Proyecto 1: Plataforma de seguimiento de datos COVID-19 para Colombia

Laura Camila Blanco Gómez
Escuela de Ciencias Exactas e Ingeniería
Universidad Sergio Arboleda - Bogotá, Colombia
laura.blanco01@correo.usa.edu.co

Santiago Cáceres Linares
Escuela de Ciencias Exactas e Ingeniería
Universidad Sergio Arboleda - Bogotá, Colombia
santiago.caceres01@correo.usa.edu.co

Andrés Camilo López Ramírez
Escuela de Ciencias Exactas e Ingeniería
Universidad Sergio Arboleda - Bogotá, Colombia
andres.lopez01@correo.usa.edu.co

Resumen

En el presente proyecto se busca extraer diversos datos sobre el virus COVID-19 de una página web, guardarlos en una base de datos y representar de forma selectiva mediante gráficos de tortas, gráficos de dos dimensiones y diagramas de barras de los datos que se han extraidos de la página web.

Palabras cláve:

COVID19, Colombia, Web Scraping, Python, html.

1. Marco teórico

1.1. COVID19

El COVID-19 es la enfermedad infecciosa causada por el coronavirus que se ha descubierto más recientemente y afecta a muchos países de todo el mundo.Razon por la cual fue ´ clasificado como Pandemia por la Organizacion Mundial de la Salud (OMS).

La mayoría de las personas (alrededor del 80 %) se recuperan de la enfermedad sin necesidad de tratamiento hospitalario. Alrededor de 1 de cada 5 personas que contraen la COVID-19 acaba presentando un cuadro grave y experimenta dificultades para respirar.

Actualmente en Colombia se cuentan con 818.203 casos confirmados, de los cuales 68.308 son casos activos, 722.536 personas recuperadas y 25.641 fallecidos.

1.2. Web scraping

Es una técnica que permite extraer datos e información de una web. Este tutorial es una guía de inicio al web scraping con Python, utilizando para ello la librería Beautiful Soup.

Beautiful Soup es una librería Python que permite extraer información de contenido en formato HTML o XML.

Se deben seguir los siguentes pasos para un buen funcionamiento del web scraping:

- Identifica los elementos de la página de los que extraer la información
- Descarga el contenido de la página
- Crear la «sopa»
- Busca los elementos en la «sopa» y obtén la información deseada

2. Resultados

Para realizar la extracción de datos se usa la técnica **Web Scraping**, con la cual se extrajo información desde **Wikipedia**, ya que cuenta con tablas que muestran los valores de contagios, muertes, supervivencia y casos activos del COVID-19, estos se actualizan constantemente. Para esto se hizo uso de las librerías de **BeautifulSoup** y **requests**.

```
from bs4 import BeautifulSoup
import requests
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

url = 'https://es.wikipedia.org/wiki/Pandemia_de_enfermedad_por_coronavirus_de_2020_en_Colombia'
ruta = "/Users/andre/DatosCovid.CSV"

page_response = requests.get(url, timeout=5)
page_content = BeautifulSoup(page_response.content, "html.parser")
ciudadtxt = list()
numerosArray = list()

tablaCol = page_content.find(
    "table", attrs={"class": "wikitable sortable collizq"})
ciudadRows = tablaCol.find_all("tr")

for row in ciudadRows:
    numerostd = row.find_all("td")[1:]
    for i in numerostd:
        numerosArray.append(sinEspacios(i.text))
    ciudadLink = row.find_all("a")
    for i in ciudadLink:
        ciudadtxt.append(i.text)
```

Listing 1: Código del web scraping

Después de hacer la extración de datos se hace una "división"de esta información que se encuentra en **numerosArray**, esto se hizo por medio de varias listas como se ve en el siguiente código, para después guardar toda esta información de un **DataFrame**

```

for i in range(0, 330, 10):
    total.append(numerosArray[i])
total.reverse()

for i in range(1, 330, 10):
    cxMhad.append(numerosArray[i])
cxMhad.reverse()

for i in range(2, 330, 10):
    muertesT.append(numerosArray[i])
muertesT.reverse()

for i in range(3, 330, 10):
    porjeMT.append(numerosArray[i])
porjeMT.reverse()

for i in range(4, 330, 10):
    falleMhab.append(numerosArray[i])
falleMhab.reverse()

for i in range(5, 330, 10):
    recuT.append(numerosArray[i])
recuT.reverse()

for i in range(6, 330, 10):
    porjeRT.append(numerosArray[i])
porjeRT.reverse()

for i in range(7, 330, 10):
    CasActT.append(numerosArray[i])
CasActT.reverse()

for i in range(8, 330, 10):
    porjeACT.append(numerosArray[i])
porjeACT.reverse()

for i in range(9, 330, 10):
```

```
    AxMhab.append(numerosArray[i])
AxMhab.reverse()


df = pd.DataFrame({'Nombre': ciudadtxt, 'Casos Confirmados Totales': total, ...
                   'Casos por millon de habitantes': cxMhad,
                   'Muertes Totales': muertesT, 'Porcentaje de muertes': porjeMT, 'Fallecido por ...
                       millon de habitantes': falleMhab,
                   'Recuperados Totales': recuT, 'Porcentaje de recuperados': porjeRT, 'Casos ...
                       Activos Totales': CasActT,
                   'Porecentaje Casos Activos': porjeACT, 'Casos Activos por millon de ...
                       habitantes': AxMhab})
```

Listing 2: Código división de información y DataFrame

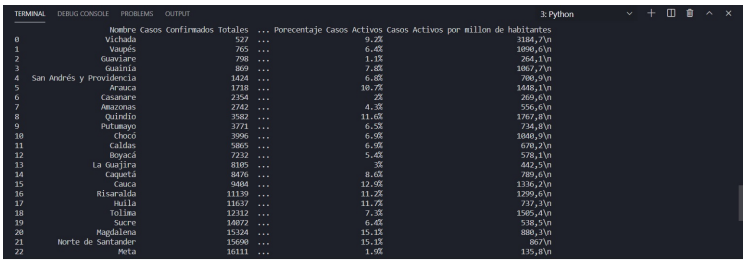

Figura 1: DataFrame

Para enviar los datos a la base de datos creamos una base de datos en MySQL, y desde Python se le enviaran los datos que debe crear en este caso en forma de tabla y definiendo el tipo de valor que se le envio y esta se llenara a partir de un for con los datos que se exportaron inicialmente desde la pagina web.

```
#Nombre de las columnas
Ciudad=flist[0][0]; Total_Casos_Confirmados=flist[0][1]; CasosxMillon=flist[0][2];  ...
    Total_Fallecimientos=flist[0][3]; Porcentaje_Fallecidos=flist[0][4];  ...
    FallecidosxMillon=flist[0][5]; Total_Recuperados=flist[0][6];  ...
    Porcentaje_Recuperados=flist[0][7]; Casos_Activos=flist[0][8];  ...
    Porecentaje_CasosActivos=flist[0][9]; Casos_ActivosxMillon=flist[0][10]

queryCreateTable = """create table datoscovid(
                      {} varchar (50),
                      {} double,
                      {} double,
                      {} double,
                      {} varchar (50),
                      {} double,
                      {} double,
                      {} varchar (50),
                      {} double,
                      {} varchar (50),
                      {} double
                      )""".format(Ciudad, Total_Casos_Confirmados, CasosxMillon, ...
                          Total_Fallecimientos, Porcentaje_Fallecidos, FallecidosxMillon, ...
                          Total_Recuperados, Porcentaje_Recuperados, Casos_Activos, ...
                          Porecentaje_CasosActivos, Casos_ActivosxMillon)

cursor.execute(queryCreateTable)
del flist[0]
rows = ''
for i in range(len(flist)-1):
    rows += "('{}', '{}', '{}', '{}','{}', '{}', ...
        '{}','{}','{}','{}','{}')".format(flist[i][0],flist[i][1],flist[i][2],flist[i][3],
                                          flist[i][4],flist[i][5],flist[i][6],flist[i][7],
                                          flist[i][8],flist[i][9],flist[i][10],)
    if i != len(flist) - 2:
        rows +=','
dataInsert = "insert into datoscovid values" + rows

```

```
try:
    cursor.execute(dataInsert)
    db.commit()
    print("Inserci n de datos exitosa!")
except:
    print("Error en conexi n :c")
    db.rollback()

db.close()
```

Listing 3: Código base de datos

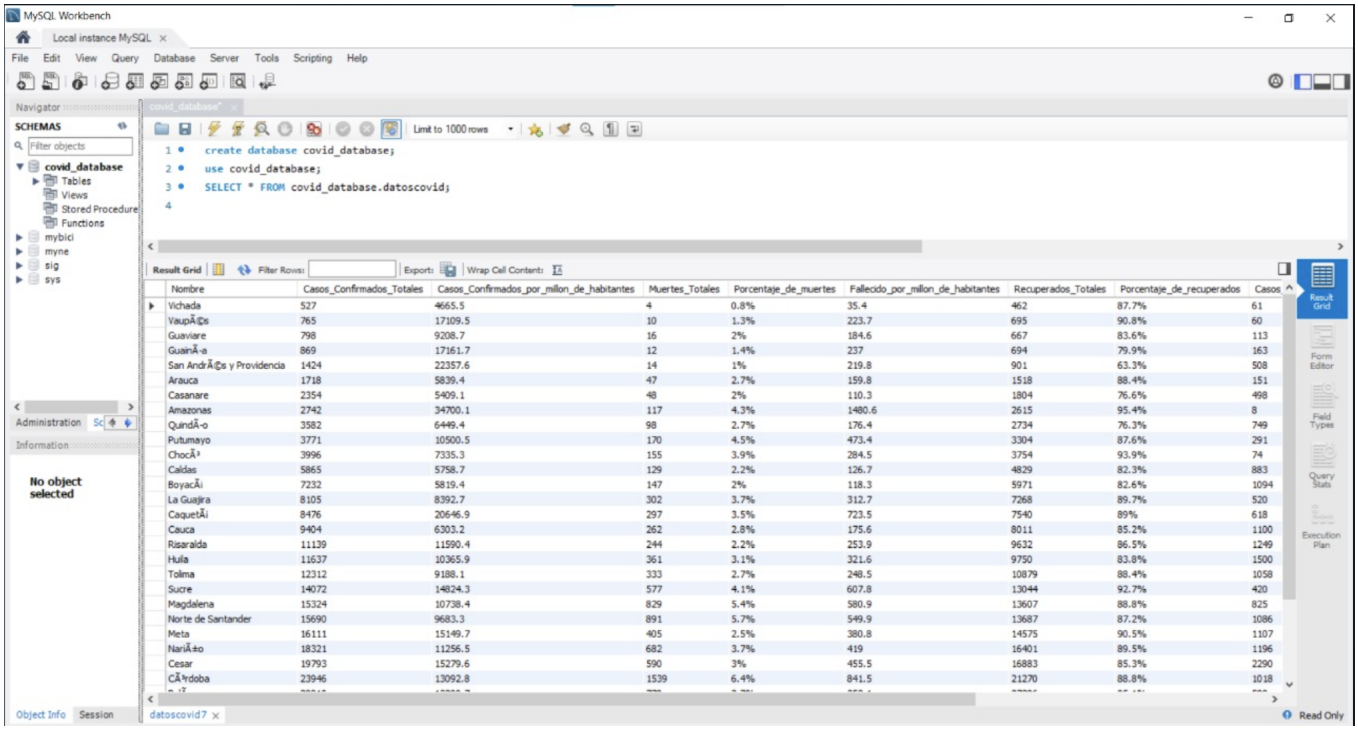

Figura 2: Base de datos

Para representar la información extraída en graficas usamos el DataFrame para seleccionar los datos que queremos graficar y se guardan en variables *x*. y *y*. Además se hace uso de la librería **matplotlib** para definir el tipo de grafica, en este caso es *plt.pie()*.

```
# Grafica Barras
x = df['Nombre'].iloc[28:33]
y = df['Casos Confirmados Totales'].iloc[28:33]
plt.bar(x, y)
plt.xlabel('Departamenro')
plt.ylabel('Casos Confirmados Totales')
plt.grid()
plt.show()
```

Listing 4: Código gráfica de barras

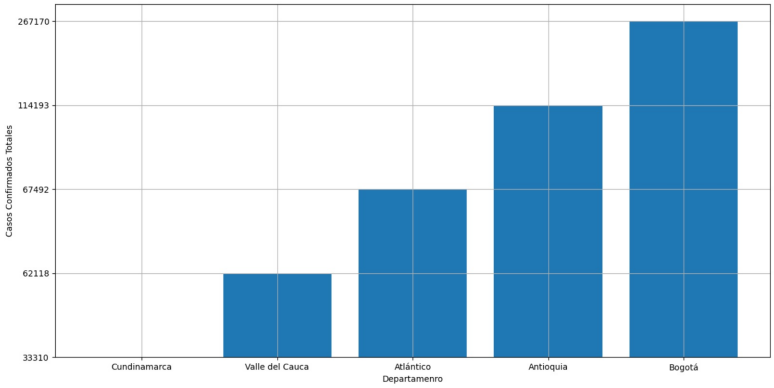

Figura 3: Grafico de barras

En el caso de la gráfica de torta se hace uso del **plt.pie()** en el cual se encontrara *autopct = "%0.1f%%"* el cual servirá para convertir los valores en porcentajes, aclarando que realizara esta conversián únicamente con los datos que se grafiquen.

```
#Grafico Torta
x = df['Muertes Totales'].iloc[28:33]
y = df['Nombre'].iloc[28:33]
plt.title('Muertes Totales')
plt.pie(x,labels= y, autopct = "%0.1f%%  )
plt.show()
```

Listing 5: Código gráfica de torta

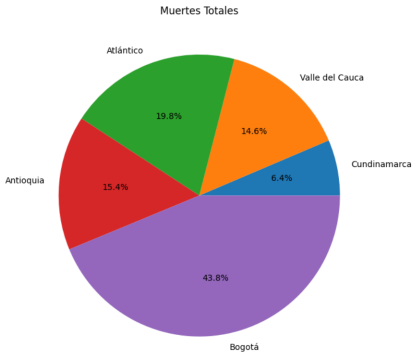

Figura 4: Grafico de torta

Y por ultimo en el grafico 2D se usa **plt.plot()** y en nuestro caso hacemos uso de **plt.grid** para que se puedan apreciar las intersecciones de los valores en la recta.

```
#Grafico 2D
x = df['Muertes Totales'].iloc[28:33]
y = df['Nombre'].iloc[28:33]
plt.plot(x,y)
plt.xlabel('Casos Confirmados Totales')
plt.ylabel('Departamenro')
plt.grid()
plt.show()
```

Listing 6: Código gráfica 2D

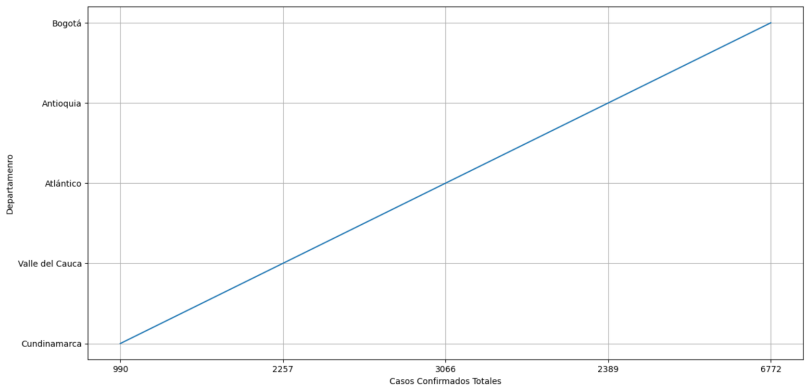

Figura 5: Grafico 2D

3. Conclusiones

- Conociendo la técnica de web scraping se facilito para la extracción desde la página web a Python.
- Se utilizan las librerías de Python Pandas, beautifulsoap y request, ya que nos permiten crear DataFrames y archivos CSV, buscar información y buscar información del html respectivamente.
- Se hace necesario a la hora de extraer los datos buscar páginas confiables que mantengan sus datos actualizados, ya que con estre se tendra un panorama más realista de lo que esta pasando.
- Gracias al uso de los **DataFrame** se pudo almacenar de una manera facíl la información extraida para despues poder llevarla a un archivo CSV con el cual se lleno la base de datos posteriormente

4. Link repositorio

https://github.com/ACLXRD/SA2020_G02_CovidScraper_6I3I11

Referencias

[1] OMS, *Preguntas y respuestas sobre la enfermedad por coronavirus (COVID-19)*. Organización mundial de la salud, 2020.

[2] J. J. L. Gómez, *Web scraping con Python. Extraer datos de una web. Guía de inicio de Beautiful Soup*. J2LOGO, 2018.

[3] I. nacional de salud, *COVID-19 en Colombia*. MinSalud, 2020.

[4] W. L. enciclopedia libre, *Pandemia de enfermedad por coronavirus de 2020 en Colombia*. Wikipedia, 2020.