

Plataforma de seguimiento de datos COVID-19 para Colombia

Juan Sebastian Sierra

Escuela de Ciencias Exactas e Ingeniería
Universidad Sergio Arboleda - Bogotá, Colombia
juan.sierra01@correo.usa.edu.co

Leyder Jesus Pacheco

Escuela de Ciencias Exactas e Ingeniería
Universidad Sergio Arboleda - Bogotá, Colombia
leyder.pacheco01@correo.usa.edu.co

Luis Felipe Velasquez

Escuela de Ciencias Exactas e Ingeniería
Universidad Sergio Arboleda - Bogotá, Colombia
luis.velasquez01@correo.usa.edu.co

Resumen

En el presente laboratorio se recolectaran los datos de Covid-19 para Colombia y se mostraran estos en diferentes diagramas para entender el comportamiento en el país y en los ciudadanos colombianos, mediante python y sus diferentes librerías.

Palabras clave:

Python, Covid-19, Datos, OpenData.

1. Marco teórico

En esta pandemia, los datos han sido los grandes protagonistas. La crisis ha resaltado la importancia de generar estadísticas oficiales oportunas y confiables para monitorear el avance de la enfermedad, detectar grupos vulnerables, medir el impacto de las políticas de aislamiento en la vida de las personas y en la economía, y proyectar las necesidades a futuro.

En el informe se buscará obtener todos los datos posibles acerca de el Covid-19 en Colombia y como este ha ido evolucionando a través del tiempo, esto se hará mediante gráficas de diferentes tipos para lograr así un seguimiento en diferentes ámbitos del país.

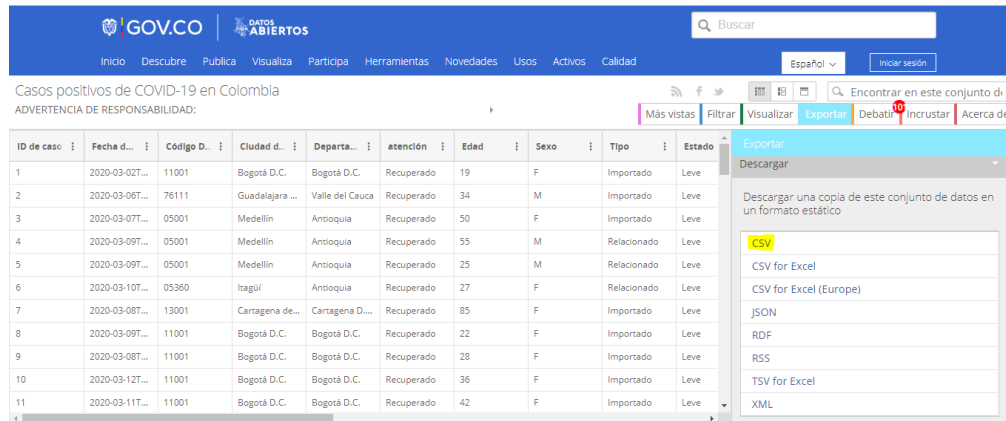
Inicialmente necesitamos obtener los datos de una fuente confiable y libre, en este caso elegimos datos.gov.co la cual es la pagina de los datos abiertos de Colombia que facilitan la descarga de los datos.

Estas son :

1. **pandas:** El cual es un paquete de Python que proporciona estructuras de datos similares a los dataframes de R. Pandas depende de Numpy, la librería que añade un potente tipo matricial a Python. Pandas proporciona herramientas que permiten: [1].
 - leer y escribir datos en diferentes formatos: CSV, Microsoft Excel, bases SQL y formato HDF5
 - seleccionar y filtrar de manera sencilla tablas de datos en función de posición, valor o etiquetas
 - fusionar y unir datos
 - transformar datos aplicando funciones tanto en global como por ventanas
 - manipulación de series temporales
 - hacer gráficas
2. **Data frames:** Los data frames son estructuras de datos de dos dimensiones (rectangulares) que pueden contener datos de diferentes tipos, por lo tanto, son heterogéneas. Esta estructura de datos es la más usada para realizar análisis de datos [2].
3. **Numpy:** Es una librería en la que se define un tipo de dato que representa matrices multidimensionales [3].
4. **Matplotlib:** Matplotlib es una librería que permite hacer gráficos. Se puede utilizar junto a Numpy o independientemente. Los gráficos pueden ser guardados en ficheros (png, svg, pdf, etc.) o ser utilizados interactivamente. El módulo principal de uso de matplotlib es pyplot [4].
5. **SQLite:** Es una base de datos server-less que se puede utilizar en casi todos los lenguajes de programación, incluido Python [5].

2. Procedimiento

Para la extracción de los Datos del Covid-19 en Colombia, se obtuvo en la página web de datos abiertos de Colombia en www.datos.gov.co donde se encuentran los datos en una plantilla de celdas y de allí se pudo obtener un link que hace referencia a un archivo .csv que contiene todos los datos necesitados



ID de caso	Fecha d...	Código D...	Ciudad d...	Departa...	atención	Edad	Sexo	Tipo	Estado
1	2020-03-02T...	11001	Bogotá D.C.	Bogotá D.C.	Recuperado	19	F	Importado	Leve
2	2020-03-06T...	76111	Guadalajara ...	Valle del Cauca	Recuperado	34	M	Importado	Leve
3	2020-03-07T...	05001	Medellín	Antioquia	Recuperado	50	F	Importado	Leve
4	2020-03-09T...	05001	Medellín	Antioquia	Recuperado	55	M	Relacionado	Leve
5	2020-03-09T...	05001	Medellín	Antioquia	Recuperado	25	M	Relacionado	Leve
6	2020-03-10T...	05360	Itagüí	Antioquia	Recuperado	27	F	Relacionado	Leve
7	2020-03-08T...	13001	Cartagena de...	Cartagena D...	Recuperado	85	F	Importado	Leve
8	2020-03-09T...	11001	Bogotá D.C.	Bogotá D.C.	Recuperado	22	F	Importado	Leve
9	2020-03-08T...	11001	Bogotá D.C.	Bogotá D.C.	Recuperado	28	F	Importado	Leve
10	2020-03-12T...	11001	Bogotá D.C.	Bogotá D.C.	Recuperado	36	F	Importado	Leve
11	2020-03-11T...	11001	Bogotá D.C.	Bogotá D.C.	Recuperado	42	F	Importado	Leve

Figura 1: Extracción de datos del Covid-19 en Colombia

La implementación se desarrolló en código python y utilizando su base de datos por defecto mysqlite, además se utilizaron dos archivos de python, llamados main.py y myutils.py

- **Importación de librerías**

Como primer paso se importó las siguientes librerías las cuales facilitan y ayudan en el proceso

```
1 import os
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import matplotlib.ticker as ticker
5 import numpy as np
```

Código 1: Librerías utilizadas para la implementación

- **myutils.py** La siguiente sección contiene los códigos utilizados en el archivo myutils.py

Se descargan los datos con una url e formato CSV

```
1 def download_data(url):
2     """
3     Descarga los datos en formato csv
4     """
5     return pd.read_csv(url, low_memory=False)
```

Código 2: Descarga de datos en formato CSV

Conexión a la base de datos SQLITE, la cual está creada como un archivo.bd

```
1 def connect_db(database_name):
2     """
3     Crea una conexión con la base de datos
4     In: database_name - nombre de la base de datos
5     Out: conn - conexión, objeto que representa la base de datos.
6     """
7     conn = None
```

```

8     try:
9         conn = db.connect(database_name)
10        print(f'database connected {db.version}')
11        return conn
12    except Error as e:
13        print(e)
14        exit(1)

```

Código 3: Creación de conexión a la base de datos

En esta sección de código se implementó una función que realice un query y retorne su resultado en un DataFrame

```

1 def make_query(query_statement, conn):
2     return pd.read_sql(query_statement, conn)

```

Código 4: Funcion del desarrollo de querys

■ main.py

La siguiente sección contiene los códigos utilizado en el archivo main.py, la cual invoca a las funciones de myutils.py para conectarse a la base de datos, además de generar las gráficas de estadísticas del Covid-19 en Colombia

En esta función se actualizan los datos en caso de que el usuario lo desee, esto lo realiza con el mismo url y la conexión a la base de datos, luego se almacenara en la misma

```

1 def update_database(csv_url, conn):
2     data = download_data(csv_url)
3     print('Datos descargados ...')
4     data = rename_columns(data)
5     save_data(data, table_name, conn)
6     print('Datos almacenados en la base de datos ...')

```

Código 5: Actualización de datos Covid-19

En esta función se valida si una tabla existe en la base de datos y se desarrolla mediante un query que devuelve la respuesta

```

1 def exists_table(conn, table_name):
2     tables = make_query(
3         f"SELECT name FROM sqlite_master WHERE type='table'", conn)
4     return tables['name'].str.contains(table_name).any()

```

Código 6: Validación si una tabla existe en la BD

En esta sección, se invoca a las funciones que grafican las estadísticas del Covid-19 donde recibe y envía a cada una de ellas a conexión y el nombre de la tabla de la base de datos

```

1 def plot_data(conn, table_name):
2     print('Generando las graficas, por favor espere ...')
3     torta_por_genero(conn, table_name)
4     muertos_por_Depto(conn, table_name)
5     activos_por_Depto(conn, table_name)
6     recuperados_por_Depto(conn, table_name)
7     contagiados_por_edad(conn, table_name)
8     torta_por_Tipo_Contagio(conn, table_name)

```

Código 7: Invocación a las funciones graficadoras

En el siguiente código se muestran las funciones que dan como resultado una gráfica por cada una de ellas, donde se desarrollaron queries y el resultado del mismo se gráfico mediante la librería matplotlib:

```

1
2
3 def diarios_comparacion(conn, table_name):
4     datainf = make_query(
5         f"SELECT strftime('%m-%d', f_notificacion) as mes_not, count(*) as cantidad FROM ...
           {table_name} GROUP BY mes_not ORDER BY mes_not", conn)
6     datarec = make_query(
7         f"SELECT strftime('%m-%d', f_recuperado) as mes_not, count(*) as cantidad FROM ...
           {table_name} WHERE mes_not IS NOT NULL GROUP BY mes_not ORDER BY mes_not", conn)
8     datamuer = make_query(
9         f"SELECT strftime('%m-%d', f_muerte) as mes_not, count(*) as cantidad FROM ...
           {table_name} WHERE mes_not IS NOT NULL GROUP BY mes_not ORDER BY mes_not", conn)
10
11     xinf = datainf['mes_not']
12     yinf = datainf['cantidad']
13     xrec = datarec['mes_not']
14     yrec = datarec['cantidad']
15     xmuer = datamuer['mes_not']
16     ymuer = datamuer['cantidad']
17
18     fig, ax = plt.subplots()
19     plt.title('Curva de contagiados en el tiempo(diario)')
20     ax.plot(xinf, yinf)
21     ax.plot(xrec, yrec)
22     ax.plot(xmuer, ymuer)
23     plt.legend(['Contagiados', 'Recuperados', 'Muertos'])
24     plt.title(
25         'Comparacion contagios, recuperados y muertos en el tiempo(diarios)')
26     # this locator puts ticks at regular intervals
27     loc = ticker.MultipleLocator(base=12)
28     ax.xaxis.set_major_locator(loc)
29     plt.show()
30
31     xinf = datainf['mes_not']
32     yinf = datainf['cantidad']
33     xrec = datarec['mes_not']
34     yrec = datarec['cantidad']
35     plt.title('Curva de contagiados')
36     fig, ax = plt.subplots()
37     ax.plot(xinf, yinf)
38     # this locator puts ticks at regular intervals
39     loc = ticker.MultipleLocator(base=12)
40     ax.xaxis.set_major_locator(loc)
41     plt.show()
42
43
44 def torta_por_genero(conn, table_name):
45     data = make_query(
46         f"SELECT sexo, count(sexo) as cantidad FROM {table_name} WHERE sexo = 'F' OR sexo ...
           = 'M' GROUP BY sexo", conn)
47     y = data['cantidad']
48     plt.figure()
49     plt.title('Procentaje de contagiados por sexo')
50     plt.pie(y,
51             labels=['Femenino', 'masculino'],
52             autopct='%1.1f%%',
53             shadow=True)
54     plt.show()
55
56
57 def muertos_por_Depto(conn, table_name):
58     data = make_query(
59         f"select departamento ,count(*) as total from {table_name} WHERE ...
           atencion='Fallecido' GROUP BY departamento ORDER BY total", conn)
60     dept = data['departamento']
61     cont = data['total']
62     plt.figure()
63     plt.title('Diez Departamentos con mas Fallecimientos')
64     plt.barh(dept[-10:], cont[-10:])
65     plt.show()
66

```

```

67
68 def activos_por_Depto(conn, table_name):
69     data = make_query(
70         f"select departamento , count(*) as total from {table_name} GROUP BY departamento ...
            ORDER BY total", conn)
71     dept = data['departamento']
72     cont = data['total']
73     plt.figure()
74     plt.title('Diez Departamentos con mas Contagiados')
75     plt.barh(dept[-10:], cont[-10:])
76     plt.show()
77
78
79 def recuperados_por_Depto(conn, table_name):
80     data = make_query(
81         f"select departamento ,count(*) as total from {table_name} WHERE ...
            atencion='Recuperado' GROUP BY departamento ORDER BY total", conn)
82     dept = data['departamento']
83     cont = data['total']
84     plt.figure()
85     plt.title('Diez Departamentos con mas recuperados')
86     plt.barh(dept[-10:], cont[-10:])
87     plt.show()
88
89
90 def contagiados_por_edad(conn, table_name):
91     # Dispersion
92     data = make_query(
93         f"SELECT edad ,count(*) as total FROM {table_name} GROUP BY Edad", conn)
94     fig, ax = plt.subplots()
95     plt.title('Dispersi n por edad de infectados')
96     plt.xlabel('Edad(a os)')
97     plt.ylabel('Infectados')
98     ax.scatter(data['edad'], data['total'])
99     plt.show()
100
101
102 def torta_por_Tipo_Contagio(conn, table_name):
103     data = make_query(
104         f"SELECT atencion,count(*) as cantidad FROM {table_name} where atencion ...
            !='Recuperado' and atencion !='Fallecido' GROUP BY atencion", conn)
105     y = data['cantidad']
106     plt.figure()
107     plt.title('Porcentaje de atenci n de contagiados')
108     plt.pie(y,
109             labels=['', 'En casa', 'Hospital', 'UCI'],
110             autopct='%1.1f%%',
111             shadow=True)
112     plt.show()

```

Código 8: Funciones que muestran las estadísticas del Covid-19

El contenido de la siguiente función es el que va a ejecutar el usuario, donde puede seleccionar si actualizar nuevamente los datos, o no hacerlo, además de invocar el método que muetsra la gráficas

```

1
2
3 def main(csv_url, database_name, table_name):
4     conn = connect_db(database_name)
5
6     if exists_table(conn, table_name):
7         print('Los datos ya estan en la base de datos')
8         y = input('Desea actualizar la base de datos? ([y]/[n]): ').lower()
9         if y == 'y':
10             print('Descargando los datos de Covid-19, por favor espere ...')
11             update_database(csv_url, conn)
12     else:
13         print('Descargando los datos de Covid-19, por favor espere ...')
14         update_database(csv_url, conn)

```

```

15
16     plot_data(conn, table_name)
17     close_db(conn)

```

Código 9: Funcion main del programa

```

1
2 def plot_data(conn, table_name):
3 if __name__ == '__main__':
4     direc_a = os.getcwd()
5     print(direc_a)
6     csv_url = "https://www.datos.gov.co/api/views/gt2j-8ykr/rows.csv?accessType=DOWNLOAD"
7     if 'codes' in direc_a:
8         database_name = "datasets/covid.db"
9     else:
10        database_name = "codes/datasets/covid.db"
11    table_name = "covidt"
12
13    main(csv_url, database_name, table_name)

```

Código 10: Datos iniciales de la url y archivo de la base de datos SQLITE

3. Resultados

A partir del procedimiento, se ejecuto el archivo python main.py, el cual descargo el archivo csv de la pagina de OpenData Colombia, los inserto en una base de datos SQLITE, realiza varias consultas y las gráficas dieron de a siguiente manera:

1. Gráfica que representa la curva de contagios por Covid-19

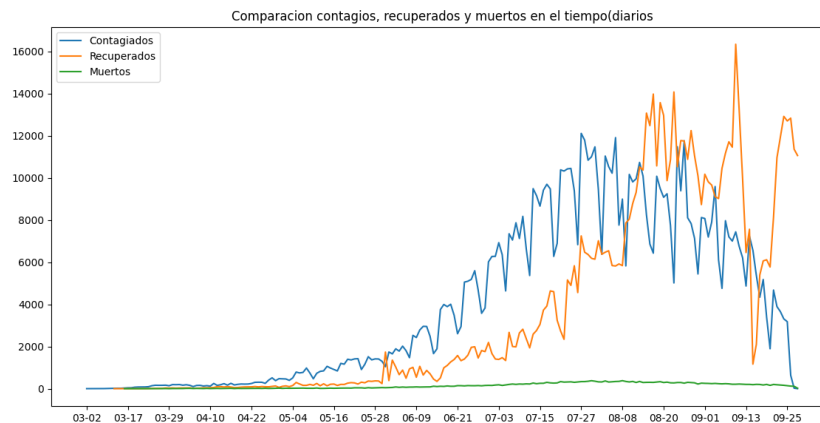


Figura 2: Extracción de datos del Covid-19 en Colombia

2. Gráfica de dispersión que demuestra los contagios de Covid-19 por edad

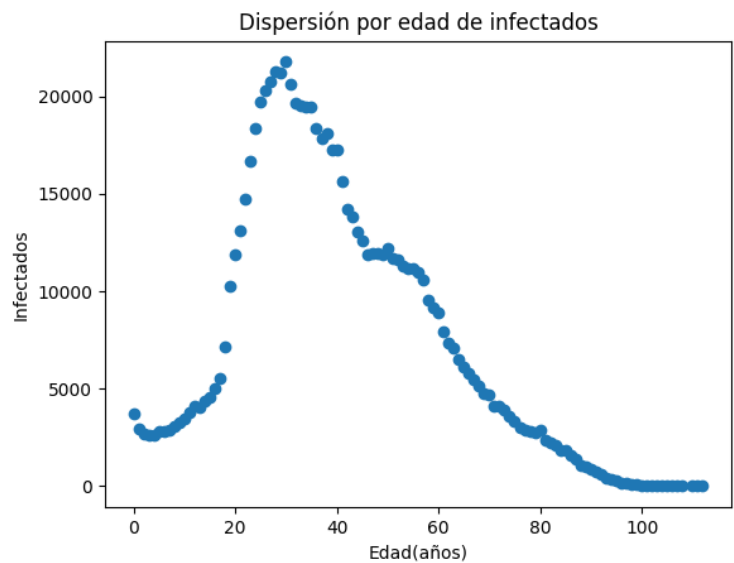


Figura 3: Extracción de datos del Covid-19 en Colombia

3. Gráfica de barras con los diez departamentos con mas contagios por covid-19:

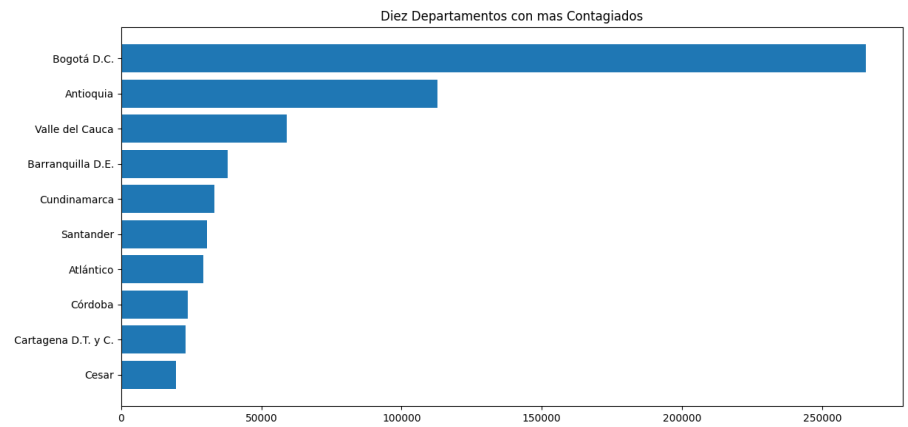


Figura 4: Extracción de datos del Covid-19 en Colombia

4. Gráfica de barras con los diez departamentos con mas presonas recuperadas por covid-19:

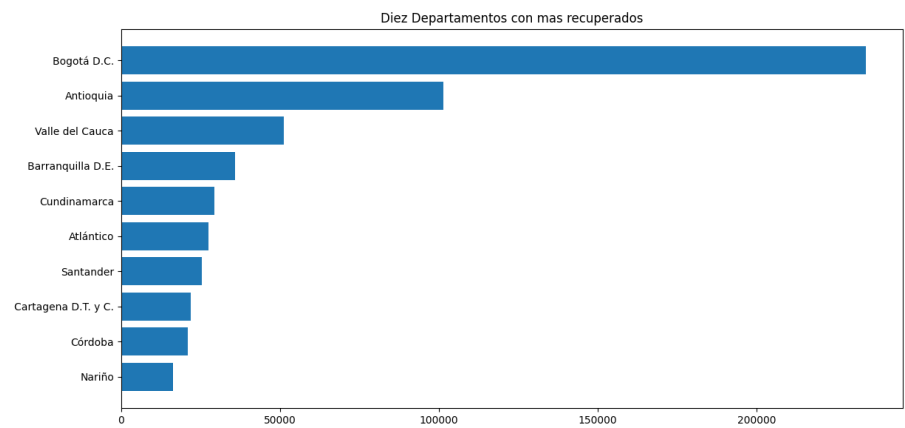


Figura 5: Extracción de datos del Covid-19 en Colombia

5. Gráfica de barras con los diez departamentos con mas fallecimientos por covid-19:

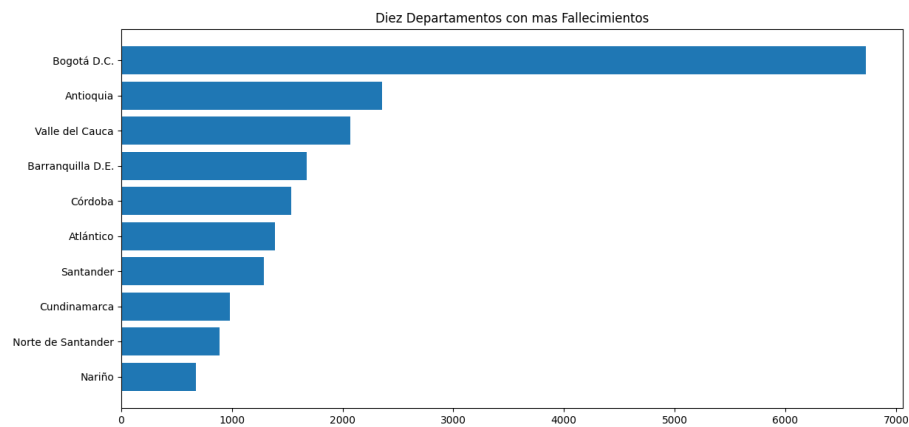


Figura 6: Extracción de datos del Covid-19 en Colombia

6. Gráfica Tipo Torta que especifica el porcentaje de contagiados por genero:

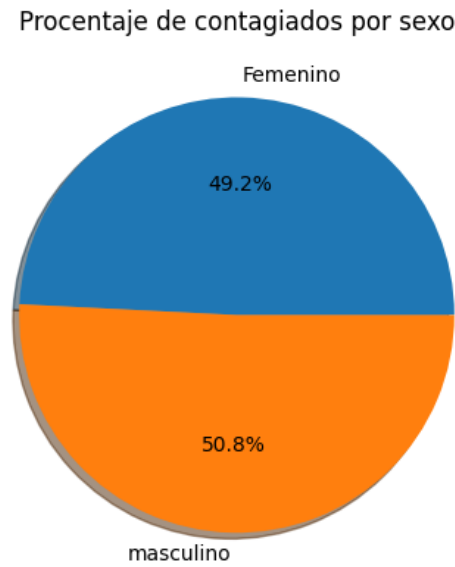


Figura 7: Extracción de datos del Covid-19 en Colombia

7. Gráfica Tipo Torta que especifica el porcentaje de cada una de los tipo de atención a las personas contagiadas

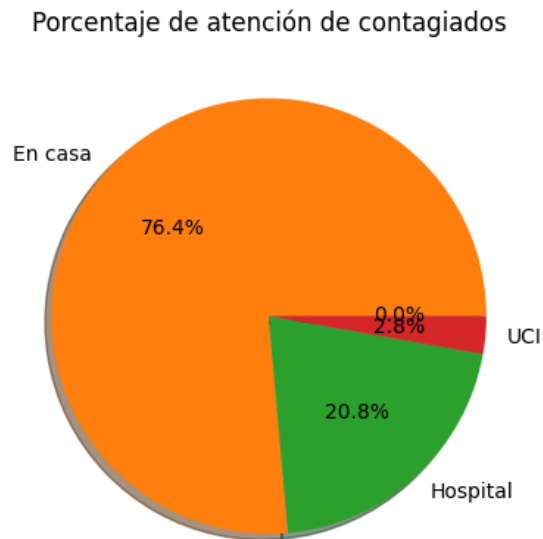


Figura 8: Extracción de datos del Covid-19 en Colombia

4. Conclusiones

Con esta primera parte del proyecto para la creación de una plataforma de seguimiento de datos COVID-19 para Colombia, se logro evidenciar la facilidad con la que se puede extraer datos del internet y poder convertirlos en información visible y útil para las personas. Además, con las tecnologías a disposición como el lenguaje de programación *Python*, el cual facilita y permite automatizar el proceso de extracción y representación de los datos en diferentes tipos de gráficas para tener el conocimiento de los datos de una manera mas clara y concisa.

Python cuenta con una gran cantidad de librerías a disposición que permiten realizar diversos programas, especialmente en la ciencia de datos, librerías como numpy, pandas, matplotlib, entre otras han permitido que la gran cantidad de datos que se tienen a disposición hoy en día.

Repositorio del proyecto

https://github.com/SierraJuanSe/COVID_tracker

5. Bibliografia

Referencias

- [1] Bioinf. Pandas python. [Online]. Available: <https://bioinf.comav.upv.es/courses/linux/python/pandas.html>
- [2] BookDown. Data frames. [Online]. Available: <https://bookdown.org/jboscomendoza/r-principiantes4/data-frames.html>
- [3] Deustoformacion. Data frames. [Online]. Available: <https://www.deustoformacion.com/blog/programacion-diseno-web/que-son-datasets-dataframes-big-data>
- [4] BioInf. Matplotlib. [Online]. Available: <https://bioinf.comav.upv.es/courses/linux/python/scipy.html#matplotlib>
- [5] LikeGeeks. Sqlite. [Online]. Available: <https://likegeeks.com/es/tutorial-de-python-sqlite3/>
- [6] BioInf. Numpy. [Online]. Available: <https://bioinf.comav.upv.es/courses/linux/python/scipy.html>