Estructura del proyecto

```
ruta_app/
├── app.py
├── templates/
│   └── index.html
├── static/
│   └── styles.css
└── dijkstra.py
```

Logistica del algoritmo Dijkstra.py

```
import heapq

def dijkstra(graph, start):
    distances = {node: float('inf') for node in graph}
    distances[start] = 0
    queue = [(0, start)]
    previous_nodes = {}

    while queue:
        current_distance, current_node = heapq.heappop(queue)

        for neighbor, weight in graph[current_node].items():
            distance = current_distance + weight

            if distance < distances[neighbor]:
                distances[neighbor] = distance
                previous_nodes[neighbor] = current_node
```

```
            heapq.heappush(queue, (distance, neighbor))

    return distances, previous_nodes

def get_path(previous_nodes, start, end):
    path = []
    current = end
    while current != start:
        path.insert(0, current)
        current = previous_nodes.get(current)
        if current is None:
            return []
    path.insert(0, start)
    return path
```

appy.py-Aplicaciòn Flask

```
from flask import Flask, render_template, request, jsonify
from dijkstra import dijkstra, get_path

app = Flask(_name_)

# Grafo de ejemplo (nodos = lugares, pesos = distancias)
graph = {
    'A': {'B': 5, 'C': 2},
    'B': {'D': 4},
    'C': {'B': 1, 'D': 7},
```

```
    'D': {}
}

# Coordenadas ficticias para mapa
locations = {
    'A': (40.7128, -74.0060),
    'B': (40.7138, -74.0030),
    'C': (40.7148, -74.0090),
    'D': (40.7158, -74.0050)
}

@app.route("/")
def index():
    return render_template("index.html", locations=locations)

@app.route("/ruta", methods=["POST"])
def ruta():
    data = request.get_json()
    origen = data["origen"]
    destino = data["destino"]
    _, prev = dijkstra(graph, origen)
    ruta = get_path(prev, origen, destino)
    coordenadas = [locations[nodo] for nodo in ruta]
    return jsonify({"ruta": ruta, "coordenadas": coordenadas})

if _name_ == "_main_":
```

```
  app.run(debug=True)
```

Interfaz- templates/index.html

```
<!DOCTYPE html>
<html>
<head>
 <title>Planificador de Rutas</title>
 <link rel="stylesheet" href="https://unpkg.com/leaflet/dist/leaflet.css" />
 <style>#map { height: 500px; }</style>
</head>
<body>
 <h1>Planificador de Rutas</h1>
 <form id="formulario">
  Origen:
  <select id="origen">
   {% for nombre in locations.keys() %}
    <option value="{{ nombre }}">{{ nombre }}</option>
   {% endfor %}
  </select>
  Destino:
  <select id="destino">
   {% for nombre in locations.keys() %}
    <option value="{{ nombre }}">{{ nombre }}</option>
   {% endfor %}
  </select>
  <button type="submit">Planificar Ruta</button>
```

```
 </form>

 <div id="map"></div>

 <script src="https://unpkg.com/leaflet/dist/leaflet.js"></script>
 <script>
  const mapa = L.map('map').setView([40.7138, -74.0060], 14);
  L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png').addTo(mapa);

  document.getElementById('formulario').onsubmit = async (e) => {
   e.preventDefault();
   const origen = document.getElementById('origen').value;
   const destino = document.getElementById('destino').value;
   const res = await fetch('/ruta', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ origen, destino })
   });
   const data = await res.json();
   const ruta = data.coordenadas.map(p => [p[0], p[1]]);
   L.polyline(ruta, {color: 'blue'}).addTo(mapa);
   mapa.fitBounds(ruta);
  };
 </script>
</body>
</html>
```