**UNIVERSIDAD CONTINENTAL**

**FACULTAD DE INGENIERÍA**

**ESCUELA ACADÉMICO PROFESIONAL DE INGENIERÍA DE SISTEMAS E INFORMÁTICA**

**CURSO:**

TALLER DE PROYECTOS - I - ING. DE SISTEMAS E INFORMÁTICA

**DOCENTE:**

AMERICO ESTRADA SANCHEZ

**TEMA:**

“SafeRoute: Desarrollo de un Sistema Web para optimización de rutas seguras en Cusco; 2025”

**PRESENTADO POR:**

|  |  |
| --- | --- |
| **APELLIDOS Y NOMBRES** | **CÓDIGO** |
| CASTILLO CCANTO FRANK | 74534208 |
| ESPETIA MAMANI JHON CRISTIAN | 74988478 |
| RAFAELE HUAMAN LUIS CRISTIAN | 74653997 |

**AVANCE SPRINT 4**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **HU** | **Actividad** | **Tiempo(h)** |
| 39 | HU-10 | Mostrar en mapa | 1.46 |
| 40 | HU-11 | Integrar reportes ciudadanos | 2.00 |
| 41 | HU-11 | Algoritmo de rutas seguras | 2.15 |
| 42 | HU-11 | Comparar con ruta rápida | 1.50 |
| 43 | HU-12 | Mostrar opciones (rápida vs segura) | 0.55 |
| 44 | HU-12 | Permitir selección | 0.55 |
| 45 | HU-12 | Mostrar métricas (tiempo/distancia/riesgo) | 0.55 |
| 46 | HU-19 | Configurar notificaciones | 4.00 |
| 47 | HU-19 | Asociar a nivel de riesgo | 2.50 |
| **Total** |  |  | **14.26** |

**Mostrar en mapa**

private void procesarRespuestaDeRutas(JSONObject response)



**Integrar reportes ciudadanos**

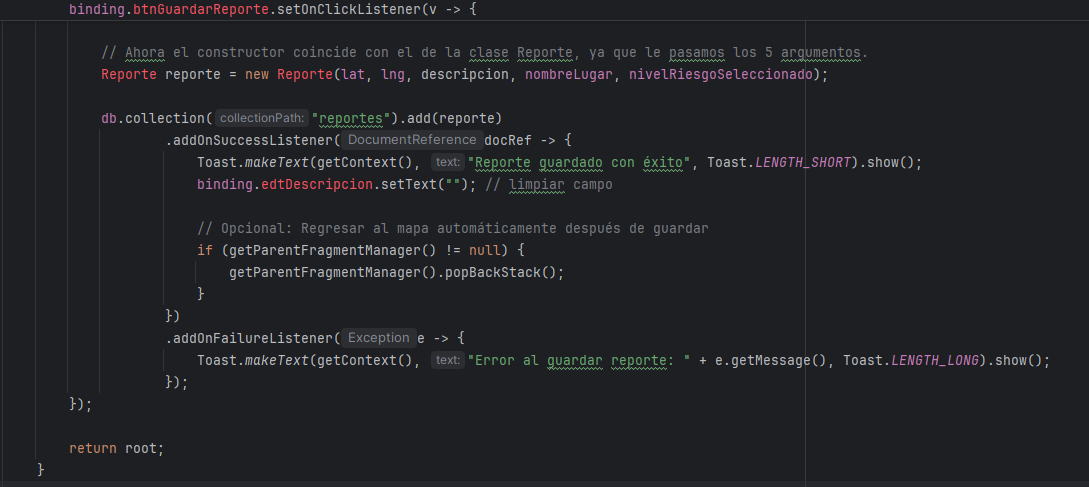
En el fragmento GalleryFragment.java, dentro del método onCreateView(), se implementa un setOnClickListener sobre el botón btnGuardarReporte.

Este bloque de código obtiene los datos ingresados por el usuario (coordenadas, descripción y nivel de riesgo) y crea un objeto de tipo Reporte.

Posteriormente, el reporte se almacena en Firebase Firestore dentro de la colección reportes.

Si el proceso es exitoso, se muestra un mensaje de confirmación al usuario y se limpia el formulario.

En caso de error, se muestra una notificación con el mensaje correspondiente.

****

****

**Algoritmo de rutas seguras**

En este bloque se obtiene la ruta más segura entre dos puntos utilizando los datos recibidos desde la API de Google Maps (Directions API).  
 El código extrae el *polyline* (la codificación de la ruta en el mapa) de la ruta seleccionada como la más segura, la decodifica y la dibuja en el mapa con un color verde distintivo.  
 Además, se muestra una etiqueta (tagSegura) con el tiempo estimado de viaje y el nivel de riesgo calculado.

* polylineSegura: contiene la representación codificada de la ruta.
* rutaMasSeguraIndex: corresponde al índice de la ruta evaluada como más segura según los reportes ciudadanos y los niveles de riesgo.
* dibujarRuta(): función encargada de pintar la línea en el mapa.
* Color.GREEN: representa visualmente la ruta más segura.
* mejorPuntajeRiesgo: valor calculado que determina la seguridad de la ruta.

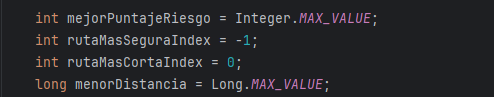
String polylineSegura = routes.getJSONObject(rutaMasSeguraIndex)

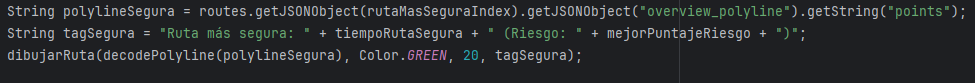
.getJSONObject("overview\_polyline").getString("points");

String tagSegura = "Ruta más segura: " + tiempoRutaSegura +

" (Riesgo: " + mejorPuntajeRiesgo + ")";

dibujarRuta(decodePolyline(polylineSegura), Color.GREEN, 20, tagSegura);

****

****

**Comparar con ruta rápida**

### Decidiendo la Mejor Ruta

Lo que hace es:

1. Recorre todas las rutas alternativas que ofrece Google.
2. Para cada una, calcula la distancia total para encontrar la **más corta**.
3. También calcula un **"puntaje de riesgo"** revisando si la ruta pasa cerca de alguna zona reportada como insegura. La que tenga el puntaje más bajo es la **más segura**.
4. Finalmente, compara ambas y decide cómo dibujarlas.

// En HomeFragment.java

private void procesarRespuestaDeRutas(JSONObject response) {

try {

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

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

// ... manejo de errores ...

return;

}

JSONArray routes = response.getJSONArray("routes");

if (routes.length() == 0) {

// ... manejo si no hay rutas ...

return;

}

int mejorPuntajeRiesgo = Integer.*MAX\_VALUE*;

int rutaMasSeguraIndex = -1;

int rutaMasCortaIndex = 0;

long menorDistancia = Long.*MAX\_VALUE*;

// --- 1. Bucle para analizar cada ruta alternativa ---

for (int i = 0; i < routes.length(); i++) {

JSONObject route = routes.getJSONObject(i);

JSONObject leg = route.getJSONArray("legs").getJSONObject(0);

// --- 2. Encontrar la ruta MÁS CORTA (por distancia) ---

long distanciaActual = leg.getJSONObject("distance").getLong("value");

if (distanciaActual < menorDistancia) {

menorDistancia = distanciaActual;

rutaMasCortaIndex = i;

}

// --- 3. Calcular el RIESGO de esta ruta ---

String polyline = route.getJSONObject("overview\_polyline").getString("points");

List<LatLng> puntos = decodePolyline(polyline);

int puntajeRiesgo = 0;

for (LatLng punto : puntos) {

for (Reporte reporte : listaDeReportes) {

float[] distancia = new float[1];

Location.*distanceBetween*(punto.latitude, punto.longitude, reporte.getLat(), reporte.getLng(), distancia);

if (distancia[0] < 75) { // Si el punto de la ruta está a menos de 75m de un reporte

if (reporte.getRiesgo() == 2) puntajeRiesgo += 1; // Suma 1 por riesgo moderado

if (reporte.getRiesgo() == 3) puntajeRiesgo += 5; // Suma 5 por riesgo alto

}

}

}

// --- 4. Encontrar la ruta MÁS SEGURA (menor puntaje de riesgo) ---

if (puntajeRiesgo < mejorPuntajeRiesgo) {

mejorPuntajeRiesgo = puntajeRiesgo;

rutaMasSeguraIndex = i;

}

}

String tiempoRutaCorta = routes.getJSONObject(rutaMasCortaIndex).getJSONArray("legs").getJSONObject(0).getJSONObject("duration").getString("text");

String tiempoRutaSegura = routes.getJSONObject(rutaMasSeguraIndex).getJSONArray("legs").getJSONObject(0).getJSONObject("duration").getString("text");

// --- 5. Lógica para MOSTRAR LAS OPCIONES ---

if (rutaMasCortaIndex == rutaMasSeguraIndex) {

// Si la más corta es también la más segura, solo se muestra una en verde

tvTiempo.setText("Ruta Óptima (corta y segura): " + tiempoRutaCorta);

String polyline = routes.getJSONObject(rutaMasCortaIndex).getJSONObject("overview\_polyline").getString("points");

String tagUnica = "Ruta Óptima: " + tiempoRutaCorta + " (Riesgo: " + mejorPuntajeRiesgo + ")";

dibujarRuta(decodePolyline(polyline), Color.*GREEN*, 20, tagUnica);

} else {

// Si son diferentes, se muestran ambas opciones

tvTiempo.setText("Comparando Rutas | Azul: Más Corta (" + tiempoRutaCorta + ") | Verde: Más Segura (" + tiempoRutaSegura + ")");

// Dibuja la ruta corta en azul

String polylineCorta = routes.getJSONObject(rutaMasCortaIndex).getJSONObject("overview\_polyline").getString("points");

String tagCorta = "Ruta más corta: " + tiempoRutaCorta;

dibujarRuta(decodePolyline(polylineCorta), Color.*BLUE*, 15, tagCorta);

// Dibuja la ruta segura en verde

String polylineSegura = routes.getJSONObject(rutaMasSeguraIndex).getJSONObject("overview\_polyline").getString("points");

String tagSegura = "Ruta más segura: " + tiempoRutaSegura + " (Riesgo: " + mejorPuntajeRiesgo + ")";

dibujarRuta(decodePolyline(polylineSegura), Color.*GREEN*, 20, tagSegura);

}

} catch (Exception e) {

// ... manejo de errores ...

}

}

**Mostrar (rápida vs segura)**

Se añadió un tag para que, cuando se toque la línea, te muestre un mensaje con su información.

// En HomeFragment.java

private void dibujarRuta(List<LatLng> puntos, int color, float ancho, String tag) {

if (mMap == null) return;

Polyline polyline = mMap.addPolyline(new PolylineOptions()

.addAll(puntos)

.color(color) // Color (Verde o Azul)

.width(ancho) // Grosor de la línea

.clickable(true)); // Hacemos que se pueda tocar

polyline.setTag(tag); // Le asignamos un texto para mostrarlo al hacer clic

polylinesActuales.add(polyline);

}



