

# Interfaces web rápidas

Disciplina: Programação aplicada à engenharia cartográfica

Maurício C. M. de Paulo - D.Sc.

27 de fevereiro de 2026

# HTML Form → Flask (WSGI)

## Fluxo da aplicação:

- 1 Usuário preenche formulário HTML
- 2 Navegador envia requisição HTTP (POST)
- 3 Flask processa via rota (@app.route)
- 4 Servidor retorna HTML como resposta

## Flask:

- Microframework web em Python
- Baseado em WSGI

## Execução:

- pixi add flask
- pixi run python app.py

# Streamlit

Isso tudo é muito bacana, mas não parece prático. Tem algo mais prático?

# Streamlit

Isso tudo é muito bacana, mas não parece prático. Tem algo mais prático?

## Backend e Frontend integrados: Streamlit

# Exemplo — HTML

```
<!DOCTYPE html>
<html>
<body>

<form action="/" method="post">

Nome:
<input type="text" name="nome"><br><br>

Curso:
<select name="curso">
<option value="python">Python</option>
<option value="geoprocessamento">Geoprocessamento</option>
<option value="web">Web</option>
</select><br><br>

<input type="submit" value="Enviar">

</form>

</body>
</html>
```

# Exemplo — Servidor Flask

```
from flask import Flask, request
app = Flask(__name__)

@app.route("/", methods=["GET", "POST"])
def index():
    if request.method == "POST":
        nome = request.form.get("nome", "")
        curso = request.form.get("curso", "")
        return f"""<h2>Dados recebidos no servidor</h2>
Nome: {nome}<br>
Curso: {curso}
"""
    else: return """<form method="post">
Nome: <input type="text" name="nome"><br><br>
Curso:
<select name="curso">
    <option value="python">Python</option>
    <option value="geoprocessamento">Geoprocessamento</option>
    <option value="web">Web</option>
</select><br><br>
<input type="submit" value="Enviar">
</form>"""
if __name__ == "__main__":
    app.run(debug=True)
```

# 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', 'Tiles', 'Load Map using URL', and 'Load from Storage'. A central modal window 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, specifically the San Francisco Bay Area and surrounding regions like Sonoma, Napa, and Solano counties. Labeled cities include Petaluma, Fairfield, Vallejo, Sausalito, San Francisco, Alameda, Berkeley, Walnut Creek, Concord, Pleasant Hill, Martinez, Benicia, Vallejo, Vacaville, Fairfield, Rohnert Park, Santa Rosa, Sebastopol, Healdsburg, Geyserville, and Hopland. The map also shows various rivers and landmarks.

# Dash e Plotly

Dash

Rerun

Rerun