RecyclerView

Crea la aplicación **App_UT2_04**

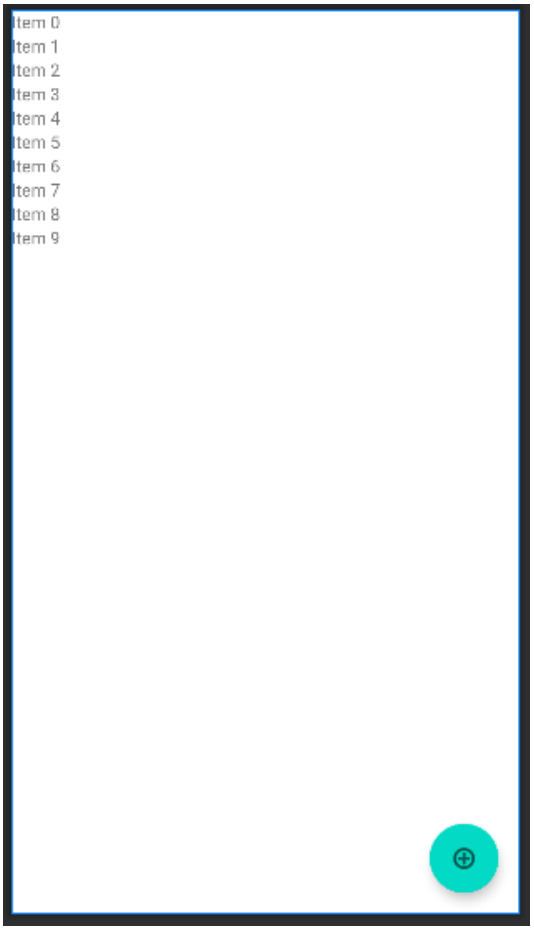
DISEÑO

En el activity_main.xml añade un **RecyclerView** que ocupe todo el layout.
Añade también un **FloatingActionButton** y posiciónalo abajo a la derecha.

Establece un margen de 16dp y como imagen del botón podemos añadir un icono ya definido (ic_manu_add)

```
android:layout_margin="16dp"
app:srcCompat="@android:drawable/ic_menu_add"
```

El diseño puede ser similar al siguiente:



El RecyclerView tendrá varios ítems. Cada ítem se representa en la pantalla con un layout específico para el ítem.
En nuestro caso el layout simplemente tendrá un TextView.

Añade en res/layout un layout con nombre **item_animal.xml** con un LinearLayout de orientación vertical y altura ajustada al contenido.
Dentro añade un TextView con tamaño de letra 22sp, negrita y cuyo id sea **textViewNombreAnimal.** Establece un margen de 8dp

DATOS

Crearemos un paquete llamado **datos** y en él crearemos un objeto Kotlin llamado DatosAnimales que contenga una función **getDatosAnimales** que devuelva un ArrayList<String>.
Simplemente tendrá que crearse una colección de nombres de animales

```
object DatosAnimales
{
   fun getDatosAnimales(): ArrayList<String>
   {
      return arrayListOf<String>(
         "Ballena",
         "Bisonte",
         "Camaleón",
         "Cebra",
         "Cocodrilo",
         "Elefante",
         "Hipopótamo",
         "Jirafa",
         "Mono",
         "Venado",
         "Zorro"
      )
   }
}
```

ADAPTADOR

En el paquete modelo crea una clase que se llame **AnimalAdapter** que herede de RecyclerView.Adapter.
Este adaptador tendrá todos los datos que se representan en el RecyclerView.

Crea también la clase interna que represente el ViewHolder. Esta clase se encargará de gestionar cada ítem o elemento del adaptador. Heredará de RecyclerView.ViewHolder y la llamaremos **AnimalViewHolder**

```
class AnimalAdapter: RecyclerView.Adapter<AnimalAdapter.AnimalViewHolder>()
{
   class AnimalViewHolder(view: View): RecyclerView.ViewHolder(view)
   {
```

```
    }

}
```

En la clase AnimalViewHolder crea un atributo que referencia a cada vista del layout de los ítems. En este caso sólo tendremos un TextView

```
class AnimalViewHolder(view: View): RecyclerView.ViewHolder(view)
{
    val textViewNombreAnimal = view.findViewById<TextView>(R.id.textViewNombreAnimal)
}
```

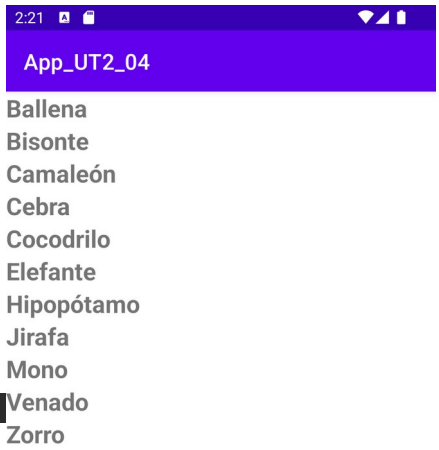
Dentro de la clase AnimalAdapter crea un atributo para contener la fuente de datos usada por el adaptador (la lista definida antes).

```
class AnimalAdapter: RecyclerView.Adapter<AnimalAdapter.AnimalViewHolder>()
{
    private val listaAnimales = DatosAnimales.getDatosAnimales()

    class AnimalViewHolder(view: View): RecyclerView.ViewHolder(view)

    //...
```

Por último, En la clase AnimalAdapter implementa los métodos **onCreateViewHolder**, **onBindViewHolder** y **getItemCount**.



No te olvides de **establecer el adapter** creado al RecyclerView en la clase MainActivity.
Si no has establecido la propiedad layoutManager en el diseño del RecyclerView ponséla por código.

```
recyclerView.layoutManager = LinearLayoutManager(this)
```

Podemos poner un separador en el recyclerView:

```
recyclerView.addItemDecoration(DividerItemDecoration(this,
DividerItemDecoration.VERTICAL))
```

Comprueba la funcionalidad. Debería mostrarte la lista.

BOTÓN FLOATINGACTIONBUTTON

Vamos a añadir funcionalidad al botón para que al ser pulsado añada un nuevo ítem al RecyclerView.
Los datos se añaden o modifican en el adaptador y éste se encarga de mostrarlos en el RecyclerView.
Crea un método **addAnimal** en el adaptador que reciba un String y lo añada a la colección.
Además, cada vez que se hace una actualización dinámica en el adaptador, se debe sincronizar el cambio:

```
fun addAnimal(animal: String)
{
   listaAnimales.add(animal)
   notifyDataSetChanged()
}
```

Desde el método onClick de respuesta del botón invocamos al método para que añada el animal.

```
val botonAnadir = findViewById<FloatingActionButton>(R.id.boton_añadir)
botonAnadir.setOnClickListener {
   (recyclerView.adapter as AnimalAdapter).addAnimal("Nuevo Animal")
}
```

Ahora bien, no siempre vamos a querer añadir el texto “Nuevo Animal”. Debemos dar la opción al usuario de introducir un nuevo animal.
Para ello usaremos un componente que veremos más adelante: los diálogos.
Crea el siguiente método y modifica la llamada desde el evento onClick:

```
private fun dialogoNuevoAnimal()
{
   val inputEditTextField = EditText(this)
   val dialog = MaterialAlertDialogBuilder(this)
      .setTitle("Nuevo animal")
      .setMessage("Introduce el nombre de un nuevo animal")
      .setView(inputEditTextField)
      .setPositiveButton("Añadir") { _, _ ->
          val nuevoAnimal = inputEditTextField .text.toString()
        (recyclerView.adapter as AnimalAdapter).addAnimal(nuevoAnimal)
      }
       .setNegativeButton("Cancelar", null)
      .create()
   dialog.show()
}
```

Para llamar al recyclerView desde el método vamos a establecerlo como atributo de la clase

```
class MainActivity : AppCompatActivity()
{
   private lateinit var recyclerView: RecyclerView
   override fun onCreate(savedInstanceState: Bundle?)
```