**EIF411 DISEÑO Y DESARROLLO DE ACPLICACIONES MOVILES**

**I Ciclo 2023**

**Integrantes**

**Norman Betancourtt Mairena**

**Andrés Rojas Rojas**

**Alejandro Barrantes Castro**

**Instrucciones y Objetivos de la Actividad**

**Objetivos de Aprendizaje**

El objetivo principal de esta actividad es aprender a utilizar la API de Google Maps y las funcionalidades de ubicación en Android, específicamente:

* Acceder a la ubicación del dispositivo a través del GPS.
* Solicitar permisos en tiempo de ejecución.
* Recibir y manejar actualizaciones de la ubicación.
* Mostrar la ubicación en un mapa utilizando Google Maps.
* Mover un marcador en el mapa y obtener la nueva ubicación.

**Instrucciones**

* Formación de Grupos: Formen parejas o grupos de tres. Cada grupo deberá desarrollar su propia versión de la aplicación.
* Entender el Proyecto: Lean y comprendan el código proporcionado. Es fundamental entender cómo funciona el código antes de empezar a trabajar con él. Si tienen dudas, no duden en preguntar.
* Desarrollo de la Aplicación: Utilicen el código proporcionado para desarrollar la aplicación. Asegúrense de seguir todos los pasos descritos en la guía tutorial.
* Pruebas: Una vez que hayan terminado de desarrollar la aplicación, realicen pruebas para asegurarse de que funcione correctamente. Intenten mover el marcador en el mapa y verifiquen que la nueva ubicación se refleje correctamente.
* Desafío: Preparen una breve explicación sobre su complemento. En su presentación, deben explicar cómo funciona su aplicación, demostrar su funcionamiento y discutir cualquier desafío que hayan enfrentado y cómo lo superaron.

**Pasos para seguir**

1. Configuración inicial
2. Añadir permisos en AndroidManifest.xml
3. Añadir dependencias en build.gradle
4. Diseño de la actividad principal (activity\_main.xml)
5. Desarrollar la lógica en MainActivity.kt
6. Probar la aplicación
7. Desafío.

**Paso 1: Configuración inicial**

Primero, se debe asegurar que el entorno de desarrollo esté correctamente configurado. Esto implica tener instalado Android Studio con la última versión del SDK de Android y Kotlin.

**Paso 2: Añadir permisos en AndroidManifest.xml**

En el archivo AndroidManifest.xml, se deben añadir tres permisos: ACCESS\_FINE\_LOCATION, ACCESS\_COARSE\_LOCATION y ACCESS\_BACKGROUND\_LOCATION. Estos permisos permiten a la aplicación acceder a la ubicación del dispositivo del usuario.

Además, se debe añadir en la sección de aplicación una meta-data con la clave de la API de Google Maps para permitir la utilización de sus servicios, esta fue generada con las credenciales del profesor. Analice le cierre de las etiquetas.

<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION"/>  
<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION"/>  
<uses-permission android:name="android.permission.ACCESS\_BACKGROUND\_LOCATION"/>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

<meta-data  
 android:name="com.google.android.geo.API\_KEY"  
 android:value="AIzaSyAA0cAvSPA\_vgTRn2EXAwV51RHJ88UZsZU"/>  
</application>

**Paso 3: Añadir dependencias en build.gradle**

Para poder utilizar los servicios de localización y mapas de Google, es necesario añadir las dependencias correspondientes en el archivo build.gradle del módulo.

implementation 'com.google.android.gms:play-services-maps:18.1.0'  
implementation 'com.google.android.gms:play-services-location:18.0.0'

**Paso 4: Diseño de la actividad principal (activity\_main.xml)**

En el archivo activity\_main.xml, se incluye un FragmentContainerView que albergará el mapa de Google. Este fragmento ocupa toda la pantalla gracias a las restricciones establecidas.

*<?*xml version="1.0" encoding="utf-8"*?>*<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity">  
  
 <androidx.fragment.app.FragmentContainerView  
 android:id="@+id/mapa"  
 android:name="com.google.android.gms.maps.SupportMapFragment"  
 android:layout\_width="0dp"  
 android:layout\_height="0dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintLeft\_toLeftOf="parent"  
 app:layout\_constraintRight\_toRightOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 tools:context=".MainActivity" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>

**Paso 5: Desarrollar la lógica en MainActivity.kt**

En MainActivity.kt se desarrolla la lógica principal de la aplicación.

**5.1 Inicialización**

Al principio de la clase MainActivity, se inicializan varias variables, incluyendo el cliente de ubicación (FusedLocationProviderClient), el LocationCallback que nos permitirá recibir actualizaciones de la ubicación, un GoogleMap para interactuar con el mapa, un Marker para el marcador actual en el mapa, y un booleano para verificar si es la primera vez que obtenemos la ubicación.

class MainActivity : AppCompatActivity(), OnMapReadyCallback {  
  
 private val codigoPermisoUbicacion = 999  
 private lateinit var clienteUbicacion: FusedLocationProviderClient  
 private lateinit var callbackUbicacion: LocationCallback  
 private lateinit var mapa: GoogleMap  
 private var marcador: Marker? = null  
 private var primeraUbicacionObtenida = false *// Ubicación correcta de la variable*

**5.2 Configuración en onCreate**

En el método onCreate, se realiza la configuración del mapa y se solicita el permiso de ubicación al usuario.

override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
  
 *// Configuración del mapa* val fragmentoMapa = *supportFragmentManager*.findFragmentById(R.id.*mapa*) as SupportMapFragment  
 fragmentoMapa.getMapAsync(this)  
  
 *// Configuración del cliente de ubicación* clienteUbicacion = LocationServices.getFusedLocationProviderClient(this)  
  
 *// Solicitar permiso de ubicación* solicitarPermisoUbicacion()  
}

**5.3 Solicitar permisos**

El método solicitarPermisoUbicacion solicita permiso de ubicación al usuario. Si se concede el permiso, se llama al método obtenerUbicacion para obtener la ubicación actual.

private fun solicitarPermisoUbicacion() {  
 if (ActivityCompat.checkSelfPermission(  
 this,  
 Manifest.permission.*ACCESS\_FINE\_LOCATION* ) != PackageManager.*PERMISSION\_GRANTED* ) {  
 ActivityCompat.requestPermissions(  
 this,  
 *arrayOf*(Manifest.permission.*ACCESS\_FINE\_LOCATION*),  
 codigoPermisoUbicacion  
 )  
 } else {  
 obtenerUbicacion()  
 }  
}

**5.4 Procesar los permisos concedidos**

En el método onRequestPermissionsResult, se verifica si el permiso de ubicación ha sido concedido. Si se concede el permiso, se llama al método obtenerUbicacion para obtener la ubicación actual.

override fun onRequestPermissionsResult(  
 requestCode: Int,  
 permissions: Array<String>,  
 grantResults: IntArray  
) {  
 if (requestCode == codigoPermisoUbicacion) {  
 if (grantResults.*isNotEmpty*() && grantResults[0] == PackageManager.*PERMISSION\_GRANTED*) {  
 obtenerUbicacion()  
 } else {  
 Toast.makeText(this, "Permiso de ubicación denegado", Toast.*LENGTH\_LONG*).show()  
 }  
 }  
  
 super.onRequestPermissionsResult(requestCode, permissions, grantResults)  
}

**5.5 Obtener ubicación**

El método obtenerUbicacion obtiene la ubicación actual del dispositivo y añade un marcador en esa ubicación en el mapa. En esta función puede salir un error de seguridad, esto depende de la versión utilizada, ignórelo y continue.

private fun obtenerUbicacion() {  
 if (!primeraUbicacionObtenida) {  
 *// Para obtener la ubicación inicial del dispositivo* clienteUbicacion.*lastLocation*.addOnSuccessListener **{** location **->** if (location != null) {  
 *// Actualizar el marcador en el mapa con la ubicación inicial* val ubicacionInicial = LatLng(location.*latitude*, location.*longitude*)  
 marcador = mapa.addMarker(MarkerOptions().position(ubicacionInicial).title("Tu ubicación"))  
 mapa.moveCamera(CameraUpdateFactory.newLatLngZoom(ubicacionInicial, 15f))  
 }  
 **}** primeraUbicacionObtenida = true  
 }  
}

**5.6 Manejar la interacción con el mapa**

En el método onMapReady, se configura el mapa para hacer los marcadores arrastrables, y se establece un OnMarkerDragListener para actualizar la ubicación del marcador cuando se arrastra a una nueva ubicación.

override fun onMapReady(googleMap: GoogleMap) {  
 mapa = googleMap  
  
 mapa.setOnMarkerClickListener **{** marcador **->** marcador.*isDraggable* = true  
 false  
 **}** mapa.setOnMarkerDragListener(object : GoogleMap.OnMarkerDragListener {  
 override fun onMarkerDragStart(marcador: Marker) {}  
  
 override fun onMarkerDrag(marcador: Marker) {}  
  
 override fun onMarkerDragEnd(marcador: Marker) {  
 *// Obtener la nueva posición del marcador después de moverlo* val nuevaUbicacion = marcador.*position  
  
 // Actualizar la posición del marcador en el mapa  
 // (aquí puedes realizar otras acciones con la nueva ubicación, si es necesario)* }  
 })  
 }  
}

**Paso 6: Probar la aplicación**

Finalmente, se puede probar la aplicación en el emulador.

**Responda:**

* ¿Cómo se podría mejorar la experiencia del usuario al interactuar con el mapa y el marcador de ubicación?
  + Agregar la funcionalidad para buscar una ubicación especifica mediante un cuadro de búsqueda, para que el usuario pueda encontrar los lugares específicos en el mapa.
  + Permitir que los usuarios interactúen con el marcador de ubicación. Por ejemplo, podrían hacer clic en el marcador para ver más detalles o abrir una ventana emergente con información adicional.
  + Implementar un zoom y una panorámica suaves en el mapa para que los usuarios puedan explorar fácilmente diferentes áreas y acercarse a la ubicación deseada. Esto crea una experiencia más fluida y agradable.
  + Permitir que los usuarios puedan personalizar el aspecto del marcador de ubicación, como el color, la forma o el icono. Esto les dará la oportunidad de personalizar la apariencia del marcador según sus preferencias.
* ¿Cómo manejaría la aplicación situaciones donde el GPS no está disponible o la señal es débil?

Se pueden mostrar mensajes informativos al usuario, utilizar fuentes de ubicación alternativas, como la red móvil o el Wi-Fi, y aprovechar la última ubicación conocida. Además, es recomendable implementar funcionalidad offline, ajustar la interfaz de usuario en ausencia de señal, y diseñar capacidades de recuperación automática cuando la señal se restablezca. Probar la aplicación en escenarios de señal débil o inexistente es fundamental para asegurar una experiencia de usuario óptima.

* ¿Cómo se podría integrar la funcionalidad de seguimiento de ubicación con otras funcionalidades o servicios?
  + Servicios de navegación: Se puede integrar la funcionalidad de seguimiento de ubicación con servicios de navegación, como Google Maps Directions API. Esto permite a los usuarios obtener rutas y direcciones precisas en tiempo real basadas en su ubicación actual.
  + Compartir ubicación en redes sociales: Se puede agregar la opción de compartir la ubicación del usuario en redes sociales populares. Esto permite a los usuarios compartir su ubicación con amigos y contactos, ya sea para coordinar encuentros o simplemente para compartir su ubicación en tiempo real.
  + Integración con servicios de clima: Se puede utilizar la ubicación del usuario para proporcionar información meteorológica en tiempo real. Por ejemplo, se puede mostrar el pronóstico del tiempo para la ubicación actual del usuario o alertarlo sobre condiciones climáticas adversas.

****

**Paso 7. Desafío (Para entregar)**

Complete el código con la siguiente parte:

Notificaciones basadas en la ubicación, esto implicaría detectar cuando el usuario entra en una cierta área y enviar una notificación en respuesta. Los estudiantes aprenderían cómo trabajar con geofencing y cómo enviar notificaciones.