



DESARROLLO DE APLICACIONES MÓVILES AVANZADAS



“USO DE LOS SERVICIOS DE GEOLOCALIZACIÓN EN ANDROID”

Resultados de Aprendizaje:

- Comprender los elementos necesarios para iniciarse en el desarrollo de aplicaciones basadas en Android.
- Desarrollar proyectos haciendo uso de librerías avanzadas que proporciona Android.

INDICE

INTRODUCCIÓN	3
CREANDO NUESTRO PROYECTO	3
PRÁCTICA POR REALIZAR	9

INTRODUCCIÓN

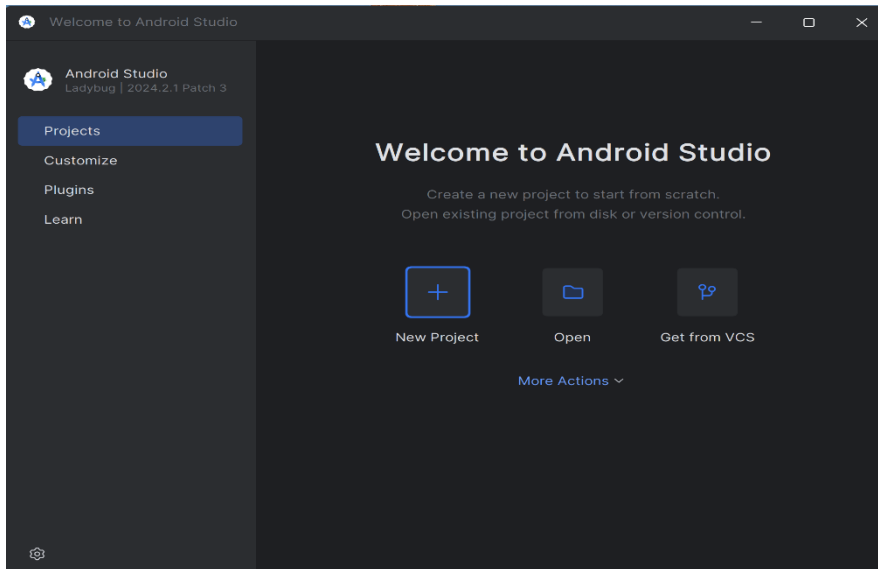
Para desarrollar esta práctica usted deberá tener los siguientes paquetes de software:

- Android Studio Versión 4 o superior
- Equipo Móvil con Android (Puede ser Tablet o Smartphone) con **modo desarrollador habilitado**.

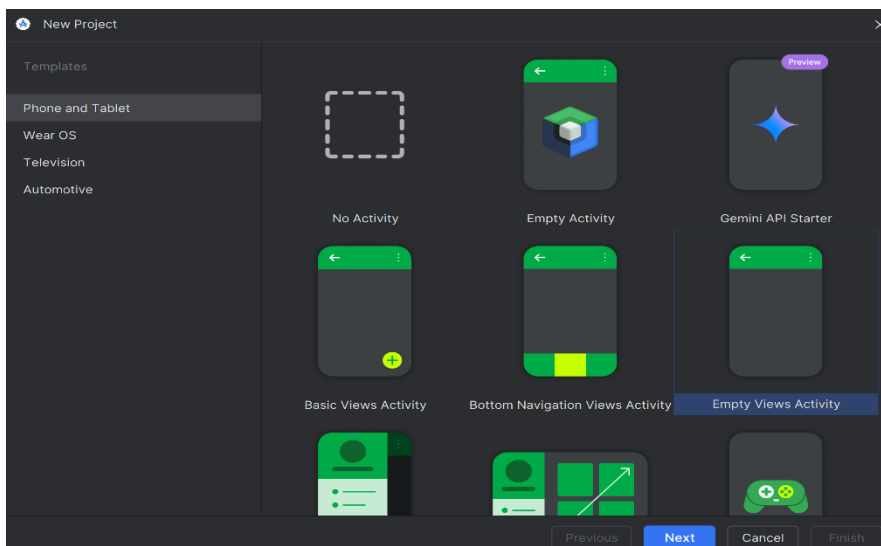
(Ver <https://www.youtube.com/watch?v=wLJS8CKo95o>)

CREANDO NUESTRO PROYECTO

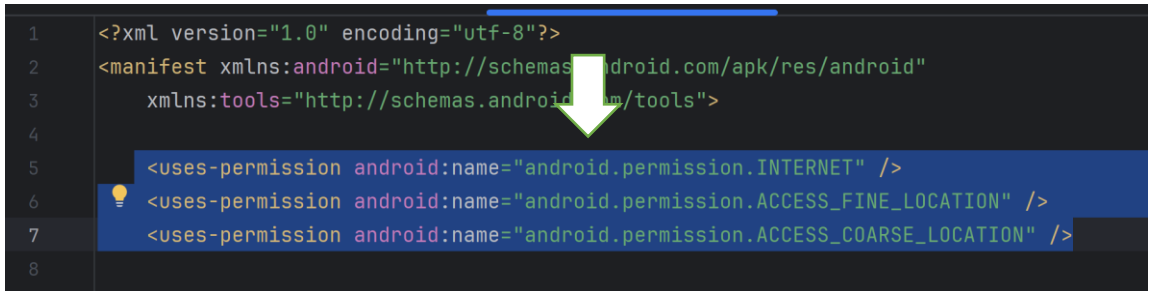
1. Iniciamos nuestro Android Studio.



2. Elegimos la plantilla **Empty Views Activity**.



3. Luego establezca las siguientes propiedades al proyecto:
 - a. Nombre: Geolocalizacion
 - b. Package Name: **net.lrivas.geolocalizacion**
 - c. Lenguaje: **Kotlin**
 - d. Minimum Api: **33**
4. Implemente los permisos en la aplicación, esto servirá para poder acceder a la geolocalización y servicios de internet.

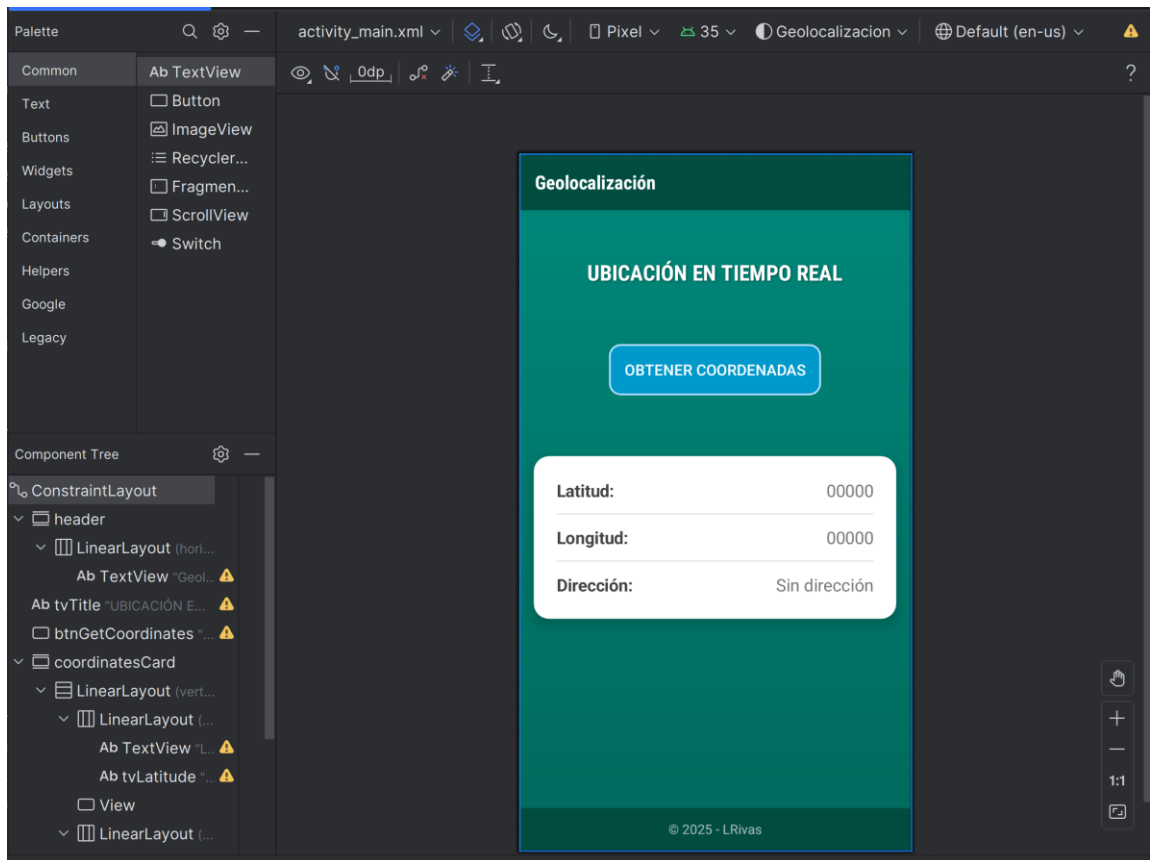


```

1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3       xmlns:tools="http://schemas.android.com/tools">
4
5     <uses-permission android:name="android.permission.INTERNET" />
6     <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
7     <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
8

```

5. Ahora crearemos la siguiente interfaz gráfica en el archivo **activity_main.xml**.



6. Luego agregamos las siguientes líneas de código:

Declaramos los objetos a nivel de clase así como las referencias de cada uno de los objetos a utilizar.

```

class MainActivity : AppCompatActivity() {

    private lateinit var botonObtenerCoordenadas: Button
    private lateinit var textoLatitud: TextView
    private lateinit var textoLongitud: TextView
    private lateinit var gestorUbicacion: LocationManager
    private val lanzadorSolicitudPermiso = registerForActivityResult(ActivityResultContracts.RequestPermission()) { isGranted ->
        if (isGranted) {
            obtenerUltimaUbicacion()
        } else {
            Toast.makeText(context, this, text: "Permiso de ubicación denegado", Toast.LENGTH_SHORT).show()
        }
    }

    private lateinit var localizacion: Localizacion
    private var rastreandoUbicacion = false

```

Ahora crearemos el **onCreate**.

```

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    botonObtenerCoordenadas = findViewById(R.id.btnGetCoordinates)
    textoLatitud = findViewById(R.id.tvLatitude)
    textoLongitud = findViewById(R.id.tvLongitude)

    gestorUbicacion = getSystemService(LOCATION_SERVICE) as LocationManager
    localizacion = Localizacion(textoLatitud, textoLongitud, MainActivity::this)

    botonObtenerCoordenadas.setOnClickListener {
        if (!rastreandoUbicacion) {
            verificarPermisoUbicacion()
            botonObtenerCoordenadas.text = "Detener rastreo"
            rastreandoUbicacion = true
        } else {
            detenerRastreo()
            botonObtenerCoordenadas.text = "Obtener Coordenadas"
            rastreandoUbicacion = false
        }
    }
}

```

Ahora vamos a crear una serie de funciones para el funcionamiento de la aplicación:

Creando la función `verificarPermisoUbicacion()`

```

private fun verificarPermisoUbicacion() {
    when {
        ContextCompat.checkSelfPermission(context, Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED -> {
            obtenerUltimaUbicacion()
        }
        else -> {
            shouldShowRequestPermissionRationale(Manifest.permission.ACCESS_FINE_LOCATION) -> {
                Toast.makeText(context, this, text: "Se necesita permiso de ubicación para continuar", Toast.LENGTH_LONG).show()
                lanzadorSolicitudPermiso.launch(Manifest.permission.ACCESS_FINE_LOCATION)
            }
            else -> {
                lanzadorSolicitudPermiso.launch(Manifest.permission.ACCESS_FINE_LOCATION)
            }
        }
    }
}

```

Creando la función obtenerUltimaUbicacion()

```
private fun obtenerUltimaUbicacion() {
    if (ContextCompat.checkSelfPermission(context: this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
        Toast.makeText(context: this, text: "Permiso de ubicación no concedido", Toast.LENGTH_SHORT).show()
        return
    }

    try {
        val ultimaUbicacion = gestorUbicacion.getLastKnownLocation(LocationManager.GPS_PROVIDER)
            ?: gestorUbicacion.getLastKnownLocation(LocationManager.NETWORK_PROVIDER)

        if (ultimaUbicacion != null) {
            textolatitud.text = String.format("%.6f", ultimaUbicacion.latitude)
            textolongitud.text = String.format("%.6f", ultimaUbicacion.longitude)
            localizacion.setLastLocation(ultimaUbicacion)
        } else {
            textolatitud.text = "No disponible"
            textolongitud.text = "No disponible"
            Toast.makeText(context: this@MainActivity, text: "No se encontró ubicación reciente", Toast.LENGTH_SHORT).show()
        }

        gestorUbicacion.requestLocationUpdates(
            LocationManager.GPS_PROVIDER,
            minTimeMs: 1000,
            minDistanceM: 0.5f,
            localizacion
        )

        if (gestorUbicacion.isProviderEnabled(LocationManager.NETWORK_PROVIDER)) {
            gestorUbicacion.requestLocationUpdates(
                LocationManager.NETWORK_PROVIDER,
                minTimeMs: 1000,
                minDistanceM: 0.5f,
                localizacion
            )
        }

        Toast.makeText(context: this, text: "Rastreo de ubicación iniciado", Toast.LENGTH_SHORT).show()
    } catch (e: SecurityException) {
        textolatitud.text = "Error"
        textolongitud.text = "Error"
        Toast.makeText(context: this@MainActivity, text: "Error de seguridad: ${e.message}", Toast.LENGTH_SHORT).show()
    } catch (e: Exception) {
        textolatitud.text = "Error"
        textolongitud.text = "Error"
        Toast.makeText(context: this@MainActivity, text: "Error al obtener ubicación: ${e.message}", Toast.LENGTH_SHORT).show()
    }
}
```

Creando la función detenerRastreo()

```
private fun detenerRastreo() {
    gestorUbicacion.removeUpdates(localizacion)
    Toast.makeText(context: this, text: "Rastreo de ubicación detenido", Toast.LENGTH_SHORT).show()
}

override fun onPause() {
    super.onPause()
    if (rastreandoUbicacion) {
        detenerRastreo()
        botonObtenerCoordenadas.text = "Obtener Coordenadas"
        rastreandoUbicacion = false
    }
}
```

En la clase **Localizacion**, **setLastLocation** es una función pública para almacenar la última ubicación, mientras que **onLocationChanged**, **onProviderDisabled**, **onProviderEnabled** y **onStatusChanged** sobrescriben métodos de **LocationListener** para manejar actualizaciones de ubicación, desactivación/activación de proveedores y estados, usando **Geocoder** para obtener la dirección.

```
class Localizacion(
    private val textoLatitud: TextView,
    private val textoLongitud: TextView,
    private val mainActivity: MainActivity
) : LocationListener {

    private var lastLocation: Location? = null
    private val geocoder: Geocoder by lazy { Geocoder(mainActivity, Locale.getDefault()) }

    fun setLastLocation(location: Location) {
        lastLocation = location
        updateAddress(location)
    }

    override fun onLocationChanged(loc: Location) {
        textoLatitud.text = String.format("%.6f", loc.latitude)
        textoLongitud.text = String.format("%.6f", loc.longitude)
        updateAddress(loc)
        lastLocation = loc
    }

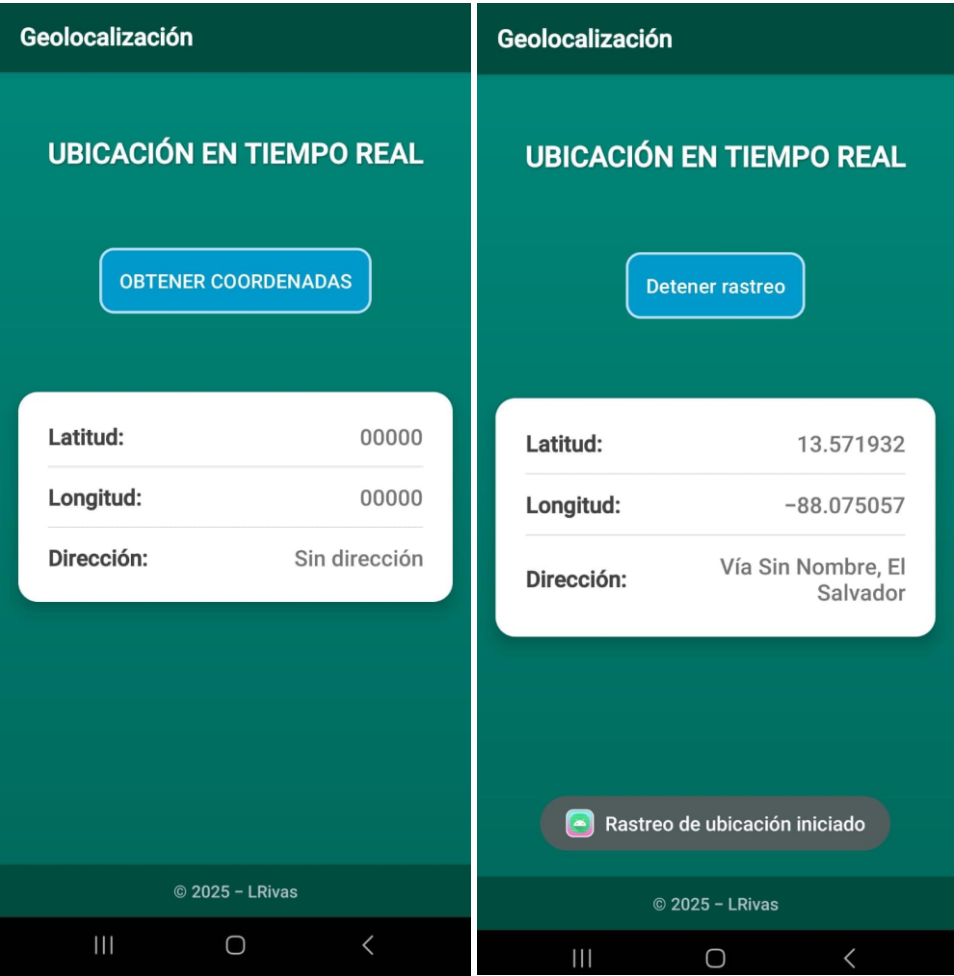
    private fun updateAddress(location: Location) {
        try {
            val addresses: List<Address> = geocoder.getFromLocation(location.latitude, location.longitude, maxResults: 1) ?: emptyList()
            if (addresses.isNotEmpty()) {
                val address = addresses[0]
                val addressText = address.getAddressLine(0) ?: "Sin dirección"
                mainActivity.runOnUiThread {
                    mainActivity.findViewById<TextView>(R.id.tvAddress)?.text = addressText
                }
            } else {
                mainActivity.runOnUiThread {
                    mainActivity.findViewById<TextView>(R.id.tvAddress)?.text = "Sin dirección"
                }
            }
        } catch (e: Exception) {
            mainActivity.runOnUiThread {
                mainActivity.findViewById<TextView>(R.id.tvAddress)?.text = "Error al obtener dirección"
            }
        }
    }

    override fun onProviderDisabled(provider: String) {
        Toast.makeText(mainActivity, "GPS Desactivado", Toast.LENGTH_SHORT).show()
    }

    override fun onProviderEnabled(provider: String) {
        Toast.makeText(mainActivity, "GPS Activado", Toast.LENGTH_SHORT).show()
    }

    override fun onStatusChanged(provider: String, status: Int, extras: Bundle) {
        when (status) {
            LocationProvider.AVAILABLE -> {}
            LocationProvider.OUT_OF_SERVICE -> {}
            LocationProvider.TEMPORARILY_UNAVAILABLE -> {}
        }
    }
}
```

Si todo anda bien, tendremos el funcionamiento de la siguiente manera:



PRÁCTICA POR REALIZAR

NOMBRE: Uso de Geolocalización

Indicación: haciendo uso del contenido visto en la semana, de solución al siguiente requerimiento:

- A. A la aplicación actual, agregue un botón que permita compartir la localización a través de WhatsApp, para ello la aplicación deberá mandar un mensaje con este formato:
 “Hola, te adjunto mi ubicación:
<https://maps.google.com/?q=CoordenadaLatitud,CoordenadaLongitud>”
- B. Investigue como **incrustar un mapa** según una coordenada específica y a partir de eso, modifique la aplicación para que se muestre un mapa con la ubicación obtenida, pero en otra Activity.

FORMA DE ENTREGA: Se deberá enviar al buzón de tarea llamado **Uso de Geolocalización**, de manera individual.

TIPO DE ENTREGA: Enviar un documento en **PDF o DOCX** al final de la semana antes de las **23:59**.

RUBRICA DE EVALUACIÓN

Indicación: a continuación, se establecen los criterios de evaluación para la actividad de la semana.

#	CRITERIO	PTS.
1	Entrega el proyecto en la fecha establecida.	2.0
2	Diseña la interfaz principal de acuerdo con el requerimiento.	0.5
3	Envía la ubicación obtenida por WhatsApp.	2.0
4	Genera el mapa a partir de la ubicación obtenida en una nueva activity.	2.0
5	Elabora el ejemplo de la práctica.	1.0
6	Sube el proyecto a git y anexa el enlace en el documento.	0.5
7	En la plantilla en la sección de comentarios, el estudiante elabore un pequeño resumen de 100 palabras como mínimo, en el que explique cuál fue su experiencia aprendida en la elaboración tanto del ejemplo de la práctica como del ejercicio solicitado.	1.0
8	El documento está ordenado según la plantilla y sin errores de ortografía.	1.0