Nina Nájera - 231088
Mishell Ciprian – 231169
José Rodríguez-21060
Gabriel Quan - 23479

Demostración de las API’s

Documentación de la API a utilizar

OpenStreetMap

Este API fue elegido por su capacidad de localización abierta. Esta es open source y es gratis en su uso; esto mientras se acredite a los autores, así evitando el plagio. Elegimos este porque no tiene requisitos de tanta complejidad.

Este fue inicialmente construido por una comunidad con el propósito de documentar la geografía del mundo. Este llega a ofrecer una gran oportunidad a los proyectos de este tipo, pues inicialmente fue pensado para bloggers, uso diario y fundaciones OSM. Dado a que no se cambiará ninguna información dentro del API, sino que solamente se usará en base a sus datos, no es necesario tener una licencia o operar en los términos privados de su uso.

Ejemplos de cómo hacer llamadas

Documentación

Al usar OpenStreetMap, se necesitará usar Overpass API. Esta es una interfaz de datos flexible y detallada. Esta API permite funciones avanzadas a la hora de pedir un dato en específico. Esta es una base de datos, un lenguaje de consulta y una consulta de criterios

Un ejemplo para la búsqueda de hospitales cercanos podría ser:

```
[out:json];
node["amenity"="hospital"](around:5000,14.6349,-90.5069);
out;
```

Aquí, amenity=hospital: Filtra los nodos que tienen el atributo "amenity" con el valor "hospital". around:5000,14.6349,-90.5069: Busca dentro de un radio de 5000 metros alrededor de la latitud 14.6349 y longitud -90.5069 (Guatemala City). Y por último, out;: Devuelve los resultados en formato JSON.

Código base

```
import okhttp3.OkHttpClient
import okhttp3.Request
import okhttp3.Response
import android.util.Log

val client = OkHttpClient()

// Coordenadas para la consulta (por ejemplo, Guatemala City)
val lat = 14.6349
val lon = -90.5069
val radius = 5000 // Radio de búsqueda en metros (5 km)

// URL de la API Overpass con la consulta para hospitales cercanos
val overpassUrl = "https://overpass-
api.de/api/interpreter?data=[out:json];node[amenity=hospital](around:$radius,$lat,$lon);out;"

// Crear la solicitud HTTP
val request = Request.Builder()
    .url(overpassUrl)
    .build()
```

```
// Ejecutar la solicitud en un hilo separado (recomendado para evitar bloquear el UI thread)
Thread {
    try {
        // Ejecutar la solicitud y obtener la respuesta
        val response: Response = client.newCall(request).execute()
        if (response.isSuccessful) {
            // Obtener la respuesta JSON
            val jsonResponse = response.body?.string()
            // Procesar la respuesta (en este caso, mostrar en el log)
            Log.d("OverpassAPI", jsonResponse ?: "No response")
        } else {
            Log.e("OverpassAPI", "Request failed with code: ${response.code}")
        }
    } catch (e: Exception) {
        Log.e("OverpassAPI", "Error: ${e.message}")
    }
}.start()
```

Hay varias partes por cubrir en este código, pero las partes principales son estas:

- El overpassUrl es la que busca los nodos en el atributo amenity=hospital. En este caso, este fue puesto a 5 kilómetros.

- El OkHttpClient: Este se usa directamente para crear la solicitud al HTTP. Esto combinado con el request, solicitará el HTTP con la URL generada.

- Thread: Usaremos un hilo para evitar el bloqueo del hilo principal de la interfaz del usuario

- Response: Si la solicitud es exitosa (response.isSuccessful), se obtendrá la respuesta en formato JSON. En caso de este simple programa, este imprimirá los logs

Con todo lo anterior mencionado, esta pequeña fracción de código de ejemplo debería de consultar a los hospitales más cercanos y a sus respectivos nombres.

Envío de parámetros

Los parámetros a utilizar en base al código ejemplo serían:

- amenity=hospital: Representa hospitales generales.
- amenity=clinic: Se usa para clínicas que proporcionan atención médica, generalmente de menor escala que un hospital.
- emergency=yes: Etiqueta adicional para instalaciones que ofrecen servicios de emergencia médica

Todos estos en combinación deberán proveer al usuario la información necesaria para poder llamar a emergencias de algún tipo.

Manejo de errores

HTTP

Los errores más comunes relacionados al HTTP podrían llegar a ser:

- 400 (Bad Request): Ocurre cuando está mal formulada la consulta
- 403 (Forbidden): Cuando el servidor API bloquea el acceso a la solicitud
- 404 (Not Found): Cuando la URL es incorrecta o no se encuentra disponible
- 429 (Too many Requests): Ocurre cuando hay muchas solicitudes en un corto período de tiempo
- 500 (Internal Server Error): Error de servidor. Puede ocurrir cuando el servidor de Overpass tiene problemas o está sobrecargado.

Errores de red y conectividad

Dado a que la aplicación se basa en ubicaciones y llamadas de emergencia, entonces es importante que la aplicación esté conectada a internet. Para ello, se implementará de forma automática una conexión a internet. Dado el caso de que esto no pueda ser posible, se solucionaría con el reinicio de la solicitud automática.