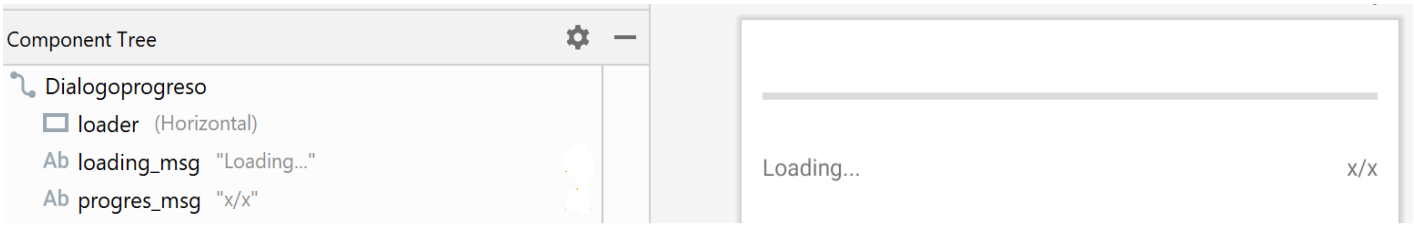
Exercici resolt Progress indicators

Descarregar aquests apunts

Desenvoluparem una aplicació que simule la càrrega d'un conjunt d'arxius, simularem la càrrega mitjançant `AsyncTask` 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. Tot el procés començarà quan premem el botó de `Load` en la pantalla principal.

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:

```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...)

Tasques asíncrones

Un fil és una unitat d'execució associada a una aplicació. És l'estructura de programació concurrent que té com a objectiu donar la percepció a l'usuari que el sistema executa múltiples tasques alhora.

Des de **Androd** podem executar tasques en segon pla utilitzant `thread` (fils), amb `AsyncTask` i amb corrutines `Kotlin`.

El problema amb els fils és que no podem accedir a cap element de la interfície gràfica des de dins d'aquest.

La finalitat de `AsyncTask` és unificar l'actualització de la vista associada a la tasca en segon pla amb l'execució d'aquesta. Està marcada com `deprecated` des de la versió 11 de **Android (API 30)**. A partir d'aquesta versió es recomana l'ús de corrutines.

`AsyncTask` és una interfície que ens permetrà crear un fil secundari en el qual fer tasques en background.

La definició genèrica d'una tarea asíncrona segueix el següent esquema:

```
class MiTarea:AsyncTask<T1,T2,T3>(){
    override fun onPreExecute(){
        //...
    }

    override fun doInBackground(vararg params:T1):T3{
        //...
        return T3
    }

    override fun onProgressUpdate(vararg params:T2){
        //...
    }

    override fun onPostExecute(result:T3){
        //...
    }

    override fun onCanceled(){
        //...
    }
}
```

El mètode encarregat d'executar el fil secundari és `doInBackground()`, la resta s'executen en el fil principal.

El mètode `onPreExcuted()` s'executa abans de començar la tasca en segon pla.

En la definició s'especifiquen tres tipus de dades:

- El primer és el que rep el mètode `doInBackGround()`. La notació `vararg params:T1` indica que es poden rebre un nombre indeterminat de paràmetres indeterminat del tipus `T1`.
- El segon és que el que permet comunicar l'avanç de la tasca. La comunicació amb el fil principal es realitzarà a través de la invocació del mètode `publicProgress()` que farà que s'execute el mètode `onProgressUpdate()` que s'encarregarà de l'actualització de la interfície.
- El tercer és el tipus de dades que es retornarà en finalitzar el fil secundari `doInBackground()` i que serà rebut en `onPostExecuted()`.

El mètode `onCanceled()` s'executa si la tasca és cancel·lada, i fa que no s'execute el `onPostExecuted()`.

En general es recomana treballar amb les corrutines enfront de `AsyncTask` i `Thread`, són més senzilles d'utilitzar i generen un codi més clar en interactuar amb la interfície.

És important, no obstant això, conéixer com funcionen, sobretot `AsyncTask`, ja que trobarem multitud d'exemples de codi implementats amb aquesta interfície.