SENSOR DE PROXIMIDAD

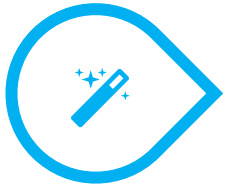
DESARROLLO DE APLICACIONES MÓVILES AVANZADAS

LUIS HUMBERTO RIVAS RODRÍGUEZ
INGENIERO EN SISTEMAS INFORMÁTICOS
Y MÁSTER EN DIRECCIÓN ESTRATÉGICA DE EMPRESAS.

Agenda

1 **Introducción a Sensores**
Conceptos Básicos

Implementación 2
Registro de Sensores



3 **Uso de Sensores**
Configuración

Cierre 4
Valoraciones Finales

RESULTADOS DE APRENDIZAJE

Competencias a desarrollar al finalizar la sesión.

INTERPRETA
Los componentes de software necesarios para construir una interfaz gráfica en Android.

COMPRENDE
La relación que tienen los componentes y su importancia en el desarrollo de aplicaciones basadas en Android.

CONSTRUYE
Aplicaciones Android haciendo uso de los componentes avanzados como son los sensores en Android.

INTRODUCCIÓN A SENSORES

CONCEPTOS BÁSICOS

- ¿Qué son los sensores en Android? Dispositivos electrónicos integrados que permiten al sistema operativo y las aplicaciones obtener información del entorno físico.

- Los sensores pueden medir variables como **movimiento**, **orientación**, **luz**, **proximidad**, etc.

INTRODUCCIÓN A SENSORES

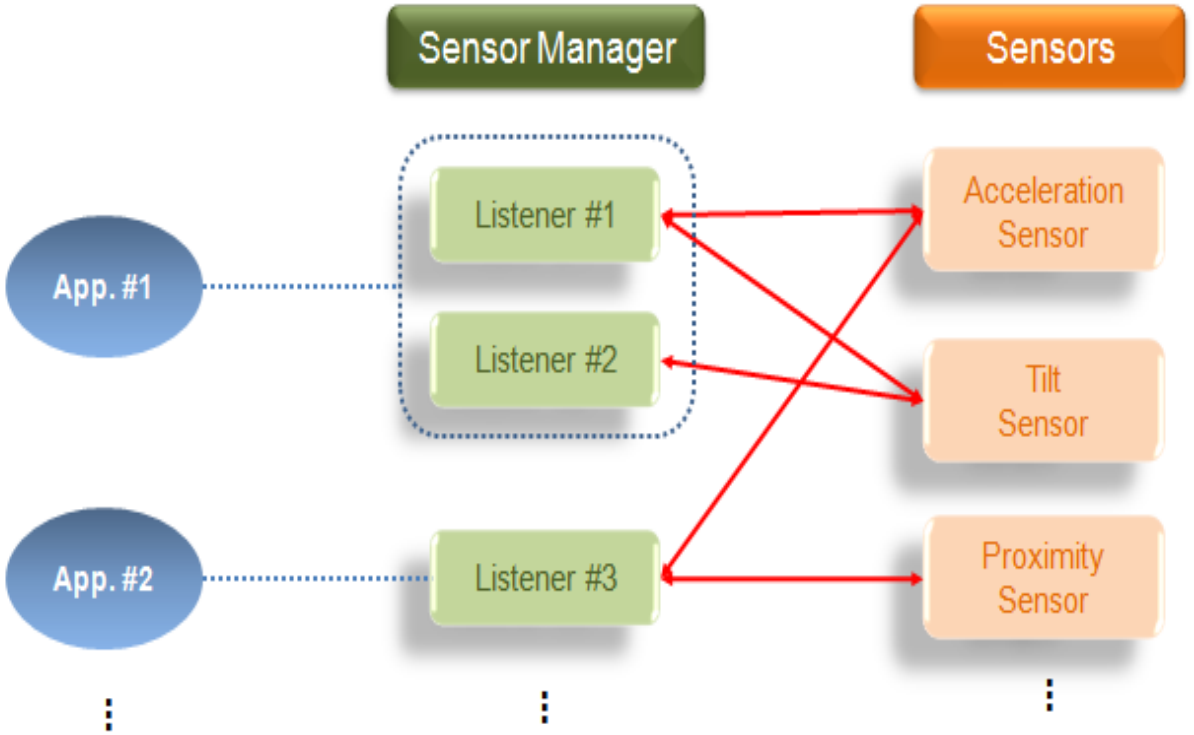
CONCEPTOS BÁSICOS

- Sensor: Dispositivo que capta magnitudes físicas (variaciones de luz, temperatura, sonido, etc.) u otras alteraciones de su entorno.
- En el caso de los dispositivos Android, estos dependiendo de la gama de teléfono pueda que traiga uno o muchos. El infalible en cualquier teléfono es el **sensor de proximidad**.

IMPLEMENTACIÓN

REGISTRO DE SENSORES

- Para poder hacer uso de los sensores en Android Studio, será necesario usar al menos 3 clases imprescindibles que son:
  - **SensorManager:** clase que permite gestionar los sensores instalados en un teléfono.
  - **Sensor:** clase que representa un sensor.
  - **SensorEventListener:** clase que permite determinar cuándo hay cambios en los valores de un sensor.



IMPLEMENTACIÓN

REGISTRO DE SENSORES

Pasos para el Registro de Sensores

1. Obtener una instancia de SensorManager.
2. Identificar el sensor que deseas usar.
3. Crear un listener para manejar los eventos del sensor.
4. Registrar el listener.
5. Desregistrar el listener cuando no sea necesario (e.g., en onPause).

```
    private lateinit var sensorManager: SensorManager
    private var accelerometer: Sensor? = null

    private val sensorEventListener = object : SensorEventListener {
        override fun onSensorChanged(event: SensorEvent) {
            // Procesar valores del sensor
            val x = event.values[0]
            val y = event.values[1]
            val z = event.values[2]
            println("X: $x, Y: $y, Z: $z")
        }
        override fun onAccuracyChanged(sensor: Sensor?, accuracy: Int) {}
    }
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        // Obtener SensorManager y el acelerómetro
        sensorManager = getSystemService(SENSOR_SERVICE) as SensorManager
        accelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER)

        // Registrar el listener
        accelerometer?.let {
            sensorManager.registerListener(sensorEventListener, it, SensorManager.SENSOR_DELAY_NORMAL)
        }
    }
    override fun onPause() {
        super.onPause()
        // Desregistrar el listener
        sensorManager.unregisterListener(sensorEventListener)
    }
}
```

IMPLEMENTACIÓN

REGISTRO DE SENSORES

```
class MainActivity : AppCompatActivity() {
    val sensorManager = getSystemService(SENSOR_SERVICE) as SensorManager
    val accelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER)
    val gyroscope = sensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE)
    val sensorEventListener = object : SensorEventListener {
        override fun onSensorChanged(event: SensorEvent) {
            when (event.sensor.type) {
                Sensor.TYPE_ACCELEROMETER -> {
                    val x = event.values[0]
                    val y = event.values[1]
                    val z = event.values[2]
                    println("Acelerómetro - X: $x, Y: $y, Z: $z")
                }
                Sensor.TYPE_GYROSCOPE -> {
                    val x = event.values[0]
                    val y = event.values[1]
                    val z = event.values[2]
                    println("Giroscopio - X: $x, Y: $y, Z: $z")
                }
            }
        }
        override fun onAccuracyChanged(sensor: Sensor?, accuracy: Int) {}
    }
    override fun onResume() {
        super.onResume()
        sensorManager.registerListener(sensorEventListener, accelerometer, SensorManager.SENSOR_DELAY_UI)
        sensorManager.registerListener(sensorEventListener, gyroscope, SensorManager.SENSOR_DELAY_UI)
    }
    override fun onPause() {
        super.onPause()
        sensorManager.unregisterListener(sensorEventListener)
    }
```

Es posible registrar y manejar varios sensores al mismo tiempo, si la aplicación requiere datos de varios sensores, puedes registrarlos todos con sus respectivos listeners.

Cuando tienes varios sensores que necesitan ser monitoreados simultáneamente, puedes registrar cada sensor con su propio listener o un único listener que maneje todos los eventos de los sensores.

USO DE SENSORES

CONFIGURACIÓN

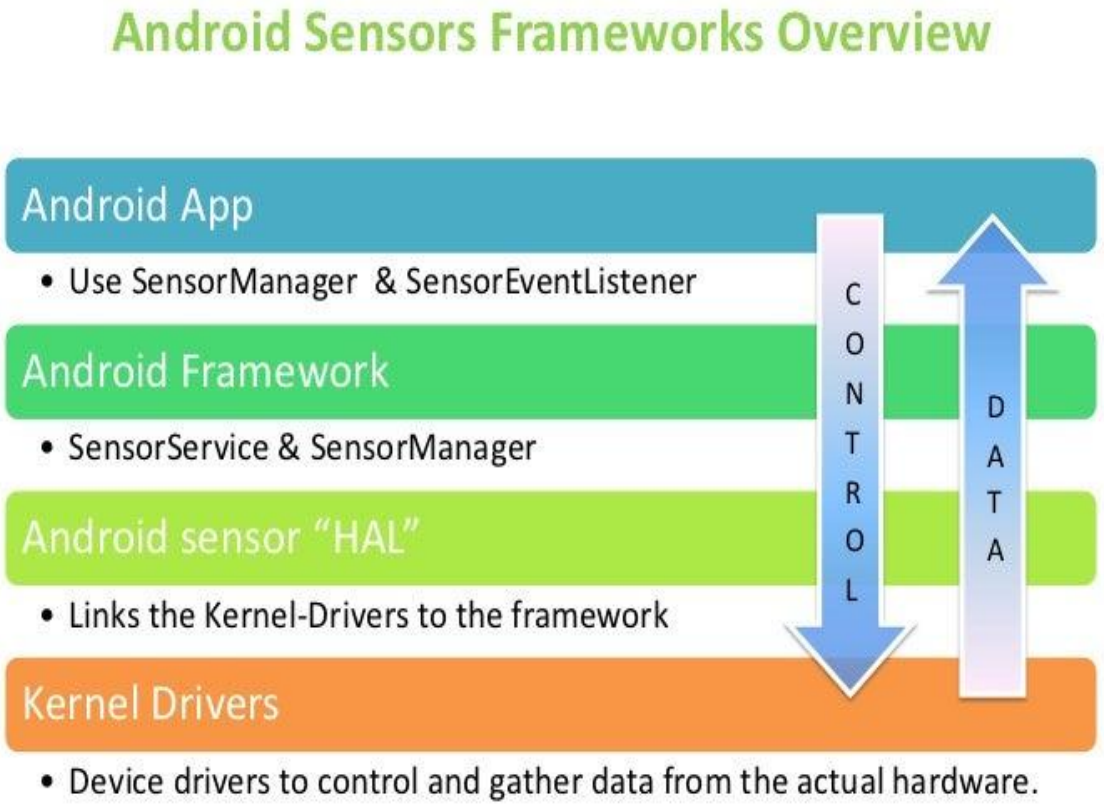
- Al recibir una llamada y acercar nuestro dispositivo móvil, este automáticamente apaga la pantalla para aumentar la vida de uso de la batería.
- Al alejar nuestro dispositivo móvil, la pantalla se vuelve a activar.
- Otro uso por ejemplo es la gestión de gestos en los dispositivos, para capturar pantalla o tomar una fotografía.
- También podría utilizarse en otros escenarios, integrando por ejemplo un sensor de proximidad y un sensor de temperatura, así se podría determinar la temperatura que tiene una persona al acercarse al dispositivo.

USO DE SENSORES

CONFIGURACIÓN

El uso de sensores en aplicaciones Android requiere configuraciones específicas para controlar aspectos como la frecuencia de lectura de los datos o el comportamiento del sensor en diferentes estados de la aplicación.

Estas configuraciones optimizan el rendimiento y la eficiencia del dispositivo.



USO DE SENSORES

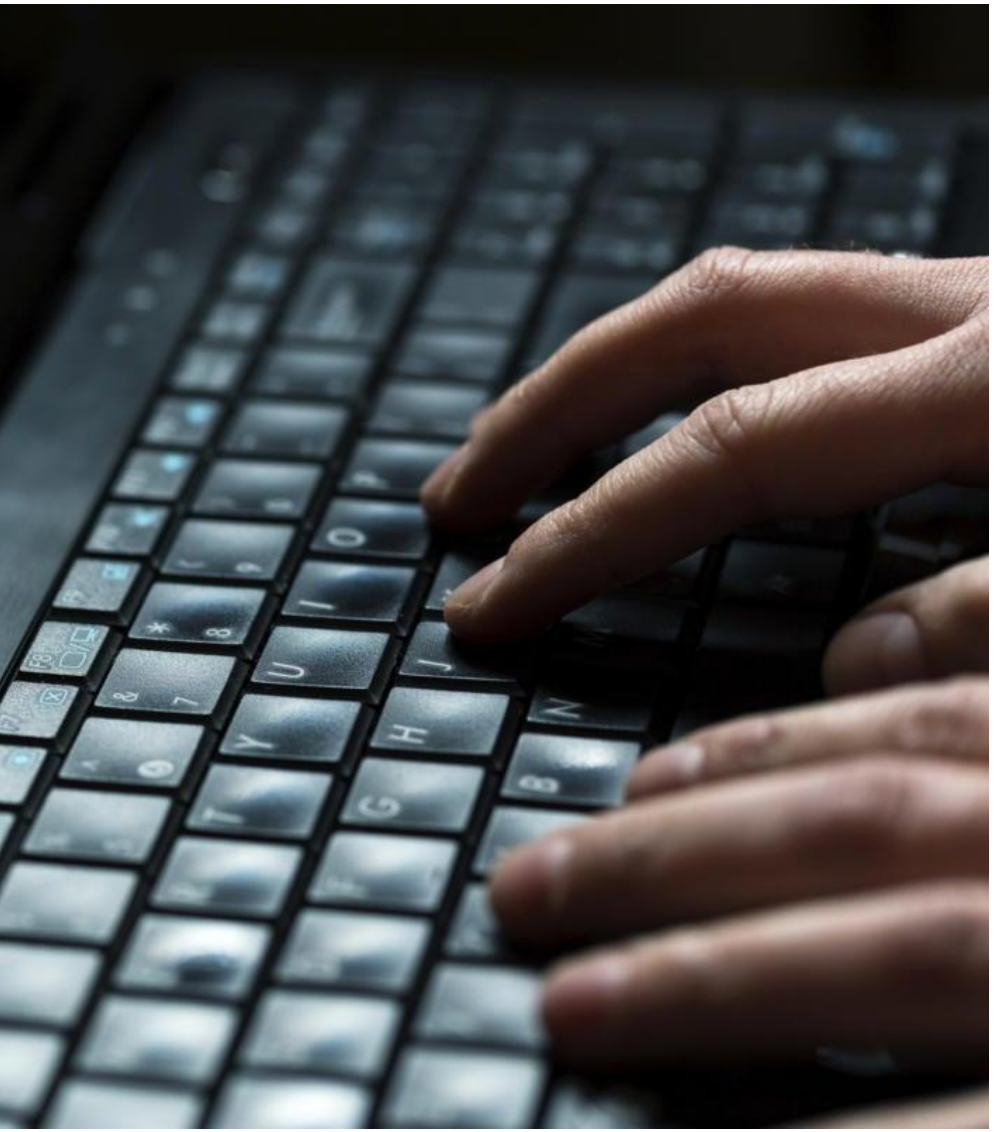
CONFIGURACIÓN

Al registrar un sensor, puedes elegir entre cuatro frecuencias predefinidas:

- **SENSOR_DELAY_FASTEST:** Máxima frecuencia, ideal para tiempo real.
- **SENSOR_DELAY_GAME:** Frecuencia para juegos.
- **SENSOR_DELAY_UI:** Frecuencia para actualizaciones de UI.
- **SENSOR_DELAY_NORMAL:** Frecuencia estándar, equilibrada en precisión y consumo.

LIVE CODING

TRABAJO A REALIZAR



- Tomando como referencia los contenidos visto en la sesión, se elaborará una aplicación móvil que permita manejar el uso de los sensores de proximidad en Android.

VALORACIONES **FINALES**

Comentarios sobre el tema.

1

ABSTRACCIÓN

Las aplicaciones Android se ensamblan a partir de clases, objetos.

2

INTENT

La acción de invocar una ventana se le denomina Intent

3

CREATIVO

Saber distribuir y seleccionar los diferentes métodos y propiedades de una clase, permitirá al programador trabajar de forma rápida y eficiente

¿PREGUNTAS?

DESARROLLO DE APLICACIONES MÓVILES AVANZADAS

LUIS HUMBERTO RIVAS RODRÍGUEZ
INGENIERO EN SISTEMAS INFORMÁTICOS
Y MÁSTER EN DIRECCIÓN ESTRATÉGICA DE EMPRESAS.