

Interfaces web rápidas

Disciplina: Programação aplicada à engenharia cartográfica

Maurício C. M. de Paulo - D.Sc.

13 de fevereiro de 2026

Introdução ao Streamlit

- Framework Python para aplicações web interativas
- Foco em simplicidade e prototipação rápida
- Executa com:

```
pixi shell  
pixi add streamlit  
streamlit run app.py
```

Principais conceitos:

- Layout declarativo
- Widgets interativos
- Atualização automática da interface

Documentação ([link](#))

Estrutura Básica da Aplicação

```
import streamlit as st
import matplotlib.pyplot as plt
import leafmap.foliumap as leafmap

st.title("Exemplo Streamlit + Matplotlib + Leafmap")

st.sidebar.header("Controles")

lat = st.sidebar.number_input("Latitude", value=-15.78)
lon = st.sidebar.number_input("Longitude", value=-47.93)
zoom_btn = st.sidebar.button("Zoom na coordenada")
```

- Sidebar contém widgets
- Valores são reavaliados a cada interação

Adicionando um Gráfico Matplotlib

```
st.subheader("Gráfico Matplotlib")

fig, ax = plt.subplots()
ax.plot([1, 2, 3, 4], [10, 20, 25, 30])
ax.set_title("Exemplo de gráfico")

st.pyplot(fig)
```

- Criamos a figura normalmente
- Usamos st.pyplot() para renderizar

Adicionando Mapa com Leafmap

```
st.subheader("Mapa Leafmap")

m = leafmap.Map(center=[lat, lon], zoom=4)

if zoom_btn:
    m.set_center(lon, lat, zoom=10)

m.add_marker(location=[lat, lon],
popup="Coordenada selecionada")

m.to_streamlit(height=500)
```

- Leafmap usa base Folium
- Botão altera o centro do mapa

Exercício de Streamlit

Analisar:

- HTML gerado
- Requests abertos
- Debug

<https://kepler.gl/>

<https://kepler.gl/> Trabalho de alunos do 2º ano 2025:
<https://mauriciodev.github.io/progcart/ipe2.html>

The screenshot shows the Kepler.gl web application interface. On the left, there's a sidebar with 'Datasets' and 'Layers' sections, and buttons for 'Load Files', 'Tileset', 'Load Map using URL', and 'Load from Storage'. A central dialog box titled 'Add Data To Map' is open, prompting the user to 'Upload CSV, Json, GeoJSON, Arrow, Parquet or saved map Json'. It includes a file upload area with five document icons, a download icon, and a 'Drag & Drop Your File(s) Here' field. Below this, a note states: "'kepler.gl' is a client-side application with no server backend. Data lives only on your machine/browser. No information or map data is sent to any server." The main map view on the right shows a grayscale map of Northern California with various cities labeled: Petaluma, Fairfield, Vallejo, Sausalito, San Francisco, Alameda, Berkeley, Walnut Creek, Concord, Pleasant Hill, Martinez, Benicia, Vallejo, Napa, Rohnert Park, Santa Rosa, Sebastopol, Healdsburg, Geyserville, Sonoma, Petaluma, Sebastopol, Santa Rosa, and Hopland. A message in the top right corner says 'Kepler.gl 3.1 + DuckDB is here! Click here to check out the preview of Kepler.gl 3.1 with DuckDB enabled.'

Dash e Plotly

Dash

Rerun

Rerun