CFGS DAM

IES MONTE NARANCO

UT-2

Programación de aplicaciones Android

Interfaz de Usuario – RecyclerView

RecyclerView - Eventos

- Para asociar un evento de click a cada elemento del RecyclerView haremos lo siguiente:
  - Crearemos en el **Adapter** una **variable** opcional (que pueda tener valor nulo) que almacenará una **función lambda** que será lo que queremos que se haga al hacer click sobre un elemento.
  - En el **ViewHolder** añadiremos el **listener** del evento **click** para que llame a la función lambda del atributo que hemos definido.
  - Desde **MainActivity** daremos valor al atributo de la **función lambda** pasándole la definición de función.

RecyclerView – Eventos – Atributo lambda

- Definimos el atributo en el Adapter

```
class PersonaAdapter(private val lista:List<Persona>): RecyclerView.Adapter<PersonaAdapter.PersonaViewHolder>() {

        var onItemClick: ((Persona) -> Unit)? = null
```

- El atributo **onItemClick** almacena una función lambda
- Recibe como parámetro un objeto **Persona**
- Devuelve **Unit** es decir no devuelve nada
- La **interrogación** indica que el atributo puede tomar valor **null**

RecyclerView – Eventos – listener

- Modificamos el ViewHolder para añadir el listener
- **Importante**: hay que añadir el módificador **inner** para que **PersonaViewHolder** sea una clase interna y pueda acceder a los atributos del adapter

```
inner class PersonaViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
    val nombre: TextView? = itemView.findViewById(R.id.tvNombre)
    val ciudad : TextView?= itemView.findViewById(R.id.tvCiudad)
    val estado : TextView? = itemView.findViewById(R.id.tvEstado)
    val icono : ImageView? = itemView.findViewById(R.id.ivIcono)
    val foto: ImageView? = itemView.findViewById(R.id.ivFoto)

    init {
        // El click del item completo
        itemView?.setOnClickListener {
            val pos = absoluteAdapterPosition
          if (pos != RecyclerView.NO_POSITION) {
              val item = lista[absoluteAdapterPosition]
               onItemClick?.invoke(item)
            }
        }
    }
}
```

RecyclerView – Eventos – listener

```
init {
        // El click del item completo
        itemView?.setOnClickListener {
            val pos = absoluteAdapterPosition
          if (pos != RecyclerView.NO_POSITION) {
              val item = lista[absoluteAdapterPosition]
               onItemClick?.invoke(item)
            }
        }
    }
```

- Le añadimos a **itemView** que es el **View** que contiene todo el elemento el **listener** del **evento Click**
- Con RecyclerView.NO_Position comprobamos si el RecyclerView está en un estado inconsistente.
- En la variable item almacenamos el objeto persona del dataset correspondientes a la posición en la que se ha hecho click (**absoluteAdapterPosition**)
- Llamamos a la función **lambda** con **invoke** y le pasamos el **objeto Persona** que obtuvimos antes.

RecyclerView – Eventos – MainActivity

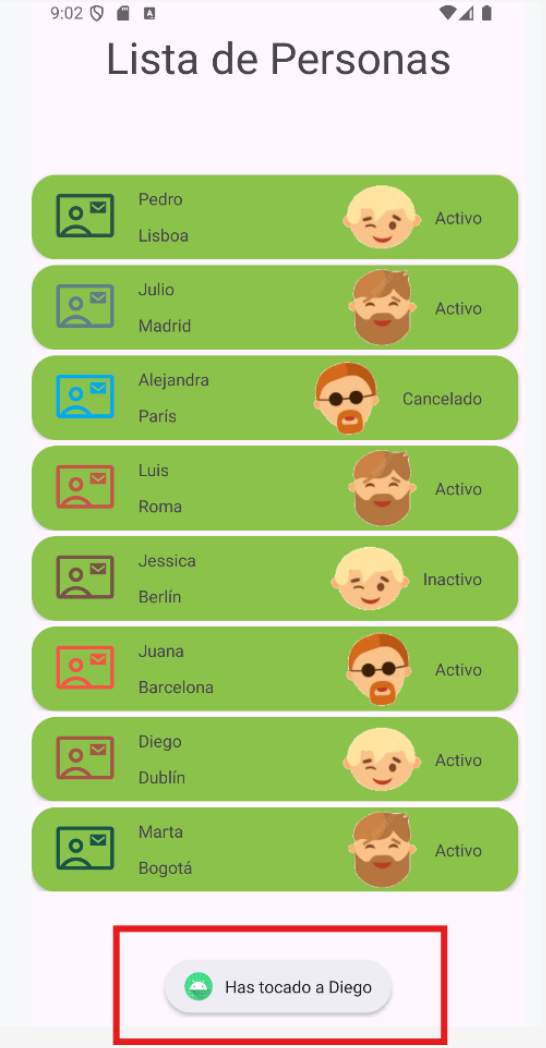
```
//Creamos el adapter
val adapter = PersonaAdapter(personas)


//Asignamos un adapter al RecyclerView
rv.adapter = adapter

//Le damos valor a la variable onItemClik que almacena la función que se ejecutará al hacer click en un elemento
adapter.onItemClick = { persona ->
    Toast.makeText(this, "Has tocado a ${persona.nombre}", Toast.LENGTH_SHORT).show()
}
```

- Asignamos al atributo **onItemClick** del **Adapter** una **función lambda** que recibe un objeto **Persona** y muestra un **toast** que muestra el nombre correspondiente al elemento pulsado.

RecyclerView – Eventos – Resultado



RecyclerView - Eventos

- Podemos asociar el evento a una view en concreto del ViewHolder en vez de a todo el elemento.

```
init {
    // El click del item completo
    foto?.setOnClickListener {
        val item = lista[absoluteAdapterPosition]
        onItemClick?.invoke(item)
    }
}
```

- Ahora para que se muestre el Toast Tendremos que hacer click en la imagen de la cara al asociar el listener a **foto** en vez de a **itemView**.

RecyclerView – Eventos - Ejercicio

- Edita el layout del elemento para añadir dos botones: uno que ponga Borrar y otro Editar:

- Añade a cada botón un evento de click que con un Toast indique para el botón de borrar que persona se va a borrar y para el botón editar que persona se va a editar

RecyclerView – Eliminar elemento

- Para borrar un elemento del ReciclerView tenemos que:
  - Borrar el elemento del dataset.
  - Notificar al Adapter que se ha modificado el Dataset.

```
lista.removeAt(position)

notifyItemRemoved(position)
```

RecyclerView – Eliminar elemento

- Vamos a crear en el Adapter un método que borre un elemento indicando su posición:

```
fun eliminarItem(position: Int) {
    lista.removeAt(position)
    notifyItemRemoved(position)
}
```

- Le pasamos al método un entero que indica la posición del elemento a borrar
- Borramos del Dataset el elemento indicado por la posición
- Notificamos al Adapter que se ha eliminado el elemento

RecyclerView – Eliminar elemento

- Modificamos el atributo que guarda la **función lambda** para el **botón de borrar** de modo que la función reciba como parámetro un **Int** que indique la **posición** en vez de un objeto **Persona**

```
var onBorrarClick: ((Int) -> Unit)? = null
```

RecyclerView – Eliminar elemento

- Modificamos la asignación del onclick en el ViewHolder

```
init {
    // El click del item completo
    btBorrar?.setOnClickListener {
        //val item = lista[absoluteAdapterPosition]
        val pos = absoluteAdapterPosition
        if (pos != RecyclerView.NO_POSITION) {
            onBorrarClick?.invoke(pos)
        }

    }
}
```

- Creamos un bloque **init** dentro de la clase **PersonaViewHolder** después del código que crea las variables con referencias a los elementos.

RecyclerView – Eliminar elemento

```
init {
    // El click del item completo
    btBorrar?.setOnClickListener {
        //val item = lista[absoluteAdapterPosition]
        val pos = absoluteAdapterPosition
        if (pos != RecyclerView.NO_POSITION) {
            onBorrarClick?.invoke(pos)
        }

    }
}
```

- En la variable **pos** obtenemos la posición del elemento pulsado
- El if comprueba que el valor de pos no sea RecyclerView.NO_POSITION que indica que el ReciclerView tiene un valor de posición incongruente. Si toma este valor no invocamos al método que hace el borrado.
- Por último llamamos a la función lambda que manejará el evento onclick.

RecyclerView – Eliminar elemento

- Por último desde **MainActivity** asignamos al **adapter** la función lambda que tendrá el código para **invocar el borrado** del elemento:

```
adapter.onBorrarClick = { position ->
    adapter.eliminarItem(position)
}
```

- La **función lambda** que asignamos **recibe** la **posición** y llama al método **eliminarItem** que creamos en el **adapter** para borrar un elemento.

RecyclerView – Añadir confirmación de borrado

- Ejercicio: Añade un AlertDialog en el botón de borrar que pida confirmación al usuario de que se quiere borrar el elemento.