### **Sprint 3 – Actividades realizadas**

1️⃣ **Filtrar información relevante** Se implementaron filtros en el mapa para mostrar zonas de riesgo, lugares turísticos o ambos combinados. Se utilizó un Spinner (spinnerFiltro) que llama a los métodos dibujarZonasDesdeFirestore() y mostrarLugaresTuristicos(LatLng location) para actualizar el mapa según la selección del usuario.

**Fragmento de código:**

**// Spinner Filtro**

**String[] filtros = {"Ninguno", "Zonas de riesgo", "Lugares turísticos", "Ambos"};**

**ArrayAdapter<String> adapterFiltro = new ArrayAdapter<>(requireContext(),**

**android.R.layout.*simple\_spinner\_item*, filtros);**

**adapterFiltro.setDropDownViewResource(android.R.layout.*simple\_spinner\_dropdown\_item*);**

**spinnerFiltro.setAdapter(adapterFiltro);**

**spinnerFiltro.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {**

**@Override**

**public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {**

**if (mMap == null || ultimaUbicacion == null) return;**

**limpiarRuta();**

**switch (position) {**

**case 0:**

**dibujarZonasDesdeFirestore();**

**break;**

**case 1:**

**mostrarLugaresTuristicos(new LatLng(ultimaUbicacion.getLatitude(), ultimaUbicacion.getLongitude()));**

**break;**

**case 2:**

**dibujarZonasDesdeFirestore();**

**mostrarLugaresTuristicos(new LatLng(ultimaUbicacion.getLatitude(), ultimaUbicacion.getLongitude()));**

**break;**

**}**

**}**

**// Spinner Modo de transporte**

**Spinner spinnerModo = root.findViewById(R.id.*spinnerModo*);**

**String[] modos = {"A pie", "Bicicleta", "Carro"};**

**ArrayAdapter<String> adapterModo = new ArrayAdapter<>(requireContext(),**

**android.R.layout.*simple\_spinner\_item*, modos);**

**adapterModo.setDropDownViewResource(android.R.layout.*simple\_spinner\_dropdown\_item*);**

**spinnerModo.setAdapter(adapterModo);**

**spinnerModo.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {**

**@Override**

**public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {**

**switch (position) {**

**case 0: modoTransporte = "walking"; break; //**

**case 1: modoTransporte = "bicycling"; break; //**

**case 2: modoTransporte = "driving"; break; //**

**}**

**}**

**@Override**

**public void onNothingSelected(AdapterView<?> parent) { }**

**});**

**2️⃣ Configurar API de rutas  
 Se configuró la Google Directions API para calcular rutas entre un origen y un destino, utilizando Places API para autocompletado. La URL de la API se construye dinámicamente según el modo de transporte y el idioma seleccionado, y la ruta se dibuja en el mapa con Polyline.**

**Fragmento de código:**

**private void calcularRuta(LatLng origen, LatLng destino) {**

**// Limpiar ruta anterior**

**if (rutaActual != null) {**

**rutaActual.remove();**

**rutaActual = null;**

**}**

**// Limpiar marcadores anteriores**

**if (marcadorOrigen != null) marcadorOrigen.remove();**

**if (marcadorDestino != null) marcadorDestino.remove();**

**// Agregar nuevos marcadores**

**marcadorOrigen = mMap.addMarker(new MarkerOptions().position(origen).title("Origen"));**

**marcadorDestino = mMap.addMarker(new MarkerOptions().position(destino).title("Destino"));**

**// Construir URL según modo de transporte**

**String url = construirUrlDirecciones(origen, destino, modoTransporte);**

**RequestQueue queue = Volley.*newRequestQueue*(requireContext());**

**JsonObjectRequest request = new JsonObjectRequest(Request.Method.*GET*, url, null,**

**response -> {**

**try {**

**String status = response.getString("status");**

**Log.*d*("DebugRuta", "Directions API status: " + status);**

**if (!status.equals("OK")) {**

**// Si no hay ruta y el modo era driving, intentar walking**

**if (modoTransporte.equals("driving")) {**

**Log.*d*("DebugRuta", "No se encontró ruta en driving, intentando walking...");**

**String walkingUrl = construirUrlDirecciones(origen, destino, "walking");**

**JsonObjectRequest walkingRequest = new JsonObjectRequest(Request.Method.*GET*, walkingUrl, null,**

**walkingResponse -> procesarRespuestaRuta(walkingResponse, origen, destino),**

**error -> {**

**error.printStackTrace();**

**calcularTiempoAproximado(origen, destino); // fallback si walking también falla**

**});**

**queue.add(walkingRequest);**

**} else {**

**calcularTiempoAproximado(origen, destino); // mostrar tiempo aproximado**

**}**

**return;**

**}**

**// Procesar la respuesta normalmente**

**procesarRespuestaRuta(response, origen, destino);**

**} catch (Exception e) {**

**e.printStackTrace();**

**calcularTiempoAproximado(origen, destino);**

**}**

**},**

**error -> {**

**error.printStackTrace();**

**calcularTiempoAproximado(origen, destino);**

**}**

**);**

**queue.add(request);**

**}**

**3️⃣ Calcular tiempo aproximado  
 Se implementó un cálculo del tiempo estimado de traslado según el modo de transporte (walking, bicycling, driving). Se consideran velocidad promedio y factores de ajuste para simular condiciones reales y se muestra en un TextView.**

**Fragmento de código:**

**private void calcularTiempoAproximado(LatLng origen, LatLng destino) {**

**float[] distancia = new float[1];**

**Location.*distanceBetween*(**

**origen.latitude, origen.longitude,**

**destino.latitude, destino.longitude,**

**distancia**

**);**

**double velocidadMps;**

**switch (modoTransporte) {**

**case "driving":**

**velocidadMps = 4.17; // ~15 km/h promedio urbano realista**

**break;**

**case "bicycling":**

**velocidadMps = 2.78; // ~10 km/h**

**break;**

**default:**

**velocidadMps = 1.11; // ~4 km/h caminando**

**break;**

**}**

**// Factor por curvas, semáforos, tráfico**

**double factorCurvas = 1.10 + Math.*random*() \* 0.15; // +10% a +25% en carro**

**double factorAleatorio = 0.95 + Math.*random*() \* 0.10; // ±5%**

**velocidadMps \*= factorAleatorio / factorCurvas;**

**int tiempoSegundos = (int)(distancia[0] / velocidadMps);**

**int horas = tiempoSegundos / 3600;**

**int minutos = (tiempoSegundos % 3600) / 60;**

**String tiempoTexto = (horas > 0) ? (horas + " h " + minutos + " min") : (minutos + " min");**

**tvTiempo.setText("Tiempo estimado aprox.: " + tiempoTexto);**

**}**