**Sesión 4-02 Clase del 12 de febrero**

En esta sesión vamos a realizar una programación básica del uso de sensores del dispositivo. Un dispositivo puede contar con varios sensores como:

* Acelerómetro
* Gravedad
* Giroscopio
* Orientación
* Proximidad
* Temperatura ambiente
* Detector de pasos
* Luz ambiental

Cuando se activa un sensor en una app, lee y notifica valores a la app con una altísima frecuencia. Esto hace que el uso de sensores provoca un alto consumo de energía y que no se debe abusar en el uso de sensores en las apps. Sin embargo, es evidente que muchas aplicaciones los usan y algunas, como los juegos que usan físicas de movimientos necesitan que la frecuencia de las lecturas sean lo más altas posibles para simular la realidad de los movimientos.

1.- En primer lugar, vamos a realizar una app muy sencilla que muestra una lista de los sensores disponibles para el dispositivo. LO vamos a hacer todo en MainActivity:

1.1.- Crea un composable **SensoresView** para visualizar la lista de sensores.

1.2.- En **SensoresView** crea una variable **context** para el contexto de la aplicación.

1.3.- Crea una variable de estado para gestionar el acceso a los sensores:

// Obtener el SensorManager del sistema  
val sensorManager = remember **{** // getSystemService se usa para acceder al servicio de sensores  
 context.getSystemService(*SENSOR\_SERVICE*) as SensorManager  
**}**

1.4.- Obtener una lista de objetos **Sensor** con información de los sensores del dispositivo:

// Obtener la lista de sensores disponibles  
val sensorList = remember **{**

// con TYPE\_ALL Devuelve todos los sensores disponibles  
sensorManager.getSensorList(Sensor.*TYPE\_ALL*)

**}**

1.5.- Sabiendo que cada elemento de esa lista es un objeto Sensor, escribir en pantalla una lista con el texto del tipo y nombre del tipo de cada sensor.

Se trata de comprobar si el dispositivo tiene acceso a Internet y si lo tiene a través de la red wifi y/o a través de la red de datos.

2.- Crea un nuevo proyecto para probar el sensor de proximidad. El sensor de proximidad es unidimensional, es decir, solo lee y devuelve un valor cada vez que hace una lectura (la distancia del sensor a un objeto cercano).

En este proyecto también vamos a tener una sola pantalla dentro de MainActivity. Para probar en el emulador los sensores, tendrás que acceder a los controles extendidos del dispositivo emulado.

2.1.- En la declaración de la clase MainActivity tenemos que indicar que implemente el interface **SensorEventListener** (escuchador de eventos de actualización de sensores).

class MainActivity : ComponentActivity(), SensorEventListener

2.2.- Al implementar el interface se exigirá implementar las funciones miembro de ese interface:

* **onSensorChanged** (que se hace cada vez que el sensor notifica un valor)
* **onAccuracyChanged** (que se hace cuando cambia la configuración del sensor)

Más adelante programaremos **onSensorChanged**. La segunda función no necesitaremos programarla.

2.3.- En la clase MainActivity declaramos tres variables miembro con los objetivos indicados en los comentarios:

// lateinit sirve para inicializar variables más tarde  
// SensorManager da acceso a los sensores del dispositivo  
private lateinit var sensorManager: SensorManager  
// Variable para el Sensor de proximidad  
private var proximitySensor: Sensor? = null  
// Estado para almacenar la lectura del sensor  
private var proximityValue by *mutableStateOf*("No disponible")

2.4.- En el método **onCreate** del ciclo de vida de MainActivity, iniciamos los objetos **sensorManager** y **proximitySensor.**

// Inicializar SensorManager  
sensorManager = getSystemService(*SENSOR\_SERVICE*) as SensorManager  
//obtener una instancia del sensor de proximidad  
proximitySensor = sensorManager.getDefaultSensor(Sensor.*TYPE\_PROXIMITY*)

2.5.- Programa ya el método **onSensorChanged** para que el valor de la variable **proximityValue** cambie cada vez que este método se ejecuta (debido a que el sensor notifique de un nuevo valor). Hay que tener en cuenta que ese método recibe un objeto **evento.** En un objeto evento, en el array values hay los valores medidos de tipo (Float). En este sensor hay un único valor en el array.

// event puede ser nulo, verifica antes de usarlo  
event?.*let* **{** // Actualizar el valor del sensor en la UI  
 proximityValue = **it**.values[0].toString()  
**}**

2.6.- Hasta aquí todavía el sensor no enviaría datos a la app. Para se comunique con **MainActivity** enviando datos, hay que registrarlo. Esto normalmente se hace en el método o**nResume** del ciclo de vida de la Activity (Cuando el usuario puede interactuar con la pantalla).

override fun onResume() {  
 super.onResume()  
 // Registrar el listener del sensor cuando la actividad está activa  
 proximitySensor?.*let* **{** sensor **->** sensorManager.registerListener(this, sensor, SensorManager.*SENSOR\_DELAY\_NORMAL*)  
 **}**}

La constante SENSOR\_DELAY\_NORMAL tiene el valor 3, con este valor se indica que cada 3 múltiplos de un tiempo base (que depende del hardware del dispositivo) se toman lecturas en el sensor. Con ese valor aproximadamente se tiene un intervalo de lectura de 200 mseg. Hay otros valores prestablecidos o se puede indicar con otro valor (el que queramos en múltiplo de ese tiempo base). Por ejemplo, si usáramos el sensor de temperatura, serían interesantes tiempos mayores.

2.6.- Para que el sensor no esté consumiendo energía cuando la aplicación esté en segundo plano o para que no siga activado aunque la aplicación se cierre, se debe “desregistrar” el sensor en el método **onPause** del ciclo de vida de la Activity.

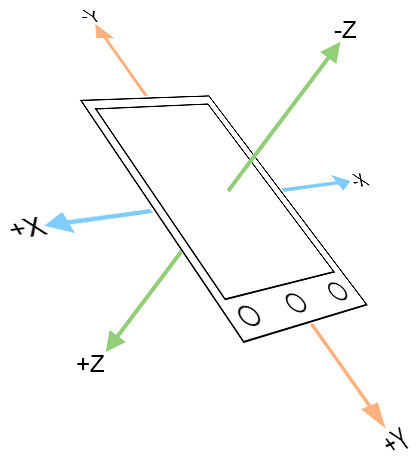
override fun onPause() {  
 super.onPause()  
 // Desregistrar el sensor para ahorrar batería  
 sensorManager.unregisterListener(this)  
}

2.7.- Programar que en pantalla del dispositivo se muestre en un Text de letra grande el valor medido por el sensor.

2.8.- Programar que cuando el sensor detecte menos de 2 cm. la pantalla se ponga en color negro y cuando detecte más de 2 cm tenga la pantalla el color rojo.

3.- Debes realizar una app que obtiene valores de como esta colocado tridimensionalmente el dispositivo. Esto se puede realizar con el sensor de rotación o con el acelerómetro.

El acelerómetro detecta la aceleración del dispositivo en los tres ejes de posición del dispositivo.



Por tanto, **onSensorChanged** toma los valores en un array de 3 float.