Proyecto: COVID-19.

Johan Sebastian Manjarres Diaz[1], Diego Alejandro Amado Rodriguez[2], Luis Alberto Velasquez Zea[3]
Escuela de Ciencias Exactas e Ingenierias, Ingenieria Electronica
Universidad Sergio Arboleda - Bogotá, Colombia
johan.manjarres@correo.usa.edu.co[1]
diego.amado01@correo.usa.edu.co[2]
luis.velasquez02@correo.usa.edu.co[3]

1. Objetivos.

- Desarrollar una aplicación para extraer los datos actualizados de COVID-19 en Colombia y Bogotá, de páginas de Internet y los guarde en una base de datos.
- Mantener informado al usuario sobre estadísticas de la actual pandemia COVID-19 en Colombia y Bogotá, por medio de visualización gráfica interactiva que brinde información suficiente y de fuentes confiables.
- Por medio de lenguaje de programación Python importar y hacer el procesamiento de datos para su manejo.
- La información suministrada al usuario deber ser entendible e interactiva para una mayor comprensión.

2. Diseño.

Libreria	Funcionamiento
`import matplotlib.pyplot as plt`	Es una colección de funciones de estilo comando que hacen que *matplotlib* funcione como MATLAB. Cada función *pyplot* realiza algún cambio en una figura: Por ejemplo, crea una figura, crea un área de trazado en una figura, traza algunas líneas en un área de trazado, decora el trazado con etiquetas, etc.
`import plotly.graph_objects as go`	`plotly.py` es una biblioteca de trazado interactiva de código abierto que admite más de 40 tipos de gráficos únicos que cubren una amplia gama de casos de uso estadísticos, financieros, geográficos, científicos y tridimensionales, en este caso `plotly.graph_objects` se usa para que los objetos gráficos se almacenen en una jerarquía de módulos.
`import plotly.express as px`	*Plotly Express* es la interfaz de alto nivel fácil de usar para Plotly, que opera con datos .ºrdenadosz produce figuras fáciles de diseñar. Cada función Plotly Express devuelve un objeto `graph_objects.Figure` cuya data y se layout se ha rellenado previamente de acuerdo con los argumentos proporcionados.
`from mpl_toolkits.mplot3d import Axes3D`	Los gráficos tridimensionales se habilitan importando el *mplot3dkit* de herramientas. Una vez que se importa este submódulo, se pueden crear ejes tridimensionales, a lo que se da soporte para los graficos y para utilizarlos, se debe importar la extensión para 3D, el objeto Axes3D.
`import matplotlib.cm as cm`	Se utiliza para realizar mapas de colores incorporados, utilidades de manejo de mapas de colores. Matplotlib tiene una serie de mapas de color incorporados accesibles a través de `matplotlib.cm.get_cmap`. También hay bibliotecas externas como palettable que tienen muchos mapas de color adicionales.
`import pandas as pd`	Pandas significa "Biblioteca de análisis de datos de Python; Pandas toma datos y crea un objeto Python con filas y columnas llamado *marco de datos* que se parece mucho a la tabla en un software estadístico, es más fácil trabajar con él en comparación con trabajar con listas y/o diccionarios a través de bucles o comprensión de listas.
`import numpy as np`	Numpy es la biblioteca central para la computación científica en Python. Proporciona un objeto de matriz multidimensional de alto rendimiento y herramientas para trabajar con estas matrices.
`from sodapy import Socrata`	Socrata alberga más de cien catálogos de datos diferentes para gobiernos, organizaciones sin fines de lucro y ONG de todo el mundo, por lo que es fácil encontrar un catálogo de datos abierto para trabajar.
`import folium`	Folium es una biblioteca de Python que nos permite visualizar datos espaciales de manera interactiva, directamente dentro del entorno de los cuadernos. La biblioteca es altamente intuitiva de usar y ofrece un alto grado de interactividad con una curva de aprendizaje baja.

Cuadro 1: Librerias usadas en el proyecto.

Mediante la creación de un *notebook* en Jupyter, se realizó la programación de los gráficos a partir de los datos obtenidos de la página del gobierno nacional Colombiano, descargados en archivos tipo `.csv` para ordenarlos en una base de datos en Excel. Utilizando y relacionando datos tales como, los números de casos de contagiados por departamentos, sexo, su estado actual y día del reporte de contagio, esto para su lectura mediante esquemas. Tambien mplementamos la plataforma spyder quien permitió la visualizacion de datos complementarios a jupyter.

3. Resultados.

Listing 1: Librerías.

```
#EXTRACCION
from sodapy import Socrata

#VISUALIZACION
import matplotlib.pyplot as plt
import plotly.graph_objects as go
import plotly.express as px
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.cm as cm,import math, import requests
from scipy.optimize import curve_fit
import datetime ,import folium
from folium import Choropleth, Circle, Marker
from folium.plugins import HeatMap, MarkerCluster

#MANEJO DE TABLAS Y DATOS
import pandas as pd, import numpy as np
```

Listing 2: Código para covertir excel y llamar el excel.

```
df_result.to_excel("data.xlsx",sheet_name="datos")
df=pd.read_excel('data.xlsx')
```

Listing 3: Codigo para generar nuevos datos

```
anastasio=np.arange(0,(len(b)),1), fechasan=[], an_tot=[]
deparn=[], anit=0, j=0, corte=0
for i in anastasio:
    newde=depart[corte]
    newfe=diagnostico[corte]
    anit=anit+b[i]
    an_tot.insert(i,anit)
    deparn.insert(i,newde)
    fechasan.insert(i,newfe)
    j+=1
    if j==lendep[corte]:
        anit=0, corte+=1, j=0
        if corte>34:
            corte=0

newfe=anim.index.values
```

A partir de los datos que se tienen en el archivo de Excel, se procede a realizar un manejo de estos datos, creando algunos nuevos y cambiarlo de formato de otros, consiguiendo asi la facilitación a la hora de realizar la visualización.

Listing 4: Código para las figuras 1.

```
fig = go.Figure(go.Indicator(mode = "gauge+number",value = contagios,title = {'text':"Contagios"},
    domain = {'x': [0, 1], 'y': [0, 1]}))
fig.show()

fig = go.Figure(go.Indicator(mode = "gauge+number",value = recupera2,title = {'text':"Recuperados"},
    domain = {'x': [0, 1], 'y': [0, 1]}))
fig.show()

fig = go.Figure(go.Indicator(mode = "gauge+number",value = muer2,title = {'text':"Fallecidos"},
    domain = {'x': [0, 1], 'y': [0, 1]}))
fig.show()
```

((a)) Contagios totales.

((b)) Recuperados totales.

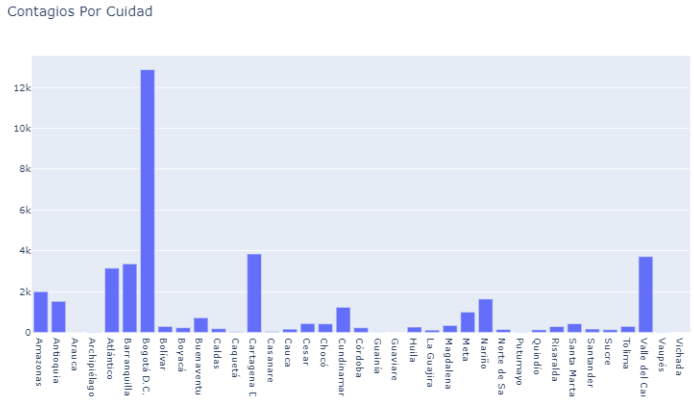
((c)) Muertes totales.

Figura 1: Visualizacion totales.

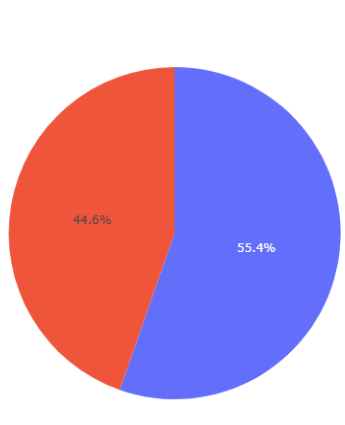
Se generan gráficas de los datos principales y de mayor interes general, como; los contagios por ciudad, segun el sexo y el estado en el que se encuntran estos, ya que estos datos son aquellos que engoblan todo y los más llamativos al público géneral.

Listing 5: Código para realizar gráfica de barras y de pastel.

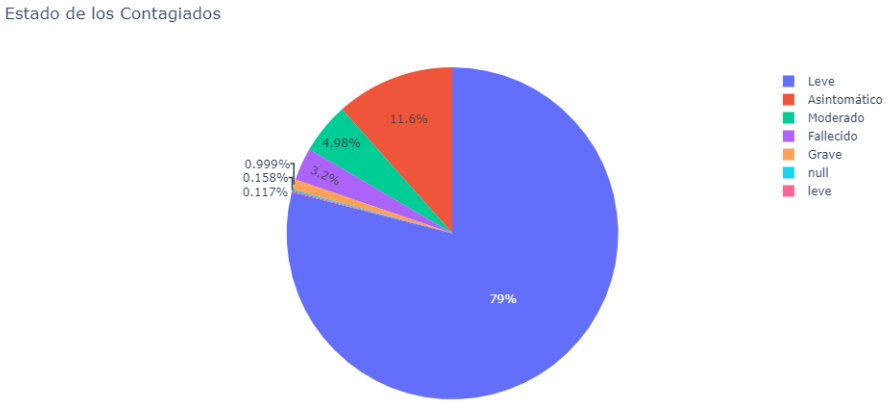
```
    #Barras
    fig=go.Figure(data=[go.Bar(x=departamentos, y=ciudades)],layout_title_text='Contagios Por ...
        Cuidad')
    fig.show()

    #Casos Registrados por Sexo
    fig=px.pie(df_result,values='casos', names='sexo', title='Casos Registrados Sexo')
    fig.show()

    #Estado de los contagios
    fig=px.pie(df_result,values='casos', names='estado', title='Estado de los Contagiados')
    fig.show()
```



((a)) Contagios por ciudad.



((b)) Casos segun el sexo.



((c)) Estado de los contagiados.

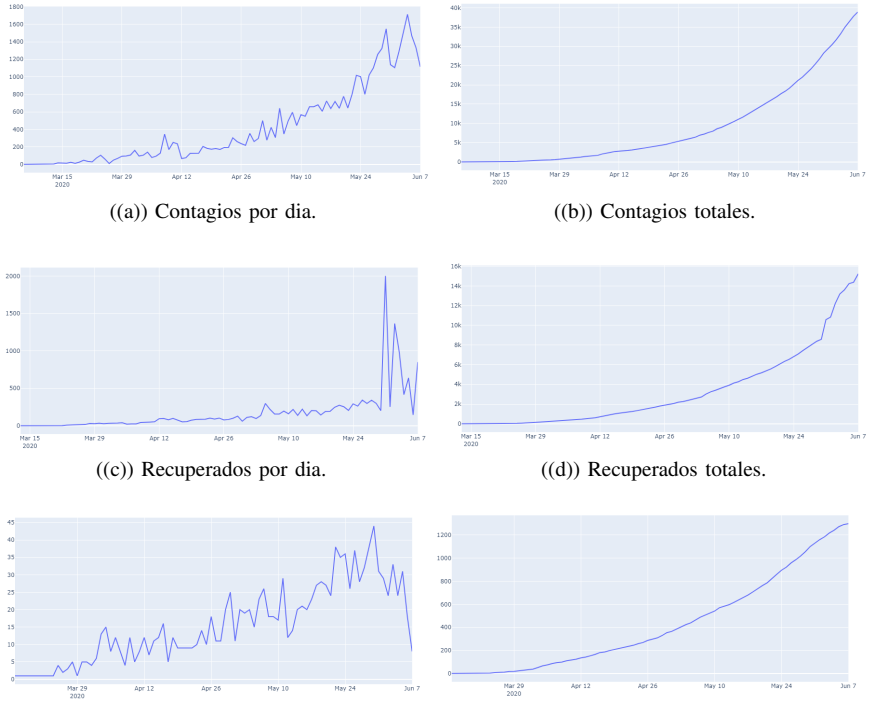
Figura 2: Seccionamiento principal.

La curva es un termino que se popularizado, haciendo referencia a la sumatoria de los contagios en el territorio nacioanl, por tanto se generan gráficas para visualizar esto, pero no solo de contagios sino tambien de recuperados y fallecidos.

Listing 6: Código para gráficas 2D interactivas.

```
#Contagios por dia
fig=go.Figure()
fig.add_trace(go.Scatter(y=casos_d,x=diagnostico,mode='lines',name='Contagios Por Dia'))
fig.show()

#Casos totales por dia
fig=go.Figure()
fig.add_trace(go.Scatter(y=con_tot,x=diagnostico,mode='lines',name='Contagios Totales Por Dia'))
fig.show()

#Recuperados por dia
fig=go.Figure()
fig.add_trace(go.Scatter(y=casos_r,x=recuperado,mode='lines',name='Recuperados por dia'))
fig.show()

#Recuperados totales
fig=go.Figure()
fig.add_trace(go.Scatter(y=rec_tot,x=recuperado,mode='lines',name='Recuperados Totales por dia'))
fig.show()
```

```
20
21      #Muertes por dia
22      fig=go.Figure()
23      fig.add_trace(go.Scatter(y=casos_m,x=muerto,mode='lines',name='Fallecidos por Dia'))
24      fig.show()
25
26      #Muertes totales
27      fig=go.Figure()
28      fig.add_trace(go.Scatter(y=dead_tot,x=muerto,mode='lines',name='Fallecidos Totales por dia'))
29      fig.show()
```



((a)) Contagios por dia.

((b)) Contagios totales.

((c)) Recuperados por dia.

((d)) Recuperados totales.

((e)) Muertes por dia.

((f)) Muertes totales.

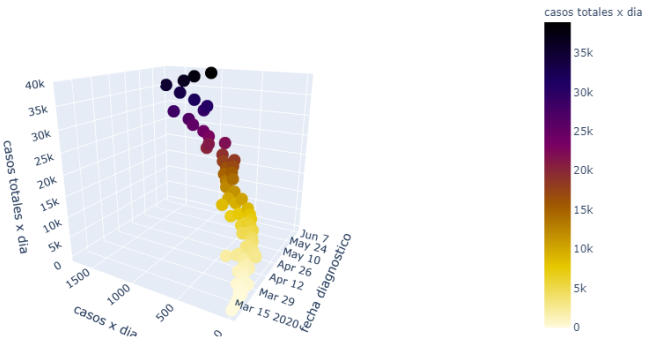
Figura 3: Histogramas 2D.

En las gráficas 3D se visualiza en el eje X las fechas, en el eje Y la cantidad de contagiados por dias, muertes por dia y recuperados por dia, y en el Z el acomulado de los contagiados, recuperados y fallecidos por covid-19, la gráfica en tres dimensiones es de tipo dispersión para poder realizar análisis de estos datos.

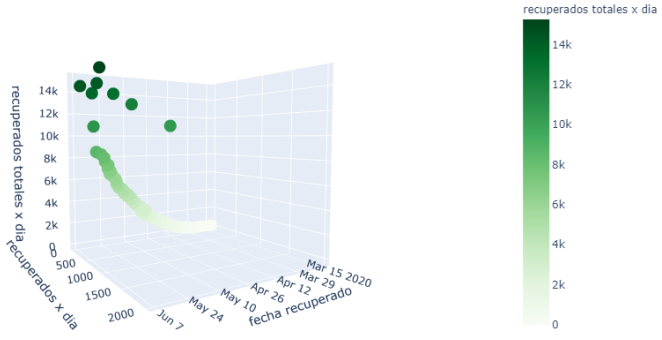
Listing 7: Código para Graficas 3D.

```
1  #Diagnosticos
2  tresd=pd.DataFrame({'fecha diagnostico':diagnostico,'casos totales x dia':con_tot,'casos x ...
       dia':casos_d})
3  fig = px.scatter_3d(tresd, x='fecha diagnostico', y='casos x dia', z='casos totales x dia', ...
       color='casos totales x dia',
4                      color_continuous_scale='electric_r')
5  fig.show()
6
7  #Recuperados
8  tresd1 = pd.DataFrame({'fecha recuperado':recuperado,'recuperados totales x ...
       dia':rec_tot,'recuperados x dia':casos_r})
9  fig = px.scatter_3d(tresd1, x='fecha recuperado', y='recuperados x dia', z='recuperados totales x ...
       dia', color='recuperados totales x dia',
10                     color_continuous_scale='greens')
```

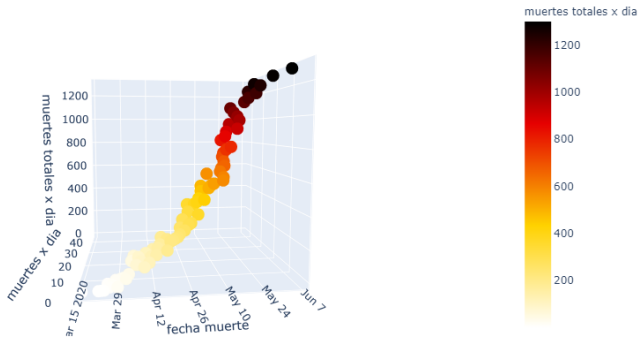
```
fig.show()

#Fallecidos
tresd2 = pd.DataFrame({'fecha muerte':muerto,'muertes totales x dia':dead_tot,'muertes x dia':casos_m})
tresd2
fig = px.scatter_3d(tresd2, x='fecha muerte', y='muertes x dia', z='muertes totales x dia',
                    color='muertes totales x dia',
                    color_continuous_scale='hot_r')
fig.show()
```



((a)) Grafica 3D contagios.



((b)) Grafica 3D recuperados.



((c)) Grafica 3D fallecidos.

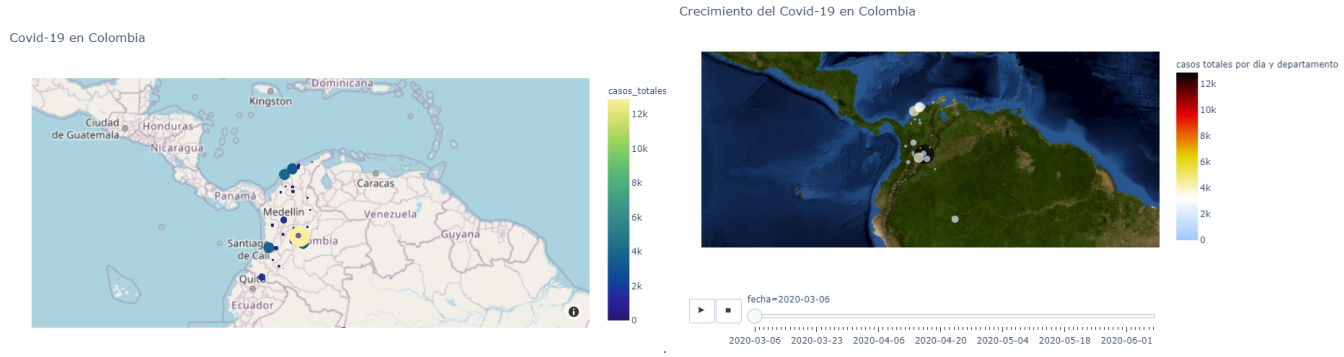
Figura 4: Graficas 3D.

Para tener una visualización más interactiva y amigable con el usuario que las necesite, se realiza un gráfico en donde se puede ver la distribución de los contagiados en el territorio nacional, una gráfica de los contagios registrados por departamento y ciudad, una de la condesacion de los casos y una referente al mapa de calor (para ver las zonas de mayor vulnerabilidad); se observan diversas ”burbujas”las cuales corresponden a un lugar y su tamaño depende de los contagios que se registren allí, adicionalmente se muestran las coordenadas del lugar, expresadas en terminos de *longitud* y *latitud*.

Listing 8: Código para mapas interactivos nivel Colombia.

```
#Mapa estatico
fig = px.scatter_mapbox(df_result, lat='lat', lon='lon', hover_name='departamento',
                        color='casos_totales',
                        size='casos_totales',color_continuous_scale='haline', zoom=3,
                        title= 'Covid-19 en Colombia')

fig.update_layout(mapbox_style="open-street-map")
fig.show()

#Mapa animado
```

```
fig = px.scatter_mapbox(anichi, lat='lat', lon='lon', hover_name='departamento', color='casos ...
    totales por dia y departamento',
                        size='casos totales por dia y ...
                            departamento',color_continuous_scale=px.colors.sequential.Blackbody_r,
                        zoom=3,animation_frame='fecha',title= 'Crecimiento del Covid-19 en Colombia')
fig.show()

#Mapa de condensacion
m_3 = folium.Map(location=[4,-72], tiles='stamenterrain', zoom_start=4.5)
mc = MarkerCluster()
for idx, row in df_result.iterrows():
    if not math.isnan(row['lon']) and not math.isnan(row['lat']):
        mc.add_child(Marker([row['lat'], row['lon']]))
m_3.add_child(mc)
m_3

#Mapa de calor
m_5 = folium.Map(location=[4,-72], tiles='stamenterrain', zoom_start=5.5)
HeatMap(data=df_result[['lat', 'lon']], radius=25).add_to(m_5)
m_5
```



((a)) Mapa estatico.

((b)) Mapa animado.

((c)) Mapa de condensacion.

((d)) Mapa de calor.

Figura 5: Mapas nivel Colombia.

Sifguiendo la finalidad del proyecto, se realizó un procesos similar, de visualización de datos, pero esta vez un poco mas seccionado, referente a las cifras de Bogotá.

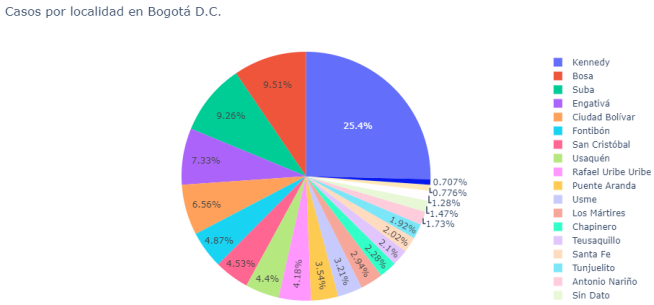
Listing 9: Extraccion de datos Bogota.

```
params = params={
        'resource_id': 'b64ba3c4-9e41-41b8-b3fd-2da21d627558',
        'limit': limite,
```

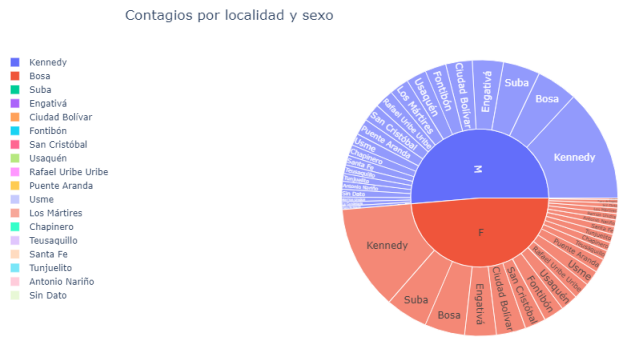
```
    }
url = 'https://datosabiertos.bogota.gov.co/api/3/action/datastore_search'
r = requests.get(url, params=params).json()
datbog = pd.DataFrame(r['result']['records'])
```

Listing 10: Código para Diagramas de pastel y dona

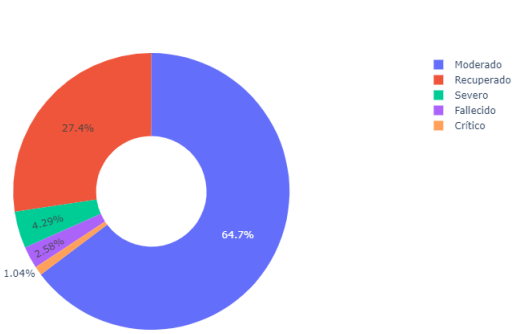
```
#Casos por localidad
fig=px.pie(datbog,values='casos', names='Localidad de residencia', title='Casos por localidad en Bogot  D.C.')
fig.show()

#Casos por localidad y sexo
fig = px.sunburst(datbog, path=['Sexo', 'Localidad de residencia'], values='casos',title='Contagios por localidad y sexo')
fig.show()

#Estado de los contagiados
fig=px.pie(donut,values='Casos', names='Estado', hole=.4,title='Estado de los Contagiados en Bogot ')
fig.show()

#Ubicacion de los contagiados
ubi=datbog.iloc[:,6], donutcho=pd.DataFrame({'Ubicacion':ubi,'Casos':casosd}), donutcho['Ubicacion'].replace({'Casa':'Casa','Fallecido No aplica No causa Directa':'Fallecido'},inplace=True)
fig=px.pie(donutcho,values='Casos', names='Ubicacion',hole=.4 ,title='Ubicacion de los Contagiados')
fig.show()
```



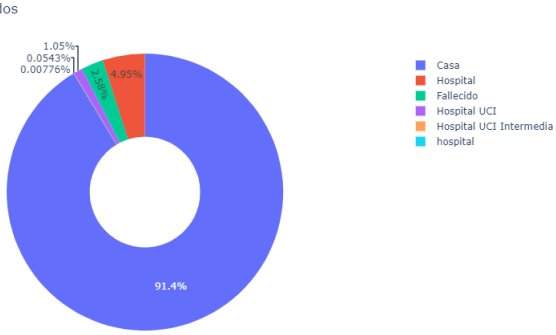
((a)) Casos por localidad.



((b)) Casos por localidad y sexo.



((c)) Estado de los contagiados.



((d)) Ubicacion de los contagiados.

Figura 6: Diagramas de pastel Bogota.

Listing 11: Código para la ubicacion geografica de los casos en Bogota.

```
#Mapa bogota
fig = px.scatter_mapbox(casbo, lat='Lat', lon='Lon', hover_name='Localidad', color='Casos',
                        size='Casos',color_continuous_scale='cividis', zoom=9,
                        title= 'Covid-19 en Bogota')

fig.update_layout(mapbox_style="open-street-map")
fig.show()
```

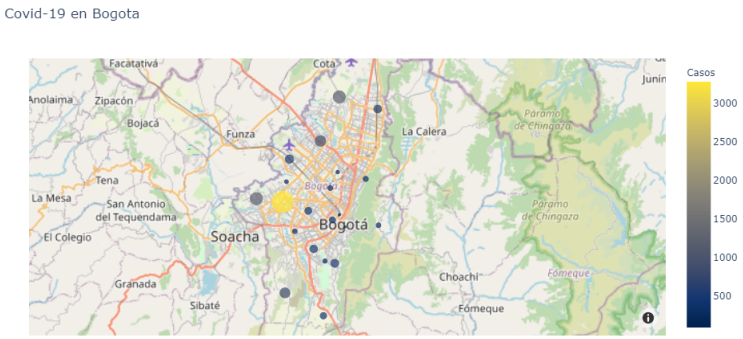
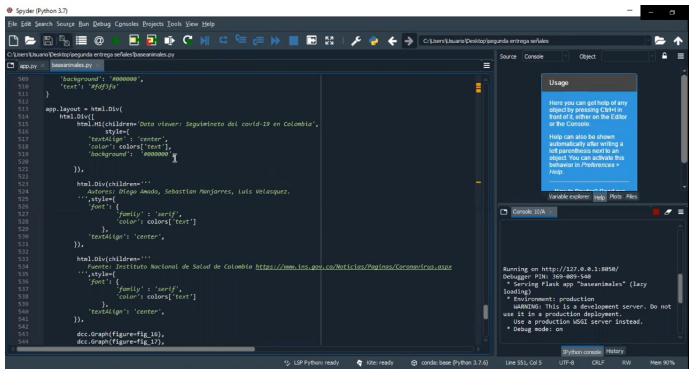


Figura 7: Mapa contagios Bogota.

Como parte final del trabajo se subieron las gráficas a un local host mediante el uso de la libreria *Dash*, para emular el uso de la aplicación que se creara en un futuro



((a)) Código para subir las imagenes, haciendo uso de la libreria *dash*

((b)) Resultado final.

Figura 8: Simulación página web.

4. Trabajo futuro.

- Es imposible, pero que diego deje de ser tan gei
- Implementar una base de datos con alguna plataforma que guarde y procese información en una nube; como la proporcionada por amazon, para no depender de Excel.
- No limitar la información de contagios, recuperados y fallecidos por Covid-19 a Colombia, sino por el contrario extenderlo a todo Sur America y el mundo.
- Crear una pagina web abierta al público, mejorando la interfaz con el usuario, haciendola amigable e intuitiva.

5. Enlace GIT

$https://github.com/Diego-Amado/Proyecto-Final_Signal_Analysis$

Referencias

[1] Plotly. Plotly python open source graphing library. [Online]. Available: https://plotly.com/python/

[2] socrata. Socrata. [Online]. Available: https://www.tylertech.com/products/socrata

[3] Python. pandas documentation. [Online]. Available: https://pandas.pydata.org/

[4] Numpy. Numpy v1.18 manual. [Online]. Available: https://numpy.org/doc/stable/

[5] A. C. Jessica Li. Interactive maps. [Online]. Available: https://www.kaggle.com/alexisbcook/interactive-maps