



UNIVERSIDAD DE GUAYAQUIL



INTEGRANTES:

VERA LENIS BRYAN

MATAMOROS FERNANDO JOAQUÍN

FERNANDO MARCILLO

CURSO:

SOF-S-MA-3-1

MATERIA:

PROCESOS DE SOFTWARE

DOCENTE:

INGENIERO MIGUEL BOTTO TOBAR

TEMA:

PROYECTO – APLICACIÓN DE MODELO DE DESARROLLO

FECHA:

16/08/2020

CICLO CI – 2020-2021

Contenido

MODELO DE DESARROLLO POR PROTOTIPOS RÁPIDO	3
1.- RECOLECCION Y REFINAMIENTO DE REQUISITOS.....	3
REQUERIMIENTOS:.....	3
2.- DISEÑO RAPIDO Y MODELADO	4
Uso de Beautiful Soup.....	4
Librería Requests	4
2.1 Primer diseño de generación de nube de palabras.	5
3.- CONSTRUCCION DEL PROTOTIPO	6
Evaluación	6
4.- DESARROLLO, ENTREGA Y RETROALIMENTACION.....	7
CONCLUSIONES Y ENTREGA FINAL.....	8

MODELO DE DESARROLLO POR PROTOTIPOS RÁPIDO

1.- RECOLECCION Y REFINAMIENTO DE REQUISITOS.

El proyecto tiene como objetivo generar una nube de palabras de etiquetas por usuario de la plataforma Stack Overflow en español.

Para ello, el usuario debe ingresar el ID del Usuario de Stack Overflow en español, y la aplicación debe ser capaz de generar su nube de palabras (imagen) de etiquetas.

REQUERIMIENTOS:

Lenguaje de programación: PYTHON

Ingresar y leer el código de usuario con el fin de obtener así su etiqueta.

Generar un archivo .txt para almacenar las etiquetas.

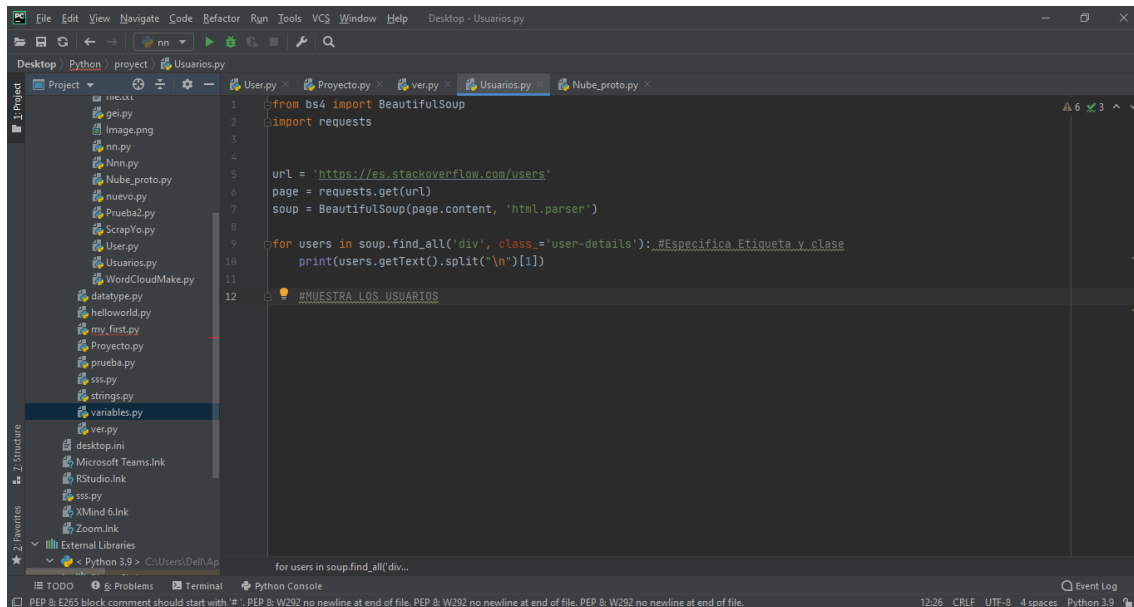
Leer el archivo Excel y generar una nube de palabras con las etiquetas en un .JPG.

Tiempo: 2 semanas

2.- DISEÑO RAPIDO Y MODELADO

El diseño rápido se centra en una representación de aquellos aspectos del software que serán visibles para el cliente o el usuario final.

Primer diseño de extracción de usuarios de StackOverflow.



```
1 from bs4 import BeautifulSoup
2 import requests
3
4 url = 'https://es.stackoverflow.com/users'
5 page = requests.get(url)
6 soup = BeautifulSoup(page.content, 'html.parser')
7
8
9 for users in soup.find_all('div', class_='user-details'): #Especifica Etiqueta y clase
10     print(users.getText().split("\n")[1])
11
12 #MUESTRA LOS USUARIOS
```

Ilustración 1. Diseño Rápido

Uso de Beatiful Soup

Beatiful Soup es una librería de Python para analizar documentos HTML (incluyendo los que tienen un marcado incorrecto). Esta librería crea un árbol con todos los elementos del documento y puede ser utilizado para extraer información. Por lo tanto, esta librería es útil para hacer web scraping (extraer información de sitios web) (Richardson, s.f.)

Librería Requests

La librería requests la utilizaremos para realizar las peticiones a la página de la que vamos a extraer los datos.

2.1 Primer diseño de generación de nube de palabras.

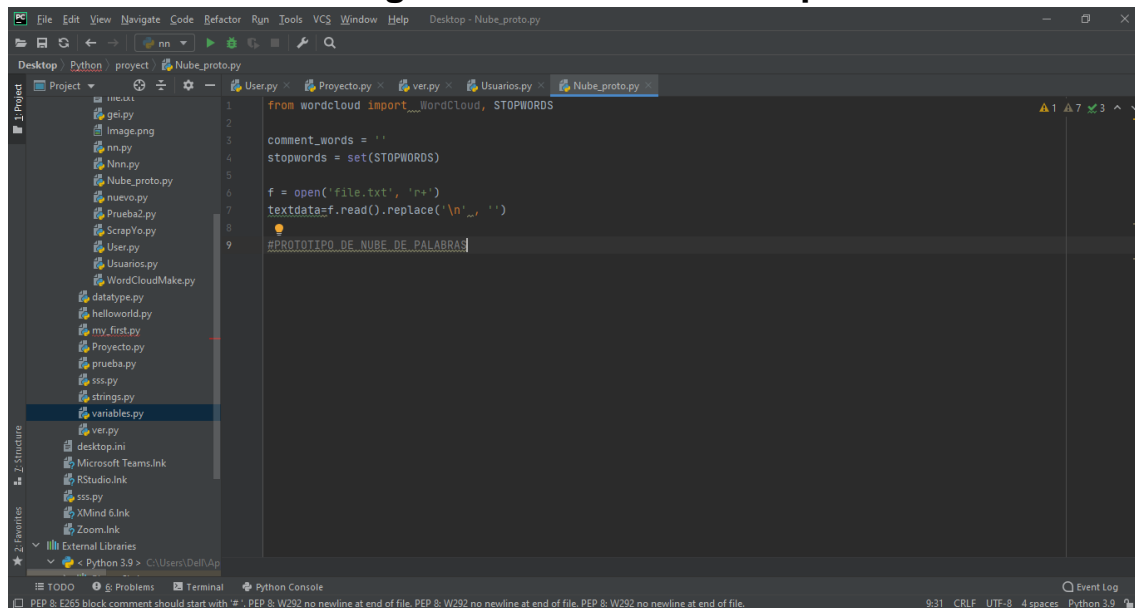


Ilustración 2. Primer Diseño de Generación de Nube de Palabras

Word Cloud es una técnica de visualización de datos que se utiliza para representar datos de texto en los que el tamaño de cada palabra indica su frecuencia o importancia. Los puntos de datos de texto importantes se pueden resaltar usando una nube de palabras. Las nubes de palabras se utilizan ampliamente para analizar datos de sitios web de redes sociales.

Para generar nubes de palabras en Python, los módulos necesarios son: matplotlib, pandas y wordcloud. (Python, s.f.)

3.- CONSTRUCCION DEL PROTOTIPO

Como estamos usando Modelo de Desarrollo de Prototipos rápido, nos desviamos hacia el uso de la librería Scrapy, la cual es una librería que permite hacer web scraping de manera vertical y horizontal, es decir, vamos a acceder a todas las páginas web donde se encuentran los usuarios de StackOverflow.



Ilustración 4. Paginación

Es decir, vamos a recorrer todas las paginaciones para extraer la

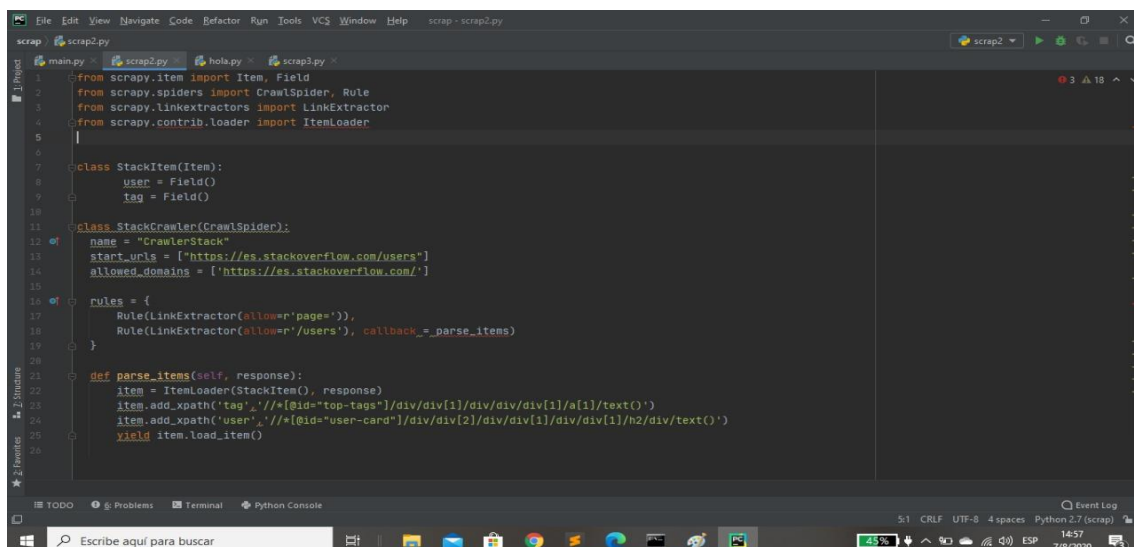
Ilustración 3. Prototipo descartado

información.

Evaluación

Después de varias pruebas se llegó a una serie de conclusiones y nos encontramos con los siguientes problemas:

- No se puede hacer demasiados requerimientos ya que nos lleva a un baneo temporal o permanente de la IP
- Extrae la información incompleta



```
1 from scrapy.item import Item, Field
2 from scrapy.spiders import CrawlSpider, Rule
3 from scrapy.linkextractors import LinkExtractor
4 from scrapy.contrib.loader import ItemLoader
5
6
7 class StackItem(Item):
8     user = Field()
9     tag = Field()
10
11
12 class StackCrawler(CrawlSpider):
13     name = "CrawlerStack"
14     start_urls = ["https://es.stackoverflow.com/users"]
15     allowed_domains = ['https://es.stackoverflow.com/']
16
17     rules = (
18         Rule(LinkExtractor(allow=r'page=')),
19         Rule(LinkExtractor(allow=r'/users/'), callback=_parse_items)
20     )
21
22 def _parse_items(self, response):
23     item = ItemLoader(StackItem(), response)
24     item.add_xpath('tag', '//*[@id="top-tags"]/div/div[1]/div/div[1]/a[1]/text()')
25     item.add_xpath('user', '//*[@id="user-card"]/div/div[2]/div/div[1]/h2/div/text()')
26     yield item.load_item()
```

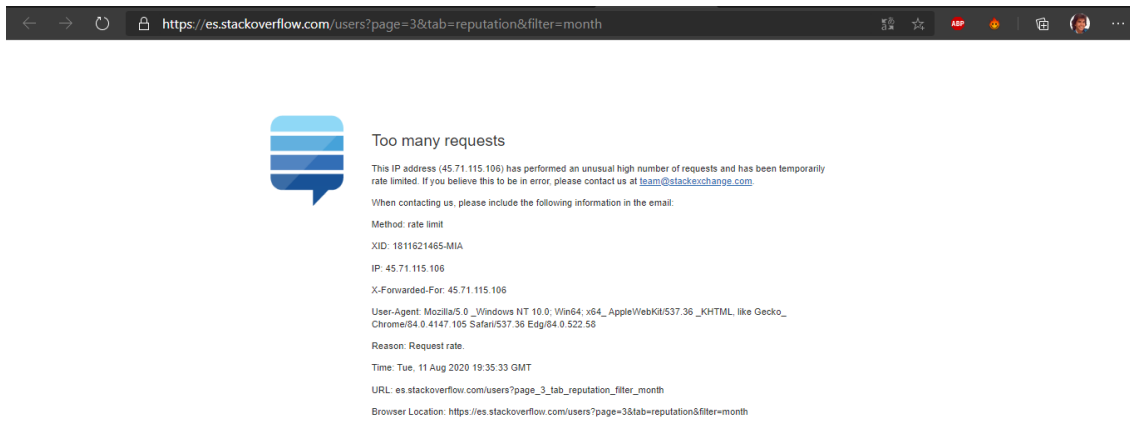


Ilustración 5. Baneo temporal de la IP



Ilustración 6. Información incompleta

4.- DESARROLLO, ENTREGA Y RETROALIMENTACION

Haciendo una retroalimentación entre todos los integrantes del grupo, se llegó a una conclusión, y mediante varios errores nos dimos cuenta de lo siguiente:

Si copias la URL de un usuario, y la pegas en otra ventana, es suficiente con que cumpla la siguiente sintaxis para que funcione:

```
url = url + user + "?tab=tags"
```

Lo cual en código nos quedó así:

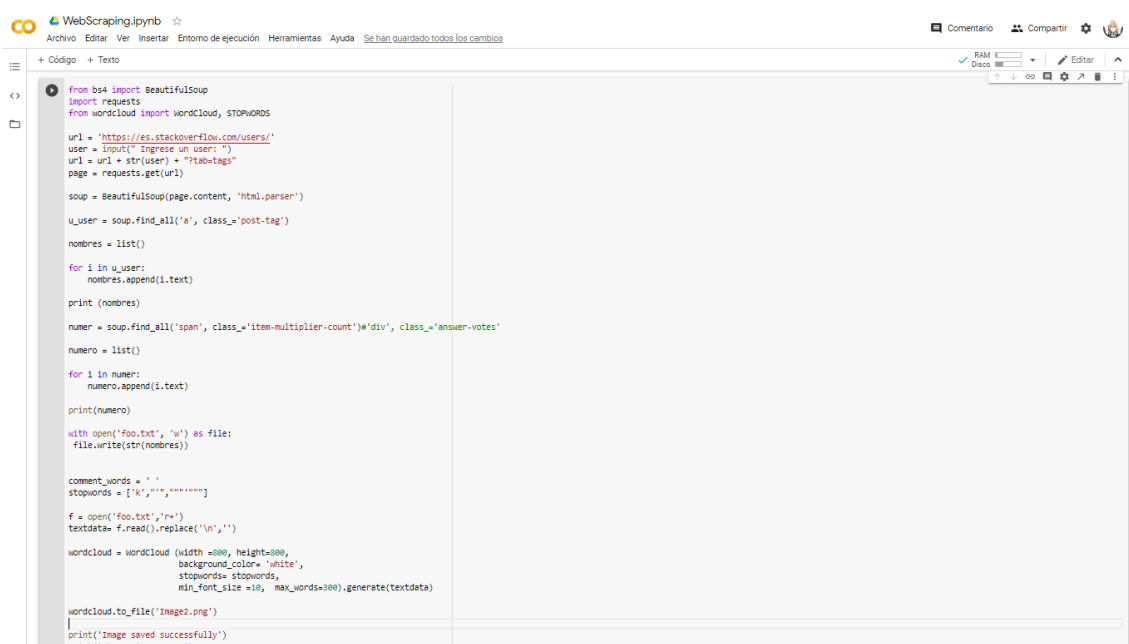
```
url = url + str(user) + "?tab=tags"
```

Es decir, cuando le damos un user correcto, el programa hace una extracción de datos única, con lo cual optimizamos tiempo y espacio de memoria, además de que obtendremos siempre información actualizada.

Así que descartamos por completo el segundo prototipo y regresamos al primer programa, modificando cierta parte de su estructura, además concatenando este con el de la nube, la cual recibe un archivo .txt y genera la nube sin ningún inconveniente.

CONCLUSIONES Y ENTREGA FINAL

En cada fase de nuestro trabajo obtuvimos retroalimentación y analizamos ambos prototipos, y aunque se haya perdido tiempo en el desarrollo del segundo, nos sirvió para hallar un factor importante en el producto final.



```
from bs4 import BeautifulSoup
import requests
from wordcloud import WordCloud, STOPWORDS

url = 'https://es.stackoverflow.com/users/'
user = input(" Ingrese un user: ")
url = url + str(user) + "?tab=tags"
page = requests.get(url)

soup = BeautifulSoup(page.content, 'html.parser')
u_user = soup.find_all('a', class_='post-tag')
nombres = list()

for i in u_user:
    nombres.append(i.text)

print (nombres)

numeros = soup.find_all('span', class_='item-multiplier-count')*div', class_='answer-votes'
numeros = list()

for i in numeros:
    numeros.append(i.text)

print(numeros)

with open('foo.txt', 'w') as file:
    file.write(str(nombres))

comment_words = ''
stopwords = ['!', ',', '.', ':']

f = open('foo.txt', 'r')
textdata= f.read().replace('\n','')

wordcloud = WordCloud(width=800, height=800,
    background_color= 'white',
    stopwords= stopwords,
    min_font_size=10, max_words=300).generate(textdata)

wordcloud.to_file('image2.png')
print("Image saved successfully")
```

Ilustración 7. Código final en el entorno Google Colab


```
Ingrese un user: 99424
['pandas', 'pandas', 'python-3.x', 'condiciones', 'python', 'dataframe', 'numpy']
['5', '3', '5', '3']
Image saved successfully
Change image name
```

Ilustración 8. Programa ejecutado en Google Colab

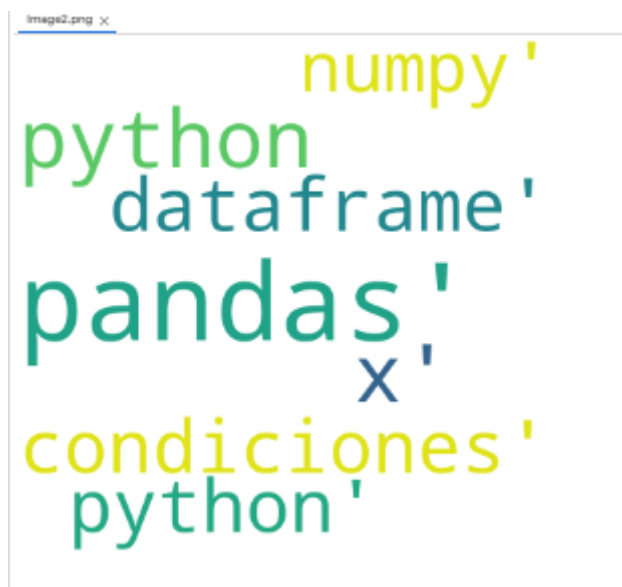


Ilustración 9. Imagen generada

GITHUB:

<https://github.com/alexisevergarden/wordcloud>

ENLACE A GOOGLE COLAB:

<https://colab.research.google.com/drive/1Wq-bEGcdJAxfekieAnYrwlq76L3-a-Eb?usp=sharing>