

# PROYECTO TECNOLÓGICO INTEGRADOR

## Profesor:

- Kevin Arce Kessler

## Equipo:

- Duarte, Carolina Alejandra
- Ferreyra, Santiago Nicolás
- Martins, Cesar Lucas
- Trejo, Ernesto Gaston
- Urcola, Ma. Victoria



---

Cohorte 2023

# Scraping

El web scraping, o raspado web, es una técnica que permite adquirir información estructurada desde un sitio web. Esto se logra mediante la simulación de la navegación humana, con el propósito de realizar análisis de datos a gran escala (Big Data) o automatizar procesos web.

A nivel mundial, diversas industrias y profesionales hacen uso del scraping, incluyendo:

- Empresas: Lo utilizan para recopilar datos competitivos, analizar el mercado, rastrear precios de productos y obtener información sobre clientes y usuarios.
- Investigadores: Esta técnica es fundamental para obtener datos relevantes en investigaciones académicas y científicas en diversas disciplinas.
- Periodistas: Los periodistas la emplean para acceder a información pública en línea y realizar investigaciones periodísticas.
- Desarrolladores: Usan el scraping para crear aplicaciones web que agregan información de múltiples fuentes.
- Gobierno: Algunos organismos gubernamentales recurren al scraping para recopilar datos públicos y mejorar la toma de decisiones.

El web scraping se utiliza ampliamente para obtener grandes volúmenes de información de forma automatizada. Sus principales aplicaciones y ventajas incluyen:

- Análisis Competitivo: Las empresas pueden analizar a sus competidores, ajustar estrategias de marketing, mejorar el SEO y tomar decisiones comerciales fundamentadas.



# Scraping

- Automatización de Tareas: Permite la automatización de tareas como reservas en línea, agregación de datos bancarios, procesos de pago y verificación de servicios web.
- Optimización de Marketing y SEO: Facilita la comprensión del mercado y mejora los recursos dedicados al marketing y al SEO.
- Toma de Decisiones Basada en Datos: Proporciona datos actualizados y relevantes para tomar decisiones informadas.

La legalidad del scraping web depende de factores como la disponibilidad pública de los datos y la protección por inicio de sesión. En Argentina, la Ley de Protección de Datos Personales (Ley 25.326) regula el scraping de datos personales, requiriendo consentimiento adecuado. También se deben respetar leyes de derechos de autor y términos y condiciones de los sitios web.

## Lenguajes de Programación y Librerías Relevantes:

- Python: Es altamente recomendado debido a su facilidad de uso. Librerías populares incluyen BeautifulSoup para extracción de datos personalizable y Scrapy para descargas masivas.
- JavaScript: Útil para sitios web con interfaces dinámicas, aunque menos adecuado para extracciones repetitivas.
- Node.js: Ideal para sitios avanzados con contenido dinámico.
- C++: Aunque eficiente, se evita en negocios debido a la complejidad.
- PHP: Efectivo para extraer datos y programar tareas en múltiples sitios web.



# Scraping

## Ejemplos Prácticos de Aplicación

- Comparación de Precios en Comercio Electrónico: Monitorizar y comparar precios de productos en diferentes tiendas en línea.
- Análisis de Noticias Financieras: Recopilar noticias y comentarios sobre eventos económicos para la toma de decisiones financieras.
- Gestión de Reputación en Redes Sociales: Recopilar comentarios y reseñas de usuarios en redes sociales para evaluar la percepción de la marca en línea.

## Ventajas de Web Crawling y Data Scraping:

- Reducción de la carga de trabajo y costos de personal.
- Aumento significativo de la velocidad de los procesos.
- Minimización del error humano en la recopilación de datos.
- Capacidad para manejar grandes volúmenes de información.
- Obtención de datos en formatos procesables y listos para su análisis.

## Bibliografía

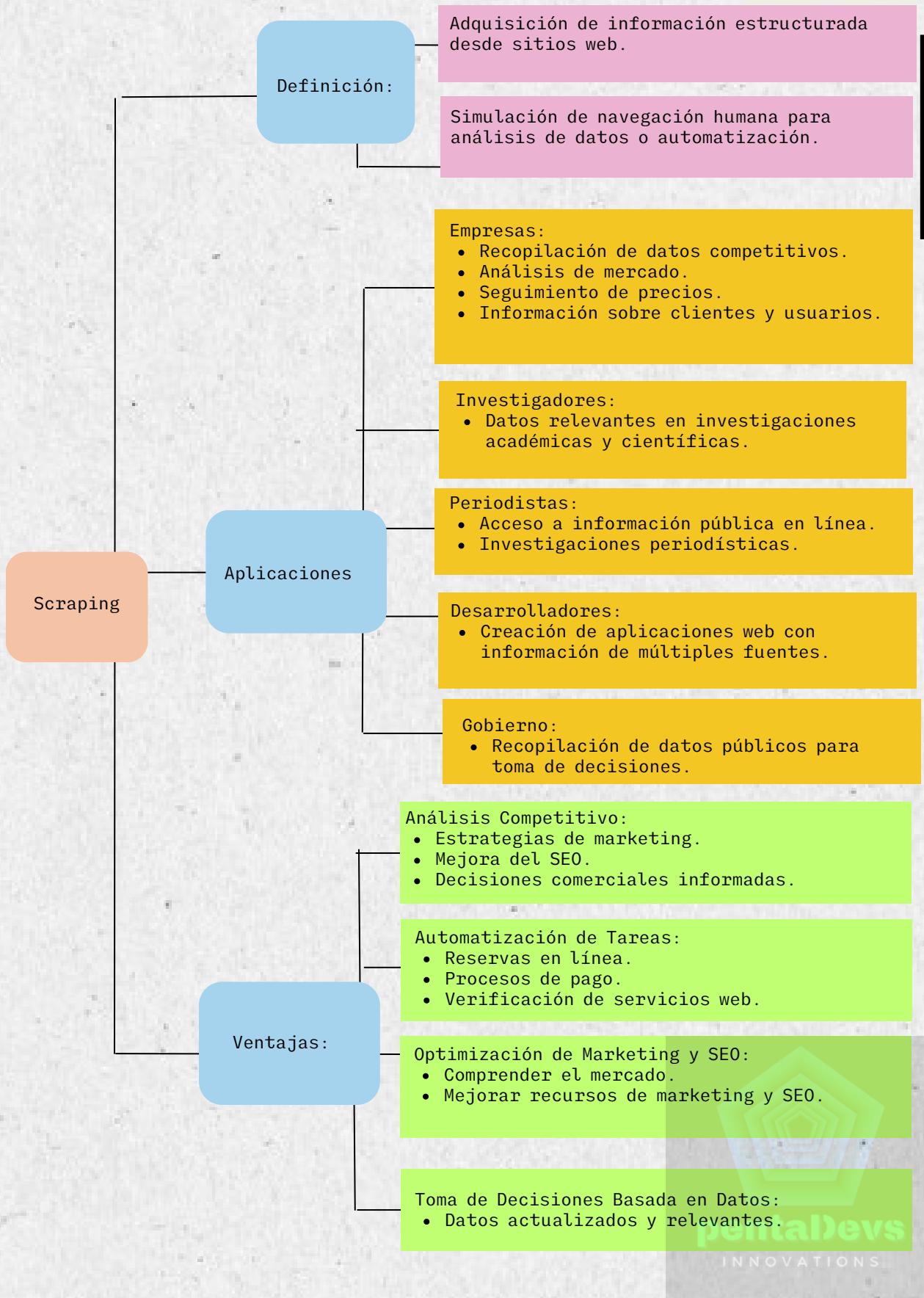
<https://www.amnislabs.com/web-scraping>

<https://datstrats.com/blog/que-es-scraping-usos-aplicaciones/>

<https://www.iubenda.com/es/help/111939-es-legal-el-web-scraping-lo-que-hay-que-saber>

<https://www.areacucuta.com/que-lenguajes-son-empleados-en-web-scraping/>

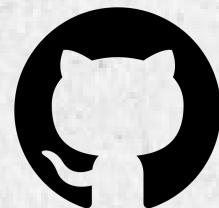




# Proyecto Scraping

## Proyecto scraping : Stack Overflow

repositorio ->



Trello ->



Ctrl + C

Ctrl + V



stack overflow



## Scraping V1.0.0 Página Seleccionada: stackoverflow.com

### • Desarrollo

Para comenzar con el proyecto decidimos repasar la información sobre Scraping para entender más su propósito, investigando nos encontramos con tres posibles librerías a través de las cuales podríamos extraer datos de la página asignada , las librerías son:  
Beautiful Soup, Selenium y Scrapy

Analizamos por separado cada una , sus ventajas y desventajas y entre otros puntos entendimos que principalmente debemos asignar la librería adecuada contemplando la estructura de la página a la que le realizaremos Scraping.

Primero investigamos e instalamos scrapy pero nos dimos con que resulta complejo a la hora de extraer los datos. En cuanto al armado de la araña para poder hacer el raspado. Constantemente daba error y no lográbamos dar con la sintaxis debido a las actualizaciones; ya que contamos con poco tiempo para el estudio de esta librería que es compleja, decidimos investigar otra librería como Beautifulsoup que sirva para nuestro proyecto.



**Beautifulsoup**



# Proyecto Scraping

Beautiful Soup:

- Ventajas

- Facilidad de Uso: Sintaxis simple y amigable para principiantes.
- Compatibilidad con HTML y XML: Excelente para analizar documentos en estos formatos.
- Integración con Análisis de Datos: Se puede combinar fácilmente con otras bibliotecas.

- Desventajas

- Problemas con JavaScript: No es adecuada para páginas web altamente dependientes de JavaScript.
- Menos Eficiente para Grandes Conjuntos de Datos: Puede ser menos eficiente que Scrapy en proyectos de gran escala.
- Sensibilidad a Cambios en la Página: Requiere ajustes si la estructura de la página cambia frecuentemente.

- Investigación y limitaciones

Una vez elegida la librería a utilizar comenzamos a profundizar en las políticas de uso de la página, en sus limitaciones , para poder entender mejor ingresamos a sus archivos robots.txt de la siguiente manera, en su link de pagina en el final colocamos /robots.txt, <https://es.stackoverflow.com/robots.txt>, de esta manera nos arrojó todos su archivos y las limitaciones que tienen los bots de scraping, analizamos cada uno de ellos para comprender mejor las restricciones.

# Proyecto Scraping



The screenshot shows a dark-themed web browser window. At the top, the address bar displays "stackoverflow.com/robots.t..." and "stackoverflow.com". Below the address bar, the main content area shows the text of the Stack Overflow robots.txt file. The file contains numerous "Disallow" entries for various URLs across different sections like posts, users, questions, answers, and jobs. It also includes "Allow" and "User-agent" directives. In the bottom right corner of the browser window, there is a watermark for "pentadevs INNOVATIONS" featuring a stylized blue pentagon logo.

```
User-Agent: *
Disallow: /posts/
Disallow: /posts?
Disallow: /amzn/click/
Disallow: /questions/ask/
Disallow: /questions/ask?
Disallow: /search/
Disallow: /search?
Disallow: /feeds/
Disallow: /feeds?
Disallow: /users/login/
Disallow: /users/login?
Disallow: /users/logout/
Disallow: /users/logout?
Disallow: /users/filter/
Disallow: /users/filter?
Disallow: /users/signup/
Disallow: /users/signup/
Disallow: /users/signup?
Disallow: /users/authenticate/
Disallow: /users/authenticate?
Disallow: /users/oauth/*
Disallow: /users/flag-summary/
Disallow: /users/flair/
Disallow: /users/flair?
Disallow: /users/activity/
Disallow: /users/activity?
Disallow: /users/stats/
Disallow: /users/*?tab=accounts
Disallow: /users/*?tab=activity
Disallow: /users/rep/show
Disallow: /users/rep/show?
Disallow: /users/prediction-data
Disallow: /users/prediction-data/
Disallow: /users/prediction-data?
Disallow: /unanswered/
Disallow: /unanswered?
Disallow: /u/
Disallow: /messages/
Disallow: /api/*
Disallow: /review/*
Disallow: /*/ivc/*
Disallow: /*?lastactivity
Disallow: /users/login/global/request/
Disallow: /users/login/global/request?
Disallow: /questions/*answertab=
Disallow: /questions/tagged/*
Disallow: /questions/tagged/%20*
Disallow: /questions//answer/submit
Disallow: /tags/*
Disallow: /tags/%20*
Disallow: /suggested-edits/
Disallow: /suggested-edits?
Disallow: /ajax/
Disallow: /plugins/
Disallow: /error
Disallow: /gps/*
Disallow: /10m
Disallow: /jobs/cv/sign-up-and-create
Disallow: /jobs/n/*
Disallow: /jobs/a/*
Disallow: /jobs/apply/*
Disallow: /jobs/companies/n/*
Disallow: /jobs/companies/a/*
Disallow: /jobs/*more-jobs-count
Disallow: /jobs/email-job
Disallow: /emails/*
Disallow: /_/*
Allow: /
User-agent: Yahoo Pipes 1.0
Disallow: /
User-agent: 008
Disallow: /
User-agent: voltron
Disallow: /
User-agent: Bytespider
Disallow: /
User-agent: GPTBot
Disallow: /
User-agent: Googlebot-Image
Disallow: /*/ivc/*
Disallow: /users/flair/
Disallow: /jobs/n/*
Disallow: /jobs/a/*
Disallow: /jobs/companies/n/*
Disallow: /jobs/companies/a/*
Sitemap: https://stackoverflow.com/sitemap.xml
```

# Proyecto Scraping

## • Código + Requests

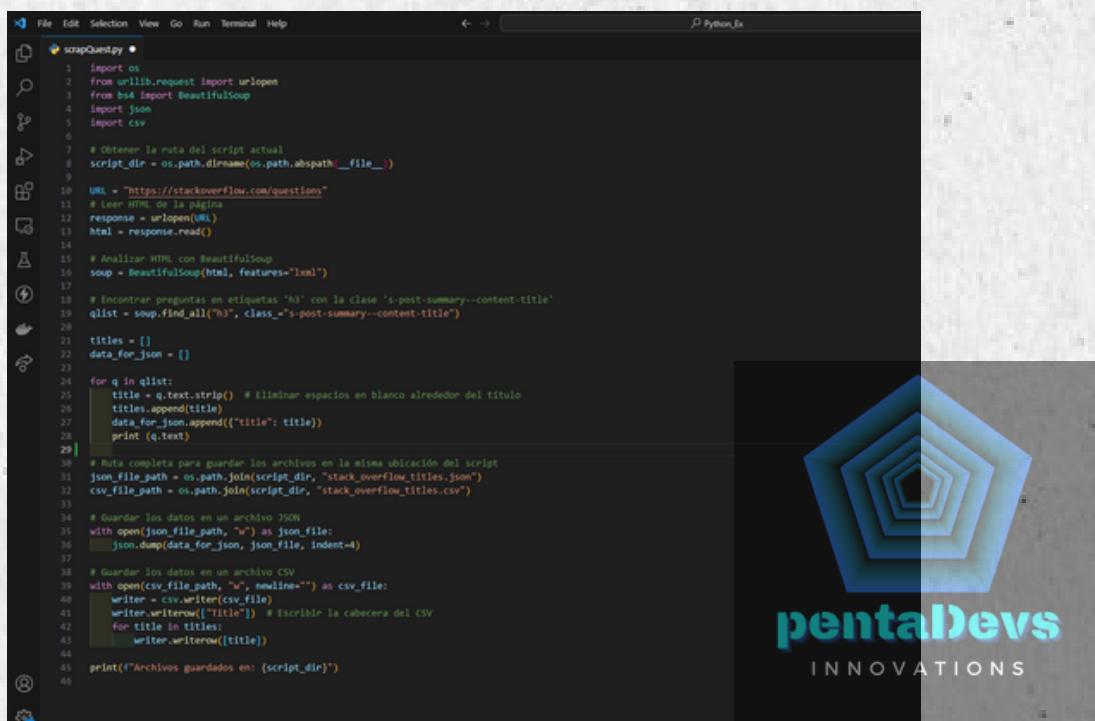
### Descubrimiento de sección

Luego de revisar las restricciones, nos dedicamos a generar el programa en python utilizando la librería y herramientas elegidas en Visual Studio Code para poder comenzar a scrapear Stack Overflow

En un principio intentamos extraer preguntas y respuestas privadas, pero sólo obtuvimos las columnas vacías.

Luego probamos realizar Scraping en la sección de bolsa de empleo , pero al revisar nos dimos cuenta que la sección no estaba operando por lo cual no tendríamos datos para extraer

En una tercera instancia decidimos probar sólo en preguntas públicas y funcionó, obtuvimos la extracción de las primeras 50 preguntas por limitación de la página.



```
File Edit Selection View Go Run Terminal Help
scrapQuestpy •
1 import os
2 from urllib.request import urlopen
3 from bs4 import BeautifulSoup
4 import json
5 import csv
6
7 # Obtener la ruta del script actual
8 script_dir = os.path.dirname(os.path.abspath(__file__))
9
10 URL = "https://stackoverflow.com/questions"
11 # Leer HTML de la página
12 response = urlopen(URL)
13 html = response.read()
14
15 # Analizar HTML con BeautifulSoup
16 soup = BeautifulSoup(html, features="lxml")
17
18 # Encontrar preguntas en etiquetas 'h3' con la clase 's-post-summary--content-title'
19 qlist = soup.find_all("h3", class_="s-post-summary--content-title")
20
21 titles = []
22 data_for_json = []
23
24 for q in qlist:
25     title = q.text.strip() # Eliminar espacios en blanco alrededor del título
26     titles.append(title)
27     data_for_json.append({"title": title})
28     print(q.text)
29
30 # Ruta completa para guardar los archivos en la misma ubicación del script
31 json_file_path = os.path.join(script_dir, "stack_overflow_titles.json")
32 csv_file_path = os.path.join(script_dir, "stack_overflow_titles.csv")
33
34 # Guardar los datos en un archivo JSON
35 with open(json_file_path, "w") as json_file:
36     json.dump(data_for_json, json_file, indent=4)
37
38 # Guardar los datos en un archivo CSV
39 with open(csv_file_path, "w", newline="") as csv_file:
40     writer = csv.writer(csv_file)
41     writer.writerow(["title"]) # Escribir la cabecera del CSV
42     for title in titles:
43         writer.writerow([title])
44
45 print(f"Archivos guardados en: {script_dir}")
```

# Proyecto Scraping

```
Go Run Terminal Help scrapQuest.py ⓘ stack_overflow_titles (2).json X ⌂ Python_Ex
1 [
2   {
3     "title": "Github not pushing changes to my github.io site"
4   },
5   {
6     "title": "What is type annotation for \"any callable but a class\" in Python using Mypy?"
7   },
8   {
9     "title": "NS_BINDING_ABORTED - why browser unexpectedly transform POST request to GET request"
10  },
11  {
12    "title": "experiencing an error while connecting to server"
13  },
14  {
15    "title": "Error when trying to connect a JAI AD_132GE#1 camera with Python, using the Harvester library"
16  },
17  {
18    "title": "Missing Tailwind Style on OME file whr?? ( EDS / Express / Vanilla JS )"
19  },
20  {
21    "title": "Draggable element does not apply border radius correctly in Chrome"
22  },
23  {
24    "title": "Why won't this unordered_map emplace?"
25  },
26  {
27    "title": "Python TypeError: unsupported operand type(s) for :: 'str' and 'str' error"
28  },
29  {
30    "title": "How is copy function implemented for slices in go?"
31  },
32  {
33    "title": "fetch() with json object, print out key in an object in an array in an object"
34  },
35  {
36    "title": "How to validate json with some conditions"
37  },
38  {
39    "title": "Powershell / Multiple Checkboxes, one event handler"
40  },
41  {
42    "title": "The tab key does not work in Visual Studio code"
43  },
44  {
45    "title": "Convert generic object T into a list<T>"
46  },
47  {
48    "title": "Ordering of GitHub operations when matrix is involved"
49  },
```



# Proyecto Scraping

## • Error 403

Luego de haber estructurado bien el código y haber podido extraer las primeras 50 preguntas públicas, intentamos repetir varias veces las reques para obtener las siguientes 50, pero nos devolvió en la terminal un error, el error 403.

Al investigar de qué se trataba resultó que una de las posibilidades es de que la página haya detectado que la solicitud provenía de un script automatizado y no de un navegador web, dejándonos sin posibilidad de utilizar nuestra librería y extracción de datos, probamos otras secciones y obtuvimos el mismo resultados, el error 403.

```
Traceback (most recent call last):
  File "c:\Users\Ferreyyra Santiago\Documents\GitHub\python\scraping\scraping.py", line 12, in <module>
    response = urlopen(URL)
               ^^^^^^^^^^
  File "C:\Users\Ferreyyra Santiago\AppData\Local\Programs\Python\Python311\Lib\urllib\request.py", line 216, in urlopen
    return opener.open(url, data, timeout)
           ^^^^^^^^^^^^^^^^^^
  File "C:\Users\Ferreyyra Santiago\AppData\Local\Programs\Python\Python311\Lib\urllib\request.py", line 525, in open
    response = meth(req, response)
               ^^^^^^^^^^^^^^
  File "C:\Users\Ferreyyra Santiago\AppData\Local\Programs\Python\Python311\Lib\urllib\request.py", line 634, in http_response
    response = self.parent.error(
               ^^^^^^^^^^
  File "C:\Users\Ferreyyra Santiago\AppData\Local\Programs\Python\Python311\Lib\urllib\request.py", line 563, in error
    return self._call_chain(*args)
           ^^^^^^^^^^
  File "C:\Users\Ferreyyra Santiago\AppData\Local\Programs\Python\Python311\Lib\urllib\request.py", line 496, in _call_chain
    result = func(*args)
           ^^^^^^^^^^
  File "C:\Users\Ferreyyra Santiago\AppData\Local\Programs\Python\Python311\Lib\urllib\request.py", line 643, in http_error_default
    raise HTTPError(req.full_url, code, msg, hdrs, fp)
urllib.error.HTTPError: HTTP Error 403: Forbidden
PS C:\Users\Ferreyyra Santiago\Documents\GitHub\python\scraping>
```



## 🚫 ¿Qué es el error 403 Prohibido y cómo arreglarlo?

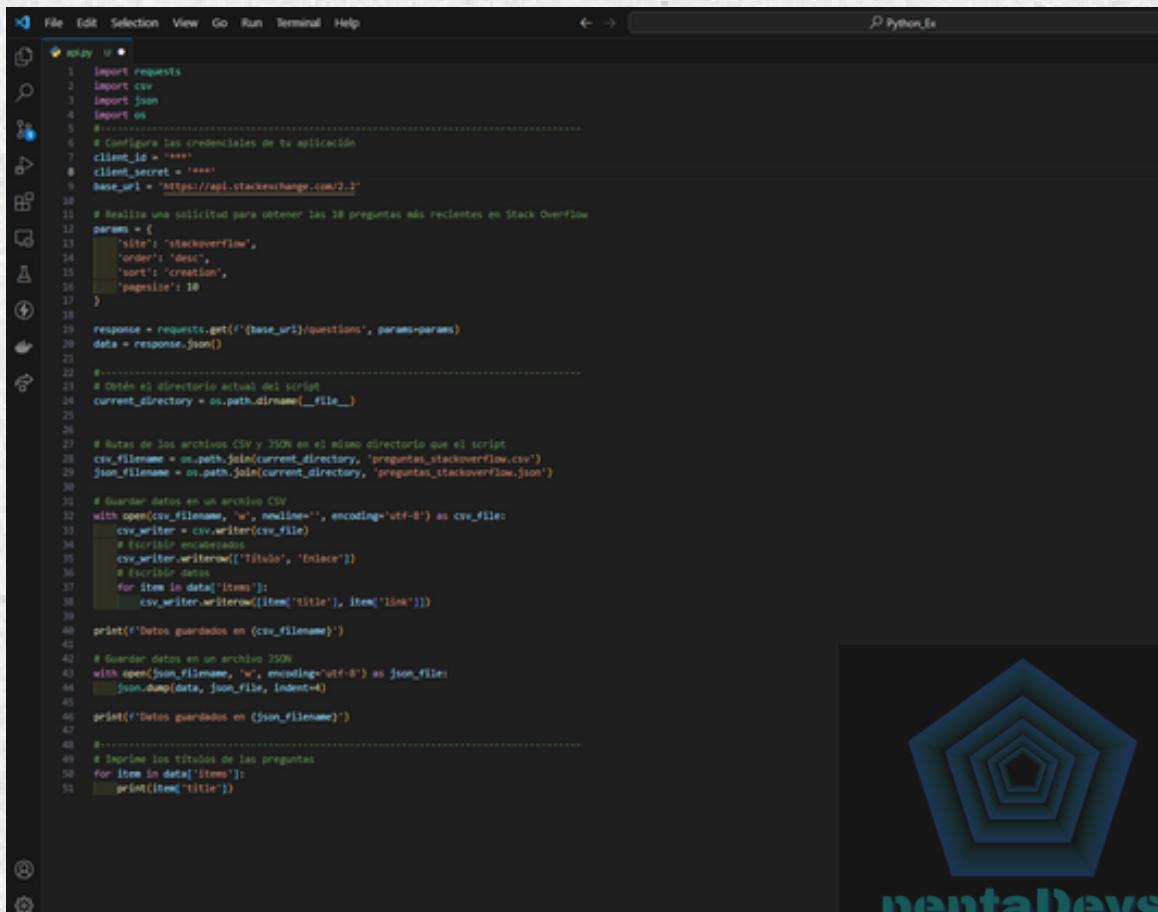
### ¿Qué es el error 403 Prohibido?

Cuando aparece el error "403 Prohibido - No tiene permiso para acceder a / en este servidor", se trata de un código de estado HTTP que indica que el servidor web ha recibido la solicitud de acceso, pero la rechaza y niega el permiso para acceder a la página solicitada.



- API StackExchange y limitaciones

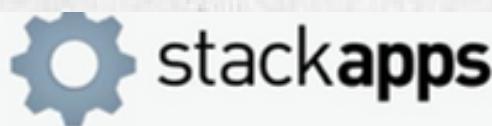
Al encontrarnos limitados para extraer datos de la página asignada comenzamos a investigar otras alternativas y nos encontramos con que Stack Overflow posee una API a través de la cual te permite realizar de forma limitada y organizada sus datos por lo que decidimos ingresar a la API llamada Stack Exchange y loguearnos , te solicita crear una aplicación asignarle un nombre una descripción y al crearla te asigna un ID y una contraseña , la cual modificando el código podes ingresarlas y te permite extraer datos de la mayoría de sus secciones, pero al utilizar esta API nos restringe el uso de la librería Beautiful Soup.



```
File Edit Selection View Go Run Terminal Help
apipy v ●
1 import requests
2 import csv
3 import json
4 import os
5 #-----
6 # Configura las credenciales de tu aplicación
7 client_id = "****"
8 client_secret = "****"
9 base_url = "https://api.stackexchange.com/2.1"
10
11 # Realiza una solicitud para obtener las 10 preguntas más recientes en Stack Overflow
12 params = {
13     'site': 'stackoverflow',
14     'order': 'desc',
15     'sort': 'creation',
16     'page-size': 10
17 }
18
19 response = requests.get(f'{base_url}/questions', params=params)
20 data = response.json()
21
22 # Obtenemos el directorio actual del script
23 current_directory = os.path.dirname(__file__)
24
25
26 # Rutas de los archivos CSV y JSON en el mismo directorio que el script
27 csv_filename = os.path.join(current_directory, 'preguntas_stackoverflow.csv')
28 json_filename = os.path.join(current_directory, 'preguntas_stackoverflow.json')
29
30 # Guardar datos en un archivo CSV
31 with open(csv_filename, 'w', newline='', encoding='utf-8') as csv_file:
32     csv_writer = csv.writer(csv_file)
33     # Encabezados
34     csv_writer.writerow(['Título', 'Enlace'])
35     # Encabezados
36     for item in data['items']:
37         csv_writer.writerow([item['title'], item['link']])
38
39 print(f'Datos guardados en {csv_filename}')
40
41 # Guardar datos en un archivo JSON
42 with open(json_filename, 'w', encoding='utf-8') as json_file:
43     json.dump(data, json_file, indent=4)
44
45 print(f'Datos guardados en {json_filename}')
46
47 #-----
48 # Imprime los títulos de las preguntas
49 for item in data['items']:
50     print(item['title'])
```



# Proyecto Scraping



ISPC

Client Id

27327



This Id identifies your application to the Stack Exchange API. Your application client id is **not** secret, and may be safely embeded in distributed binaries.

Pass this as `client_id` in our OAuth 2.0 flow.

Client Secret ([reset](#))

cSMFZIp1UHJZuHwCM)i)CQ((



Pass this as `client_secret` in our OAuth 2.0 flow if your app uses the explicit path.

This **must be** kept secret. Do not embed it in client side code or binaries you intend to distribute. If you need client side authentication, use the implicit OAuth 2.0 flow.

Key

BO)8kc5a6D)QMW3Cy6sC8Q((

Pass this as `key` when making requests against the Stack Exchange API to receive a [higher request quota](#).

This is not considered a secret, and may be safely embed in client side code or distributed binaries.

Description

Ejercicios sobre SCRAPING

This **text-only** blurb will be shown to users during authentication.

OAuth Domain

ISPC.com

Whenever a redirect occurs during an [authentication sessions](#) (as specified by `redirect_uri` ) it must reside under this domain.



# Proyecto Scraping

## Ejercicios sobre SCRAPING

This **text-only** blurb will be shown to users during authentication.

### OAuth Domain

ISPC.com

Whenever a redirect occurs during an **authentication sessions** (as specified by `redirect_uri`) it must reside under this domain.

For the purposes of redirection, subdomains are considered to be under their parent domain. Registering `example.com` would allow a `redirect_uri` of `foo.example.com` for example.

### Application Website

ISPC.com

A link to this website will accompany most displays of your application on the Stack Exchange network.

### Application Icon

Not Set

This image will accompany most displays of your application on the Stack Exchange network.

### Stack Apps Post

Not Set

When you've published your application, it should be listed on Stack Apps with the `app` or `script` tags.

### Client Side Flow Is Enabled

An application can either be configured for client side or server side authentication flows.

Changing to one will disable the other flow.

### Desktop OAuth Redirect Uri Is Enabled

Applications that have the client side flow enabled can use `https://stackexchange.com/oauth/login_success` as their `redirect_uri` by default.

This is provided so non-web clients can participate in OAuth 2.0 without requiring a full fledged web server. Applications that do not need this behavior can disable it.



# Proyecto Scraping

---

## • Conclusión

Debido a que la página Stack Overflow posee una API para realizar Scraping dentro de la misma, nos encontramos limitados para utilizar nuestra librería propuesta, Beautiful Soup, para extraer datos.

Razón por la cual deberemos realizar el scrapeo sobre otra página.



# Proyecto Scraping



## Proyecto Scraping: La Voz del Interior

repositorio ->



Trello ->



# Proyecto Scraping

## Scraping V2.0.0 Página Seleccionada: lavoz.com

### - Desarrollo

Comenzamos nuestro trabajo en el nuevo sitio web asignado y rápidamente investigamos las limitaciones del sitio. Decidimos en qué sección y tipo de información íbamos a trabajar y establecimos un alcance , utilizamos la librería Beautiful Soup para extraer los datos que necesitábamos, almacenándolos en formato JSON y CSV.

Sin embargo, lo que creíamos que sería una página más bien estática , se transformó en dinámica, así nos dimos cuenta de la limitación de nuestra librería y lo tedioso que resultaba tener que acceder al sitio web, navegar por cada sección y título para extraer los datos manualmente. Después de debatir con el equipo, decidimos buscar una solución más eficiente.

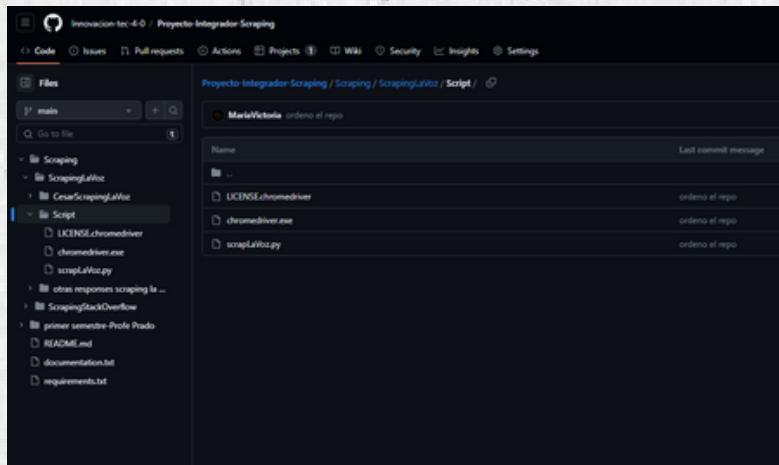
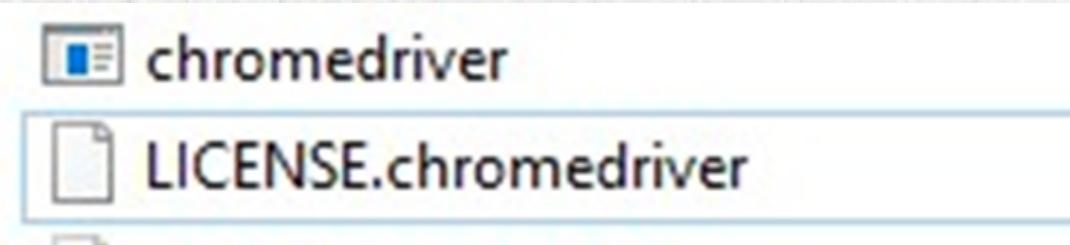
De esta forma hicimos uso de la librería Selenium, diseñada específicamente para la automatización de navegadores web. Con esta librería combinada con Beautiful Soup, automatizamos el proceso de acceso al sitio web, navegación, ingreso a contenidos, y clics.



# Proyecto Scraping

Al instalarla, enfrentamos el problema de encontrar un controlador compatible con nuestros navegadores. Algunos miembros del equipo tenían versiones diferentes de Chrome, lo que complicó la búsqueda del controlador adecuado y la automatización para todos.

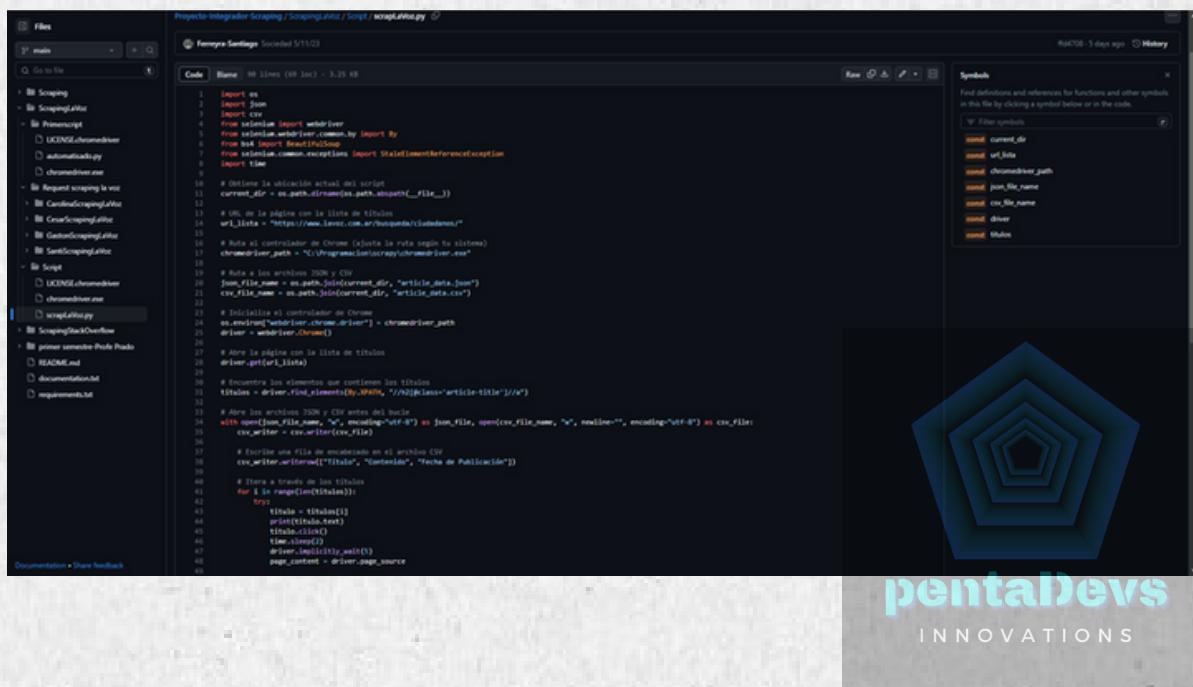
Información de Chrome



# Proyecto Scraping

Finalmente por recomendación del profesor Kevin establecimos un límite de consultas más corto, ya que un gran número de peticiones estaba sobrecargando el sitio web. Al limitar a 10 consultas, pudimos extraer títulos, contenido y fechas. Sin embargo, nos encontramos con un nuevo desafío: la aparición de anuncios publicitarios que interrumpen el programa.

En el proyecto en general nos dimos cuenta que hay muchos factores que pueden interferir en la extracción de datos y requiere de muchos filtros previos para poder garantizar las buenas prácticas, por ejemplo, la estructura del código y sus peticiones en forma gradual hasta encontrar los límites o sea la relación entre los bots de extracción y los bots que limitan el acceso para poder obtener mejores resultados, la posible combinación de librerías, el tiempo entre peticiones, realizar y revisar las actualizaciones en los navegadores con periodicidad y tener un alcance más flexible o general en un principio para poder llegar a un objetivo de una forma menos invasiva.



The screenshot shows a code editor with a Python script named `scrappingnews.py`. The script uses Selenium to open a URL, find elements containing article titles, and then extract the title and publication date into a CSV file. The code includes imports for `selenium`, `BeautifulSoup`, and `urllib`. It defines variables for the driver path, CSV file names, and the URL to scrape. The script then loops through the titles, opening each one in the browser and extracting the title and date, which are then written to a CSV file.

```
import os
import json
from selenium import webdriver
from selenium.webdriver.common.by import By
from bs4 import BeautifulSoup
from selenium.common.exceptions import StaleElementReferenceException
import time

# Define la ubicación actual del script
current_dir = os.path.dirname(__file__)
url = "https://www.lavanguardia.com/periodicos/ciudadano/"

# Ruta al controlador de Chrome (abre la ruta según tu sistema)
chromedriver_path = "C:/Program Files/Google/Chrome/chromedriver.exe"

# Ruta a los archivos JSON y CSV
json_file_name = os.path.join(current_dir, "article_data.json")
csv_file_name = os.path.join(current_dir, "article_data.csv")

# Inicializa el controlador de Chrome
driver = webdriver.Chrome(chromedriver_path)

# Abre la página con la lista de títulos
driver.get(url)

# Encuentra los elementos que contienen los títulos
títulos = driver.find_elements(By.XPATH, '//a[@class="article-title"]')

# Abre los archivos JSON y CSV antes del bucle
with open(json_file_name, "w", encoding="utf-8") as json_file, open(csv_file_name, "w", newline="", encoding="utf-8") as csv_file:
    json_file.write("[]")
    csv_file.write("title,fecha_de_publicacion\n")

# Escribir una fila en el archivo JSON
for título in títulos:
    try:
        título.click()
        time.sleep(2)
        driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
        page_content = driver.page_source
        # Escribir una fila en el archivo CSV
        csv_file.write(f'{título.text},{page_content}\n')
    except StaleElementReferenceException:
        continue
```

# Proyecto Scraping

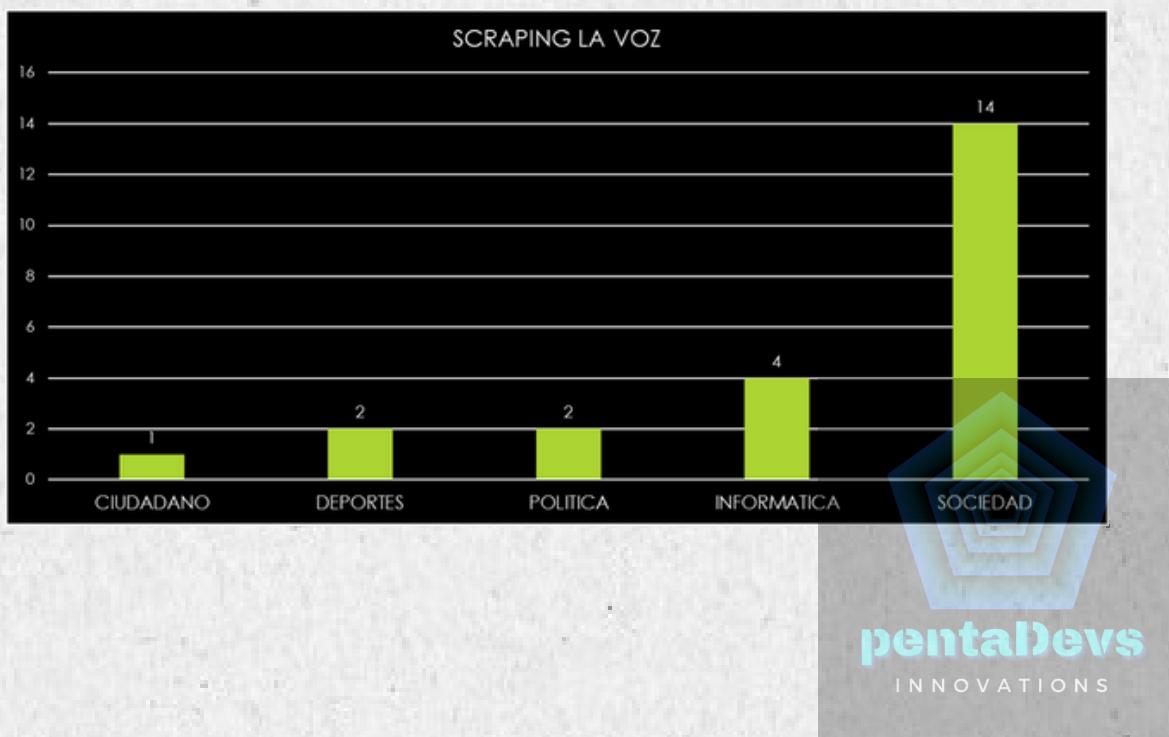
Así entendemos la importancia de la técnica Scraping y su enorme capacidad de usos

En nuestras extracciones pudimos recabar información sobre tecnología e informática pudiendo utilizarla para determinar el auge de la misma en los últimos meses en comparación con el principio de año,

También pudimos extraer información sobre deportes con posibilidad de obtener los resultados del campeonato vigente,

Pudimos obtener información sobre la sociedad , información que se puede utilizar para entender el comportamiento social , analizando el impacto que pueden tener las noticias en las personas que las leen, evaluando descubrimos que la mayoría de las noticias no son de tipo positivas sino con un tono pesimista , característica puntual por ejemplo para el inicio de un estudio.

De esta manera queremos reflejar el enorme abanico de posibilidades para realizar análisis de todo tipo en cualquier tipo de datos que podamos extraer . que nos permite esta gran metodología llamada SCRAPING.



# Proyecto Scraping

---

