Juan Menduiña

Trabajo Práctico N° 5

Ejercicio 1.

Investigar el uso de los diferentes sensores del dispositivo en la documentación oficial de Android: https://developer.android.com/guide/topics/sensors/sensors_environment (Sección: Use the light, pressure and temperature sensors).

(a) *¿Cuál es la función de la clase SensorManager?*

La función de la clase *SensorManager* es responsable de acceder a los sensores físicos disponibles en un dispositivo *Android*. Proporciona métodos para:

- Obtener una lista de sensores disponibles (*getSensorList()*).
- Obtener un sensor específico (*getDefaultSensor()*).
- Registrar y eliminar *listeners* para recibir datos de sensores (*registerListener()* y *unregisterListener()*).

En resumen, gestiona el acceso a los sensores y facilita la comunicación entre la aplicación y los sensores físicos.

(b) *¿Cómo se genera una instancia de un Sensor?*

Una instancia de un *Sensor* se genera a través del *SensorManager*, usando el método *getDefaultSensor()*. Este método devuelve un objeto Sensor correspondiente al tipo de sensor solicitado (por ejemplo, luz, presión, temperatura, etc.).

(c) *¿Para qué sirven los métodos registerListener y unregisterListener de la clase Sensor?*

- *registerListener:* Registra un *listener* (*SensorEventListener*) para empezar a recibir datos del sensor seleccionado. También permite establecer la frecuencia de muestreo.
- *unregisterListener:* Detiene el envío de datos desde el sensor al *listener* registrado, lo cual es útil para ahorrar batería y recursos del sistema cuando no se necesita monitorear el sensor.

(d) *¿Por qué se utilizan las transiciones de estado onResume y onPause para registrar/eliminar el listener del Sensor?*

Las transacciones de estado *onResume* y *onPause* se utilizan para registrar/eliminar el *listener* del *Sensor* porque gestionan, eficientemente, los recursos del dispositivo. En particular:

- En *onResume()*, se registra el *listener* para empezar a recibir datos del sensor cuando la actividad está visible y en primer plano.
- En *onPause()*, se elimina el *listener* para detener la recepción de datos cuando la actividad ya no está activa, reduciendo, así, el consumo de batería y uso del procesador.

Esta práctica garantiza que los sensores sólo trabajen cuando, realmente, se necesitan.

Ejercicio 2: Sensor de Luminosidad.

(a) *Implementar una aplicación que muestre, en un TextView, el valor del Sensor en tiempo real.*

En *activity_main.xml*:

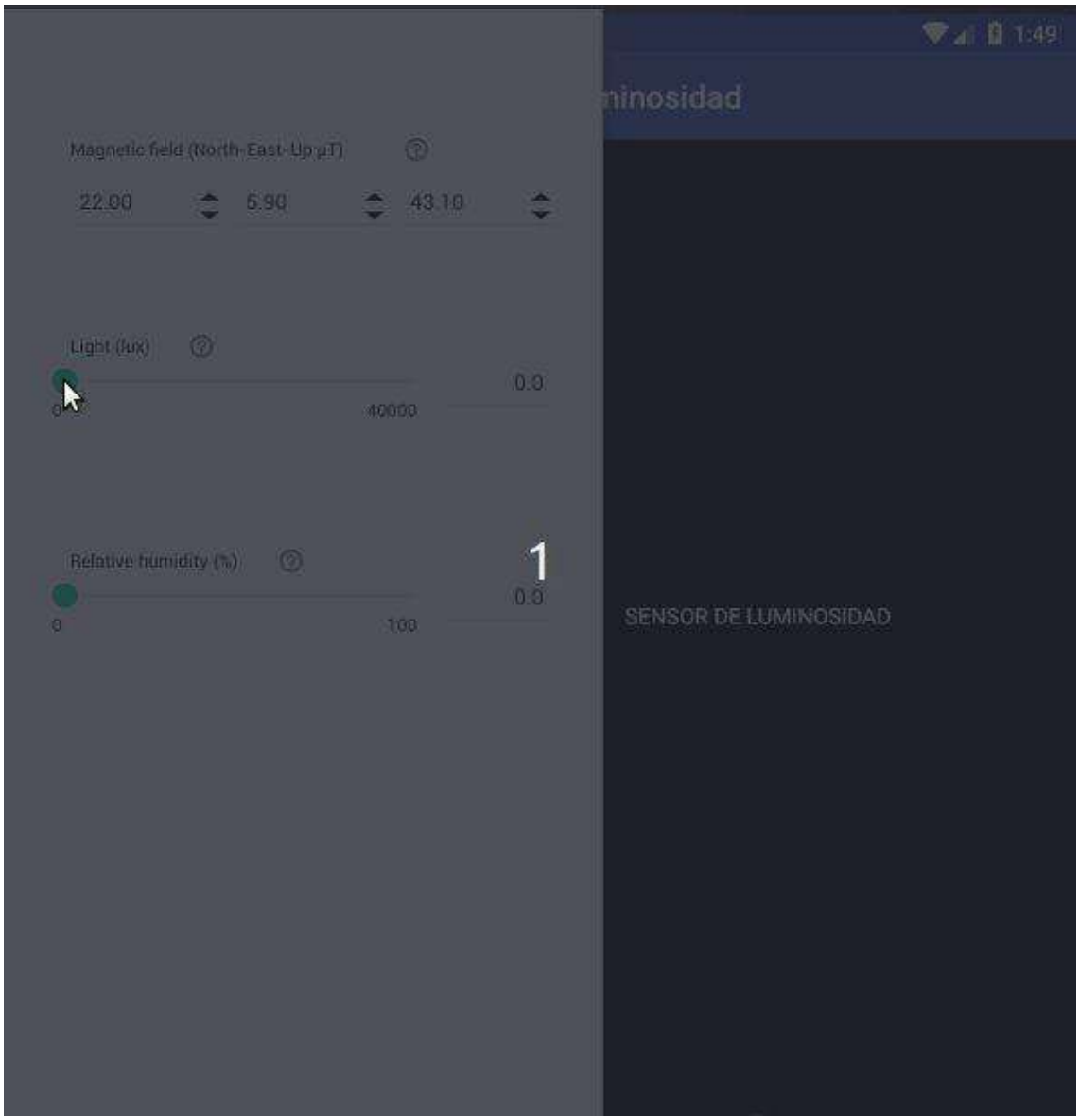
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/txtSensor"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:textColor="@color/black"
        android:textSize="24sp"
        android:textStyle="bold"
        android:gravity="center" />
</LinearLayout>
```

En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity(), SensorEventListener {
    private lateinit var txtSensor: TextView
    private lateinit var sensorManager: SensorManager
    private var lightSensor: Sensor? = null
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) {
v, insets ->
            val systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom)
            insets
        }
        txtSensor = findViewById(R.id.txtSensor)
        sensorManager = getSystemService(SENSOR_SERVICE) as
SensorManager
        lightSensor =
sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT)
        if (lightSensor == null) {
            txtSensor.text = "Sensor de luz no disponible"
        }
    }
    override fun onResume() {
        super.onResume()
```

```
        if (lightSensor != null) {
            sensorManager.registerListener(this, lightSensor,
SensorManager.SENSOR_DELAY_NORMAL)
        }
    }
    override fun onPause() {
        super.onPause()
        sensorManager.unregisterListener(this)
    }
    override fun onSensorChanged(event: SensorEvent) {
        txtSensor.text = "SENSOR DE LUMINOSIDAD:\n${event.values[0]}
lux"
    }
    override fun onAccuracyChanged(sensor: Sensor?, accuracy: Int) {

    }
}
```

(b) *Implementar una aplicación con un TextView que ocupe toda la pantalla y que contenga un texto fijo. El color de fondo y el texto del TextView deberá reaccionar al nivel de luminosidad de manera que se vea el fondo blanco y letras negras en condiciones de mucha luz, mientras que, en condiciones de poca luz, se visualizará el fondo negro con letras blancas.*



En *activity_main.xml*:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/txtSensor"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:textSize="24sp"
        android:textStyle="bold"
```

```
        android:gravity="center" />
</LinearLayout>
```

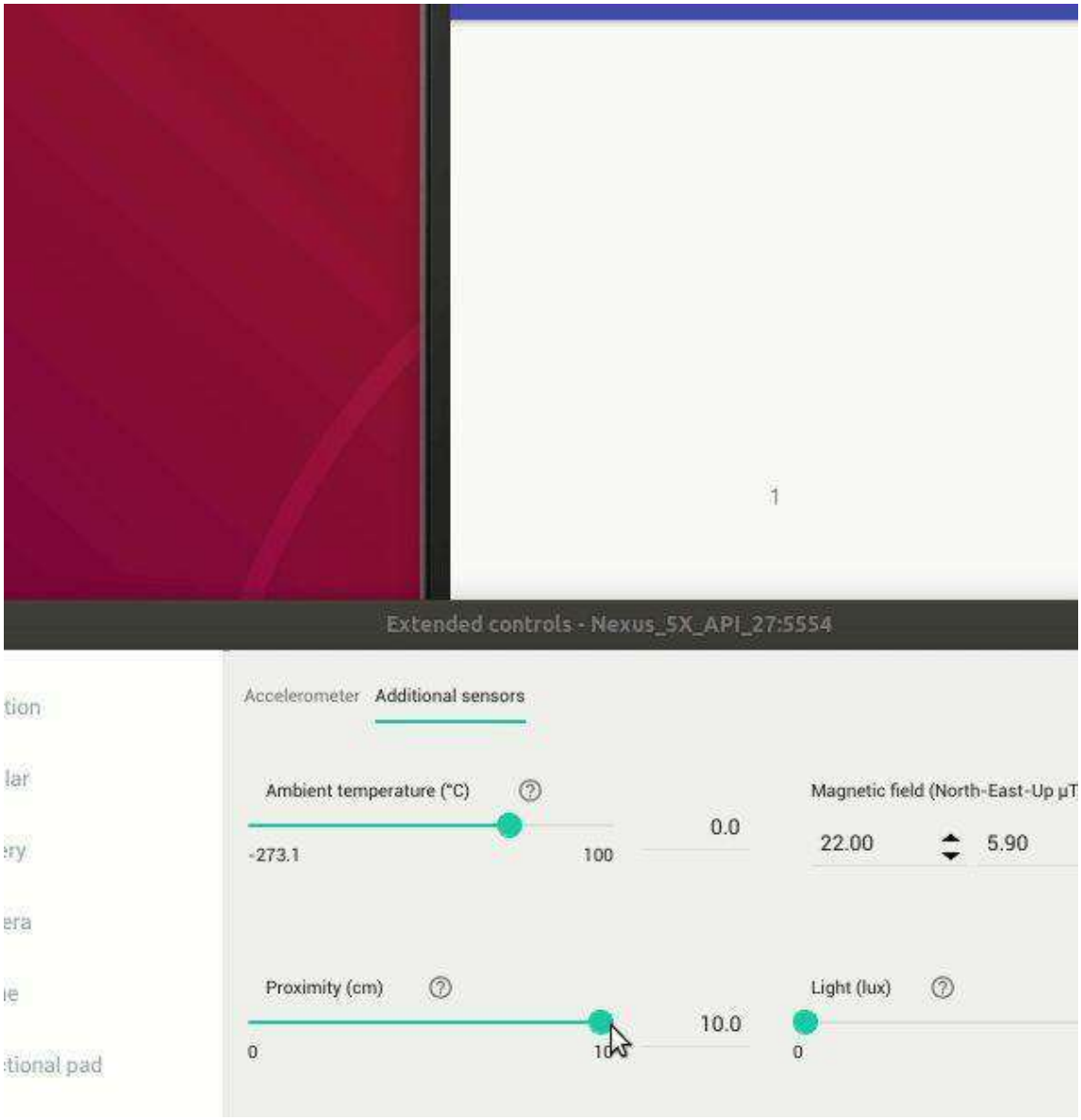
En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity(), SensorEventListener {
    private lateinit var layout: LinearLayout
    private lateinit var txtSensor: TextView
    private lateinit var sensorManager: SensorManager
    private var lightSensor: Sensor? = null
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) {
v, insets ->
            val systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom)
            insets
        }
        layout = findViewById(R.id.main)
        txtSensor = findViewById(R.id.txtSensor)
        sensorManager = getSystemService(SENSOR_SERVICE) as
SensorManager
        lightSensor =
sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT)
        if (lightSensor == null) {
            txtSensor.text = "Sensor de luz no disponible"
        }
    }
    override fun onResume() {
        super.onResume()
        if (lightSensor != null) {
            sensorManager.registerListener(this, lightSensor,
SensorManager.SENSOR_DELAY_NORMAL)
        }
    }
    override fun onPause() {
        super.onPause()
        sensorManager.unregisterListener(this)
    }
    override fun onSensorChanged(event: SensorEvent) {
        txtSensor.text = "SENSOR DE LUMINOSIDAD:\n${event.values[0]}
lux"
        if (event.values[0] > 20000) {
            layout.setBackgroundColor(Color.WHITE)
            txtSensor.setTextColor(Color.BLACK)
        }
        else {
            layout.setBackgroundColor(Color.BLACK)
            txtSensor.setTextColor(Color.WHITE)
        }
    }
    override fun onAccuracyChanged(sensor: Sensor?, accuracy: Int) {

    }
}
```

Ejercicio 3: Sensor de Proximidad.

Implementar una aplicación que muestre un TextView con un contador numérico. El contador debe incrementarse cada vez que el usuario pase su mano por encima del dispositivo.

Nota: Para determinar que el usuario realizó el gesto de pasar su mano por encima del dispositivo, se deberá detectar un estado de "no proximidad", seguido de un estado de "proximidad".



En *activity_main.xml*:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/txtSensor"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="SENSOR DE PROXIMIDAD:\n0"
        android:textColor="@color/black"
        android:textSize="24sp"
        android:textStyle="bold"
        android:gravity="center" />
</LinearLayout>
```

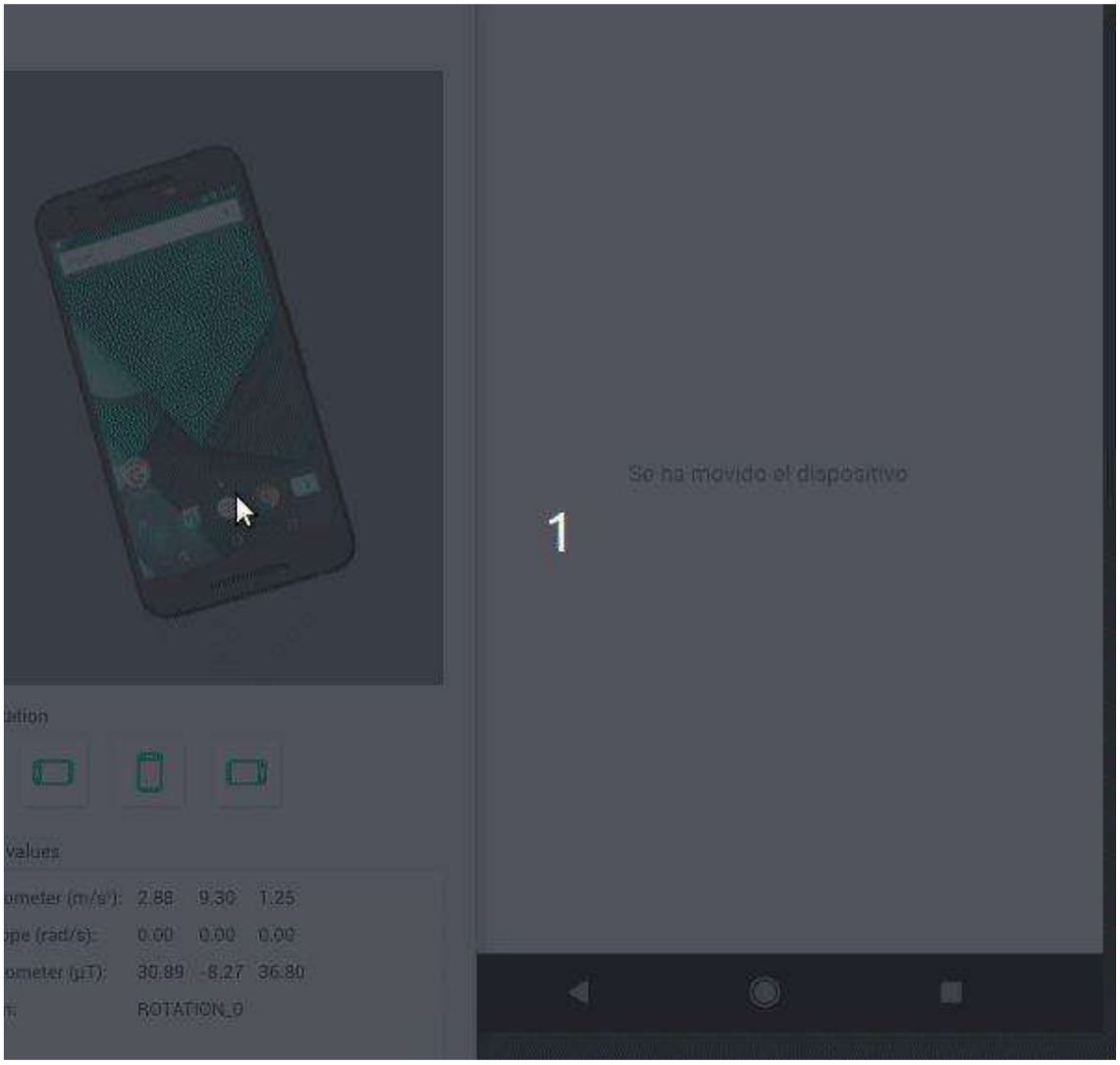
En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity(), SensorEventListener {
    private lateinit var txtSensor: TextView
    private lateinit var sensorManager: SensorManager
    private var proximitySensor: Sensor? = null
    private var counter = 0
    private var wasFar = false
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) {
v, insets ->
            val systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom)
            insets
        }
        txtSensor = findViewById(R.id.txtSensor)
        sensorManager = getSystemService(SENSOR_SERVICE) as
SensorManager
        proximitySensor =
sensorManager.getDefaultSensor(Sensor.TYPE_PROXIMITY)
        if (proximitySensor == null) {
            txtSensor.text = "Sensor de proximidad no disponible"
        }
    }
    override fun onResume() {
        super.onResume()
        if (proximitySensor != null) {
            sensorManager.registerListener(this, proximitySensor,
SensorManager.SENSOR_DELAY_NORMAL)
        }
    }
    override fun onPause() {
        super.onPause()
        sensorManager.unregisterListener(this)
    }
    override fun onSensorChanged(event: SensorEvent) {
        val distance = event.values[0]
        if (distance > 0) {
```

```
            wasFar = true
        }
        else if ((distance == 0f) && (wasFar)) {
            wasFar = false
            counter++
            txtSensor.text = "SENSOR DE PROXIMIDAD:\n$counter"
        }
    }
    override fun onAccuracyChanged(sensor: Sensor?, accuracy: Int) {

    }
}
```

Ejercicio 4: Acelerómetro.

Nota: Para estos ejercicios, se deberá utilizar el sensor de Aceleración LINEAL, dado que el mismo no tiene en cuenta la fuerza de gravedad.



(a) *Generar una aplicación que muestre, en un TextView, los valores de aceleración en los tres ejes X, Y, Z.*

En *activity_main.xml*:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp"
    tools:context=".MainActivity">
```

```
    <TextView
        android:id="@+id/txtSensor"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="ACELERACIÓN:\nX: 0.0\nY: 0.0\nZ: 0.0"
        android:textColor="@color/black"
        android:textSize="24sp"
        android:textStyle="bold"
        android:gravity="center" />
</LinearLayout>
```

En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity(), SensorEventListener {
    private lateinit var txtSensor: TextView
    private lateinit var sensorManager: SensorManager
    private var linearAccelerationSensor: Sensor? = null
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) {
v, insets ->
            val systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom)
            insets
        }
        txtSensor = findViewById(R.id.txtSensor)
        sensorManager = getSystemService(SENSOR_SERVICE) as
SensorManager
        linearAccelerationSensor =
sensorManager.getDefaultSensor(Sensor.TYPE_LINEAR_ACCELERATION)
        if (linearAccelerationSensor == null) {
            txtSensor.text = "Sensor de aceleración lineal no
disponible"
        }
    }
    override fun onResume() {
        super.onResume()
        if (linearAccelerationSensor != null) {
            sensorManager.registerListener(this,
linearAccelerationSensor, SensorManager.SENSOR_DELAY_NORMAL)
        }
    }
    override fun onPause() {
        super.onPause()
        sensorManager.unregisterListener(this)
    }
    override fun onSensorChanged(event: SensorEvent) {
        val x = event.values[0]
        val y = event.values[1]
        val z = event.values[2]
        val txt = """
            ACELERACIÓN:
            X: ${"%.2f".format(x)} m/s²
            Y: ${"%.2f".format(y)} m/s²
            Z: ${"%.2f".format(z)} m/s²
        """.trimIndent()
```

```
            txtSensor.text = txt
        }
        override fun onAccuracyChanged(sensor: Sensor?, accuracy: Int) {

        }
}
```

(b) *Cuando el dispositivo está en reposo, ¿los valores son, estrictamente, cero?*

Cuando el dispositivo está en reposo, los valores no son, estrictamente, cero, ya que, si bien este sensor no tiene en cuenta la fuerza de gravedad, está sujeto a:

- Ruido electrónico: Los sensores físicos tienen imprecisiones mínimas por vibraciones internas, corriente, temperatura, etc.
- Pequeñas vibraciones del entorno.
- Resolución del sensor: Algunos sensores detectan cambios tan pequeños que parecen fluctuaciones aleatorias, aunque no haya movimiento.

(c) *Generar una aplicación que sea capaz de detectar el estado de reposo (completamente quieto) del dispositivo y que genere un aviso en un TextView cuando el mismo pierda su estado de reposo (alguien mueva el dispositivo).*

En *activity_main.xml*:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/txtSensor"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="ESTADO DE REPOSO"
        android:textColor="@color/black"
        android:textSize="24sp"
        android:textStyle="bold"
        android:gravity="center" />
</LinearLayout>
```

En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity(), SensorEventListener {
    private lateinit var txtSensor: TextView
    private lateinit var sensorManager: SensorManager
    private var linearAccelerationSensor: Sensor? = null
```

```
    private val threshold = 0.1f
    private var isAtRest = true
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) {
v, insets ->
            val systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom)
            insets
        }
        txtSensor = findViewById(R.id.txtSensor)
        sensorManager = getSystemService(SENSOR_SERVICE) as
SensorManager
        linearAccelerationSensor =
sensorManager.getDefaultSensor(Sensor.TYPE_LINEAR_ACCELERATION)
        if (linearAccelerationSensor == null) {
            txtSensor.text = "Sensor de aceleración lineal no
disponible"
        }
    }
    override fun onResume() {
        super.onResume()
        if (linearAccelerationSensor != null) {
            sensorManager.registerListener(this,
linearAccelerationSensor, SensorManager.SENSOR_DELAY_NORMAL)
        }
    }
    override fun onPause() {
        super.onPause()
        sensorManager.unregisterListener(this)
    }
    override fun onSensorChanged(event: SensorEvent) {
        val x = event.values[0]
        val y = event.values[1]
        val z = event.values[2]
        val movementMagnitude = sqrt((x * x + y * y + z *
z).toDouble()).toFloat()
        if (movementMagnitude < threshold) {
            if (!isAtRest) {
                isAtRest = true
                txtSensor.text = "ESTADO DE REPOSO"
            }
        }
        else {
            if (isAtRest) {
                isAtRest = false
                txtSensor.text = "¡MOVIMIENTO DETECTADO!"
            }
        }
    }
    override fun onAccuracyChanged(sensor: Sensor?, accuracy: Int) {

    }
}
```

Ejercicio 5: Permisos.

(a) *¿Por qué algunos sensores requieren solicitar permisos en tiempo de ejecución y otros pueden usarse directamente? Investigar los niveles de los permisos en https://developer.android.com/guide/topics/permissions/overview#normal-dangerous.*

Algunos sensores requieren solicitar permisos en tiempo de ejecución y otros pueden usarse directamente porque los primeros tienen un nivel de permiso *Dangerous* y los otros *Normal*.

Los sensores que no comprometen información personal (como luz, proximidad o aceleración) pueden usarse directamente, mientras que los sensores que sí podrían exponer datos sensibles (como ubicación o salud) necesitan permisos en tiempo de ejecución y, por lo tanto, deben ser solicitados, explícitamente, al usuario en tiempo de ejecución.

(b) *Implementar una aplicación que solicite permisos en tiempo de ejecución para usar la ubicación del usuario. Se pueden utilizar los métodos para consultar y pedir los permisos definidos a continuación. Recordar declarar el permiso tanto en el Manifest como en la Activity.*

```
override fun onStart() {
    super.onStart()
    if (!chequearPermisosLocalizacion()) {
        pedirPermisosLocalizacion()
    } else {
        obtenerLocalizacion()
    }
}

override fun onRequestPermissionsResult(
    requestCode: Int,
    permissions: Array<String?>,
    grantResults: IntArray
) {
    super.onRequestPermissionsResult(requestCode, permissions,
grantResults)
    if (requestCode == PERMISSION_GRANTED) {
        if (grantResults.size > 0
            && grantResults[0] ==
PackageManager.PERMISSION_GRANTED
        ) {
            // El usuario concedió el permiso, ya podemos usar
la ubicación
            obtenerLocalizacion()
        }
    }
}

private fun chequearPermisosLocalizacion(): Boolean {
    val permissionState =
        ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION)
    return permissionState == PackageManager.PERMISSION_GRANTED
}

private fun pedirPermisosLocalizacion() {
    ActivityCompat.requestPermissions(
        this@MainActivity,
arrayOf(Manifest.permission.ACCESS_COARSE_LOCATION),
        PERMISSION_GRANTED
    )
}

fun obtenerLocalizacion() {

    //Acá usamos el método de geolocalización que hayamos
elegido
}
```

En *build.gradle.kts*:

```
implementation("com.google.android.gms:play-services-location:21.2.0")
```

En *AndroidManifest.xml*:

```
<uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
```

En *activity_main.xml*:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    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:id="@+id/main"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/txtSensor"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:textColor="@color/black"
        android:textSize="24sp"
        android:textStyle="bold"
        android:gravity="center" />
</LinearLayout>
```

En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity() {
    private lateinit var txtSensor: TextView
    companion object { private const val PERMISSION_REQUEST_CODE =
1001 }
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) {
v, insets ->
            val systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom)
            insets
        }
        txtSensor = findViewById(R.id.txtSensor)
    }
    override fun onStart() {
        super.onStart()
        if (!chequearPermisosLocalizacion()) {
            pedirPermisosLocalizacion()
        }
        else {
            obtenerLocalizacion()
        }
    }
    override fun onRequestPermissionsResult(requestCode: Int,
permissions: Array<out String>, grantResults: IntArray) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults)
        if (requestCode == PERMISSION_REQUEST_CODE) {
            if ((grantResults.isNotEmpty()) && (grantResults[0] ==
PackageManager.PERMISSION_GRANTED)) {
                Toast.makeText(this, "Permiso de ubicación concedido",
Toast.LENGTH_SHORT).show()
                obtenerLocalizacion()
            }
            else {
                Toast.makeText(this, "Permiso de ubicación denegado",
Toast.LENGTH_SHORT).show()
            }
        }
    }
```

```
    private fun chequearPermisosLocalizacion(): Boolean {
        return ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) ==
PackageManager.PERMISSION_GRANTED
    }
    private fun pedirPermisosLocalizacion() {
        ActivityCompat.requestPermissions(this,
arrayOf(Manifest.permission.ACCESS_COARSE_LOCATION),
PERMISSION_REQUEST_CODE)
    }
    private fun obtenerLocalizacion() {
        txtSensor.text = "Consultando ubicación..."
        if (!chequearPermisosLocalizacion()) return
        val fusedLocationClient =
LocationServices.getFusedLocationProviderClient(this)
        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
            Toast.makeText(this, "Permiso de ubicación denegado",
Toast.LENGTH_SHORT).show()
            return
        }
        val locationRequest =
LocationRequest.Builder(Priority.PRIORITY_HIGH_ACCURACY, 3000).build()
        val locationCallback = object : LocationCallback() {
            override fun onLocationResult(result: LocationResult) {
                val location = result.lastLocation
                if (location != null) {
                    txtSensor.text = String.format(
                        "Latitud: %.6f\nLongitud: %.6f",
                        location.latitude, location.longitude
                    )
                }
                else {
                    txtSensor.text = "Ubicación no disponible"
                }
            }
        }
        fusedLocationClient.requestLocationUpdates(
            locationRequest,
            locationCallback,
            Looper.getMainLooper()
        )
    }
}
```

Ejercicio 6: Cámara.

(a) *Implementar una aplicación que tome una fotografía usando un intent implícito y la muestre en un ImageView, tal como lo visto en la clase teórica.*

En *AndroidManifest.xml*:

```
<uses-feature android:name="android.hardware.camera"
android:required="true" />
```

En *activity_main.xml*:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp"
    tools:context=".MainActivity">
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="tomarFoto"
        android:text="Tomar Foto"
        android:textColor="@color/black"
        android:textSize="24sp"
        android:textStyle="bold"
        android:backgroundTint="#888888"
        android:layout_marginBottom="6dp" />
    <ImageView
        android:id="@+id/imgFoto"
        android:layout_width="250dp"
        android:layout_height="250dp"
        android:layout_gravity="center"
        android:scaleType="centerCrop" />
</LinearLayout>
```

En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity() {
    private val cameraLauncher =
registerForActivityResult(ActivityResultContracts.StartActivityForResu
lt())
    { result ->
        if (result.resultCode == RESULT_OK) {
            val imgFoto = findViewById<ImageView?>(R.id.imgFoto)
            val imgBitmap =
result.data?.extras?.getParcelable<Bitmap>("data")
            imgFoto.setImageBitmap(imgBitmap)
        }
    }
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
```

```
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) {
v, insets ->
            val systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom)
            insets
        }
    }
    fun tomarFoto(view: View?) {
        val i = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
        cameraLauncher.launch(i)
    }
}
```

(b) *Al usar la cámara con un intent implícito, no se necesita solicitar permiso al usuario en tiempo de ejecución. ¿Qué diferencia hay si se quiere usar la cámara de manera directa dentro de la app? Investigar en https://developer.android.com/guide/topics/media/camera#manifest.*

La diferencia que hay si se quiere usar la cámara de manera directa dentro de la *app* es que sí necesita declarar el permiso en el archivo *AndroidManifest.xml* y, además, solicitarlo en tiempo de ejecución.

(c) *Si se quiere guardar la imagen en la memoria de almacenamiento, ¿alcanza con haber solicitado el permiso de uso de la cámara?*

Si se quiere guardar la imagen en la memoria de almacenamiento, no alcanza con haber solicitado el permiso de uso de la cámara, sino que también hay que considerar el permiso de acceso al almacenamiento, dependiendo de la versión de *Android*.

Ejercicio 7: Posicionamiento.

(a) *Investigar las diferentes estrategias de obtener el posicionamiento del usuario (GPS, WiFi, Celular), sus ventajas y desventajas. Ver https://developer.android.com/guide/topics/location/strategies.*

Las diferentes estrategias de obtener el posicionamiento del usuario son:

- GPS: Usa satélites para calcular la ubicación geográfica precisa.
  Las ventajas son: alta precisión (5-10 metros); ideal para navegación al aire libre.
  Las desventajas son: no funciona bien en interiores; alto consumo de batería; puede tardar varios segundos en obtener la ubicación.
- WiFi: Usa torres de telefonía móvil y puntos de acceso WiFi cercanos para estimar la ubicación.
  Las ventajas son: funciona bien en interiores; bajo consumo de batería; rápida obtención de ubicación.
  Las desventajas son: menor precisión que el GPS (entre 20-100 metros); requiere conectividad a redes disponibles.
- Celular: Escucha actualizaciones de ubicación que otras *apps* ya están solicitando.
  Las ventajas son: casi sin impacto en la batería.
  Las desventajas son: no garantiza recibir actualizaciones a tiempo; no se controla cuándo ni con qué frecuencia se actualiza la ubicación.

(b) *Implementar una aplicación que muestre la latitud y longitud del usuario en un TextView, usando las APIs de ubicación de Google Play Services. Para esto, se debe agregar Google Play Services al archivo Gradle del proyecto, ubicado en directorio principal del proyecto: classpath 'com.google.gms:google-services:4.3.15'. Debe quedar de la siguiente forma:*

```
// Top-level build file where you can add configuration options common to all

buildscript {

    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.1.3'
        classpath 'com.google.gms:google-services:3.2.0'

        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        google()
        jcenter()
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
```

Luego, se agrega la dependencia de las APIs de ubicación en el archivo gradle de /app/build.gradle: implementation 'com.google.android.gms:play-services-location:15.0.1'.

```
1   apply plugin: 'com.android.application'
2
3   android {
4       compileSdkVersion 27
5       defaultConfig {
6           applicationId "com.example.alfonso.cameraexample"
7           minSdkVersion 23
8           targetSdkVersion 27
9           versionCode 1
10          versionName "1.0"
11          testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
12      }
13      buildTypes {
14          release {
15              minifyEnabled false
16              proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-ru
17          }
18      }
19  }
20
21  dependencies {
22      implementation fileTree(dir: 'libs', include: ['*.jar'])
23      implementation 'com.android.support:appcompat-v7:27.1.1'
24      implementation 'com.android.support:support-v4:27.1.1'
25      implementation 'com.android.support:support-media-compat:27.1.1'
26      implementation 'com.android.support.constraint:constraint-layout:1.1.1'
27      implementation 'com.google.android.gms:play-services-location:15.0.1'
28      testImplementation 'junit:junit:4.12'
29      androidTestImplementation 'com.android.support.test:runner:1.0.2'
30      androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
31  }
```

Una vez hecho esto, ya se incluye la librería de ubicación de Google Play Services en la aplicación. Por lo tanto, se pueden usar las clases que provee la librería para solicitar la ubicación:

```
private var client: FusedLocationProviderClient? = null
private var locationCallback: LocationCallback? = null
```

En el método onCreate, inicializamos estas variables:

```
client = LocationServices.getFusedLocationProviderClient(this)
locationCallback = object :
   LocationCallback() {
   override fun onLocationResult(locationResult: LocationResult) {
       super.onLocationResult(locationResult)
       //usamos locationResult.getLastLocation() para tener
       //la ubicación más reciente
   }
}
```

Y, una vez que se tengan los permisos, se le solicita al cliente que envíe la información de localización al callback:

```
client?.requestLocationUpdates(LocationRequest.create(),
   locationCallback as LocationCallback, null
)
```

En *build.gradle.kts*:

```
implementation("com.google.android.gms:play-services-location:21.2.0")
```

En *AndroidManifest.xml*:

```
<uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
```

En *activity_main.xml*:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center"
    android:padding="16dp"
    tools:context=".MainActivity">
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="obtenerUbicacion"
        android:text="Obtener ubicación"
        android:textColor="@color/black"
```

```
        android:textSize="24sp"
        android:textStyle="bold"
        android:backgroundTint="#888888"
        android:layout_marginBottom="20dp" />
    <TextView
        android:id="@+id/txtUbicacion"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@color/black"
        android:textSize="20sp" />
 </LinearLayout>
```

En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity() {
    private lateinit var txtUbicacion: TextView
    companion object { private const val PERMISSION_REQUEST_CODE =
1001 }
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) {
v, insets ->
            val systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom)
            insets
        }
        txtUbicacion = findViewById(R.id.txtUbicacion)
    }
    private fun checkLocationPermission(): Boolean {
        val coarseLocation = ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION)
        val fineLocation = ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION)
        if ((coarseLocation != PackageManager.PERMISSION_GRANTED) ||
(fineLocation != PackageManager.PERMISSION_GRANTED)) {
            requestLocationPermission()
            return false
        }
        return true
    }
    private fun requestLocationPermission() {
        ActivityCompat.requestPermissions(this,
arrayOf(Manifest.permission.ACCESS_COARSE_LOCATION,
Manifest.permission.ACCESS_FINE_LOCATION), PERMISSION_REQUEST_CODE)
    }
    override fun onRequestPermissionsResult(requestCode: Int,
permissions: Array<out String>, grantResults: IntArray) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults)
        if (requestCode == PERMISSION_REQUEST_CODE) {
            if ((grantResults.isNotEmpty()) && (grantResults[0] ==
PackageManager.PERMISSION_GRANTED)) {
                Toast.makeText(this, "Permiso de ubicación concedido",
Toast.LENGTH_SHORT).show()
                obtenerUbicacion(null)
            }
```

```
            else {
                Toast.makeText(this, "Permiso de ubicación denegado",
Toast.LENGTH_SHORT).show()
            }
        }
    }
    fun obtenerUbicacion(view: View?) {
        txtUbicacion.text = "Consultando ubicación..."
        if (!checkLocationPermission()) return
        val fusedLocationClient =
LocationServices.getFusedLocationProviderClient(this)
        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
            Toast.makeText(this, "Permiso de ubicación denegado",
Toast.LENGTH_SHORT).show()
            return
        }
        val locationRequest =
LocationRequest.Builder(Priority.PRIORITY_HIGH_ACCURACY, 3000).build()
        val locationCallback = object : LocationCallback() {
            override fun onLocationResult(result: LocationResult) {
                val location = result.lastLocation
                if (location != null) {
                    txtUbicacion.text = String.format(
                        "Latitud: %.6f\nLongitud: %.6f",
                        location.latitude, location.longitude
                    )
                }
                else {
                    txtUbicacion.text = "Ubicación no disponible"
                }
            }
        }
        fusedLocationClient.requestLocationUpdates(locationRequest,
locationCallback, Looper.getMainLooper())
    }
}
```

(c) *Implementar una aplicación que, al iniciar, muestre a qué distancia se encuentra el usuario de la Facultad de Informática. Investigar los métodos que provee la clase Location. La ubicación de la Facultad es -34.9037905, -57.9378442.*

En *build.gradle.kts*:

```
implementation("com.google.android.gms:play-services-location:21.2.0")
```

En *AndroidManifest.xml*:

```
<uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
```

En *activity_main.xml*:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center"
    android:padding="16dp"
    tools:context=".MainActivity">
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="obtenerDistancia"
        android:text="Obtener distancia"
        android:textColor="@color/black"
        android:textSize="24sp"
        android:textStyle="bold"
        android:backgroundTint="#888888"
        android:layout_marginBottom="20dp" />
    <TextView
        android:id="@+id/txtUbicacion"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@color/black"
        android:textSize="20sp"
        android:gravity="center" />
</LinearLayout>
```

En *MainActivity.kt*:

```
class MainActivity : AppCompatActivity() {
    private lateinit var txtUbicacion: TextView
    companion object { private const val PERMISSION_REQUEST_CODE =
1001 }
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) {
v, insets ->
            val systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom)
            insets
        }
        txtUbicacion = findViewById(R.id.txtUbicacion)
    }
    private fun checkLocationPermission(): Boolean {
        val coarseLocation = ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION)
        val fineLocation = ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION)
        if ((coarseLocation != PackageManager.PERMISSION_GRANTED) ||
(fineLocation != PackageManager.PERMISSION_GRANTED)) {
            requestLocationPermission()
            return false
```

```
        }
        return true
    }
    private fun requestLocationPermission() {
        ActivityCompat.requestPermissions(this,
arrayOf(Manifest.permission.ACCESS_COARSE_LOCATION,
Manifest.permission.ACCESS_FINE_LOCATION), PERMISSION_REQUEST_CODE)
    }
    override fun onRequestPermissionsResult(requestCode: Int,
permissions: Array<out String>, grantResults: IntArray) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults)
        if (requestCode == PERMISSION_REQUEST_CODE) {
            if ((grantResults.isNotEmpty()) && (grantResults[0] ==
PackageManager.PERMISSION_GRANTED)) {
                Toast.makeText(this, "Permiso de ubicación concedido",
Toast.LENGTH_SHORT).show()
                obtenerDistancia(null)
            }
            else {
                Toast.makeText(this, "Permiso de ubicación denegado",
Toast.LENGTH_SHORT).show()
            }
        }
    }
    fun obtenerDistancia(view: View?) {
        txtUbicacion.text = "Consultando distancia..."
        if (!checkLocationPermission()) return
        val fusedLocationClient =
LocationServices.getFusedLocationProviderClient(this)
        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
            Toast.makeText(this, "Permiso de ubicación denegado",
Toast.LENGTH_SHORT).show()
            return
        }
        val locationRequest =
LocationRequest.Builder(Priority.PRIORITY_HIGH_ACCURACY, 3000).build()
        val locationCallback = object : LocationCallback() {
            override fun onLocationResult(result: LocationResult) {
                val location = result.lastLocation
                if (location != null) {
                    val facultad = Location("").apply {
                        latitude = -34.9037905
                        longitude = -57.9378442
                    }
                    val distanciaMetros =
location.distanceTo(facultad)
                    txtUbicacion.text = String.format(
                        "Estás a %.2f kilómetros de la Facultad de
Informática\n(Latitud: %.6f, Longitud: %.6f)",
                        distanciaMetros/1000, location.latitude,
location.longitude
                    )
                }
                else {
                    txtUbicacion.text = "Ubicación no disponible"
                }
            }
        }
```

```
        fusedLocationClient.requestLocationUpdates(locationRequest,
locationCallback, Looper.getMainLooper())
    }
}
```