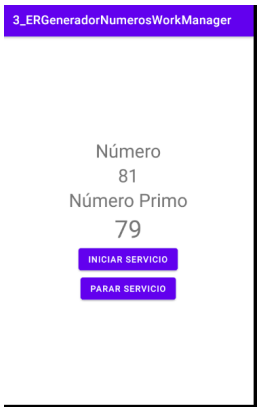
Bloc 10. Exercici Resolt Números Cosins

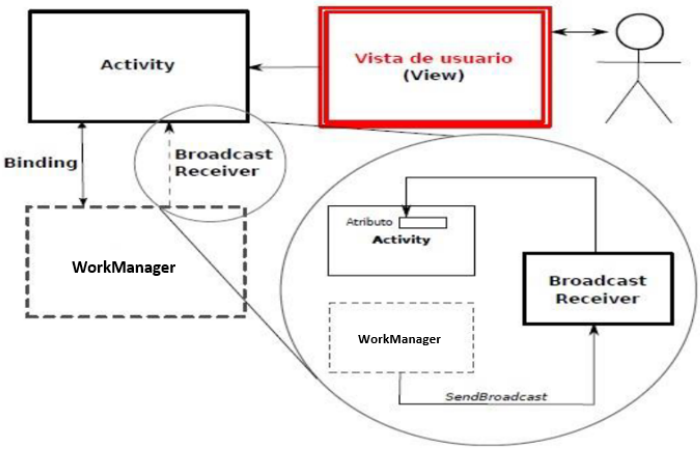
Descarregar este exercici

Realitzarem una aplicació que rep números a través d'un treball, creat amb `WorkManager`, i actualitza la interfície gràfica, concretament es visualitzarà l'últim número generat i l'últim nombre primer detectat per la pantalla del dispositiu. L'aplicació tindrà dos botons per a iniciar i detindre el servici.



Els components de la nostra aplicació seran els següents:

- Activity: amb una vista associada que permeta a l'usuari parar i reiniciar el servici i mostrar les dades corresponents.
- WorkManager: un treball que genere en background els números independentment de l'activity.
- BroadcastReceiver: un receptor de missatges de manera que cada vegada que es rep un número, el processa (per a saber si és cosí) i actualitza la vista de l'activity. El broadcast receiver pot ser definit junt amb l'activity o en una classe a part.



Comencem definint el codi del nostre treball que cridarem **GeneradorNumeros.kt**. Cal tindre en compte que quan s'implementa un `WorkManager` com una classe derivada de `Worker` és necessari definir el mètode `doWork()`, és ací on s'implementen les instruccions que permeten proporcionar el servici a través d'un fil diferent.

El codi del nostre servici:

```
    class GeneradorNumeros(var context: Context, workerParams: WorkerParameters) :
2       Worker(context, workerParams) {
        override fun doWork(): Result {
4           val datos=inputData
5           var contador= datos.getInt("INICIO",0)
6           while (!isStopped) {
7               var i = Intent(ACCION)
                i.putExtra("VALORCONTADOR", contador)
9               applicationContext.sendBroadcast(i)
                contador++
                Thread.sleep(1000)
            }
            return Result.success()
        }
15      companion object
        {
            const val ACCION = "com.ejemplos.b10.a3_ergeneradornumerosworkmanager.recibirNume
        }
    }
```

📌 **Línea 2** heretem de la classe Worker que ens obligarà a implementar el mètode necessari. En el mètode doWork() es codifica la funcionalitat del treball. **Línea 4 - 5** ací recuperem un Data per a extraure la informació manada al crear el treball, en este cas el valor d'inici del comptador. Recordeu que este l'única cosa que fa és generar números (bucle amb comptador) , és en el broadcast on es realitzaran les comprovacions de si és o no cosí i es comunicarà a l'activity principal. Observar que la constant ACCIÓ de l'intent **i**, **Línea 7** té com a valor una cadena única que identificarà el treball en el `BroadcastReceiver` , usem una variable estàtica per a fer-ho **Línea 15**. Manem el Broadcast passant com a paràmetres el valor del comptador **Línea 9**. Per a cada iteració es manarà el broadcast, per la qual cosa aniran arribant al receiver els números.

Vegem ara la implementació de la **MainActivity.kt**:

```
class MainActivity : AppCompatActivity() {
    lateinit var mreceiver: MyReceiver
    lateinit var t1: TextView
    lateinit var t2: TextView
    lateinit var idWork:UUID
    var iniciado=false
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        t1 = findViewById(R.id.textView1)
        t2 = findViewById<TextView>(R.id.textView2)
        mreceiver = MyReceiver(t1, t2)
        val b1 = findViewById<Button>(R.id.button1)
        val b2 = findViewById<Button>(R.id.button2)
        b1.setOnClickListener { iniciarServicio() }
        b2.setOnClickListener { pararServicio() }
    }
}
```

Per ara res a comentar, només creguem el broadcast receiver i associem elements a les view del layout. Implementem també la funcionalitat dels botons, cridant a dos mètodes que implementarem a continuación. Ens crearem també una propietat per a guardar l'instancia del treball i poder accedir per a la seua cancel·lació.

El botó d'inici de servici té el codi següent:

```
    fun iniciarServicio() {
2           if(!iniciado)
            {
                iniciado=true
5               val filter = IntentFilter(GeneradorNumeros.ACCION)
                this.registerReceiver(mreceiver, filter)
                val datos=Data.Builder()
8               datos.putInt("INICIO",t1.text.toString().toInt())
9               val workManager = WorkManager.getInstance(MainActivity@ this)
10              val work = OneTimeWorkRequest.Builder(GeneradorNumeros::class.java)
11                  .setInputData(datos.build())
                    .build()
                idWork=work.id
14              workManager.enqueue(work)
15
16          }
            else  Toast.makeText(this, "El trabajo ya se ha lanzado", Toast.LENGTH_LONG)
            .show()
        }
```

📌**Línea 2** es comprova si s'havia iniciat amb anterioritat el treball, per a no llançar un altre si abans no ha sigut parat. **Línea 5** es crega el `IntentFilter` per a registrar el receiver amb una cadena única que permeta al BroadcastReceiver reconéixer d'on ve la informació, registrem el receiver afegint el filter creat. **Línea 9-10** es crega un `Data.Build` per a passar la informació necessària al `Work` , en este cas un sencer perquè el comptador conserve el número pel qual estava funcionant, encara que es reinicie el treball. `Línea 11-14` es crega un treballa únic, s'afigen les dades anteriors i es construïx. `Línea 15` se guarda l'aneu generat per este treball per a poder cancel·lar-ho posteriorment. `Línea 16` s'encola el treball perquè comence el seu funcionament quan el sistema puga fer-ho.

El botó de parada de servici té el codi següent:

```
    fun pararServicio() {
2               if (iniciado) {
                        iniciado=false
4                       WorkManager.getInstance(MainActivity@ this)
                        .cancelWorkById(idWork)
6                       unregisterReceiver(mreceiver)
                        Toast.makeText(this, "Trabajo detenido", Toast.LENGTH_LONG)
                        .show()
                  } else Toast.makeText(this, "El Trabajo ya estaba detenido",
                                        Toast.LENGTH_LONG).show()
        }
```

📌**Línea 2**, comprovem que el servici encara no està parat. Localitzem el servici i ho cancel·lem **Línea 4** i anul·lem el registre del `BroadcastService()` **Línea 6**.

Passem ara a definir el codi del receptor **MyReceiver**, ho farem com a classe interna a MainActivity.

Cada vegada que el treball genera un nou número, envia un missatge de broadcast a través d'un `IntentFilter`. El receptor es va registrar per a atendre missatges a través d'un IntentFilter on s'especifica l'acció que atendrà, és per açò que cada vegada que reba un missatge el receptor MyReceiver disparará el mètode `onReceive()`. S'obtindran les dades extres de l'intent i s'actualitzaran els dos TextView. Un d'ells s'actualitza sempre i un altre només si el comptador que s'ha passat en un nombre primer.
Recordeu que en el codi de l'activity principal es passen al constructor del receptor dos paràmetres, que són les referències de les vistes que hem d'actualitzar.

```
class MyReceiver(private val t1: TextView, private val t2: TextView) : BroadcastReceiver() {
        fun esPrimo(n: Int): Boolean {
            var i: Int
            i = 2
            while (i < n) {
                if (n % i == 0) return false
                i++
            }
            return true
        }
        override fun onReceive(context: Context?, intent: Intent) {
            val s = intent.extras!!["VALORCONTADOR"].toString()
            t1.text = s
            if (esPrimo(s.toInt())) t2.text = s
        }
        init {
            t1.text = "0"
            t2.text = "0"
        }
    }
```