EJERCICIO GIT

Durante esta práctica se espera aprendáis a trabajar con Github de manera más fluida, sobre todo cuando trabajáis en equipo.

Teniendo como referencia este repositorio: https://github.com/Clapiniella/git-map
Debéis seguir los pasos definidos, por **trabajador**, hasta conseguir SIN SALTAROS NINGÚN PUNTO, que vuestro programa funcione.

El primer paso debe ser definir quién va a ser qué trabajador, ya que esto marcará el desarrollo del ejercicio.

Los comandos que vas a utilizar durante la práctica:

Para crear una rama → `` `git checkout -b nombre-rama` ``
Para ver un listado de tus ramas → `` `git branch` ``
Para fusionar dos ramas → (estando en la rama destino) `` `git merge rama-origen` ``
https://stackoverflow.com/questions/14005854/what-to-do-with-branch-after-merge
Para subir cambios de una rama a Github → `` `git push origin nombre-rama` ``
Para crear una Pull Request → Hazlo desde Github, ten cuidado al seleccionar la rama origen y la rama destino.
Si encuentras conflicto, debes resolverlo en un editor de texto como por ejemplo Visual Studio Code.
Para cambiar de rama → `` `git checkout nombre-rama` ``
Para borrar una rama → `` `git branch -d nombre-rama` ``

Si se te abre VIM en la terminal:

- `` `I` ``: Insertar texto → Ahora escribe
  Si tiene mensaje de Merge y te gusta pasa al paso 2.
- `` `Esc` + ":wq" `` para guardar y salir

Si quieres salir sin guardar → `` `Esc` + :q! ``

¡Los mensajes de commit deben ser explicativos!

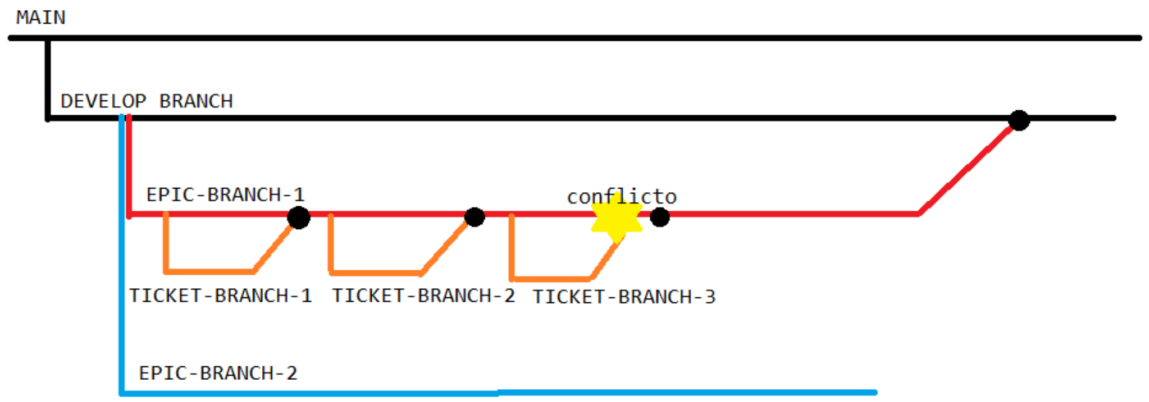
Mensaje commit de ramas ticket:

"epic branch número - ticket número - cambio realizado"

¡EL ORDEN DE LOS PASOS ES SÚPER IMPORTANTE!
¡ASEGURARTE 18 VECES DE EN QUÉ RAMA ESTÁS ANTES DE HACER CUALQUIER COSA, MAY SAVE YOUR LIFE!

Unos videotutoriales porque sé que los necesitarás.

- https://www.youtube.com/watch?v=vu4Rv1SmzwM
- https://www.youtube.com/watch?v=3-rCELg8feQ



Si quieres saber más…

https://bocoup.com/blog/git-workflow-walkthrough-feature-branches
https://bocoup.com/blog/git-workflow-walkthrough-reviewing-pull-requests

En este caso vuestra **develop branch** tendrá el propósito de crear la primera versión de una API que muestra un Dashboard & mapa.

Para la realización del ejercicio habrá que instalar las siguientes librerías.

Requirements.txt

```
Flask==1.1.2
folium==0.12.1
requests==2.24.0
pandas==1.2.0
plotly==4.9.0
```

TRABAJADOR 1

1. Crea un repositorio vacío. (Github)
2. Da permisos de **push** a Trabajador 2 y a Trabajador 3. (Github)
3. Crea desde `git bash` la **develop-branch,** esta será la receptora de los cambios que se produzcan en las epic branches de los tres trabajadores.
   (No olvides hacer `` `git push origin develop-branch` ``)
4. Comprueba que efectivamente en Github están ambas ramas [main, develop]
   Ahora Trabajador 2 y 3 pueden empezar a trabajar. No antes.

5. Crea la **epic-branch-1**, en esta te encargarás de desarrollar **`api.py`**
   Para esto deberás crear (partiendo de **epic-branch-1**) una rama ticket por cada uno de los trozos de código que añadas a **`api.py`**. ¡Revisa el repo original que no se te

pase nada o tendrás que volver atrás! Tampoco te olvides de indentar correctamente los trozos de código.

6. Crea rama ticket 1 (**branch-ticket-1**) para añadir Trozo de código 1. Cuando lo tengas, fusiona con la **epic-branch-1**.

   Trozo de código 1

```
from flask import Flask, render_template
from src import covid_dash, hospitals_tb
```

7. Crea rama ticket 2 para añadir Trozo de código 2. Cuando lo tengas, fusiona con la **epic-branch-1**. ¡la parte del `if __name__ == '__main__':` va al final del archivo!

   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)
```

8. Crea rama ticket 3 a partir de **epic-branch-1**. Inserta este código, añade y commit. ¡No fusiones aún! ¡No *git merge*!

   Trozo de código 3

```
@app.route("/dashboard")
def dashboard():
    return render_template('dashboard.html')

@app.route("/map")
def map():
    return render_template('map.html')
```

9. Vuelve a **epic-branch-1**. Inserta este código, añade y commit. Fusiona con **branch-ticket-3**. (Paso 9)

   Trozo de código 4.

```
@app.route("/dashboard")
def dashboard():
    return render_template('ÑAÑAAÑÑAÑAÑAÑAÑÑA.html')
```

```
@app.route("/map")
def map():
    return render_template('map.html')
```

10. Resuelve el conflicto, quédate con el trozo de código de la rama ticket-3.
11. Asegúrate de que has fusionado todas tus branches ticket con la **epic-branch-1** y que tu archivo **`api.py`** es idéntico al del repo original. Si no es así, haz una rama ticket para cada cosa que añadas.
12. Es el momento en el que debes mandar tu trabajo al repositorio, usa **`git push origin epic-branch-1`** para esto.

    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!

13. Eres el primero en hacer la Pull request a **develop-branch desde tu rama epic-branch-1** en Github (Te acepta los cambios Trabajador 2).

    Espera que Trabajador 2 y Trabajador 3, acaben de hacer sus PR y los tres archivos estén en la **develop-branch**. (Si les echas una mano, iréis más rápido)

14. Además, cuando ya estén todas las PR hechas y aceptadas, Trabajador 3 (que antes habrá hecho un `git pull origin develop-branch`) tiene la misión de añadir todas las carpetas y darle la misma estructura a la rama **develop-branch** que tiene el repo original. Para esto deberá crear la rama **epic-branch-4**. Cuando haya realizado los cambios, deberá hacer una PULL REQUEST a **develop-branch** que tú deberás revisar y aceptar.

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

TRABAJADOR 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.

```
clara@TA-Data-Science MINGW64 ~/Desktop/trabajador398409/git-map (main)
$ git branch
* main
```

Antes de continuar asegúrate de que Trabajador 1 ha creado la rama develop y la ha pusheado a Github. Entonces sigue los siguientes pasos:

```
clara@TA-Data-Science MINGW64 ~/Desktop/trabajador398409/git-map (main)
$ git branch -a
* main
  remotes/origin/HEAD -> origin/main
  remotes/origin/develop
  remotes/origin/main

clara@TA-Data-Science MINGW64 ~/Desktop/trabajador398409/git-map (main)
$ git checkout develop
Switched to a new branch 'develop'
Branch 'develop' set up to track remote branch 'develop' from 'origin'.

clara@TA-Data-Science MINGW64 ~/Desktop/trabajador398409/git-map (develop)
$ git branch
* develop
  main
```

https://stackoverflow.com/questions/67699/how-to-clone-all-remote-branches-in-git

3. Crea la **epic-branch-2** (desde **develop-branch**), en esta te encargaras de desarrollar `**hospitals_tb.py**`
   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**`. ¡Revisa el repo original que no se te pase nada o tendrás que volver atrás! Tampoco te olvides de indentar correctamente los trozos de código.
4. Crea rama ticket 1 (**branch-ticket-1**) para añadir Trozo de código 1. Cuando lo tengas, fusiona con la **epic-branch-2**.

   Trozo de código 1

```python
import folium
from folium.plugins import MarkerCluster
import pandas as pd
import requests
```

5. Crea rama ticket 2 para añadir Trozo de código 2. Cuando lo tengas, fusiona con la **epic-branch-2**.

   Trozo de código 2

```python
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)
```

6. Crea rama ticket 3 a partir de **epic-branch-2**. Inserta este código, añade y commit. ¡No fusiones aún! ¡No *git merge*!

Trozo de código 3

```
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/map.html')
```

7. Vuelve a **epic-branch-2**. Inserta este código, añade y commit. Fusiona con **branch-ticket-3**. (Paso 9)

Trozo de código 4

```
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/bichitosdecolores.html')
```

8. Resuelve el conflicto, quédate con el trozo de código de la rama ticket-3.
9. Asegúrate de que has fusionado todas tus branches ticket con la **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.
10. Es el momento en el que debes mandar tu trabajo al repositorio, usa `**git push origin epic-branch-2**` para esto.

    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!

11. Acepta la Pull Request que va a hacer Trabajador 1 a la **develop-branch**.
12. Ahora es tu turno de hacer una Pull request a **develop-branch** desde **epic-branch-2**. En este caso te acepta los cambios Trabajador 3, pero la primera vez te la va a rechazar y te pedirá que abras otra rama ticket a tu **epic-branch-2** y añadas un markdown con los nombres de los tres Trabajadores. Vuelve a fusionar con **epic-branch-2** y vuelve a abrir la PR. Ahora sí, Trabajador 3 deberá aceptarla.

13. Antes de probar si tu programa funciona, Trabajador 3 tiene la misión de añadir todas las carpetas y darle la misma estructura a la rama **develop-branch** que tiene el repo original. Para esto deberá crear la rama **epic-branch-4**. Cuando haya realizado los cambios, deberá hacer una PULL REQUEST a **develop-branch** que Trabajador 1 deberá revisar y aceptar.
14. ¡Ya habéis terminado! Ahora crea una rama develop-branch en tu ordenador y haz *git pull* desde esta rama **develop-branch** (desde tu ordenador) y tendrás el proyecto acabado.

TRABAJADOR 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.

```
clara@TA-Data-Science MINGW64 ~/Desktop/trabajador398409/git-map (main)
$ git branch
* main
```

Antes de continuar asegúrate de que Trabajador 1 ha creado la rama develop y la ha pusheado a Github. Entonces sigue los siguientes pasos:

```
clara@TA-Data-Science MINGW64 ~/Desktop/trabajador398409/git-map (main)
$ git branch -a
* main
  remotes/origin/HEAD -> origin/main
  remotes/origin/develop
  remotes/origin/main

clara@TA-Data-Science MINGW64 ~/Desktop/trabajador398409/git-map (main)
$ git checkout develop
Switched to a new branch 'develop'
Branch 'develop' set up to track remote branch 'develop' from 'origin'.

clara@TA-Data-Science MINGW64 ~/Desktop/trabajador398409/git-map (develop)
$ git branch
* develop
  main
```

https://stackoverflow.com/questions/67699/how-to-clone-all-remote-branches-in-git

3. Crea la **epic-branch-3** (desde **develop-branch**), en esta te encargarás de desarrollar **`covid_dash.py`**
   Para esto deberás crear (partiendo de **epic-branch-3**) una rama ticket por cada uno de los trozos de código que añadas a **`covid_dash.py`**. ¡Revisa el repo original que no se te pase nada o tendrás que volver atrás! Tampoco te olvides de identar correctamente los trozos de código.
4. Crea rama ticket 1 (**branch-ticket-1**) para añadir Trozo de código 1. Cuando lo tengas, fusiona con la **epic-branch-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"]]
```

5. Crea rama ticket 2 para añadir Trozo de código 2. Cuando lo tengas, fusiona con la **epic-branch-3**.

   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()
```

6. Crea rama ticket 3 a partir de **epic-branch-3**. Inserta este código, añade y commit. ¡No fusiones aún! ¡No *git merge*!

   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",
            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/dashboard.html')
```

7. Vuelve a **epic-branch-3**. Inserta este código, añade y commit. Fusiona con **branch-ticket-3**. (Paso 9)

   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"] + "<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",
            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')
```

8. Resuelve el conflicto, quédate con el trozo de código de la rama ticket-3.
9. Asegúrate de que has fusionado todas tus branches ticket con la **epic-branch-3** y que tu archivo `**covid_dash.py**` es idéntico al del repo original. Si no es así, haz una rama ticket para cada cosa que añadas.
10. Es el momento en el que debes mandar tu trabajo al repositorio, usa `**git push origin epic-branch-3**` para esto.

    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!

11. [Empieza Trabajador 1, eres el último en hacer la Pull Request] No podrás hacer tu PR hasta que no hayas aceptado la PR de Trabajador 2. No obstante, la primera vez que Trabajador 2 haga la PR recházala, dile que abra otra rama ticket y añada un markdown con los nombres de los tres Trabajadores y vuelva a fusionar con **epic-branch-2**. Cuando vuelva a abrir la PR. Acéptala.
12. Ahora puedes hacer tu Pull Request que Trabajador 1, aceptará sin comentario alguno. A no ser que vea algo que no estaba previsto.
13. Antes de probar si tu programa funciona, tienes la GRAN misión de añadir todas las carpetas y darle la misma estructura a la rama **develop-branch** que tiene el repo original. Para esto deberás crear la rama **epic-branch-4**, una vez hayas hecho `git pull origin **develop-branch** estando en la rama **develop-branch**. No te olvides de añadir el archivo `**index.html**` a la carpeta templates. Este archivo podrás cogerlo del repo original. Los demás archivos *html* se añadirán al ejecutarse el programa.
14. Cuando hayas realizado los cambios, deberás hacer una PULL REQUEST a **develop-branch** que Trabajador 1 deberá revisar y aceptar.
15. ¡Ya habéis terminado! Haz *git pull* desde la rama **develop-branch** (desde tu ordenador) y tendrás el proyecto acabado.

---------- x ----------

Trabajador 1, Trabajador 2 y Trabajador 3. Si has finalizado con éxito todo el procés y tu programito funciona, primero de todo debes saber que eres UN CRACK y después que puedes hacer la PR final... de develop-branch a MAIN. Esto no es parte del ejercicio, igual que no lo es mejorar el “estilo inexistente” de la landing page o modificar el código del dashboard o mapa para mejorarlo o personalizarlo, pero si lo haces, pues más CRACK aún. **<3**

---------- x ----------