Procedemos a realizar este ejercicio:

Ejercicio práctico en grupos de 3 personas

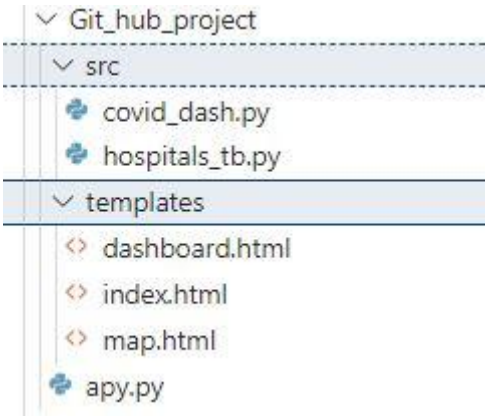
En este ejercicio, realizaremos una serie de tareas que nos ayudarán a afianzar lo explicado en el taller.

Para ello, formaremos grupos de 3 personas que representarán cada uno de los miembros del equipo: Developer_1, Developer_2, Developer_3. Cada uno de estos miembros será elegido por consenso del grupo y tendrá un papel determinado.

Deberéis seguir los pasos indicados uno por uno sin saltaros nada para que el resultado sea un proyecto funcional.

No se pedirá desarrollo de código, se os proporcionará una parte a cada uno para realizar las tareas.

Al finalizar el proyecto, la estructura debe de quedar de este modo:



Developer_1:

1. Crea un repositorio vacío en **Github**
2. Da permisos de push a Trabajador 2 y trabajador 3 en **Github**
3. Crea desde **git bash (o Visual Studio Code)** la develop-branch que será la rama que reciba los cambios que se produzcan en las epic branches de los tres trabajadores
4. **NO OLVIDES HACER git push origin develop-branch**
5. Comprueba que en **Github** están ambas ramas [main, develop-branch]
6. **AHORA TRABAJADOR 2 Y 3 PUEDEN EMPEZAR A TRABAJAR, NO ANTES**

7. Crea la epic-branch-1
   En ella te encargarás de desarrollar [api.py]

   Para ello (**Partiendo de epic-branch-1**) deberás crear una rama ticket por cada uno de los trozos de código que añadas a [api.py]

   - Crea la rama branch-ticket-1 para añadir el **Trozo de código 1**. Cuando lo tengas, fusiona con la epic-branch-1
   - Crea la rama branch-ticket-2 para añadir **Trozo de código 2**. Cuando lo tengas, fusiona con la epic-branch-1. **LA PARTE DE** if __name__ == '__main__' **** **va al final del archivo!**
   - Crea la rama epic-branch-3 a partir de epic-branch-1. Después, crea dos sub-branches dentro de branch-ticket-3 , la branch-ticket-3-1 y la branch-ticket-3-2. Inserta en cada una de ellas el trozo de código 3 y el 4, respectivamente. Fusiona primero branch-ticket-3 con branch-ticket-3-1. Luego branch-ticket-3 con branch-ticket-3-2. Al hacerlo, te dará un conflicto, dándote la opción de quedarte con el código que tienes actualmente en branch-ticket-3, branch-ticket-3-2 o una mezcla de ambas. Compara ambos códigos y elige la opción que consideres correcta.
   - Vuelve a epic-branch-1. Inserta este código, añade y commit. Fusiona con branch-ticket-3
   - Resuelve el conflicto, quédate con el trozo de código de la rama branch-ticket-3
   - Asegúrate de que has fusionado todas tus branches ticket con la epic-branch-1 y que tu archivo [api.py](<http://api.py>) es idéntico al del repo original. Si no es así, haz una rama ticket para cada cosa que añadas.
   - Es el momento en el que debes mandar tu trabajo al repositorio. usa git push origin epic-branch-1 para esto.

código developer_1:

Trozo de código 1:

```
from flask import Flask, render_template

from src import covid_dash, hospitals_tb
```

Trozo de código 2:

```
 app = Flask(__name__)

@app.route("/")

def landing_page():

        return render_template('index.html')

if __name__ == '__main__':

        app.run(host='0.0.0.0',port=1991, debug=True)
```

Trozo de código 3:

```
 @app.route("/dashboard")

def dashboard():

        return render_template('dashboard.html')

@app.route("/map")

def map():

        return render_template('map.html')
```

Trozo de código 4:

```
@app.route("/dashboard")

def dashboard():

        return render_template('ÑAÑAAÑÑAÑAÑAÑAÑÑA.html')

@app.route("/map")

def map():

        return render_template('map.html')
```

Developer_2

1. Clona el repositorio Creado por trabajador 1 en local.
2. Al clonarte el repositorio, solo habrás traído la rama main , sin embargo, necesitarás la rama develop-branch para continuar trabajando.
3. Antes de continuar, asegúrate de que Trabajador 1 ha creado la rama develop y la ha pusheado a Github, Entonces sigue los siguientes pasos https://stackoverflow.com/questions/67699/how-to-clone-all-remote-branches-in-git
4. Crea la epic-branch-2 desde develop-branch . En esta, te encargarás de desarrollar hospitals_tb.py
    1. Para esto deberás crear, partiendo de epic-branch-2 una rama ticket por cada uno de los trozos de código que añadas a hospitals_tb.py
    2. Crea la rama branch-ticket-1 para añadir **Trozo de código 1**. Cuando lo tengas, fusiona con la epic-branch-2.
    3. Crea la rama branch-ticket-2 para añadir **Trozo de código 2.** Cuando lo tengas, fusiona con la epic-branch-2
    4. Crea la rama branch-ticket-3 a partir de epic-branch-2 . Después, crea dos sub-branches dentro de branch-ticket-3 , la branch-ticket-3-1 y la branch-ticket-3-2. Inserta en cada una de ellas el trozo de código 3 y el 4, respectivamente. Fusiona primero branch-ticket-3 con branch-ticket-3-1. Luego branch-ticket-3 con branch-ticket-3-2. Al hacerlo, te dará un conflicto, dándote la opción de quedarte con el código que tienes actualmente en branch-ticket-3, branch-ticket-3-2 o una mezcla de ambas. Compara ambos códigos y elige la opción que consideres correcta.
    5. Vuelve a epic-branch-2 . Inserta este código, añade y commit. Fusiona con branch-ticket-3
5. Resuelve el conflicto, quédate con el trozo de código de la rama branch-ticket-3 .
6. Asegúrate de que has fusionado todas tus branches con epic-branch-2 y que tu archivo hospitals_tb.py es idéntico al del repo original. Si no es así, haz una rama ticket para cada cosa que añadas.
7. **ES EL MOMENTO:** manda tu trabajo al repositorio utilizando git push origin epic-branch-2 para esto.

código developer_2:

Trozo de código 1:

```
import folium
from folium.plugins import MarkerCluster
import pandas as pd
import request
```

Trozo de código 2:

```
def json_to_df(data):
    elements = data['elements']
    places = {'category': [], 'lat': [], 'lon': [], 'name': [], 'address': []}
    for i in elements:
        tipo = i.get('tags', None).get('amenity', None)
        latitude = i.get('lat', None)
        longitude = i.get('lon', None)
        name = i.get('tags', {}).get('name', "NO NAME")
        street = i.get('tags', {}).get('addr:street', "NO STREET")
        number = i.get('tags', {}).get('addr:housenumber', 9999)
        places['category'].append(tipo)
        places['lat'].append(latitude)
        places['lon'].append(longitude)
        places['name'].append(name)
        places['address'].append(street + ' ' + str(number))
    return pd.DataFrame(places)
```

Trozo de código 3:

```
list_health = [“bichitos de colores súper chulos”]

dataframes = []

for amenity in list_health:

        overpass_url = "<http://overpass-api.de/api/interpreter>"

        overpass_query = f"""[out:json]; node["amenity"= {amenity}] (40.40, -3.71,40.54, -3.60); out;"""”

# the bridge está en el Latitud: 40.421703 Longitud: -3.691725

response = requests.get(overpass_url, params={'data':overpass_query})

try:

        data = response.json()

        df = json_to_df(data)

        dataframes.append(df)

except:

        continue

health_csv = pd.concat(dataframes)

coordenadas_TB = (40.421703,-3.691725)

some_map2 = folium.Map(location=coordenadas_TB, zoom_start=14)

#for row in subset.itertuples():

some_map2.add_child(folium.Marker(location=[40.421703,-3.691725],popup="
The Bridge",icon=folium.Icon(icon='home',color='red')))

mc = MarkerCluster()

for row in health_csv.itertuples():

mc.add_child(folium.Marker(location = [row.lat, row.lon], popup=row.name,
icon=folium.Icon(icon='glyphicon glyphicon-heart-empty', color='blue')))

some_map2.add_child(mc)

some_map2.save('templates/map.html')
```

Trozo de código 4:

```
list_health = ["hospital", "clinic", "doctors"]
dataframes = []
for amenity in list_health:
overpass_url = "http://overpass-api.de/api/interpreter"
overpass_query = f"""
[out:json];
node["amenity"= {amenity}]
(40.40, -3.71,40.54, -3.60);
out;
"""
# the bridge está en el Latitud: 40.421703 Longitud: -3.691725
response = requests.get(overpass_url, params={'data':overpass_query})
try:
        data = response.json()
        df = json_to_df(data)
        dataframes.append(df)
except:
        continue
health_csv = pd.concat(dataframes)
coordenadas_TB = (40.421703,-3.691725)
some_map2 = folium.Map(location=coordenadas_TB, zoom_start=14)
#for row in subset.itertuples():
```

```
some_map2.add_child(folium.Marker(location=[40.421703,-3.691725],popup="
The Bridge",icon=folium.Icon(icon='home',color='red')))

mc = MarkerCluster()

for row in health_csv.itertuples():

mc.add_child(folium.Marker(location = [row.lat, row.lon], popup=row.name,
icon=folium.Icon(icon='glyphicon glyphicon-heart-empty', color='blue')))

some_map2.add_child(mc)

some_map2.save('templates/bichitosdecolores.html')
```

Developer_3

1. Clona el repositorio creado por Trabajador 1 en local.
2. Al clonarte el repositorio, solo habrás traído la rama main, sin embargo, necesitarás la rama develop-branch para continuar trabajando.
3. Antes de continuar, asegúrate de que Trabajador 1 ha creado la rama develop y la ha pusheado a Github. Entonces sigue los siguientes pasos: https://stackoverflow.com/questions/67699/how-to-clone-all-remote-branches-in-git
4. Crea la epic-branch-3 desde develop-branch en esta te encargarás de desarrollar covid_dash.py.
    1. Crea rama branch-ticket-1 para añadir **Trozo de código 1.** Cuando lo tengas, fusiona con la epic-branch-3
    2. Crea rama branch-ticket-2para añadir **Trozo de código 2**. Cuando lo tengas, fusiona con la epic-branch-3
    3. Crea rama branch-ticket-3a partir de epic-branch-3. Después, crea dos sub-branches dentro de branch-ticket-3 , la branch-ticket-3-1 y la branch-ticket-3-2. Inserta en cada una de ellas el trozo de código 3 y el 4, respectivamente. Fusiona primero branch-ticket-3 con branch-ticket-3-1. Luego branch-ticket-3 con branch-ticket-3-2. Al hacerlo, te dará un conflicto, dándote la opción de quedarte con el código que tienes actualmente en branch-ticket-3, branch-ticket-3-2 o una mezcla de ambas. Compara ambos códigos y elige la opción que consideres correcta.
    4. Vuelve a epic-branch-3
    5. Inserta este código, añade y commit. Fusiona con branch-ticket-3
5. Resuelve el conflicto, quédate con el trozo de código de la rama branch-ticket-3 .
6. Asegúrate de que has fusionado todas tus branches ticket con la epic-branch-3y que tu archivo covid_dash.py es **idéntico al del repo original.**
    1. Si no es así, haz una rama ticket para cada cosa que añadas.
7. Es el momento en el que debes mandar tu trabajo al repositorio, usa git push origin epic-branch-3para esto.

código developer_3:

Trozo de código 1:

```
import plotly.graph_objects as go

from plotly.subplots import make_subplots

import pandas as pd

import requests

from datetime import datetime

#<https://towardsdatascience.com/building-a-real-time-dashboard-us>

ing-python-plotly-library-and-web-service-145f50d204f0

raw=requests.get("<https://services1.arcgis.com/0MSEUqKaxRlEPj5g/arcgi>s/rest/services/Coronavirus_2019_nCoV_Cases/FeatureServer/1/query?where=1%3D1&outFields=*&outSR=4326&f=json")

raw_json = raw.json()

df = pd.DataFrame(raw_json["features"])

data_list = df["attributes"].tolist()

df_final = pd.DataFrame(data_list)

df_final.set_index("OBJECTID")

df_final = df_final[["Country_Region", "Province_State", "Lat","Long_", "Confirmed", "Deaths", "Recovered", "Last_Update"]]
```

Trozo de código 2:

```
def convertTime(t):
        t = int(t)
        return datetime.fromtimestamp(t)
df_final = df_final.dropna(subset=["Last_Update"])
df_final["Province_State"].fillna(value="", inplace=True)
df_final["Last_Update"]= df_final["Last_Update"]/1000
df_final["Last_Update"] =df_final["Last_Update"].apply(convertTime)
df_total=df_final.groupby("Country_Region",as_index=False).agg({"Confirmed"
: "sum","Deaths" : "sum","Recovered" : "sum"})
total_confirmed = df_final["Confirmed"].sum()
total_recovered = df_final["Recovered"].sum()
total_deaths = df_final["Deaths"].sum()
df_top10 = df_total.nlargest(10, "Confirmed")
top10_countries_1 = df_top10["Country_Region"].tolist()
top10_confirmed = df_top10["Confirmed"].tolist()
df_top10 = df_total.nlargest(10, "Recovered")
top10_countries_2 = df_top10["Country_Region"].tolist()
top10_recovered = df_top10["Recovered"].tolist()
df_top10 = df_total.nlargest(10, "Deaths")
top10_countries_3 = df_top10["Country_Region"].tolist()
top10_deaths = df_top10["Deaths"].tolist()
```

Trozo de código 3:

```
fig= make_subplots(rows = 4, cols = 6,specs=[[{"type": "scattergeo", "rowspan": 4, "colspan": 3},None, None, {"type": "indicator"}, {"type": "indicator"},{"type": "indicator"} ],[ None, None, None, {"type": "bar","colspan":3}, None, None],[ None, None, None, {"type": "bar","colspan":3}, None, None],[ None, None, None, {"type": "bar","colspan":3}, None, None],])

message = df_final["Country_Region"] + " " +df_final["Province_State"] + "<br>"

message += "Confirmed: " + df_final["Confirmed"].astype(str) +"<br>"

message += "Deaths: " + df_final["Deaths"].astype(str) + "<br>"

message += "Recovered: " + df_final["Recovered"].astype(str) +"<br>"

message += "Last updated: " + df_final["Last_Update"].astype(str)

df_final["text"] = message

fig.add_trace(go.Scattergeo(locationmode="country names",lon=df_final["Long_"],lat=df_final["Lat"],hovertext=df_final["text"],showlegend=False,marker=dict(size=10,opacity=0.8,reversescale=True,autocolorscale=True,symbol='square',line=dict(width=1,color='rgba(102,102,102)'),cmin=0,color=df_final['Confirmed'],cmax=df_final['Confirmed'].max(),colorbar_title="Confirmed Cases<br>Latest Update",colorbar_x = -0.05)),row=1, col=1)

fig.add_trace(go.Indicator(mode="number",value=total_confirmed,title="Confirmed Cases",),row=1, col=4)

fig.add_trace(go.Indicator(mode="number",value=total_recovered,title="Recovered Cases",),row=1, col=5)

fig.add_trace(go.Indicator(mode="number",value=total_deaths,title="Deaths Cases",),row=1, col=6)

fig.add_trace(go.Bar(x=top10_countries_1,y=top10_confirmed,name="Confirmed Cases",marker=dict(color="Yellow"),showlegend=True,),row=2, col=4)

fig.add_trace(go.Bar(x=top10_countries_2,y=top10_recovered,name="Recovered Cases",marker=dict(color="Green"),showlegend=True),row=3, col=4)

fig.add_trace(go.Bar(x=top10_countries_3,y=top10_deaths,name="Deaths Cases",marker=dict(color="crimson"),showlegend=True),row=4, col=4)

fig.update_layout(template="plotly_dark",title = "Global COVID-19 Cases (Last Updated:"+str(df_final["Last_Update"][0])+")",showlegend=True,legend_orientation="h",legend=dict(x=0.65,y=0.8),geo=dict(projection_type="orthographic",sho
```

wcoastlines=True,landcolor="white",showland=True,showocean=True,lakecolor="LightBlue"),annotations=[dict(text="Source:<https://bit.ly/3aEzxjK>",showarrow=False,xref="paper",yref="paper",x=0.35,y=0)])

fig.write_html('templates/dashboard.html')

Trozo de código 4:

fig = make_LO_QUE_TE_DE_LA_GANA(rows = 4, cols = 6,specs=[[{"type": "scattergeo", "rowspan": 4, "colspan": 3},None, None, {"type": "indicator"}, {"type": "indicator"},{"type": "indicator"}],[None, None, None, {"type": "bar","colspan":3}, None, None],[None, None, None, {"type": "bar","colspan":3}, None, None],[None, None, None, {"type": "bar","colspan":3}, None, None],])

message = df_final["Country_Region"] + " " + df_final["Province_State"] + "
"

message += "Confirmed: " + df_final["Confirmed"].astype(str) + "
"

message += "Deaths: " + df_final["Deaths"].astype(str) + "
"

message += "Recovered: " + df_final["Recovered"].astype(str) + "
"

message += "Last updated: " + df_final["Last_Update"].astype(str)

df_final["text"] = message

fig.add_trace(go.Scattergeo(locationmode="country names",lon=df_final["Long_"],lat=df_final["Lat"],hovertext=df_final["text"],showlegend=False,marker=dict(size=10,opacity=0.8,reversescale= True,autocolorscale = True,symbol = 'square',line=dict(width=1,color='rgba(102, 102, 102)'),cmin = 0,color=df_final['Confirmed'],cmax=df_final['Confirmed'].max(),colorbar_title="Confirmed Cases
Latest Update",colorbar_x = -0.05)), row=1, col=1)

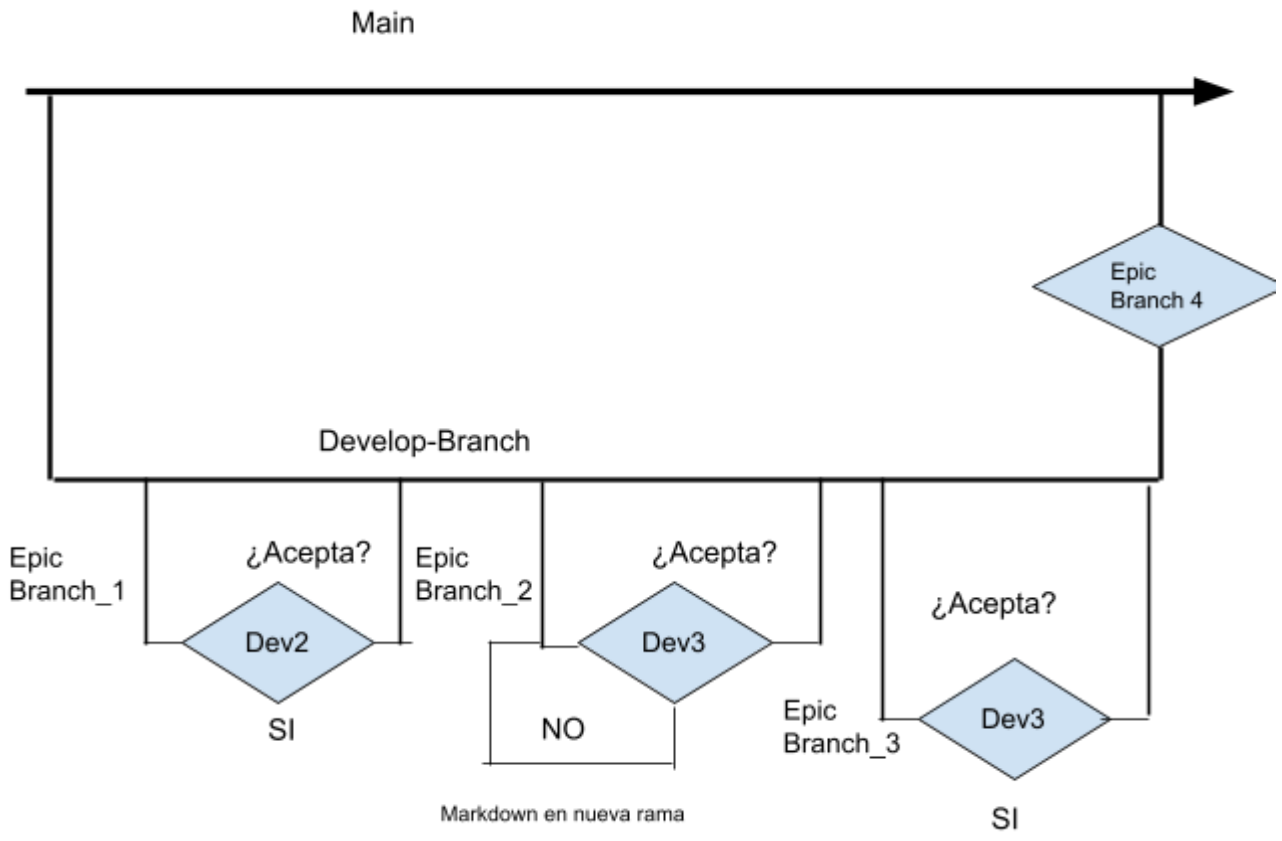
fig.add_trace(go.Indicator(mode="number",value=total_confirmed,title="Confirmed Cases",),row=1, col=4)

fig.add_trace(go.Indicator(mode="number",value=total_recovered,title="Recovered Cases",),row=1, col=5)

```
fig.add_trace(go.Indicator(mode="number",value=total_deaths,title="Deaths Cases",),row=1, col=6)

fig.add_trace(go.Bar(x=top10_countries_1,y=top10_confirmed,name="Confirmed Cases",arker=dict(color="Yellow"),showlegend=True,),row=2, col=4)

fig.add_trace(go.Bar(x=top10_countries_2,y=top10_recovered,name="Recovered Cases",marker=dict(color="Green"),showlegend=True),row=3, col=4)

fig.add_trace(go.Bar(x=top10_countries_3,y=top10_deaths,name="Deaths Cases"marker=dict(color="crimson"), showlegend=True),row=4, col=4)

fig.update_layout(template="plotly_dark",title="Global COVID-19 Cases (Last Updated:"+str(df_final["Last_Update"][0])+")",showlegend=True,legend_orientation="h",legend=dict(x=0.65,y=0.8),geo=dict(projection_type="orthographic",showcoastlines=True,landcolor="white",showland=True,showocean=True,lakecolor="LightBlue"),annotations=[dict(text="Source:<https://bit.ly/3aEzxjK>",showarrow=False,xref="paper",yref="paper", x=0.35, y=0)])

fig.write_html('templates/quedivertidoesesteejercicio.html')
```

LLEGADO A ESTE PUNTO, ES IMPORTANTE QUE TODOS LOS MIEMBROS DEL EQUIPO ESTÉIS MÁS O MENOS EN EL MISMO PUNTO, SI NO ES ASÍ, ES TAMBIÉN TU RESPONSABILIDAD QUE ESO SEA ASÍ. ÉCHALES UNA MANO!!!

Vamos a realizar las funciones de esta imagen



¿Qué modificación hay que hacer en Epic-Branch_2?

Agrega una nueva rama, donde crearás un Markdown con los nombres, apellidos y e-mails de los integrantes del desarrollo.

¿Qué tiene la Epic_Branch_4?

Develop_3 tiene que crear una nueva rama epic-branch-4 proveniente de develop-branch, donde estructurarás el programa y agregarás la carpeta de templates entregada.

Haz un pull-request que te aprobará el Develop_1.

¡¡Ya habéis terminado!! Ahora puedes hacer git pull desde develop-branch (desde tu ordenador) y ya tendrás el proyecto acabado.