perplexity

Mapas em Dashboards para Web Apps HTML com Python e Plotly

O **Plotly** é uma poderosa biblioteca Python que oferece diversas opções para criar mapas interativos em dashboards web. Para web applications HTML, existem vários tipos de mapas e abordagens que podem ser implementadas usando Plotly junto com frameworks como Dash.

Tipos de Mapas Disponíveis no Plotly

O Plotly oferece duas categorias principais de mapas para dashboards web:

1. Mapas Baseados em Tiles (Mapbox/MapLibre)

Estes mapas utilizam camadas de tiles para renderização e incluem:

- `px.scatter_map` / `go.Scattermap` - Para pontos de dispersão geográfica[1]
- `px.density_map` / `go.Densitymap` - Para mapas de densidade/calor[2][1]
- `px.choropleth_map` / `go.Choroplethmap` - Para mapas coropléticos com tiles[3]
- `px.line_map` - Para linhas conectando pontos geográficos[4]

2. Mapas Baseados em Contorno (Outline)

Estes mapas utilizam contornos geográficos naturais:

- `px.scatter_geo` / `go.Scattergeo` - Para pontos geográficos[5][6]
- `px.choropleth` / `go.Choropleth` - Para mapas coropléticos[3][7]
- `px.line_geo` - Para linhas em mapas geográficos[8]

Como Usar Mapas em Dashboards Web

1. Mapas de Densidade (Densidade/Calor)

Os mapas de densidade são ideais para visualizar concentrações de dados em áreas geográficas[9][2]:

```
import plotly.express as px
import pandas as pd

# Carregando dados
df = pd.read_csv('dados_com_lat_lon.csv')
```

```
# Criando mapa de densidade
fig = px.density_map(
    df,
    lat='latitude',
    lon='longitude',
    z='quantidade',
    radius=10,
    center=dict(lat=-14, lon=-55),  # Centro do Brasil
    zoom=3,
    map_style="open-street-map"
)

fig.show()
```

Para usar o **density_mapbox** (versão anterior):

```
fig = px.density_mapbox(
    df,
    lat='geolocation_lat',
    lon='geolocation_lng',
    z='quantidade',
    mapbox_style="open-street-map",
    zoom=3,
    radius=10
)
```

2. Mapas Coropléticos (Choropleth)

Mapas coropléticos coloriem regiões baseadas em dados estatísticos[9][3][10]:

```
import plotly.express as px
import json

# Carregando arquivo GeoJSON para delimitar regiões
with open('brasil_estados.geojson') as f:
    geojson = json.load(f)

# Criando mapa coroplético
fig = px.choropleth(
    df,
    locations='codigo_estado',
    color='valor_estatistico',
    geojson=geojson,
    featureidkey="properties.codigo",
    title='Estatísticas por Estado'
)

fig.update_geos(fitbounds="locations", visible=False)
fig.show()
```

3. Mapas de Dispersão Geográfica

Para plotar pontos específicos em mapas[6]:

```
fig = px.scatter_geo(
    df,
    locations="iso_alpha",
    size="populacao",
    color="continente",
    hover_name="pais",
    projection="natural earth"
)

fig.show()
```

Integração com Dashboards Web usando Dash

Para criar dashboards web interativos, use o **Dash**[11][12][13]:

```
from dash import Dash, html, dcc, Input, Output, callback
import plotly.express as px
import pandas as pd

# Inicializando app Dash
app = Dash(__name__)

# Layout do dashboard
app.layout = html.Div([
    html.H1("Dashboard de Mapas"),

    # Dropdown para seleção
    dcc.Dropdown(
        id='tipo-mapa',
        options=[
            {'label': 'Mapa de Densidade', 'value': 'density'},
            {'label': 'Mapa Coroplético', 'value': 'choropleth'},
            {'label': 'Scatter Geo', 'value': 'scatter'}
        ],
        value='density'
    ),

    # Componente do gráfico
    dcc.Graph(id='mapa-graph')
])

# Callback para atualizar mapa
@callback(
    Output('mapa-graph', 'figure'),
    Input('tipo-mapa', 'value')
)
def update_map(tipo_selecionado):
    if tipo_selecionado == 'density':
        fig = px.density_map(df, lat='lat', lon='lon', z='valor')
    elif tipo_selecionado == 'choropleth':
```

```
        fig = px.choropleth(df, locations='estado', color='valor')
    else:
        fig = px.scatter_geo(df, locations='codigo', size='valor')

    return fig

# Executando app
if __name__ == '__main__':
    app.run(debug=True)
```

Exportação para HTML

Salvando Mapas como Arquivos HTML

Para integrar mapas em web apps HTML estáticas[14][15]:

```
# Salvando figura como HTML completo
fig.write_html("mapa_interativo.html")

# Salvando apenas o div (para embedar em páginas existentes)
fig.write_html(
    "mapa_div.html",
    full_html=False,
    include_plotlyjs='cdn'
)
```

Configurações de Exportação HTML

- `include_plotlyjs=True` - Inclui Plotly.js completo (~3MB) para uso offline[15]
- `include_plotlyjs='cdn'` - Referencia Plotly.js via CDN (requer internet)[15]
- `include_plotlyjs='directory'` - Assume plotly.min.js no mesmo diretório[15]

Exemplos Práticos de Implementação

Dashboard de Monitoramento Regional

```
# Dashboard para dados COVID-19 ou indicadores regionais
app.layout = html.Div([
    html.H1("Monitoramento Regional"),

    dcc.DatePickerRange(
        id='date-picker',
        start_date='2020-01-01',
        end_date='2023-12-31'
    ),

    dbc.Row([
        dbc.Col([
            dcc.Graph(id='mapa-choropleth')
```

```
        ], width=8),

        dbc.Col([
            dash_table.DataTable(id='tabela-dados')
        ], width=4)
    ])
])
```

Dashboard de Análise de Vendas por Região

```
# Usando dados de e-commerce com coordenadas
fig = px.density_mapbox(
    vendas_df,
    lat='latitude_cliente',
    lon='longitude_cliente',
    z='valor_venda',
    animation_frame='mes',
    title='Vendas por Região ao Longo do Tempo'
)

fig.update_layout(mapbox_style="carto-positron")
```

Melhores Práticas

1. **Performance**: Para grandes datasets, considere agregação de dados antes da visualização[11][12]
2. **Responsividade**: Configure layouts responsivos usando Bootstrap components do Dash[13]
3. **Interatividade**: Implemente callbacks para permitir filtragem e drill-down nos dados[16]
4. **Estilização**: Use temas consistentes e estilos de mapa apropriados para o contexto[9][17]
5. **Acessibilidade**: Inclua hover information e legendas claras para melhor usabilidade[3]

Os mapas do Plotly oferecem uma solução completa para dashboards web, combinando facilidade de uso com alta interatividade. A integração com Dash permite criar aplicações web profissionais com poucas linhas de código, enquanto a capacidade de exportação HTML garante flexibilidade para diferentes tipos de deployment[18][13].