

PROYECTO TECNOLÓGICO INTEGRADOR

Integrantes:

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

Cohorte 2023

Proyecto Scraping

Página Seleccionada: <https://es.stackoverflow.com/>

Introducción al Scraping y elección de la librería.

- 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.

X  stackoverflow.com/robots.t...   

```
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: /*?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

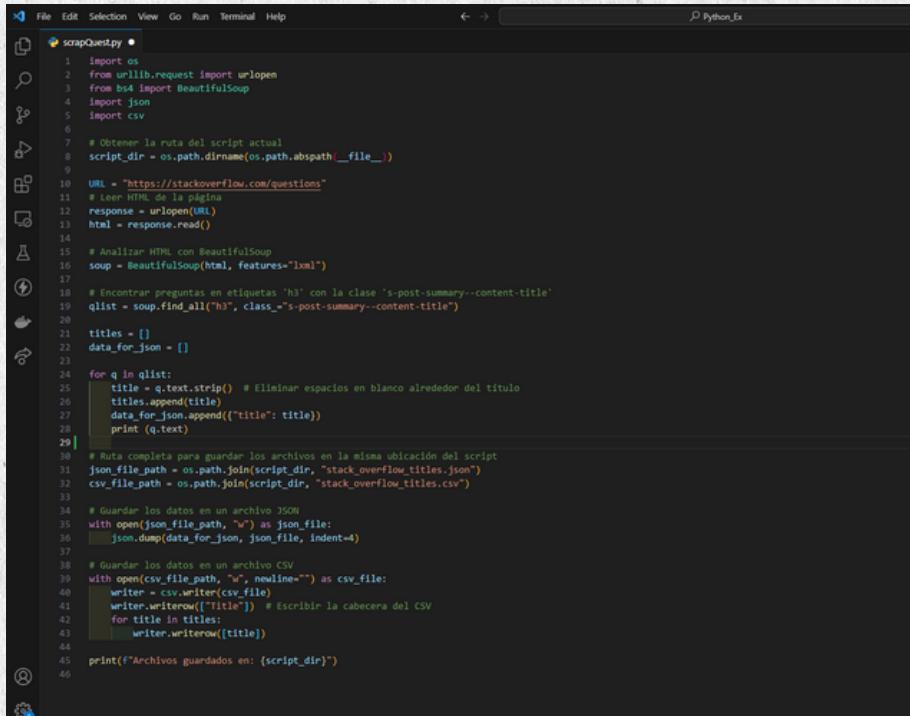
Descubrimiento de sección

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.

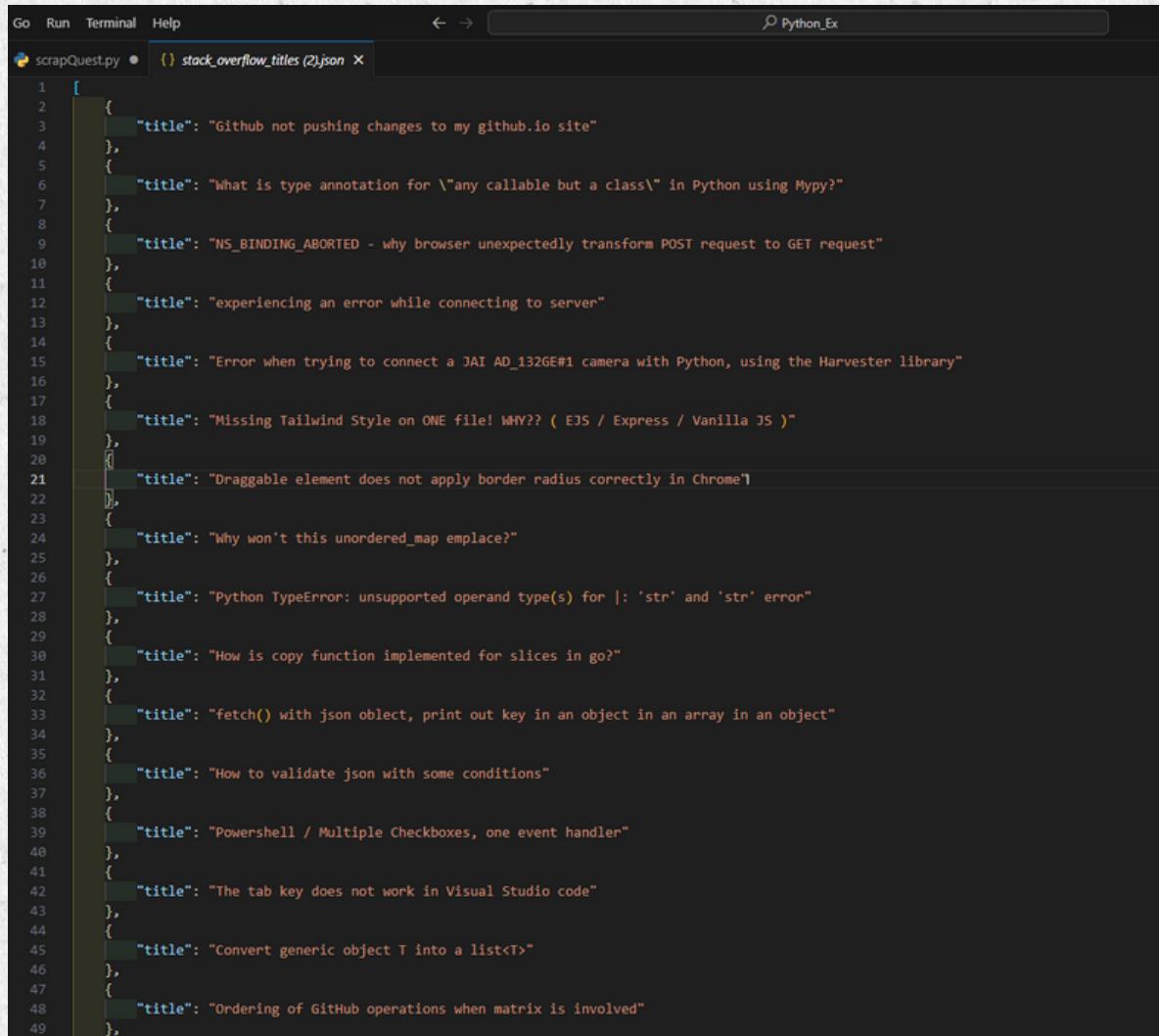
Luego probamos realizar Scraping en la sección de bolsa de empleo , pero al revisar nos dimos cuenta que la sección no estaba operando por lo cual no tendríamos datos para extraer

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 • Python_Ex
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



```
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 ONE file! WHY?? ( EJS / Express / Vanilla JS )"  
19  },  
20  {  
21    "title": "Draggable element does not apply border radius correctly in Chrome"\br/>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  },
```

• 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>
```

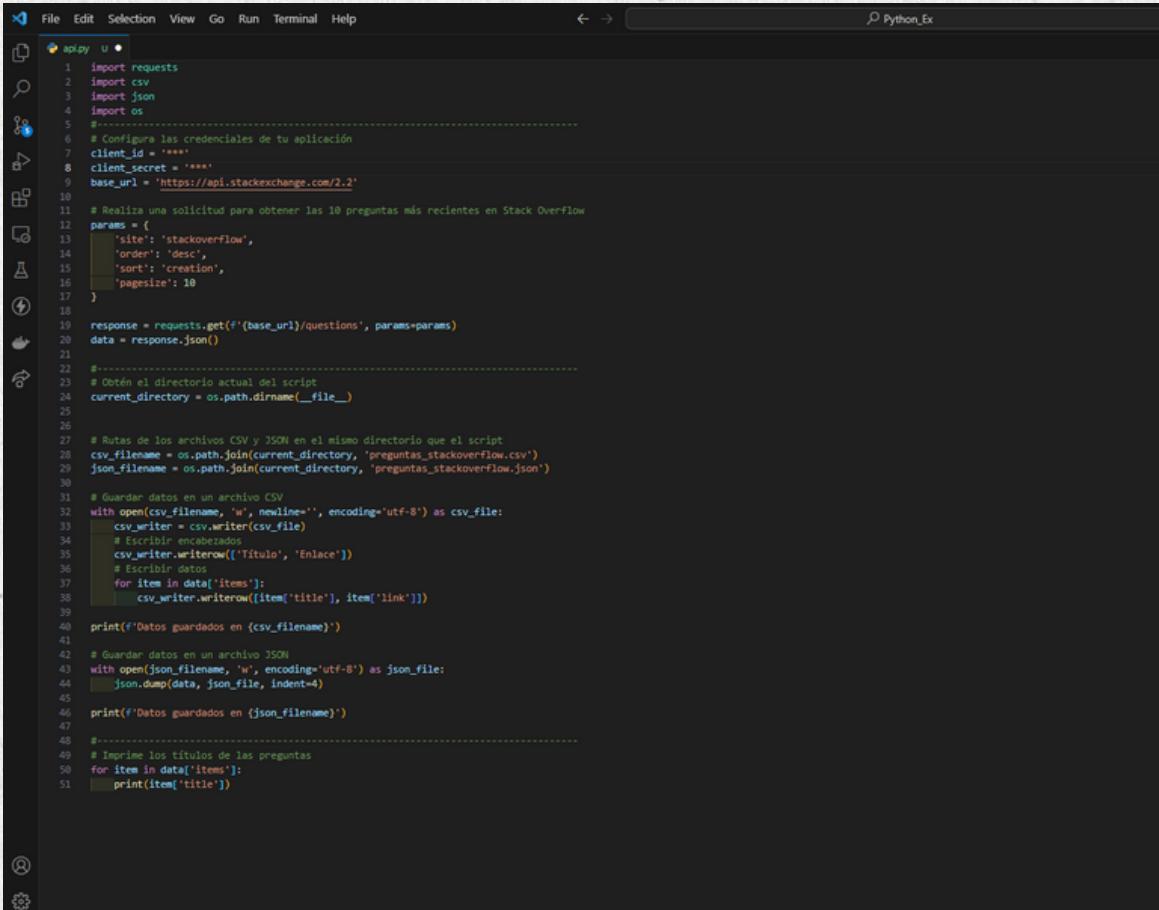
🚫 ¿Qué es el error 403 Prohibido y cómo arreglarlo?

¿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**

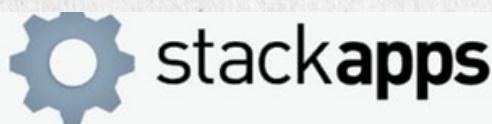
Al encontrarnos limitados para extraer datos de la página asignada comenzamos a investigar otras alternativas y nos encontramos con que Stack Overflow posee una API a través de la cual te permite realizar de forma limitada y organizada sus datos por lo que decidimos ingresar a la API llamada Stack Exchange 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 U ●
File Edit Selection View Go Run Terminal Help
apipy U ●
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/2.2'
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     'pagesize': 10
17 }
18
19 response = requests.get(f'{base_url}/questions', params=params)
20 data = response.json()
21
22 #-----
23 # Obtén el directorio actual del script
24 current_directory = os.path.dirname(__file__)
25
26
27 # Rutas de los archivos CSV y JSON en el mismo directorio que el script
28 csv_filename = os.path.join(current_directory, 'preguntas_stackoverflow.csv')
29 json_filename = os.path.join(current_directory, 'preguntas_stackoverflow.json')
30
31 # Guardan datos en un archivo CSV
32 with open(csv_filename, 'w', newline='', encoding='utf-8') as csv_file:
33     csv_writer = csv.writer(csv_file)
34     # Escribir encabezados
35     csv_writer.writerow(['Título', 'Enlace'])
36     # Escribir datos
37     for item in data['items']:
38         csv_writer.writerow([item['title'], item['link']])
39
40 print(f'Datos guardados en {csv_filename}')
41
42 # Guardar datos en un archivo JSON
43 with open(json_filename, 'w', encoding='utf-8') as json_file:
44     json.dump(data, json_file, indent=4)
45
46 print(f'Datos guardados en {json_filename}')
47
48 #-----
49 # Imprime los títulos de las preguntas
50 for item in data['items']:
51     print(item['title'])

```



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)j)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.

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.

- **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.

