COMO CONECTAR LAS VISTA DEL LAYAUT AL ACTIVUTY

En el MainActivity

Lo primero, una vez en el proyecto, hay que ir al "build.gradle(app)" y añadir el siguiente código:

```
buildFeatures{
    viewBinding = true
}
```

Lo siguiente es ir a la MainActivity y crear la variable "laterini var", que se iniciará más tarde:

```
class MainActivity : NombreProyecto() {


// Que es el nombre de la Activity Main más el Binding
private lateinit var binding: ActivityMainBinding

override fun onCreate(…
```

ActivityMainBinding es una clase de nos lleva al activity_main.xlm (al layaut), Y con esa clase ya podemos cargarnos el:

```
setContentView(R.layout.activity_main)
```

que es la vista que infla el layout.

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    binding=ActivityMainBinding.inflate((layoutInflater)) //<= nuevo inflater
    setContentView(R.layout.activity_main) //<= este no lo cargamos
```

Volvemos a crear el setContentView que nos carguemos pero pasándole el binding:

```
setContentView(binding.root)
```

Ahora para acceder a cualquiera de las vistas del layout, hay que llamar primero al binding, por ejemplo, si en el layout tenemos el TextView con la id = btn1, y un texto, accederíamos al texto así:

```
binding.tv1.text
```

Otro ejemplo, para acceder al Toast con el onClick de un Button cOn id = bt1 sería así:

```


//...
        binding.btn1.setOnClickListener { toast() }
    }
    private fun toast() {
        Toast.makeText(this, "Botón pulsado", Toast.LENGTH_SHORT).show()
    }
```

En un Fragment

En el Fragment, la variable se crea con “barra baja” delante: “_binding” y decirle que es nulo con el “?”

Se crea también un valor binding get() = binding!!

En el onCreate, retorna el layout inflado y esto hay que cambiarlo por el _binding:

```
class MyFragment : Fragment() {
    // Ques es el mismo nombre que Fragmen My más el Binding
    private var _binding: FragmentMyBinding? = null   //<= variable nueva
    private val binding get() = _binding!!            //<= valor nuevo
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        //return inflater.inflate(R.layout.fragment_my, container, false)
        _binding = FragmentMyBinding.inflate(inflater, container, false) //<= inflater
        return binding.root //<= se retorna esto
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        // Para acceder al boton
        binding.btn2.setOnClickListener{Toast.makeText(
                activity,
                "click",
                Toast.LENGTH_SHORT
                ).show()}
    }
}
```

Y retornar el binding.root.

Y ahora para acceder a nuestras vistas, sería como en le MainActivity.

```
private var _binding: FragmentMyBinding? = null
private val binding get() = _binding!!
```

Con este código se crea una variable (var) _binding del tipo (:) FragmentMyBinding que puede ser nula(?), el ? hace referencia que no sabemos si va a tener contenido o va a ser nulo. Se le da un valor nulo (= null)

También se crea una variable binding (val) , que cuando la llamemos nos va a decir “get() => dame el _binding” y las (!!) quieren decir que estoy seguro de que no es null

Esto es porque cuando se crea el fragment en el ciclo de vida, hay que crear el _binding primero y para añadirle un valor luego hay que hacelo en View?{... “donde está en return” dentro del onCreateView

En un RecyclerView

En el Adapter (MyAdapter): “Este ejemplo trae el ViewHolder en la misma clase”

```
class MyAdapter(private val list: List<String>) :
                    RecyclerView.Adapter<MyAdapter.ViewHolder>() {

    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
    val item: String = list[position]
    holder.bind(item)
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {
    val layoutInflater = LayoutInflater.from(parent.context)

    return ViewHolder(layoutInflater.inflate(
                    R.layout.item_superhero_list,
                    parent,
                    false)
                    )
    }


    override fun getItemCount() = list.size


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

        //val tv1 = view.findViewById<TextView>(R.id.tv1)<= modo viejo
        private val binding = ItemMyLayoutBinding.bind(view)

        fun bind(myVariable: String) {
            binding.tv1.text = myVariable //<= cambia el texto en el TextView

        }
    }
}
```