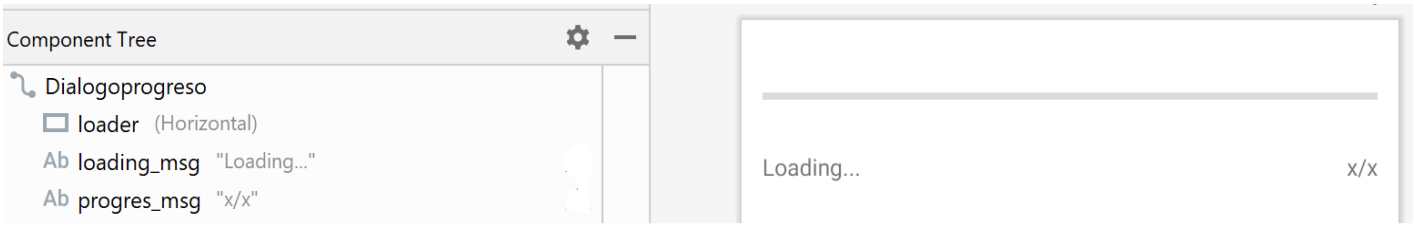
Bloc 10 . Exercici Resolt Barra Progrés

Descarregar este ejercici

Desenvoluparem una aplicació que simule la càrrega d'un conjunt d'arxius, simularem la càrrega mitjançant `Corrutinas` on especificarem un retard per cada pas de la tasca, i utilitzarem una `LinearProgressIndicator` per a veure l'evolució de l'estat de la càrrega.

El disseny del nostre **LinearProgressIndicator`** serà personalitzat, incloent uns camps de text tal com es veu en la següent imatge:



L'arxiu xml corresponent a aquesta interfície seria:

dialogo_progress.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    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:padding="13dp"
    android:id="@+id/Dialogoprogreso"
    android:layout_centerHorizontal="true"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <com.google.android.material.progressindicator.LinearProgressIndicator
        android:id="@+id/loader"
        style="@style/Widget.MaterialComponents.LinearProgressIndicator"
        android:layout_width="match_parent"
        android:layout_height="65dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"/>

    <TextView
        android:id="@+id/loading_msg"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:text="Loading..."
        android:textAppearance="?android:textAppearanceSmall"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/loader"/>

    <TextView
        android:id="@+id/progres_msg"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="x/x"
        android:textAppearance="?android:textAppearanceSmall"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/loader" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

📌 Aclariments:

- **Línies 12-18**: definició del `LinearProgressIndicator`. En la línia 16 especifiquem el text que es visualitzarà quan el `FAB` estiga premut. La resta de propietats ja estan explicades.
- **Línies 20-28**: ací definim el primer `Text` que indicarà el que fa la nostra `LinearProgressIndicator`
- **Línies 30-37**: ací definim el text associat al progrés de l'acció (1/10, 2/10...)

Tot el procés començarà quan premem el botó de `Load` en la pantalla principal. Afegirem un botó per a iniciar el funcionament de la barra de progrés.

main_activity.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/buttonProgress"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Load"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Haurem d'unflar el dialogue que conté la barra de progrés i mostrar-ho quan polsem el botó. Per a això es pot crear un mètode que s'encarregue d'este treball i que pot quedar de la manera següent:

```
private fun setDialog() {
        val builder = MaterialAlertDialogBuilder(this)
        val inflater = this@MainActivity.layoutInflater
        val v = inflater.inflate(R.layout.dialogo_progress, null)
        progressBar = v.findViewById(R.id.loader)
        loadingMessage = v.findViewById(R.id.loading_msg)
        progressMessage = v.findViewById(R.id.progres_msg)
        builder.setView(v)
        dialog = builder.create()
        dialog?.setCancelable(false)
        dialog?.show()
    }
```

📌 com es pot veure en este codi, tenim tres variables que seran unflades dins del mètode però són propietats de la classe, ja que hauran de ser accedides des de fora d'este per a modificar l'aspecte de la progressBar. El mateix ocorre amb l'objecte dialog.

El codi principal que permet crear la tasca en segon pla ho podem realitzar de la manera següent:

```
    fun barraProgreso()
    {
3       setDialog()
4       job = CoroutineScope(Dispatchers.Default).launch {
5           val inicio = progreso
6           for (i in inicio..100) {
7               Thread.sleep(100)
8               withContext(Dispatchers.Main) {
                        progressBar!!.progress = i
                        progressMessage.text = i.toString() + "/100"
                        progreso = i
12              }
            }
14          progreso = 0
            dialog?.dismiss()
16          dialog = null
        }
    }
```

📌 lanzamos una corrutina per mitjà de `launch` amb un abast definit en CoroutineScope fora del fil principal i per a treballs costosos en el temps `Dispachers.Default` **Línea 4**, (el treball costós se suposa que és el retard que produïm amb el bucle i el sleep **Líneas 6 i 7**) . Per a poder canviar les vistes relacionades amb el fil principal, llancem la funció de suspensió

> `withContext` amb `Dispachers.Main` **Líneas 8-12**.
> Les **líneas de 14 a 16** s'encarreguen de reiniciar els elements que s'han usat. El dialogue ho tanquem i iniciem la variable a null, per a evitar problemes si es produïx un gir. Usem la propietat progrés per a guardar el progrés de la barra i poder recuperar-ho posteriorment al reiniciar després d'un gir **Línea 5 i 14.**

Si volem controlar els girs de pantalla, i que la barra de progrés continue funcionant pel lloc en què s'havia quedat al produir-se el gir, haurem de guardar l'estat d'esta. Això ho fem amb la propietat de tipus sencer `progreso` que hem comentant anteriorment. Per a guardar i recuperar informació d'una activitat podem utilitzar els mètode `onSaveInstanceState` i `onRestoreInstanceState` estos mètodes seran cridats automàticament al destruir l'aplicació i al tornar-se a crear (sempre que no siga per un tancament realitzat de forma correcta per l'usuari).

```
override fun onSaveInstanceState(outState: Bundle)
{
    super.onSaveInstanceState(outState)
    outState.putInt("PROGRESO", progreso)
    if (dialog != null) {
            dialog?.dismiss()
            dialog = null
    }
}
override fun onRestoreInstanceState(savedInstanceState: Bundle)
{
    super.onRestoreInstanceState(savedInstanceState)
    progreso = savedInstanceState.getInt("PROGRESO")
}
```

> 📌 mètode `onSaveInstanceState` utilitzem el bundle que té com a paràmetre per a guardar el progrés i anul·lem el dialogue sempre que estiga creat. Mètode `onRestoreInstanceState` recuperem la informació del progrés que ens arribarà per mitjà del bundle de paràmetre. Estos mètodes guarden la informació en la memòria no volàtil per mitjà de serialización, per la qual cosa el procés és un poc més lent que si s'usara ViewModel i només s'ha d'utilitzar quan es vol guardar informació simple.

Per a acabar queda el codi de la **MainActivity**, on cridarem al mètode barraProgreso directament si hi ha una posició guardada per endavant (significa que s'ha recuperat l'estat després d'un reinicie), o caldrà polsar en el botó perquè es realitze la crida i se inicie la tasca.

```
class MainActivity : AppCompatActivity() {
    private lateinit var binding: ActivityMainBinding
    var dialog: AlertDialog?=null
    var progressBar: ProgressBar? = null
    var progreso = 0
    lateinit var loadingMessage: TextView
    lateinit var progressMessage: TextView


    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        if(savedInstanceState!=null) progreso=savedInstanceState.getInt("PROGRESO")
        binding = ActivityMainBinding.inflate(layoutInflater)
        val view = binding.root
        setContentView(view)
        if(progreso>0)  barraProgreso()
        binding.buttonProgress.setOnClickListener {barraProgreso()}
    }
    ...
}
```