje C
html5lib	BeautifulSoup(markup, "html5lib")	1.- Tolerante a fallas 2.- Analizador de documentos en un navegador 3.- Genera documentos en HTML5	1.- Es lento 2.- No depende de extensiones externas

Scrapy:

Clases	Descripción	Códigos
Spider	Son clases que definen cómo se realizará el raspado de un sitio y que datos se van a extraer.	<code>start_request()</code> <code>start_urls = ''</code> <code>parse(response)</code> <code>allowed_domains = ''</code> <code>custom_settings = { }</code> <code>from_crawler(crawler,*arg,**kwargs)</code> <code>closed(reason)</code>
LinkExtractor	Es una clase que crea un objeto, que especifica cómo extraer enlaces de la URL	<code>LinkExtractor(allow='', follow= True/False, callback='',)</code>
Selectores	Sirven para seleccionar algunas partes del código HTML raspado, para esto se utilizan expresiones XPATH y CSS	<code>response.xpath()</code> <code>response.css()</code> <code>response.relector</code>
Item	Es una clase que crea un objeto que funciona como contenedor para los datos rastreados.	<code>scrapy.Field()</code> <code>scrapy.Item</code>
Item Loader	Es un cargador de elementos, a donde llegan todos los datos que son desechados por el objeto Ítem.	<code>ItemLoader()</code>

Scrapy Shell	Es una herramienta de línea de comandos que permite probar el analizador sin tener que usar rastreador. Su propósito principal es probar el código de extracción de datos.	scrapy shell

Selenium:

Herramientas	Descripción	Ventajas	Desventajas
Selenium IDE	Es un entorno de pruebas de software para aplicaciones basadas en la web	Fácil para iniciar y realizar pruebas básicas No necesita un aprendizaje previo	Las pruebas pueden llegar a ser lentas No se puede reutilizar nada Solo funciona con Firefox o Chrome Es necesario tener el navegador arrancando
Web Driver	Es un driver que permite soportar los navegadores más usados, como Chrome, Firefox, Opera, Explorer, Opera	Fácil para comenzar a usar las API's Permite organizar el código Abrirá el navegador	El código es dependiente del navegador adecuado. No se puede ejecutar remotamente

Selenium Remote Control	Es un sistema cliente/servidor que permite utilizar el navegador de forma local o en otro ordenador	Compatible con cualquier lenguaje de programación Utiliza navegadores que permiten usar JavaScript	Presenta limitaciones en los requerimientos en navegadores que no permitan JavaScript Puede haber problemas con las opciones de seguridad de los navegadores
Selenium Grid	Es un servidor que permite realizar pruebas en varios navegadores.	Permite utilizar y realizar pruebas desde distintos navegadores, de distintas versiones, incluso con diferentes sistemas operativos. Reduce el tiempo que tarda un paquete de pruebas en completarse	No administra la infraestructura y es posible que no se adapte a las necesidades específicas del cliente.

❖ **Validación de entradas**

Al utilizar python, en ocasiones es necesario validar que las entradas que se solicitan al usuario, en los programas, sean correctas para poder ejecutar el algoritmo correspondiente.

En el caso específico del web scraping hay algunos datos que se deben de validar primeramente para poder proseguir con la extracción de la data.

-URLs

El módulo **urllib.request** de Python sirve para acceder y utilizar recursos de internet identificados por URLs (*Uniform Resource Locators*). Ofrece una interfaz muy simple, a través de la función *urlopen*. Esta función es capaz de acceder a URLs usando una variedad de protocolos diferentes. También ofrece una interfaz un poco más compleja para manejar situaciones comunes - como la autenticación básica, cookies y proxies, entre otros. Estos son proporcionados por los llamados objetos de apertura y gestores. Urllib.request soporta la obtención de recursos identificados por URLs para muchos «esquemas de URL» (identificados por la cadena de texto ubicada antes del ":" en el URL - por ejemplo "ftp" es el esquema de URL de "ftp://python.org/") usando sus protocolos de red asociados (por ejemplo FTP, HTTP). Usando esta librería es posible mandar un request al servidor donde se aloja la información de URL, siendo "200" una respuesta afirmativa que nos indica que la URL es correcta en pocas

palabras. También podemos usar la función “urllib.parse.urlparse” con la cual podemos dividir en partes la URL y validar los campos que queramos, esto si queremos que sea de un dominio en específico, un tipo de protocolo en concreto, puerto, fragmento o sección, etc.

Ejemplos:

```
import urllib
from urllib import request

respuesta=request.urlopen ( "https://es.wikipedia.org/wiki/Python ")
respuesta.code
```

Salida: 200

```
import urllib
from urllib.parse import urlparse

HOST= 'https://www.Web.com:80/sitios;params?query=valor&usuario=user2K19#seccion'

partes=urlparse(HOST)

print(partes.scheme)
print(partes.netloc)
print(partes.hostname)
print(partes.port)
print(partes.path)
print(partes.params)
print(partes.query)
print(partes.fragment)
```

Salida: https

```
www.Web.com:80
www.web.com
80
/sitios
params
query=valor&usuario=user2K19
seccion
```

-Cadenas

En el caso de la validación de cadenas de texto, python nos ofrece varios métodos nativos los cuales nos ayudan a corroborar entradas de distintos tipos. Algunos de los más utilizados son los siguientes:

isalnum() Devuelve *True* si la cadena es alfanumérica, de lo contrario *False*.

Ejemplo:

```
• >cadena_alfanumerica = "El valor es 1000"
• >cadena_alfanumerica.isalnum()
• False
• >cadena_alfanumerica = "awlpftawnju8mke4r5i9cfaw"
• >cadena_alfanumerica.isalnum()
• True
```

isalpha() Devuelve *True* si la cadena es alfabética, de lo contrario *False*.

Ejemplo:

```
• >cadena_alfabetica = "Hola mundo 1000"
• >cadena_alfabetica.isalpha()
• False
• >cadena_alfabetica = "Hola mundo"
• >cadena_alfabetica.isalpha()
• False
• >cadena_alfabetica = "Hola"
• >cadena_alfabetica.isalpha()
• True
```

isdigit() Devuelve *True* si la cadena es numérica, de lo contrario *False*.

Ejemplo:

```
• >cadena_numerica = "12345"
• >cadena_numerica.isdigit()
• True
• >cadena_numerica = "1 2 3 4 5"
• >cadena_numerica.isdigit()
• False
• >cadena_numerica = "1.2"
• >cadena_numerica.isdigit()
• False
```

islower() Devuelve *True* si la cadena contiene solamente minúsculas, de lo contrario *False*.

Ejemplo:

```
• >cadena_minusculas = "Hola mundo"
• >cadena_minusculas.islower()
• False
• >cadena_minusculas = "hola mundo"
• >cadena_minusculas.islower()
• True
```

isupper() Devuelve *True* si la cadena contiene solamente mayúsculas, de lo contrario *False*.

Ejemplo:

```
• >cadena_mayusculas = "Hola mundo"
• >cadena_mayusculas.isupper()
• False
• >cadena_mayusculas = "HOLA MUNDO"
• >cadena_mayusculas.isupper()
• True
```

isspace() Devuelve *True* si la cadena contiene solamente espacios en blanco, de lo contrario *False*.

Ejemplo:

```
• >cadena_espacios = "hola mundo"
• >cadena_espacios.isspace()
• False
• >cadena_espacios = " "
• >cadena_espacios.isspace()
• True
• >cadena_espacios = " "
• >cadena_espacios.isspace()
• True
```

Si se busca una estructura en específico para una entrada, se puede usar simplemente un ciclo `while` que valide que se cumple con un patrón o estructura en común y que no deje avanzar el algoritmo hasta que el usuario introduzca correctamente el texto.

❖ ***Métodos HTTP para envío de datos***

Para vincular sitios web escritos en HTML con códigos en el backend programados en lenguajes como Python o JavaScript, es necesario apoyarse del protocolo de transferencia de hipertexto (http, por sus siglas en inglés) y sus métodos de envío de información, específicamente “GET” y “POST”. Este protocolo de la capa de aplicación del modelo OSI y TCP/IP se encarga de mandar peticiones a los servidores web para poder cargar sitios, páginas web, archivos, etc. en nuestros browsers, por lo tanto es ideal para esta tarea.

HTTP define un conjunto de métodos de petición para indicar la acción que se desea realizar para un recurso determinado. Aunque estos también pueden ser sustantivos, estos métodos de solicitud a veces son llamados HTTP verbs. Cada uno de ellos implementa una semántica diferente, pero algunas características similares son compartidas por un grupo de ellos.

Algunos de estos métodos son los siguientes:

-GET

El método **GET** solicita una representación de un recurso específico. Las peticiones que usan el método **GET** sólo deben recuperar datos.

-HEAD

El método **HEAD** pide una respuesta idéntica a la de una petición **GET**, pero sin el cuerpo de la respuesta.

-POST

El método **POST** se utiliza para enviar una entidad a un recurso en específico, causando a menudo un cambio en el estado o efectos secundarios en el servidor.

-PUT

El modo **PUT** reemplaza todas las representaciones actuales del recurso de destino con la carga útil de la petición.

-DELETE

El método **DELETE** borra un recurso en específico.

-CONNECT

El método **CONNECT** establece un túnel hacia el servidor identificado por el recurso.

-OPTIONS

El método **OPTIONS** es utilizado para describir las opciones de comunicación para el recurso de destino.

-TRACE

El método **TRACE** realiza una prueba de bucle de retorno de mensaje a lo largo de la ruta al recurso de destino.

-PATCH

El método **PATCH** es utilizado para aplicar modificaciones parciales a un recurso.

❖ **Referencias en formato APA**

-Web Scraping - Formas de llevarlo a cabo:

- *Scrapear. Definición y cómo empezar a scrapear webs.* (2020, 10 noviembre). Enlaces en periódicos online | Prensalink. <https://prensalink.com/scrapear/>
- *API: qué es y para qué sirve.* (2019, 23 agosto). Xataka. <https://www.xataka.com/basics/api-que-sirve>
- *Herramientas de testing: introducción a Selenium.* (2020, 28 julio). DIGITAL55. <https://www.digital55.com/desarrollo-tecnologia/herramientas-testing-introduccion-selenium/>
- John Watson Rooney. (2021, 13 enero). *Beautifulsoup vs Selenium vs Scrapy - Which tool for web scraping in 2021?* YouTube. <https://www.youtube.com/watch?v=J82SxHP5SWY>
- Kite. (2020, 29 febrero). *Python Web Scraping - Should I use Selenium, BeautifulSoup or Scrapy? [2020].* YouTube. <https://www.youtube.com/watch?v=zucvHSQsKHA>
- *BeautifulSoup: todo lo que un científico de datos debe saber.* (2020, 26 noviembre). ICHI.PRO. <https://ichi.pro/es/beautifulsoup-todo-lo-que-un-cientifico-de-datos-debe-saber-260333597823834>
- Montoya, S. (2018, 22 febrero). *Extrae información de páginas web con Scrapy.* Gidahatari. <https://gidahatari.com/ih-es/extrae-informacion-de-paginas-web-con-scrapy>
- *Just a moment. . .* (2014, 4 diciembre). Just a Moment. . . <https://es.schoolofdata.org/2014/12/04/la-magia-de-import->

io/?__cf_chl_jschl_tk__=16e364db9a76c67a4f3483c9805552aa2dd604c0-
 1618329848-0-AeZ9U4B9iO-
 hkPpYgCgrENDWtJnHMPNZMiqquWlB2rfA2FwnK3VQu9AvcTBKx5FhQR7BNP
 DvO77ApQpv6myWdAE9waHE8DQBmja58GxDl6SIWSBxojNn1QmSRpcCgHKJI
 mX5e7C8AIFpAHVJ2M_D2WCLkp1acl9KiEo8d9o8HIoNgzTB8XyQVb3K4gPQd4
 jIwgG4W39P6PiGeM3ZE-4pcbIKCmr-
 0XaP9DswpHrAU_iYfdIphWnwnddMKXE4qZYw-
 ZZNdqUdNBSaNR326aOwegfVYq3apziinuoH-
 wKhSjl_c_k0GouYimQIr7p77LvealK-6_BDMkk-KrXU2JDQmktz-
 845I2iO7TarXo4fbR4zQtsmq-
 P5mMMSehAvO_Pq85_2PPl3_RW3fDIcVKwKwSzpxtUfrlwRU1EdzemmqI3-

- Capterra. (2018, 29 marzo). *import.io*.
<https://www.capterra.mx/software/135185/import-io>
- *Comparación entre Octoparse y Import.io: ¿Cuál es el mejor para el web scraping?*
 (2015, 22 agosto). Octoparse. <https://www.octoparse.es/blog/cual-es-el-mejor-para-el-web-scraping>
- Capterra. (2018, 19 julio). *ParseHub*.
<https://www.capterra.mx/software/145965/parsehub>
- *9 Raspadores Web Gratis que No Te Puedes Perder en 2021*. (2019, 13 febrero).
 Octoparse. <https://www.octoparse.es/blog/web-scraping-gratuitos-que-no-te-puedes-perder>
- GetApp. (2018, 19 diciembre). *Scrapinghub*.
<https://www.getapp.com.mx/software/116931/scrapinghub>
- colaboradores de Wikipedia. (2021, 16 enero). *XPath*. Wikipedia, la enciclopedia libre. <https://es.wikipedia.org/wiki/XPath>

-Validación de entradas en JavaScript y Python (URLs y Cadenas de texto):

- *¿Cómo puedo verificar que la URL sea correcta?* (2018, 28 febrero). Stack Overflow en español. <https://es.stackoverflow.com/questions/142273/como-puedo-verificar-que-la-url-sea-correcta#:~:text=Usa%20la%20funci%C3%B3n%20split%20de,acceder%20al%20primer%20texto%20separado>).
- *JavaScript — Tratando de validar la URL usando JavaScript.* (2009, 20 agosto). Swarm. <https://www.it-swarm-es.com/es/javascript/tratando-de-validar-la-url-usando-javascript/967002092/>
- *Código de JavaScript - función para validar si una url es correcta.* (2018, 29 junio). La Web del Programador. <https://www.lawebdelprogramador.com/codigo/JavaScript/2360-funcion-para-validar-si-una-url-es-correcta.html>
- *python — ¿Cómo validar una URL con una expresión regular en Python?* (2009, 6 mayo). Swarm. <https://www.it-swarm-es.com/es/python/como-validar-una-url-con-una-expresion-regular-en-python/957989129/>
- Socratica. (2018, 12 septiembre). *Urllib - GET Requests || Python Tutorial || Learn Python Programming*. YouTube. https://www.youtube.com/watch?v=LosIGgon_KM
- John Ortiz Ordoñez. (2019, 9 julio). *Python 3 - Receta 204: Obtener las Diferentes Partes de una URL con urllib.parse.urlparse*. YouTube. <https://www.youtube.com/watch?v=95aw6KYI7ts>
- Fazt. (2019, 4 enero). *Tu Primer Página Web con Python3*. YouTube. <https://www.youtube.com/watch?v=fxavwHPJ36o>
- *HOWTO - Cómo obtener recursos de Internet con el paquete urllib — documentación de Python - 3.9.4.* (2017, 1 septiembre). HOWTO-Python.

<https://docs.python.org/es/3/howto/urllib2.html#:%7E:text=urllib.,una%20variedad%20de%20protocolos%20diferentes>.

- García, M. (2017, 6 julio). *08. Python: validar entradas (ejemplos)*. Código or Not.
<https://codingornot.com/08-python-validar-entradas-ejemplos>

-Métodos de petición del protocolo HTTP:

- colaboradores de Wikipedia. (2021, 7 abril). *Protocolo de transferencia de hipertexto*.
Wikipedia, la enciclopedia libre.
https://es.wikipedia.org/wiki/Protocolo_de_transferencia_de_hipertexto
- *Métodos de petición HTTP - HTTP / MDN*. (2021, 21 abril). Developer-Mozilla.
<https://developer.mozilla.org/es/docs/Web/HTTP/Methods>