Visualización de Datos

Práctica 6: Visualización dinámica e interactiva

Francisco Javier Mercader Martínez Rubén Gil Martínez

Índice

En este informe se explica detalladamente el archivo `app.py`.

1 Decisiones de Diseño

1.1 Elección del conjunto de datos

- **Conjunto de datos utilizados:**

  Se hace uso del dataset *Gapminder* de Plotly a través de `px.data.gapminder()` ya que es un conjunto de datos ampliamente utilizado para visualizar indicadores sociales y económicos (como el PIB per cápita, la esperanza de vida y la población) en diferentes países a lo largo de varios años.

- **Justificación de la elección:**

  - **Cobertura y riqueza de datos:** El dataset ofrece información histórica de diversos países, lo que permite explorar tendencias, crecimientos y comparaciones de manera efectiva.
  - **Contexto interactivo:** El dataset ha resultado una buena opción para el diseño del dashboard interactivo, ya que permite que los usuarios exploren los datos geográficos y temporales mediante filtros, permitiendo una experiencia muy visual e interactiva.

1.2 Selección de tipos de gráficos y componentes interactivos

- **Gráficos utilizados:**

  - **Gráfico de líneas y puntos:**

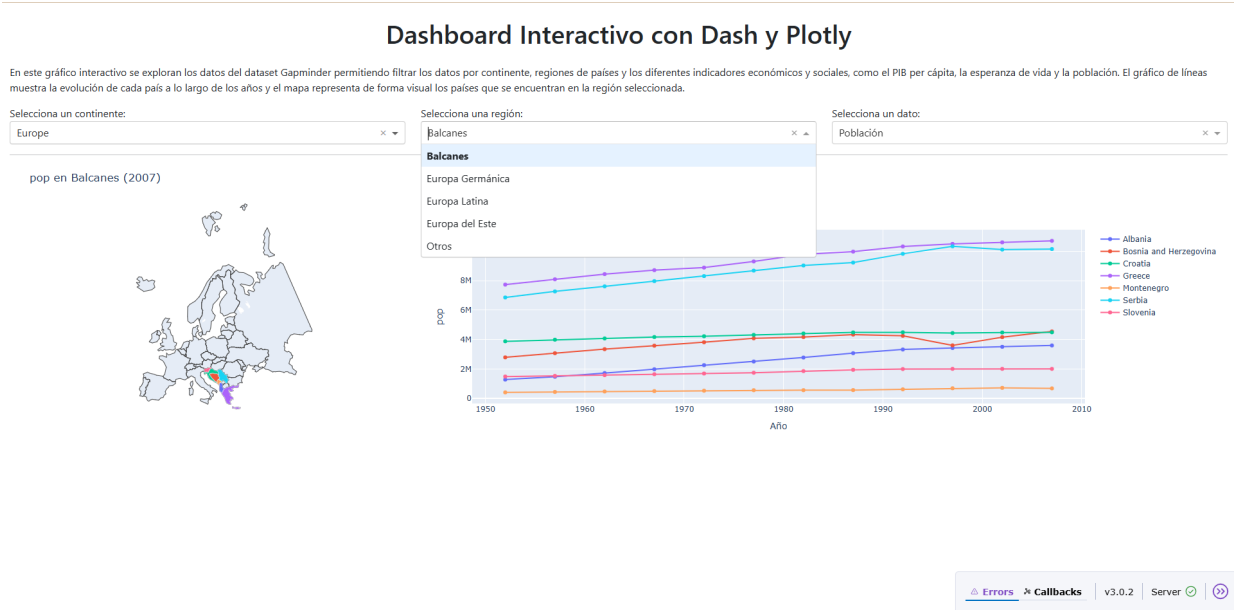
    Se utiliza para visualizar la evolución temporal de un indicador (por ejemplo, población, PIB per cápita o esperanza de vida) para cada país dentro de la región seleccionada.

  - **Mapa:**

    Se genera un mapa con el fin de representar visualmente la localización geográfica de los países filtrados. Se utiliza una traza de *Choropleth* para resaltar de forma visual los países en la región, asignando un color distinto a cada uno.

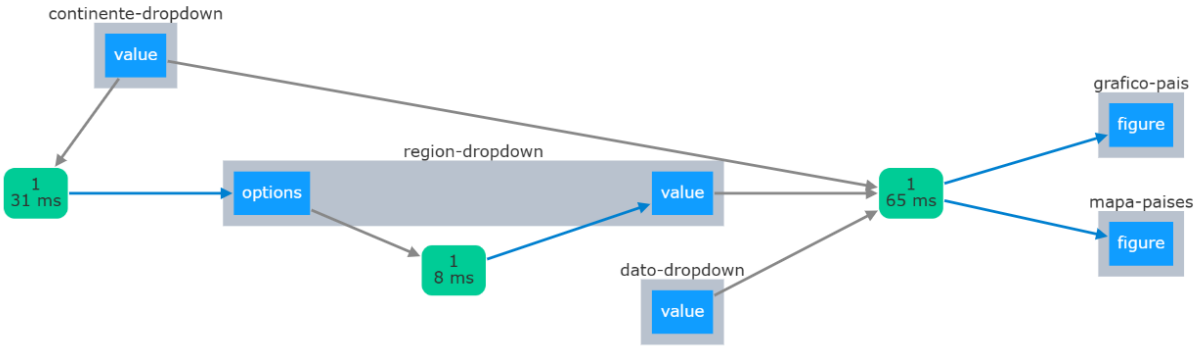
- **Componentes interactivos:**

- **Dropdowns:**
  - Uno para seleccionar el continente.
  - Uno para seleccionar la región (cuya lista se actualiza dinámicamente en función del continente elegido).
  - Uno para elegir el indicador (dato) a visualizar.



- **Interactividad con callbacks:**

  Se utiliza la funcionalidad de *callbacks* en Dash para enlazar los elementos del dashboard, permitiendo actualizaciones automáticas de los gráficos al cambiar las selecciones.



- **Justificación de la elección:**
  - **Visualización dinámica:** Los gráficos escogidos (líneas para tendencias temporales y mapas para datos geográficos) proporcionan una representación visual clara y comprensible.
  - **Interactividad mejorada:** Los componentes interactivos (dropdowns) facilitan la exploración de datos y permiten personalizar la vista según las preferencias del usuario, lo que mejora la experiencia de usuario.
  - **Consistencia visual:** El uso de *dash-bootstrap-components* junto a un tema de Bootstrap garantiza una interfaz limpia, adaptable y profesional.

1.3 Estructura y diseño del Dashboard

- **Disposición general:**

  El dashboard se organiza en un contenedor fluido (`dbc.Container`) para adaptarse a distintos tamaños de pantalla. Se presenta de la siguiente manera:

- **Encabezado y descripción:**

  Un título principal centrado y un párrafo descriptivo que explican el propósito del dashboard.

- **Filtros:**

  Una fila (`dbc.Row`) que contiene tres columnas, cada una con un dropdown para seleccionar el continente, la región y el dato a visualizar.

- **Visualización de gráficos:**

  Otra fila con dos columnas:

  * La columnas de la izquierda (ancho 4) muestra el mapa.
  * La columna de la derecha (ancho 8) muestra el gráfico de líneas.

- **Consideraciones de usabilidad y experiencia de usuario:**
  - **Interfaz intuitiva:** La disposición de los filtros y gráficos permite que el usuario entienda rápidamente cómo interactuar con el dashboard.
  - **Actualización en tiempo real:** El uso de callbacks garantiza que los gráficos se actualicen de manera instantánea al cambiar cualquiera de los parámetros.
  - **Visualización atractiva:** La integración de Bootstrap ayuda a ofrecer una interfaz limpia y moderna, que mejora la experiencia visual general.

2 Explicación de la solución

2.1 Lógica de los Callbacks

El archivo define tres callbacks esenciales para la interactividad del dashboard:

- **Callback para actualizar las opciones de Continente:**

  Este callback se activa al cambiar el continente seleccionado. Filtra el dataframe según el continente y extrae las regiones únicas (definidas en la columna `region_custom`).

```
@app.callback(
    Output('region-dropdown', 'options'),
    Input('continente-dropdown', 'value')
)
def update_region_options(continente):
    regiones = df[df['continent'] == continente]['region_custom'].unique()
    return [{'label': r, 'value': r} for r in sorted(regiones)]
```

  Explicación: Con esto, se garantiza que el dropdown de regiones siempre contenga opciones relevantes al continente seleccionado.

- **Callback para seleccionar la Región por defecto:**

  Una vez actualizadas las opciones, este callback establece por defecto la primera opción disponible en el dropdown de región.

```
@app.callback(
    Output('region-dropdown', 'value'),
    Input('region-dropdown', 'options')
)
def set_default_region(options):
    return options[0]['value'] if options else None
```

  Explicación: Esto mejora la experiencia del usuario al evitar estados vacíos en la selección de región.

- **Callback para actualizar el gráfico y el mapa:**

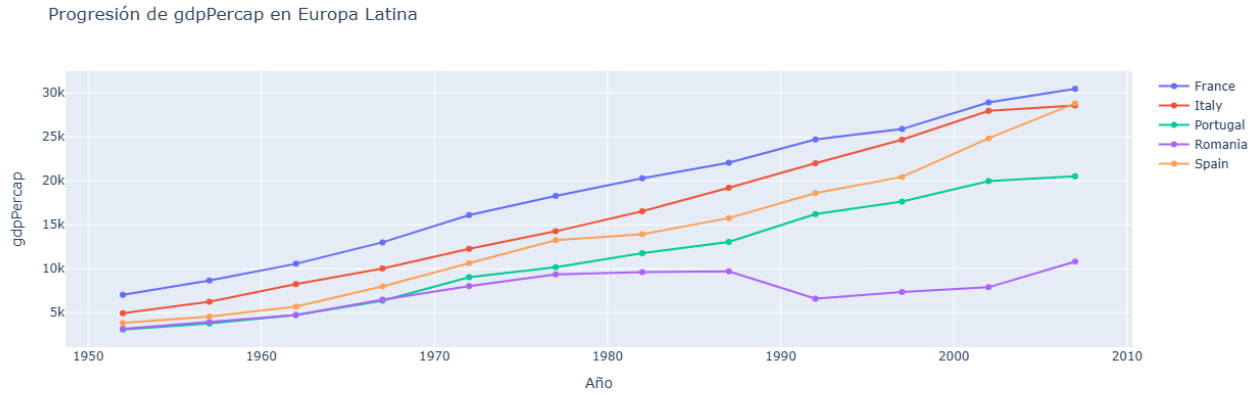
  Este callback, que depende de los tres dropdowns (continente, región y dato seleccionado), filtra el dataframe para extraer la información pertinente y actualiza dos figuras:

– **Gráfico de líneas:**

Se itera sobre cada país en la región filtrada para dibujar una línea de evolución temporal del indicador escogido.

```
for pais in paises:
    df_pais = filtered_df[filtered_df["country"] == pais]
    fig_lineas.add_trace(go.Scatter(
        x=df_pais['year'],
        y=df_pais[dato],
        mode='lines+markers',
        name=pais,
        line=dict(color=color_map[pais])
    ))
```

Explicación: De esta forma se puede comparar visualmente la evolución de cada país dentro de la región.

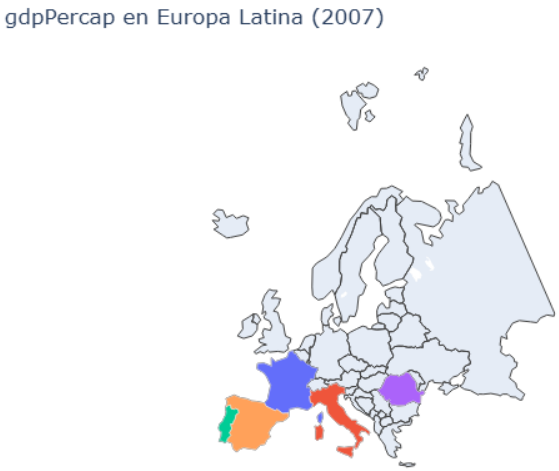


– **Mapa Choropleth:**

Se genera un mapa que destaca los países de la región en el último año registrado, aplicando una escala de colores personalizada para cada país.

```
for _, row in df_mapa.iterrows():
    fig_mapa.add_trace(go.Choropleth(
        locations=[row['country']],
        locationmode='country names',
        z=[1],
        showscale=False,
        marker=dict(line=dict(color='white', width=0.5)),
        name=row['country'],
        colorscale=[[0, color_map[row['country']]], [1, color_map[row['country']]]]
    ))
```

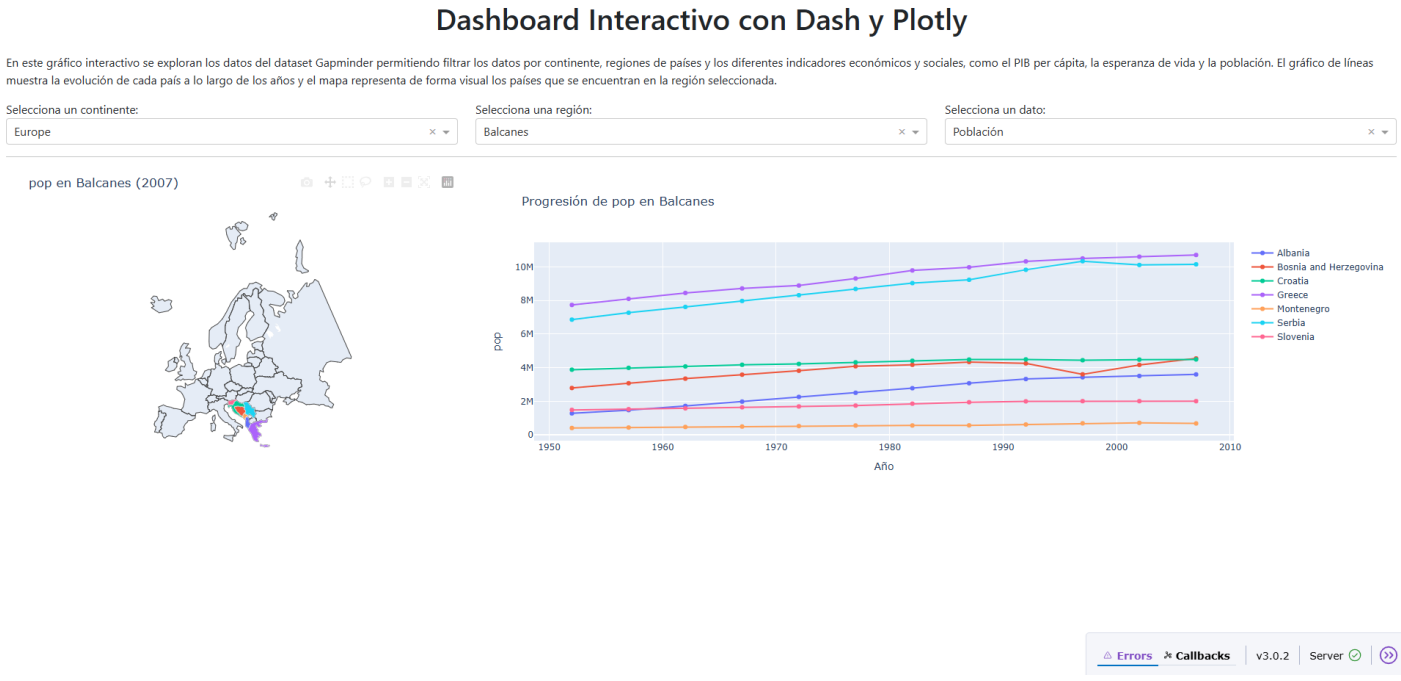
Explicación: La función `update_geos` se usa para ajustar el zoom del mapa en función del continente seleccionado, mejorando la legibilidad geográfica.



Nota: Se asigna un color único a cada país basado en la lista de colores de Plotly para mantener la coherencia visual entre el gráfico de líneas y el mapa.

2.2 Procesamiento y filtrado de datos

- **Filtrado inicial:** Se filtra el dataframe `df` de acuerdo con el continente y la región personalizada (`region_custom`) seleccionados.
- **Asignación de colores:** Los países se ordenan alfabéticamente y se asigna un color de la paleta `qualitative.Plotly` de forma cíclica para visualizarlos de manera diferenciada.
- **Selección del año en el mapa:** Se utiliza el año más reciente del dataset para filtrar los datos y reflejar la información actual en el mapa.



3 Conclusiones

3.1 Retos y desafíos encontrados

- **Sincronización de componentes:** Lograr que todos los componentes (dropdowns y gráficos) respondieran de manera sincronizada fue uno de los principales desafíos. Se resolvió utilizando callbacks encadenados, de modo que la actualización de opciones y valores predeterminados se realice de forma fluida.
- **Personalización Visual:** La asignación de colores únicos a cada país y la personalización del layout (con Bootstrap) ayudaron a mejorar la experiencia visual, aunque implicaron un manejo cuidadoso de la consistencia en ambas visualizaciones.

3.2 Posibles mejoras del Dashboard

- **Inclusión de más indicadores:** Ampliar el conjunto de indicadores (por ejemplo, tasa de alfabetización o inversión extranjera) para proporcionar una visión más completa.
- **Interactividad adicional:** Incorporar filtros adicionales o interacciones (como selección múltiple, zoom en gráficas o tooltip detallado) para ofrecer una experiencia más rica.