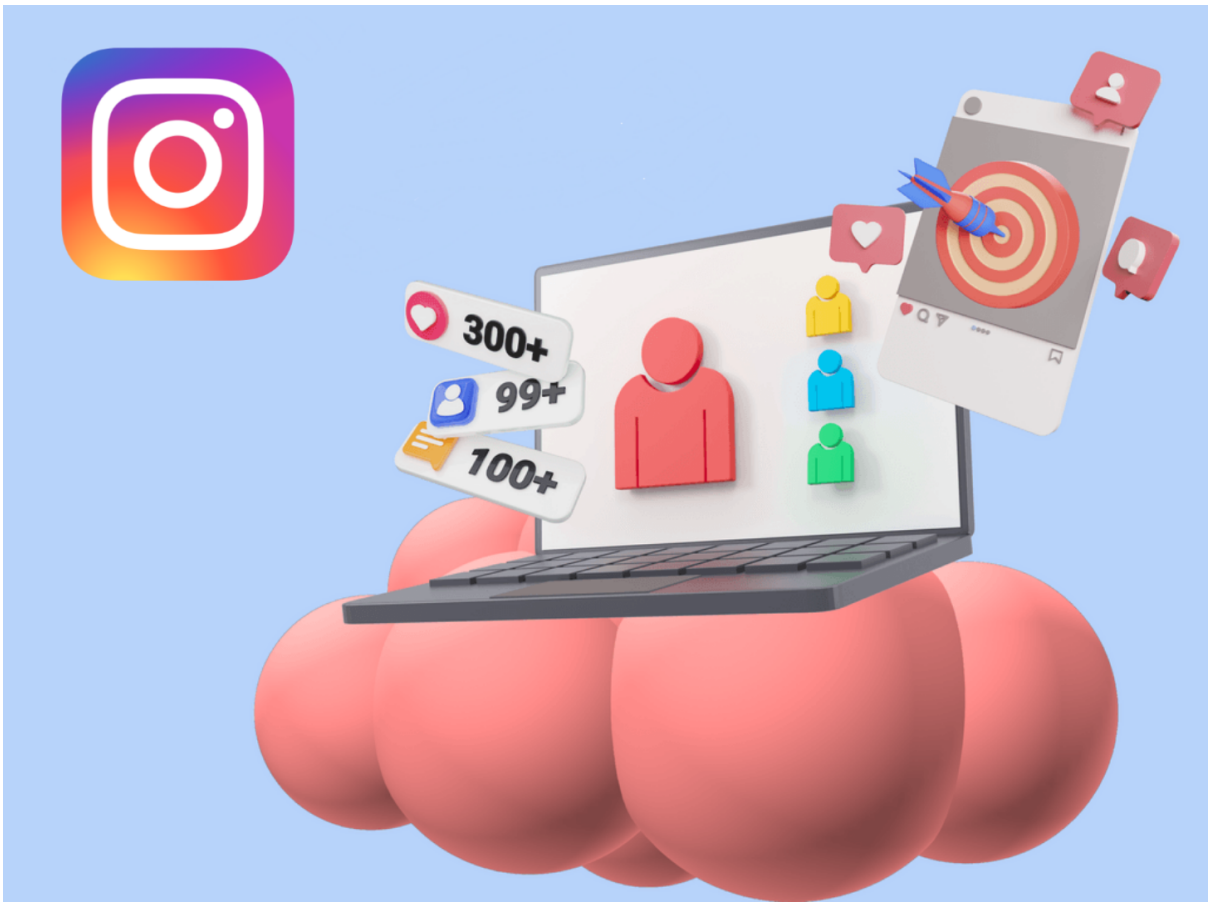
UNIVERSITAT JAUME I

EXTRACTOR DE DATOS DE INSTAGRAM



Memoria de un extractor de datos de Instagram

Nicolás Camañes Antolín

20923237E

INTRODUCCIÓN **3**

INSTALACIÓN PREVIA **3**

- LIBRERÍAS EMPLEADAS 3

SELENIUM WEBDRIVER, TIME, OS 3

WGET, WARNINGS 4

- INSTALACIÓN NECESARIA (Linux) 4

IMPLEMENTACIÓN DEL PROGRAMA **5**

- IMPORTS 5
- MÉTODOS 5

def load_instagram(): 6

def allow_cookies(): 6

def login(u, p): 6

def not_save_login_info(): 7

def no_notif(): 7

def search_user(user_name): 7

def get_username(): 8

def get_followers(): 9

def get_following(): 9

def get_name(): 9

def get_role(): 9

def get_link(): 9

def get_descrip(): 9

def scroll_posts_images(): 10

def get_images(posts, condicion): 11

def create_general_folder(user_name): 14

def save_profile_info(n_posts, n_followers, n_following, full_name, 14

def posts_txt(user_name, url_posts, likes_posts, coments_posts): 15

def create_posts_folder(user_name): 15

- PROGRAMA PRINCIPAL 16

TUTORIAL DE USO **18**

BIBLIOGRAFÍA **20**

INTRODUCCIÓN

Esta memoría consiste en la explicación del código del programa 'myextractor.py', así como las instalaciones necesarias para poder usarlo y un tutorial de cómo usarlo adecuadamente. El objetivo del programa es poder obtener información de perfiles de instagram y descargarla en diferentes tipos de formato.

INSTALACIÓN PREVIA

LIBRERÍAS EMPLEADAS

- SELENIUM WEBDRIVER

Se trata de una herramienta de código abierto muy útil para la automatización de pruebas de navegación web. Nos permite la ejecución directa de pruebas vía código fuente en el servidor. Es soportada por diferentes lenguajes de programación como Java, C#, Python, etc. En concreto Selenium WebDriver a diferencia de otras componentes de Selenium, no utiliza middleware sino que controla el navegador interactuando directamente con él. Además, se puede usar con los siguientes navegadores: Mozilla, Google Chrome, Opera, Microsoft Edge e Internet Explorer. Por un lado, las ventajas más destacables son que, su proceso de instalación es claro y simple, es gratuito, no depende de un servidor para usar el controlador web y el código escrito de las implementaciones es claro. Por otro lado, la desventaja o limitación más destacada es que sólo podemos ejecutar aplicaciones basadas en web.

- TIME

Se trata de un conjunto de funciones que permiten trabajar con fechas y horas.

- OS

Este módulo proporciona una forma portátil de utilizar la funcionalidad del sistema operativo. Los usos más comunes son, crear un directorio, obtener la ruta de un directorio, determinar el nombre del sistema operativo o eliminar un directorio entre otras.

- WGET

Se trata de un módulo muy útil para descargar archivos desde el servidor principal mediante la URL del archivo.

- WARNINGS

Cuando ejecutamos un programa, puede que nos salten mensajes de alerta para informar de errores de configuración o de la denegación de características de bibliotecas faltantes. Grácias a esta biblioteca conseguiremos que estas alertas no sean visibles para el usuario.

INSTALACIÓN NECESARIA (Linux)

Para ejecutar este programa será necesario instalar python 3. En caso de que no lo tengamos, para linux puede instalarse ejecutando las dos órdenes siguientes en el terminal:

```
$ sudo apt-get update
$ sudo apt-get install python3.6
```

También será necesario que descarguemos el programa 'myextractor.py' que se adjunta con la práctica para poder ejecutarlo.

Este programa usa unas librerías que puede que no tengamos instaladas en el ordenador. Para ver la lista de librerías, y la versión, que tenemos instaladas podemos ejecutar en el terminal el comando:

```
$ pip lists
```

Seguramente, si nunca hemos usado estas librerías, las que nos falten instalar serán Selenium y Wget. Que las podemos instalar ejecutando los siguientes comandos:

```
$ pip install selenium
$ pip install wget
```

En caso de que nos den error los dos comandos anteriores, será necesario ejecutar las dos órdenes anteriores como root, es decir, añadir sudo al principio de cada comando.

Cabe aclarar que las librerías Time, Warinings y Os son módulos de la librería de python y al tener instalado python, no será necesario instalarlas.

Por último, será necesario instalar el driver, recomiendo que sea de Chrome ya que es con el que se ha probado este programa y evitaremos errores inesperados. Esto lo podemos obtener en la siguiente web: https://chromedriver.chromium.org/downloads

Para saber la versión de Chrome que tenemos podemos ejecutar:

$ google-chrome –product-version

IMPLEMENTACIÓN DEL PROGRAMA

IMPORTS

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
import time
import os
import wget
import warnings
```

Mediante el código previo, importamos los módulos de las librerías necesarias, que hemos descrito en apartados anteriores, para usar funciones de estas posteriormente en nuestro código.

MÉTODOS

En primer lugar, explicaremos lo que hacen algunas funciones, de las librerías que hemos importado, que aparecerán a frecuentemente en algunos de los métodos que se han implementado.

- **driver.find_elemet(by, value):** Mediante este método podemos buscar lo que deseemos en el código html configurando adecuadamente los dos parámetros (by y value). El parámetro by se corresponde con el tipo de dato que estemos buscando o de qué forma vayamos a buscarlo. En este programa usaremos:`CSS_SELECTOR`, XPATH y TAG_NAME. En el parámetro value, introduciremos la ruta para acceder a dicho dato en el código html.
- **time.sleep(seconds):** Mediante este método de la librería time. Podemos hacer que nuestro programa se paralice tantos segundos como lo pasemos por parámetro. En el programa será muy útil tanto para simular el comportamiento humano de algunas

ejecuciones para que Instagram no nos bloquee la cuenta, como para esperar a que se carguen las páginas para poder ejecutar el código sobre ellas.

➢ **driver.get(url):** Con éste método cargaremos mediante el driver las páginas web que le pasemos por parámetro en la variable. Gracias a esto, podremos cargar la página principal de Instagram o cada uno de los posts de un perfil.

A continuación, se explica el funcionamiento de cada uno de los métodos que hemos implementado:

```
def load_instagram():
    driver.get("https://www.instagram.com/")
```

Este método usa la función importada driver.get() para cargar la página principal de Instagram.

```
def allow_cookies():
    time.sleep(4)
    driver.find_element(by=By.XPATH,
                        value = "//button[contains(text(), \
                       'Permitir solo cookies necesarias')]").click()
```

Mediante este método buscamos en la pantalla el botón de “Sólo permitir las cookies necesarias” y hacemos click sobre él.

```
def login(u, p):
    time.sleep(5)
    username = driver.find_element(by = By.XPATH,
                value = '//*[@id="loginForm"]/div/div[1]/div/label/input')
    password = driver.find_element(by = By.XPATH,
                value = '//*[@id="loginForm"]/div/div[2]/div/label/input')
    username.clear()
    password.clear()
    username.send_keys(u)
    password.send_keys(p)
    login = driver.find_element(by = By.XPATH,
            value = '//*[@id="loginForm"]/div/div[3]/button').click()
```

Con éste método, primero, buscamos los slots donde poner el usuario y la contraseña que los localizamos porque tienen como input: username y password. A continuación, los vaciamos para asegurarnos de que no hay ningún dato introducido previamente. Después

introducimos los el usuario (u) y la contraseña (p) que le hemos pasado cómo parámetro para que los introduzca en sus correspondientes slots. Finalmente, hacemos click en el botón de submit para acabar con la identificación en Instagram.

```
def not_save_login_info():
    time.sleep(6)
    driver.find_element(by = By.XPATH,
                        value = '//*[@id="react-root"]/section/main/\
                        div/div/div/div/button').click()
```

Una vez hemos pasado la fase de logearnos en Instagram, nos saldrá un texto emergente preguntando si queremos guardar los datos de usuario y contraseña de acceso. Mediante este método le daremos click al botón con el texto de 'Ahora no' para no guardar los datos con los que hemos accedido.

```
def no_notif():
    time.sleep(4)
        driver.find_element(by = By.CSS_SELECTOR,
        value = 'body > div.RnEpo.Yx5HN > div > div > div\
                 > div.mt3GC > button.aOOlW.HoLwm').click()
```

A continuación, nos saldrá otra ventana emergente que nos preguntará si queremos recibir notificaciones de Instagram. Mediante éste método, de una forma parecida a en el anterior. Haremos click al botón con el texto 'Ahora no'.

```
def search_user(user_name):
    time.sleep(5)
    searchbox = driver.find_element(by = By.XPATH,
                value = '//*[@id="react-root"]/section/nav/\
                div[2]/div/div/div[2]/input')
    searchbox.clear()
    searchbox.send_keys(user_name)
    time.sleep(2)
    user = get_username()
    if user == user_name:
        searchbox.send_keys(Keys.ENTER)
        time.sleep(2)
        searchbox.send_keys(Keys.ENTER)
        return('Hemos encontrado el usuario con éxito')
    else:
        return('No se ha encontrado ese usuario')
```

El objetivo de éste método es buscar el usuario (user_name) que le hemos pasado como parámetro. En primer lugar buscaremos el slot donde debemos introducir el usuario y se lo asignaremos a la variable searchbox. Del mismo modo que en métodos anteriores donde teníamos que introducir datos en la web, vaciaremos el slot para evitar errores si se ha quedado el de alguna búsqueda anterior, introduciremos el usuario que le hemos pasado al método como parámetro y hacemos doble enter. Luego, compararemos el usuario que buscamos con el devuelto por la función get_username() y, en caso de coincidir, devolveremos el texto 'Hemos encontrado el usuario con éxito'. En caso contrario, devolveremos 'No se ha encontrado ese usuario'.

```
def get_username():
    time.sleep(1)
    try:
        c_username = driver.find_element(by=By.XPATH,
        value='/html/body/div[1]/section/nav/div[2]/div/div/div[2]/div[3]\
        /div/div[2]/div/div[1]/a/div/div[2]/div/div/div/div[1]').text
    except: c_username = 'Unknown'
    return str(c_username)
```

Este es el método al que llamamos en la función anterior. Lo usamos para evitar que el programa se bloquee al introducir usuarios que no existen o asegurarnos de que el usuario encontrado se corresponde con el usuario devuelto. Lo que hace es intentar devolver el primer usuario que sale en la barra desplegable usuarios que aparece después de introducir un usuario en la caja de búsqueda (Figura). En caso de no existir ningún usuario parecido, no se desplegará ninguna barra con usuarios y por tanto saltará una excepción, en ese caso, el método devolverá el texto 'Unknown'.

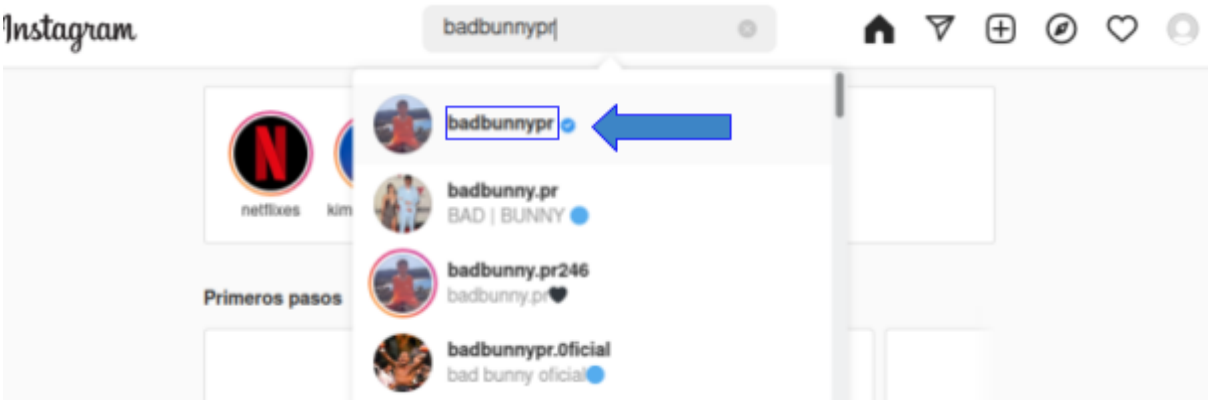


Fig. 1 Imagen de la barra desplegable emergente al buscar un usuario

```
def get_posts():
   time.sleep(3)
   try: n_posts = driver.find_element(by = By.XPATH,
       value='/html/body/div[1]/section/main/\
       div/header/section/ul/li[1]/div/span').text
   except: n_posts = '0'
   return n_posts
def get_followers():
   try: n_followers = driver.find_element(by = By.XPATH,
       value = '/html/body/div[1]/section/main/div/header\
       /section/ul/li[2]/a/div/span').text
   except:n_followers = '0'
   return n_followers
def get_following():
   try: n_following = driver.find_element(by = By.XPATH,
        value = '/html/body/div[1]/section/main/div/header/\
        section/ul/li[3]/a/div/span').text
   except: n_following = '0'
   return n_following
def get_name():
   try: full_name = driver.find_element(by = By.XPATH,
        value= '/html/body/div[1]/section/main/div/header/\
        section/div[2]/span').text
   except: full_name = 'Unknown'
   return full_name
def get_role():
   try: role = driver.find_element(by = By.XPATH,
               value = '/html/body/div[1]/section/main/div/\
               header/section/div[2]/div[1]/div').text
   except: role = 'Unknown'
   return role
def get_link():
   try: address= driver.find_element(by = By.XPATH,
                value = '/html/body/div[1]/section/main/div/\
                header/section/div[2]/a/div').text
   except: address = 'Unknown'
   return address
def get_descrip():
   try: descrip = driver.find_element(by = By.XPATH,
```

```
                value = '/html/body/div[1]/section/main/div/\
                header/section/div[2]/div[2]').text
    except:
        try: descrip = driver.find_element(by = By.XPATH,
                    value = '/html/body/div[1]/section/main/div/\
                    header/section/div[2]/div').text
        except: descrip = 'Unknown'
    return descrip
```

Para estos métodos juntaremos la explicación de su funcionamiento como si fueran uno ya que resultan muy parecidos. Lo que hacen es, mediante la función importada de selenium, driver.find.element() buscamos el elemento que deseamos obtener. Ya que, como parámetro a esta función, le pasamos el xpath de donde se encuentra el dato que deseamos. Lo guardamos en una variable y lo devolvemos. Lo que acabamos de explicar lo introducimos dentro del esquema try: except: para que en caso de no encontrar alguno de los datos, en lugar de bloquearse el programa, devuelve para ese dato el texto 'Unknown'. Esto lo hacemos para los siguietnes datos: número de posts (1), número de seguidores (2), número de usuarios seguidos (3), nombre del perfil (4), tipo de cuenta (5), descripción del perfil (6) y enlace del perfil (6). Estos datos se encuentran al principio de cada perfil tal y como se muestra en la figura. Aunque en algunos perfiles puede no aparecer alguno de ellos, por eso es importante capturar las excepciones.

Fig. 2 Dato que que extraemos del usuario de Instagram

```
def scroll_posts_images():
    time.sleep(6)
    posts = []
    scrolldown = driver.execute_script("window.scrollTo(0, \
                document.body.scrollHeight);\
                var scrolldown=document.body.scrollHeight;\
                return scrolldown;")
    match=False
    while(match==False):
        links_posts = driver.find_elements(by = By.TAG_NAME, value = 'a')
        for link in links_posts:
            post = link.get_attribute('href')
            if (post not in posts) and ('/p/' in post):
                  posts.append(post)
        last_count = scrolldown
        time.sleep(7)
        scrolldown = driver.execute_script("window.scrollTo(0, \
                    document.body.scrollHeight);\
                    var scrolldown=document.body.scrollHeight;\
                    return scrolldown;")
        time.sleep(2)
        if last_count==scrolldown: match=True
    return posts
```

Este método es crucial para obtener los enlaces a cada uno de los posts de un perfil de instagram. Para entender éste método, primero es necesario saber que cuando buscamos un perfil, a no ser que tenga muy pocas fotos, no se nos cargarán todos los posts a la vez, sino que, a medida que vayamos scroleando la página, se nos irán cargando los posts en diferentes tandas. En primer lugar, definimos la variable posts como una lista vacía. A continuación, mediante una de las funciones de selenium y gracias al bucle while de la función, iremos haciendo scroll hasta el final de la página, que no es lo mismo que hasta el final de todo el perfil de Instagram. Cada vez que hacemos scroll, es decir, por cada una de las iteraciones del bucle, lo que haremos será añadir a posts todos los enlaces que nos vayamos encontrando, gracias al driver, y que aún no hayamos añadido. Para encontrar estos enlaces en el código html, mediante la función driver.find_elements() buscaremos la etiqueta ‘a’, de donde nos quedaremos con el atributo ‘hrfef’, que es donde se encuentra el enlace al posts. Una vez encontrado un enlace, para decidir si lo añadimos a la lista comprobamos que el enlace tenga la subcadena ‘/p/’ para asegurarnos que es un post y

además también comprobamos que no lo hayamos añadido anteriormente para evitar repeticiones. El criterio con el que finalizará de scrollear, saldrá del bucle while, es que lo que devuelve la función con la que realizamos scroll sea igual a lo que devolvió la última vez que la usamos. De esta manera, el bucle finaliza cuando llegamos al final del perfil de Instagram ya que no podremos seguir scrolleando hacia abajo.

```
def get_images(posts, condicion):
    images, likes, coments = [], [], []
    for a in posts:
        driver.get(a)
        time.sleep(5)
        try: like = driver.find_element(by=By.XPATH,
                    value='//*[@id="react-root"]/section/main/divdiv[1]/\
                     article/div/div[2]/div/div[2]/section[2]/\
                    div/div/div/a/div/span').text
        except:
            try: like = driver.find_element(by=By.XPATH,
                        value='//*[@id="react-root"]/section/main/div/\
                        div[1]/article/div/div[2]/div/div[2]/section[2]/\
                        div/span/div/span').text
            except: like = 'Unknown'
        likes.append(like)
        try: coment = driver.find_element(by=By.XPATH,
                      value='//*[@id="react-root"]/section/main/div/\
                      div[1]/article/div/div[2]/div/div[2]/div[1]/ul/div/\
                      li/div/div/div[2]/div[1]/span').text
        except: coment = 'Unknown'
        coments.append(coment)
        if condicion == 'si':
            try:
                    vid = driver.find_element(by=By.TAG_NAME, value='video')
                    vid=vid.get_attribute('poster')
                    images.append(vid)
            except:
                try:
                    img = driver.find_element(by=By.TAG_NAME, value='img')
                    time.sleep(4)
                    img = img.get_attribute('src')
                    images.append(img)
                except: images.append('Not found')
```

```
return (images, likes, coments)
```

A este método le debemos pasar como parámetro la lista con los enlaces a los posts que hemos obtenido del método scroll_posts_images() y un string, la variable condicion, que en caso de ser 'si', por cada post obtendrá el enlace a la imagen (en el vector images) que aparece en cada post para poderla descargar más adelante mediane otros métodos. Además, por cada post trataremos de obtener la cantidad de likes o reproducciones (en el vector likes) y la descripción de la ùblicación (en el vector coments). El objetivo es devolver tres vectores (images, likes y coments) donde cada índice haga referencia a un post y de cada vector podamos obtener el dato correspondiente a ese post tal y como se muestra en la Figura 3 .

Fig. 3 Estructura y relación entre las tres listas que devuelve este método

Más detalladamente, antes de empezar con la 'extracción' inicializamos las variables images, likes y coments a tres listas vacías. A continuación, con un bucle for, iremos recorriendo uno a uno los enlaces que tenemos en la lista posts. Por cada uno de ellos, mediante la función dirver.get(url) cargaremos el post y realizaremos las siguientes extracciones.

En primer lugar, mediante un try, trataremos de obtener los likes. Si es el caso de que aparecen, en caso de que no podamos y salte una excepción, puede ser por dos motivos, que no haya likes sino reproducciones porque el post sea un vídeo o que el usuario tenga configurado instagram para que no aparezcan los likes de las publicaciones. Para tratar primero el caso de un video, en la excepción, hemos introducido otro try para tratar de encontrar las reproducciones. Y, si aquí, vuelve a saltar otra excepción nos guardaremos para ese post la cadena 'Unknown' en la lista posts para indicar que no los hemos podido averiguar. En los otros dos casos en la lista likes le añadiremos un string con la cantidad de likes o reproducciones según se corresponda en cada post.

En segundo lugar, haremos algo parecido pero para los comentarios. Para este caso las fotos y los videos se tratarán de la misma forma. Luego, en un try, trataremos de obtener el comentario de la foto, y en caso de saltar alguna excepción, que será porque no hemos podido obtener el dato que deseamos, tomaremos el comentario de ese post como la

cadena ‘Unknown’. Y añadiremos el comentario extraído para ese post (o la cadena ‘Unknown’ si no lo hemos encontrado) a la lista coments.

En tercer lugar, trataremos de obtener el enlace de la imagen de cada post. En este caso se realizará de formas diferentes si se trata de un vídeo o de una imagen, ya que en el código html el enlace que buscamos se encontrará en un sitio u otro. Primero tenemos un if para comprobar si la variable local condicion, que le hemos pasado como parámetro al método, es igual a la cadena ‘si’. Esto es porque en el programa principal preguntaremos al usuario si desea descargar las imágenes. Por tanto sólo necesitaremos obtener el enlace a la imagen en caso de que el usuario desee descargarlas. A continuación, tenemos un try para obtener el enlace del la imagen de la portada del post en caso de que se trate de un vídeo, la orden para realizar la extracción es, una vez más drive.find_elemet() donde buscaremos la etiqueta ‘video’, nos quedaremos con el atributo ‘poster’ y añadiremos el enlace a la lista images. En caso de saltar una excepción, será porque no ha encontrado la etiqueta ‘video’, esto es debido a que el post no se trata de un vídeo. Seguramente se trate de una imagen. Luego en el except, haremos otro try para intentar obtener el enlace a la imagen del post. Esto lo haremos de la misma forma que como hemos hecho en el caso de los vídeo, sólo que ahora buscaremos la etiqueta ‘img’, al encontrarla nos quedaremos con el atributo ‘src’ que es donde se encuentra el enlace que deseamos y lo añadiremos a la lista images. En caso de saltar otra excepción aquí, para evitar que nuestro programa se bloquee, añadiremos a la lista images la cadena ‘Unknown’.

Finalmente, el método devolverá una tupla con las tres listas en las que hemos ido guardando la información.

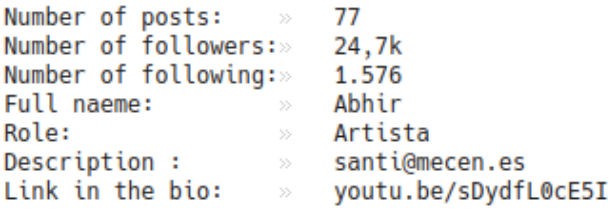
Cabe destacar que el orden hemos seguido en la extracción de los enlaces de las imágenes, primero intentar extraer la imagen de la portada del video y si falla porque no se trata de un video tratar de extraer la imagen porque seguramente se trate de una imagen es muy importante. Ya que si cambiamos el orden no lo hará bien porque si se trata de un video, y tratamos de buscar la etiqueta ‘img’ no saltará una excepción porque más adelante de donde debería dar con la etiqueta ‘video’ encontrará una etiqueta ‘img’ correspondiente con la foto de perfil del usuario. Y estaríamos devolviendo el enlace de esta imágen y no el que deseamos.

```
def create_general_folder(user_name):
    #Crear carpera si no existe
    if(os.path.isdir(user_name) == False):
            os.mkdir(user_name)
    return (os.path.dirname(os.path.abspath(__file__))+user_name)
```

En este método usamos el módulo 'os' para crear un directorio. Primero comprobamos si existe un directorio con el nombre que le pasamos como parámetro, que se corresponde con el nombre de usuario de Instagram del cuál estamos haciendo la extracción. En caso de no existir, crearemos un directorio con dicho nombre. Finalmente, el método devolverá la ruta absoluta del directorio.

```
def save_profile_info(n_posts, n_followers, n_following, full_name,
                      role, descrip, link_gived):
    f= open(str(user_name)+'/'+'profile_info'+'.txt','w')
    f.write('Number of posts:    '+'\t'+str(n_posts)+'\n')
    f.write('Number of followers:'+'\t'+str(n_followers)+'\n')
    f.write('Number of following:'+'\t'+str(n_following)+'\n')
    f.write('Full name:         '+'\t'+str(full_name)+'\n')
    f.write('Role:               '+'\t'+str(role)+'\n')
    f.write('Description :       '+'\t'+str(descrip)+'\n')
    f.write('Link in the bio:    '+'\t'+str(link_gived)+'\n')
    f.close()
```

Grácias a éste método, la información que hemos obtenido mediante los métodos: get_posts(), get_followers(), get_following(), get_name(), get_role(), get_descrip() y get_link(). La podremos almacenar en un fichero de texto al cuál llamaremos 'profile_info.txt' para que en un futuro resulte más fácil acceder a ella en caso de necesitarlo. Como parámetros al método le pasamos una variable por cada uno de los datos que previamente hemos extraído. Después, mediante la función open (fichero, 'w') crearemos el archivo, o lo abriremos si ya existe. Y finalmente empleando la función .write() escribiremos una línea por cada dato que vamos a almacenar en el fichero tal y como se muestra en la siguiente imagen.



```
def posts_txt(user_name, url_posts, likes_posts, coments_posts):
    f = open (str(user_name)+'/'+'url_of_posts'+'.txt','w')
    for i in range(len(url_posts)):
        f.write('Post '+str(i)+':'+'\t'+url_posts[i]+'\n'+
                'Likes/Reproductions:'+'\t'+str(likes_posts[i])+'\n'+
                'Description:'+'\n'+coments_posts[i]+'\n')
        f.write('\n')
    f.close()
```

El objetivo de éste método también es depositar información en un fichero. Pero en éste caso se tratará de la información de cada post en un fichero al que llamaremos 'url_of_posts.txt' y tendrá el siguiente aspecto:

```
Post 0:»https://www.instagram.com/p/CdJTqnhr9eH/
Likes/Reproductions:»   5.430.873
Description:
SOLO FALTA 1 DÍA!!!!!! UN VERANO SIN TI ❤️ 🔥🌴☀️ 🌊 EStaN REadyyyyyyy?????

Post 1:»https://www.instagram.com/p/CdEEEjDF_Pr/
Likes/Reproductions:»   4.048.669
Description:
Un Verano Sin Ti ❤️ 🌊☀️ 🌴🩳
```

```
def create_posts_folder(user_name):
    director_for_posts = user_name+'/downloaded_posts'
    if(os.path.isdir(director_for_posts) == False):
            # Create folder
            os.mkdir(director_for_posts)
    return (os.path.dirname(os.path.abspath(__file__)) + director_for_posts,
            director_for_posts)
```

Este método es muy parecido a create_general_folder(). Primero comprobamos si existe una carpeta llamada downloaded_posts dentro de la carpeta que creamos con create_general_folder(). En caso de no existir la creamos. Y, finalmente, devolvemos una tupla con la ruta absoluta del nuevo directorio y la ruta local es éste también.

```
def download_images(images, posts_folder):
    counter = 0
    for img in images:
        if img != 'Not found':
            save_as = os.path.join(posts_folder,
                                   'posts_image_'+str(counter)+".jpg")
            wget.download(img, save_as)
            counter += 1
```

A este método le pasamos por parámetro la lista con los enlaces a las imágenes y la ruta del directorio donde las queremos descargar, que es el segundo elemento de la tupla que devuelve create_posts_folder(). Por cada elemento de la lista, primero comprobaremos que se trata de un enlace ya que en el método en el que rellenamos la lista en caso de no encontrar el enlace de una imagen añadimos la cadena 'Not found' a la lista. Por tanto, comprobamos que el elemento sea diferente a 'Not found'. Luego, mediante el comando os.path.join() creamos la ruta donde guardaremos la imagen y la guardamos en la variable save_as. En tercer lugar, mediante el método wget.download() descargamos la imagen y la guardamos en la ruta que hemos creado. Cabe destacar que vamos incrementando un contador por cada iteración del bucle for para que cuando creamos el nombre de cada post podamos realizar una ordenación en el nombre de cada imagen.

PROGRAMA PRINCIPAL

Resultará bastante fácil de entender el funcionamiento del programa principal ya que como hemos separado el código en muchas funciones, aquí las iremos llamando en el orden adecuado.

En primer lugar mediante la orden warnings.filterwarnings('ignore') hacemos que los warnings se ignoren y no se impriman por pantalla. Después, en PATH definimos la ruta absoluta donde se encuentra el driver que hemos descargado en el proceso de instalación. A continuación, definimos el usuario y la contraseña de la cuenta con la que nos logueamos en instagram en las variables usuario y contra. A la variable driver, le asignaremos lo que devuelve la función webdriver.Chrome(PATH) que hemos importado de selenium. Y ejecutaremos los siguientes métodos para que se ejecuten consecutivamente.

```
load_instagram()
allow_cookies()
login(usuario, contra)
not_save_login_info()
no_notif()
```

La segunda fase del programa consiste en encontrar el usuario que deseamos scrapear. Primero, pediremos que se introduzca por teclado el nombre de usuario que deseamos scrapear. Para que no se bloquee si se introduce un usuario que no existe o se teclea mal el nombre de usuario ae ha implementado un bucle while que en caso de que lo que nos devuelva la función search_user() sea la cadena 'No se ha encontrado ese usuario' volveremos a pedir que se introduzca el nombre de la cuenta que queremos scrapear por teclado hasta que al ejecutar el método search_user() nos devuelva algo diferente a dicha

cadena. Conociendo el funcionamiento del método search_user(), lo que devolverá en ese caso será 'Hemos encontrado el usuario con éxito'.

```
encontrado = search_user(user_name)
while encontrado == 'No se ha encontrado ese usuario':
   user_name = input('No se ha encontrado ese usuario, vuelve a \
                      introducir el usuario que deseas scrapear: ')
   encontrado = search_user(str(user_name))
print("Hemos encontrado el usuario, vamos a proceder con la extracción")
```

Una vez encontrado el usuario que vamos a scrapear, llamaremos a los siguientes métodos (ya explicados) para obtener información general del perfil, crear la carpeta donde guardaremos toda la información que guardaremos y, dentro de esta, crearemos el fichero 'porfile_info.txt'.

```
number_posts = get_posts()
number_followers = get_followers()
number_following = get_following()
profile_name = get_name()
profile_role = get_role()
profile_descrip = get_descrip()
profile_link = get_link()
ruta_general = create_general_folder(user_name)

save_profile_info(number_posts, number_followers, number_following,
                  profile_name, profile_role, profile_descrip, profile_link)
```

A continuación, llamaremos al método scroll_posts_images() para obtener la lista con los enlaces a todas las publicaciones del perfil y pediremos que se responda por teclado si se desea descargar las imágenes que hemos encontrado. En caso de respuesta afirmativa, pasaremos a ejecutar los métodos que crea el directorio para las imágenes y la descarga de dichas imágenes. Además, en esta fase también crearemos el archivo 'url_of_posts.txt' y escribiremos allí toda la información que hemos extraído de cada publicación.

```
print("Estamos recorriendo el perfil mientras recogemos información")
posts = scroll_posts_images()
print("Ademaś, ¿también deseas descargear las imágenes?(si/no):", end=" ")
cond_imagenes = input()
while cond_imagenes != 'si' and cond_imagenes != 'no':
   cond_imagenes = input("Parece que no has introducido bien la resuesta,\
                          ¿deseas descargear las imágenes?(si/no): ")
if cond_imagenes == 'si': print("A conticuación, descargaremos las imagenes")
```

```
else: print('No descargaremos las imagenes')
(images, likes, coments) = get_images(posts, cond_imagenes)
posts_txt(user_name, posts, likes, coments)
print('La ruta absoluta del directorio que hemos creado para guardar la \
      informacion es: ',ruta_general)
if cond_imagenes == 'si':
   (full_path, folder_path) = create_posts_folder(user_name)
   print('La ruta absoluta del directorio que hemos creado para guardar\
         las imagenes es: ', full_path)
   download_images(images, folder_path)
print('\nLA EXTRACCIÓN HA FINALIZADO CON ÉXITO, HASTA LA PRÓXIMA')
```

TUTORIAL DE USO

Para poder usar este programa es necesario haber completado la instalación correctamente,

Después, deberemos de configurar los siguientes parámetros del archivo 'myextractor.py' para que al ejecutarlo funcione correctamente. Son 3 strings que se encuentran al principio del programa principal y se corresponden los las variables PATH, usuario y contra:

```
PATH = "/home/nicolas/PycharmProjects/Instagram-Scraper2/chromedriver"
usuario = 'MI USUARIO'
contra = 'MI CONTRASEÑA'
```

- PATH: Se corresponde con la ruta absoluta del fichero chromedriver que hemos descargado previamente.
- usuario: Se corresponde con el nombre de usuario con el que deseamos logearnos en instagram. A través de este usuario realizaremos la extracción. Por tanto, es importante que la cuenta que deseamos scrapear sea accesible, es decir, no esté bloqueada o en caso que la cuenta que queremos scrapear sea privada, debemos de seguirla.
- contra: Se corresponde con la contraseña del usuario anterior

Una vez hecho esto, ya podremos ejecutar el programa. Ejecutando en el terminal:

$ python3 'ruta de myextractor.py'

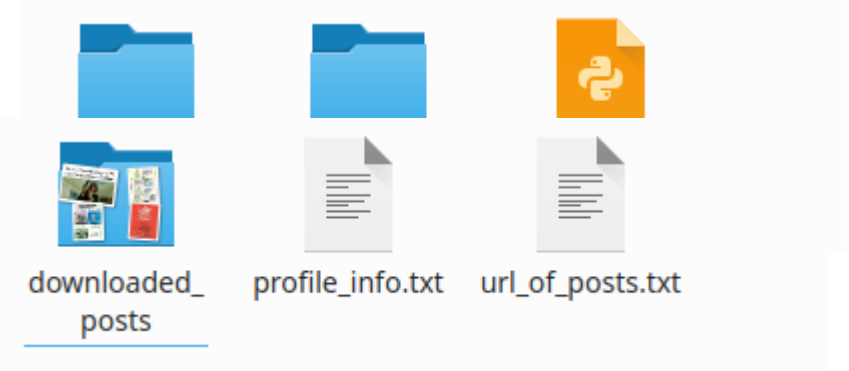
Se comenzará a ejecutar el programa y se pedirá en algunas fases que se introduzcan datos por teclado como es el nombre de perfil que deseamos scrapear o si deseamos o no descargar las imágenes. Un ejemplo de la ejecución del programa es el siguiente:

```
nicolas@nicolas:~/PycharmProjects/Instagram_Scraper$ python3 myextractor.py
BIENVENIDO AL EXTRACTOR DE INSTAGRAM
Mediante este programa vas a poder obtener información general de un perfil de ususario y además información de cada post del usuario
Ahora introduce el usuario al que vamos a realizar la extracción: tips.programadores
Hemos encontrado el usuario, vamos a proceder con la extracción
Estamos recorriendo el perfil mientras recogemos información
Además, ¿también deseas descargear las imágenes?(si/no): si
A conticuación, descargaremos las imagenes
La ruta absoluta del directorio que hemos creado para guardar la informacion es:  /home/nicolas/PycharmProjects/Instagram_Scrapertips.p
rogramadores
La ruta absoluta del directorio que hemos creado para guardar las imagenes es:  /home/nicolas/PycharmProjects/Instagram_Scrapertips.pro
gramadores/downloaded_posts
```

Y como resultado de la extracción, en el directorio donde tenemos el extractor encontraremos un nuevo directorio con el nombre de la cuenta que acabamos de scrapear:

Y dentro de este directorio encontraremos los ficheros y directorios que ya hemos mencionado:

Finalmente cabe destacar que, realizando este programa he averiguado que la cantidad de publicaciones que tiene una cuenta de instagram, en perfiles con más de 50 publicaciones no suele coincidir con la cantidad real de publicaciones. Es por esto que



puede que la cantidad de posts que nos saldrá en el fichero profile_info.txt no coincida con la cantidad de posts e imágenes que obtengamos.

BIBLIOGRAFÍA

[1] Victor Manuel Soto Morales, 'Descubre las funcionalidades de Selenium Webdriver'. Disponible: https://www.pragma.com.co/blog/descubre-las-funcionalidades-de-selenium-webdriver
[2] Gilberto Sanchez Mares, 'Selenium WebDriver en un Ambiente de Pruebas Continuas'. Disponible: https://sg.com.mx/revista/54/selenium-webdriver-un-ambiente-pruebas-continuas
[3] Arry M. Lani Anhar, 'Web Scraping Instagram with Selenium Python'. Disponible: https://medium.com/analytics-vidhya/web-scraping-instagram-with-selenium-python-b8e77af32ad4 (método scroll)
[4] Sancho Parameswara, 'Scraping Like Data In Instagram Post Using Python'. Disponible: https://blog.devgenius.io/scraping-like-data-in-instagram-post-using-python-6072bb2ef6b8
[5] Python Simplified, 'Web Scraping Instagram with Selenium'. Disponible: https://www.youtube.com/watch?v=iJGvYBH9mcY
[6] HKN MZ, 'How to Scrape Everything From Instagram Using Python', Disponible: https://python.plainenglish.io/scrape-everythings-from-instagram-using-python-39b5a8baf2e5
[7] David Amos, 'A Practical Introduction to Web Scraping in Python'. Disponible: https://realpython.com/python-web-scraping-practical-introduction/