UNIVERSIDAD MAYOR DE SAN ANDRÉS

FACULTAD DE CIENCIAS PURAS Y NATURALES

CARRERA DE INFORMÁTICA

Geolocalización y búsqueda de lugares cercanos

(Duplas 17,18)

DOCENTE: Lic. Celia Elena Tarquino Peralta

UNIVERSITARIOS:

Quispe Mamani Juan Gabriel

Quispe Tancara Aracely Katy

Toledo Lopez Milton Josue

Villarroel Garvizu Milton Alejandro

LA PAZ – BOLIVIA

2025

1. Introducción

El proyecto Geolocalización y búsqueda de lugares cercanos tiene como objetivo permitir a los usuarios encontrar lugares cercanos en Bolivia mediante el uso de la geolocalización. Utilizando Flask para el backend, Redis para almacenar los lugares y Leaflet.js para mostrar un mapa interactivo, la aplicación facilita la búsqueda de lugares como restaurantes, hospitales, tiendas, entre otros, dentro de un radio específico a partir de una ubicación dada.

2. Objetivos del Proyecto:

- Buscar lugares cercanos mediante la geolocalización del usuario.
- Interacción en tiempo real con un mapa interactivo utilizando Leaflet.js.
- Filtrar por tipo de lugar (restaurantes, tiendas, etc.) y radio de búsqueda.
- Almacenar y gestionar los datos de los lugares usando Redis como base de datos.

3. Tecnologías Utilizadas

3.1. Backend:

- Python: Lenguaje de programación principal.
- Flask: Framework web utilizado para crear el servidor y manejar las solicitudes HTTP.
- Redis: Base de datos en memoria utilizada para almacenar y consultar los lugares cercanos.

3.2. Frontend:

- Leaflet.js: Biblioteca JavaScript para crear mapas interactivos.
- Bootstrap: Framework CSS para crear una interfaz de usuario responsiva y moderna.

- JavaScript: Lenguaje de programación para la interacción del usuario en el navegador.

4. Instalación

4.1. Requisitos Previos:

- Python 3.7 o superior.
- Redis (puede ser local o en la nube, dependiendo de tu preferencia).
- Un navegador moderno para probar la aplicación.
- pip install -r requirements.txt
    - Flask
    - redis
    - requests

5. Uso de la Aplicación

5.1. Iniciar la búsqueda

- Puedes seleccionar una ubicación en el mapa o usar el GPS para obtener tu ubicación actual.
- Selecciona un tipo de lugar (por ejemplo, restaurante, tienda, etc.).
- Establece un radio de búsqueda para definir la distancia en metros.
- Haz clic en el botón Buscar para obtener los lugares cercanos a tu ubicación.

5.2. Visualización de resultados:

- Los lugares encontrados se mostrarán en el mapa interactivo.
- Además, se presentarán en una lista de resultados, con el nombre y la distancia de cada lugar desde tu ubicación.

5.3. Limpiar la búsqueda:

- Al hacer clic en el botón Limpiar todo, se eliminan todos los marcadores y resultados de la búsqueda.

6. Estructura del Proyecto

GeoRedis/

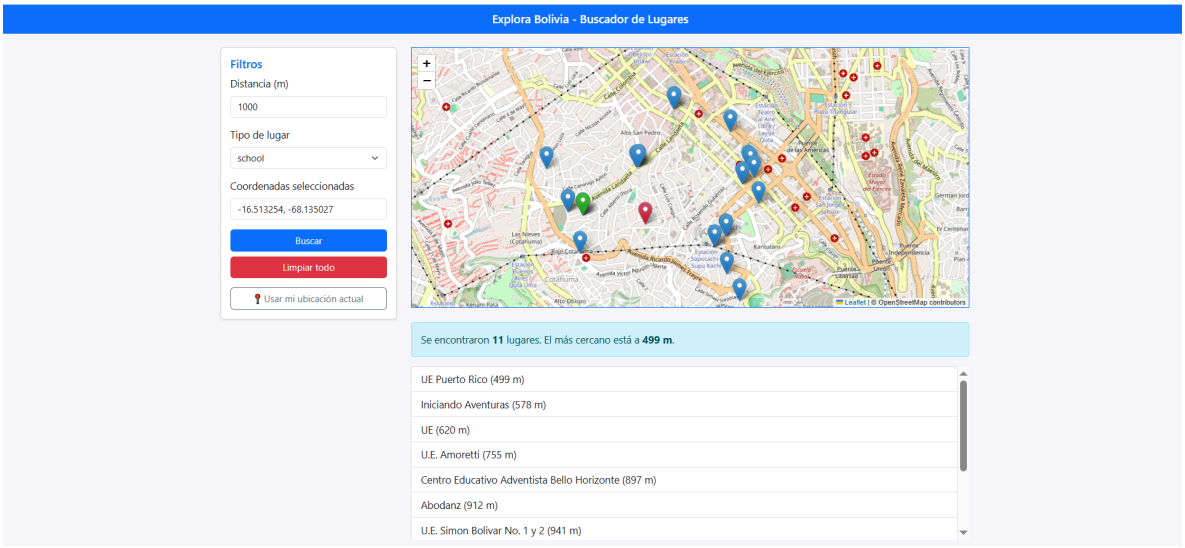
```
├── app.py            # Archivo principal de Flask para manejar la aplicación web
├── api.py          # Funciones auxiliares para manejar la lógica del backend
├── tipos.py          # Otras funciones y utilidades del proyecto
├── requerimientos.txt    # Lista de dependencias de Python
└── templates/
    └── index.html      # Archivo HTML con la interfaz de usuario
```

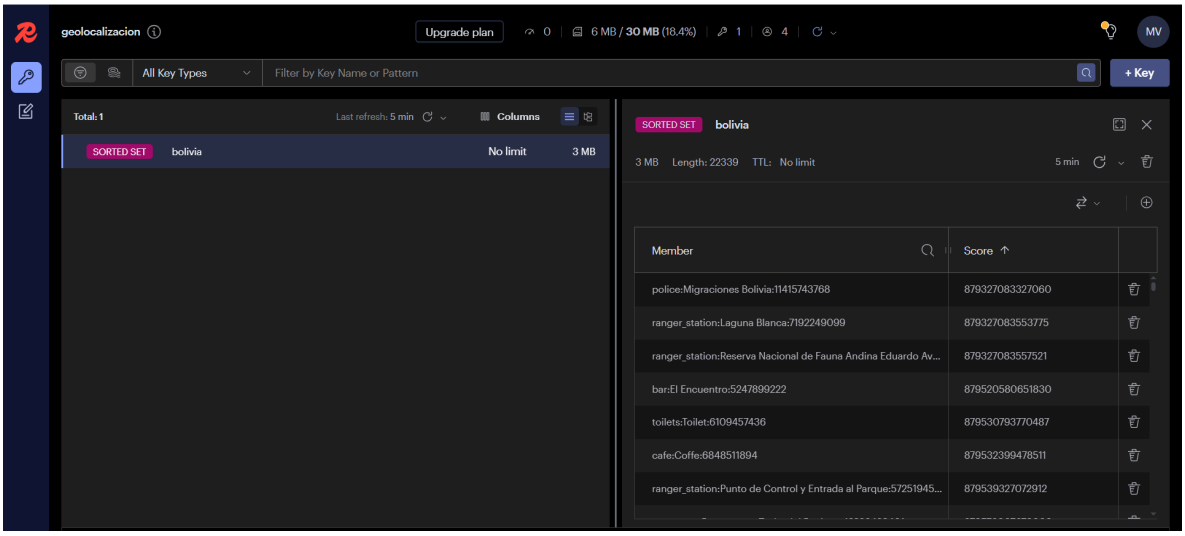
- app.py: Contiene el código de la aplicación Flask, incluyendo las rutas y la lógica para procesar las solicitudes de búsqueda y mostrar los resultados.
- script1.py y script2.py: Contienen funciones auxiliares que ayudan a procesar los datos o manejar tareas específicas (como consultar Redis).
- requirements.txt: Archivo que contiene las dependencias necesarias para ejecutar el proyecto.
- templates/index.html: Plantilla HTML que define la interfaz de usuario de la aplicación.

7. Cadena de Conexión

```
r = redis.Redis(
     host='redis-17135.c232.us-east-1-2.ec2.redns.redis-cloud.com',
     port=17135,
     decode_responses=True,
     username="default",
     password="1234",
 )
```

8. Capturas


Explora Bolivia - Buscador de Lugares
Filtros
Distancia (m)
1000
Tipo de lugar
school
Coordenadas seleccionadas
-16.513254, -68.135027
Buscar
Limpiar todo
Usar mi ubicación actual
Leaflet | © OpenStreetMap contributors
Se encontraron 11 lugares. El más cercano está a 499 m.
UE Puerto Rico (499 m)
Iniciando Aventuras (578 m)
UE (620 m)
U.E. Amoretti (755 m)
Centro Educativo Adventista Bello Horizonte (897 m)
Abodanz (912 m)
U.E. Simon Bolivar No. 1 y 2 (941 m)


geolocalizacion
Upgrade plan
6 MB / 30 MB (18.4%)
MV
All Key Types
Filter by Key Name or Pattern
+ Key
Total: 1
Last refresh: 5 min
Columns
SORTED SET
bolivia
No limit
3 MB
SORTED SET
bolivia
3 MB
Length: 22339
TTL: No limit
5 min
Member
Score
police:Migraciones Bolivia:11415743768
879327083327060
ranger_station:Laguna Blanca:7192249099
879327083553775
ranger_station:Reserva Nacional de Fauna Andina Eduardo Av...
879327083557521
bar:El Encuentro:5247899222
879520580651830
toilets:Toilet:6109457436
879530793770487
cafe:Coffe:6848511894
879532399478511
ranger_station:Punto de Control y Entrada al Parque:57251945...
879539327072912

9. Scripts

9.1. Api.py

```
import requests
import redis
def cargarDatos():
    r = redis.Redis(
        host='redis-17135.c232.us-east-1-2.ec2.redns.redis-cloud.com',
        port=17135,
        decode_responses=True,
        username="default",
        password="1234",
    )
    print("Conectando")
    # Consulta Overpass para amenities"
    query = """
    [out:json][timeout:180];
    area["name"="Bolivia"]["admin_level"="2"]->.bolivia;
    (
    node["amenity"](area.bolivia);
    );
    out center;
    """
    # Hacer la solicitud
            response   =   requests.post("http://overpass-api.de/api/interpreter",
data=query)
    data = response.json()
    # Iniciar un pipeline
    pipe = r.pipeline()
    # Procesar y agregar los comandos al pipeline
    for element in data.get("elements", []):
        tags = element.get("tags", {})
        nombre = tags.get("name")
        tipo = tags.get("amenity")
        latitud = element.get("lat")
        longitud = element.get("lon")
        id = element.get("id")

        if not tipo or not latitud or not longitud or not nombre or not id:
            continue

        miembro = f"{tipo}:{nombre}:{id}"
        pipe.geoadd("bolivia", (longitud, latitud, miembro))

    # Ejecutar todos los comandos en un solo envío
    try:
        pipe.execute()
        print("✅ Todos los lugares fueron guardados correctamente.")
    except Exception as e:
        print(f"❌ Error durante la ejecución del pipeline: {e}")
```

9.2. App.py

```
from flask import Flask, render_template, request, jsonify
import redis
from tipos import amenities
from api import cargarDatos

# cargarDatos()

app = Flask(__name__)

# Conectar a Redis
r = redis.Redis(
    host='redis-17135.c232.us-east-1-2.ec2.redns.redis-cloud.com',
    port=17135,
    decode_responses=True,
    username="default",
    password="1234",
)

@app.route('/')
def index():
    # cargarDatos()
    tipos = amenities()
    return render_template('index.html', tipos=tipos)

@app.route('/buscar', methods=['POST'])
def buscar():
    data = request.get_json()
    lat = float(data['lat'])
    lon = float(data['lon'])
    radio = int(data['radio'])
    tipo = data['tipo']

        lugares = r.georadius('bolivia', lon, lat, radio, unit='m', withdist=True,
withcoord=True)

    resultados = []
    for lugar in lugares:
        nombre = lugar[0]
        if nombre.startswith(tipo + ":"):
            partes = nombre.split(":", 2)  # tipo:nombre:id
            if len(partes) == 3:
                _, nombre_real, _ = partes
            else:
                nombre_real = nombre

            resultados.append({
                'nombre': nombre_real,
                'distancia': lugar[1],
                'lat': lugar[2][1],
                'lon': lugar[2][0]
            })

    return jsonify(resultados)

@app.route('/cerrar', methods=['POST'])
def cerrar():
    print("⚠️ El usuario cerró la página.")
    return '', 204  # Sin contenido, respuesta rápida

if __name__ == '__main__':
    try:
        app.run(debug=True)
    finally:
        print("🧹 Flask cerrando, limpiando Redis...")
```

9.3. Tipos.py

```
import redis

def amenities():
    # Conexión a Redis
    r = redis.Redis(
        host='redis-17135.c232.us-east-1-2.ec2.redns.redis-cloud.com',
        port=17135,
        decode_responses=True,
        username="default",
        password="1234"
    )

    # Obtener todos los miembros del GeoSet
    lugares = r.zrange("bolivia", 0, -1)

    # Extraer el tipo (primer segmento antes del primer ':')
    tipos_unicos = set()
    for lugar in lugares:
        partes = lugar.split(":", 1)
        if len(partes) >= 2:
            tipos_unicos.add(partes[0])

    # Mostrar tipos ordenados
    tipos_ordenados = sorted(tipos_unicos)
    return tipos_ordenados
```

9.4. Index.html

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Buscador de Lugares - Bolivia</title>
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css">
  <link rel="stylesheet" href="https://unpkg.com/leaflet/dist/leaflet.css" />
  <style>
    body {
      font-size: 1.1rem;
    }
    #mapa {
      height: 450px;
      border: 2px solid #0d6efd;
    }
    .centrado {
      text-align: center;
    }
    .scrollable-list {
      max-height: 300px;
      overflow-y: auto;
      cursor: pointer;
    }
  </style>
</head>
<body class="bg-light">
  <nav class="navbar navbar-dark bg-primary mb-4">
    <div class="container-fluid justify-content-center">
      <span class="navbar-brand mb-0 h1">Explora Bolivia - Buscador de Lugares</span>
    </div>
  </nav>

  <div class="container">
    <div class="row">
      <!-- Panel lateral de filtros -->
      <div class="col-md-3">
        <div class="card mb-4 shadow-sm">
          <div class="card-body">
            <h5 class="card-title text-primary">Filtros</h5>
            <form id="formulario">
              <div class="mb-3">
                <label for="radio" class="form-label">Distancia (m)</label>
                <input type="number" id="radio" name="radio" value="1000"
class="form-control" required>
              </div>
              <div class="mb-3">
                <label for="tipo" class="form-label">Tipo de lugar</label>
                <select id="tipo" name="tipo" class="form-select">
                  {% for t in tipos %}
                    <option value="{{ t }}">{{ t }}</option>
                  {% endfor %}
                </select>
              </div>
              <div class="mb-3">
                <label for="coords" class="form-label">Coordenadas seleccionadas</label>
                <input type="text" id="coords" class="form-control" readonly>
              </div>
              <button type="submit" class="btn btn-primary w-100">Buscar</button>
```

```
              <button type="button" id="limpiar" class="btn btn-danger w-100
mt-2">Limpiar todo</button>
            </form>
            <button id="usarGPS" class="btn btn-outline-secondary mt-3 w-100">📍 Usar mi
ubicación actual</button>
          </div>
        </div>
      </div>

      <!-- Mapa y resultados -->
      <div class="col-md-9">
        <div id="mapa" class="mb-4"></div>
        <div id="estadisticas" class="mb-3"></div>
        <div id="resultados" class="scrollable-list"></div>
      </div>
    </div>
  </div>

  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"></scri
pt>
  <script src="https://unpkg.com/leaflet/dist/leaflet.js"></script>
  <script>
    let mapa = L.map('mapa').setView([-16.5, -68.15], 13);
    let marcadorUsuario = null;
    let marcadores = [];
    let marcadorSeleccionado = null;

    // Íconos para los marcadores
    const iconoRojo = L.icon({
      iconUrl:
'https://raw.githubusercontent.com/pointhi/leaflet-color-markers/master/img/marker-icon-r
ed.png',
      shadowUrl: 'https://unpkg.com/leaflet@1.7.1/dist/images/marker-shadow.png',
      iconSize: [25, 41],
      iconAnchor: [12, 41],
      popupAnchor: [1, -34],
      shadowSize: [41, 41]
    });

    const iconoAzul = L.icon({
      iconUrl:
'https://raw.githubusercontent.com/pointhi/leaflet-color-markers/master/img/marker-icon-b
lue.png',
      shadowUrl: 'https://unpkg.com/leaflet@1.7.1/dist/images/marker-shadow.png',
      iconSize: [25, 41],
      iconAnchor: [12, 41],
      popupAnchor: [1, -34],
      shadowSize: [41, 41]
    });

    const iconoVerde = L.icon({
      iconUrl:
'https://raw.githubusercontent.com/pointhi/leaflet-color-markers/master/img/marker-icon-g
reen.png',
      shadowUrl: 'https://unpkg.com/leaflet@1.7.1/dist/images/marker-shadow.png',
      iconSize: [25, 41],
      iconAnchor: [12, 41],
      popupAnchor: [1, -34],
      shadowSize: [41, 41]
    });
```

```
    L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
      attribution: '&copy; OpenStreetMap contributors'
    }).addTo(mapa);

    // Actualiza las coordenadas en el formulario
    function actualizarCoordenadas(latlng) {
      document.getElementById("coords").value = `${latlng.lat.toFixed(6)},
${latlng.lng.toFixed(6)}`;
    }

    // Usar la ubicación GPS
    document.getElementById("usarGPS").addEventListener("click", function() {
      if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(function(pos) {
          const latlng = [pos.coords.latitude, pos.coords.longitude];
          if (marcadorUsuario) mapa.removeLayer(marcadorUsuario);
          marcadorUsuario = L.marker(latlng, { icon: iconoRojo }).addTo(mapa);
          marcadorUsuario.latlng = L.latLng(latlng);
          mapa.setView(latlng, 15);
          actualizarCoordenadas(marcadorUsuario.latlng);
        }, function() {
          alert("No se pudo obtener tu ubicación.");
        });
      } else {
        alert("Tu navegador no soporta geolocalización.");
      }
    });

    // Clic en el mapa
    mapa.on('click', function(e) {
      if (marcadorUsuario) mapa.removeLayer(marcadorUsuario);
      marcadorUsuario = L.marker(e.latlng, { icon: iconoRojo }).addTo(mapa);
      marcadorUsuario.latlng = e.latlng;
      actualizarCoordenadas(e.latlng);
    });

    // Buscar lugares
    document.getElementById("formulario").addEventListener("submit", async function(e) {
      e.preventDefault();
      if (!marcadorUsuario) {
        alert("Selecciona una ubicación en el mapa o usa el GPS");
        return;
      }

      const tipo = document.getElementById("tipo").value;
      const radio = document.getElementById("radio").value;

      const res = await fetch("/buscar", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({
          lat: marcadorUsuario.latlng.lat,
          lon: marcadorUsuario.latlng.lng,
          radio: radio,
          tipo: tipo
        })
      });

      const lugares = await res.json();
      const resultadoDiv = document.getElementById("resultados");
      const estadisticasDiv = document.getElementById("estadisticas");
      resultadoDiv.innerHTML = "";
```

```
      estadisticasDiv.innerHTML = "";
      marcadores.forEach(m => mapa.removeLayer(m));
      marcadores = [];

      // Ordenar por distancia ascendente
      lugares.sort((a, b) => a.distancia - b.distancia);

      if (lugares.length === 0) {
        resultadoDiv.innerHTML = "<div class='alert alert-warning'>No se encontraron
lugares con los filtros seleccionados.</div>";
      } else {
        estadisticasDiv.innerHTML = `
          <div class='alert alert-info'>
            Se encontraron <strong>${lugares.length}</strong> lugares. El más cercano
está a <strong>${Math.round(lugares[0].distancia)} m</strong>.
          </div>`;

        const lista = document.createElement("ul");
        lista.className = "list-group";
        lugares.forEach((lugar, index) => {
          const item = document.createElement("li");
          item.className = "list-group-item d-flex align-items-center";
          item.textContent = `${lugar.nombre} (${Math.round(lugar.distancia)} m)`;

          // Añadir clic en la lista
          item.addEventListener("click", function() {
            // Centrar el mapa en el marcador seleccionado y cambiar el ícono a verde
            mapa.setView([lugar.lat, lugar.lon], 15);
            if (marcadorSeleccionado) {
              marcadorSeleccionado.setIcon(iconoAzul);
            }
            marcadorSeleccionado = L.marker([lugar.lat, lugar.lon], { icon: iconoVerde
}).addTo(mapa)
              .bindPopup(lugar.nombre);
          });

          lista.appendChild(item);

          const marcador = L.marker([lugar.lat, lugar.lon], { icon: iconoAzul
}).addTo(mapa)
            .bindPopup(lugar.nombre);
          marcadores.push(marcador);
        });
        resultadoDiv.appendChild(lista);
      }
    });

    // Limpiar todo
    document.getElementById("limpiar").addEventListener("click", function () {
      if (marcadorUsuario) {
        mapa.removeLayer(marcadorUsuario);
        marcadorUsuario = null;
      }
      marcadores.forEach(m => mapa.removeLayer(m));
      marcadores = [];
      document.getElementById("coords").value = "";
      document.getElementById("resultados").innerHTML = "";
      document.getElementById("estadisticas").innerHTML = "";
    });
  </script>
</body>
</html>
```