Bootcamp - Android

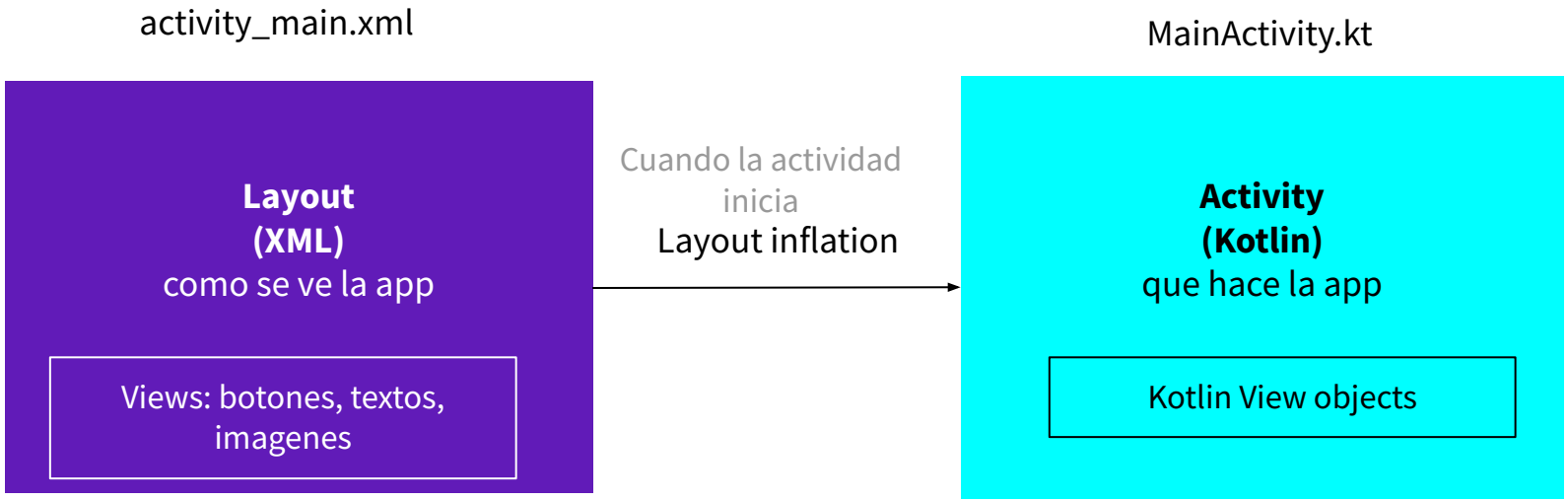
Roshka - Agosto 2022

Activity && Layout

En nuestro proyecto tenemos una **actividad**, que es una clase Android que es responsable de dibujar la interfaz de usuario y recibir los inputs del usuario.

Cuando la app se lanza, lo hace en una actividad específica. Las actividades están asociadas con un archivo layout. Los **layouts** son archivos XML que describen cómo se ve la aplicación.

Activity && Layout



Activity

En nuestro archivo ActivityMain, nuestra clase extiende de **AppCompatActivity**, que es una clase que hace que nuestra actividad sea compatible con la mayor cantidad de dispositivos.

Otra particularidad en Android, es que no contamos con el método main() de otros programas, sino que definimos en el **AndroidManifest** el punto inicial de nuestra app.

Para realizar la conexión del layout con la actividad, sobreescribimos el método *onCreate* y llamamos a la función **setContentView** que es la encargada de “inflar” nuestra actividad.

Layouts

Los layouts definen los elementos de nuestra app, su posición y sus propiedades.

En Android, estos elementos se llaman **Views**, algunos de estos son: TextView, Button, ImageView, CheckBox, RadioButton, etc.

Estas se definen en nuestros layouts (XML) con tags y sus propiedades van como propiedades de estas.

Layouts

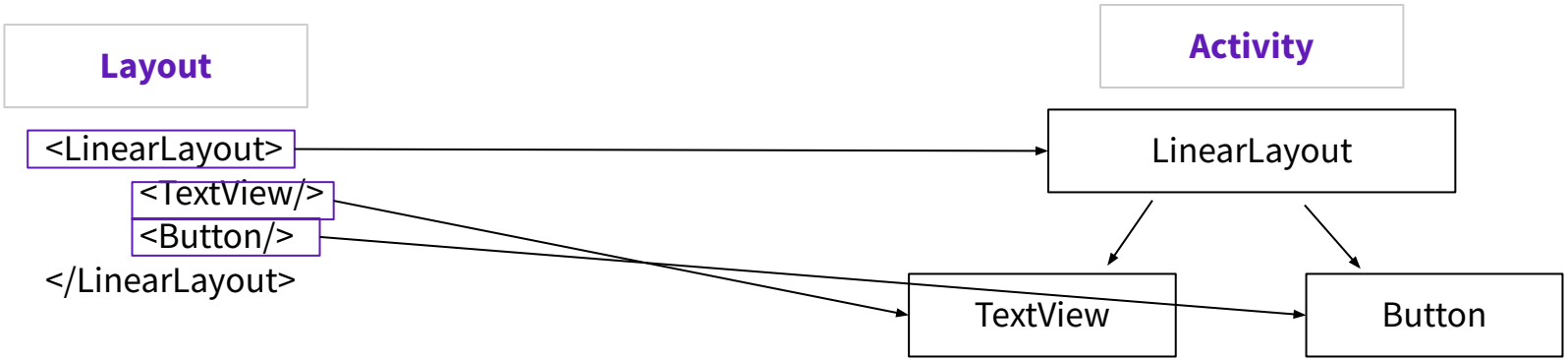
Algunos elementos de nuestro actual layout:

- LinearLayout, se trata de un View que puede contener múltiples views
- Height y width(largo y ancho), en el código vemos que usamos 2:
  - match_parent: tomará el tamaño del elemento padre
  - wrap_content: tomará el tamaño del contenido
- TextSize en general se utiliza la unidad de medida **sp** (scale independent pixel), que se utiliza para el tamaño de texto independientemente a la calidad de display del dispositivo (para que el tamaño sea parecido en múltiples pantallas)

Interactuando con las vistas

Como ya sabemos nuestro código Kotlin es el encargado de hacer interactiva nuestra aplicación.

Como en las páginas webs existe una jerarquía en forma de árbol de los elementos definidos en el HTML, aquí ocurre algo similar con el XML.



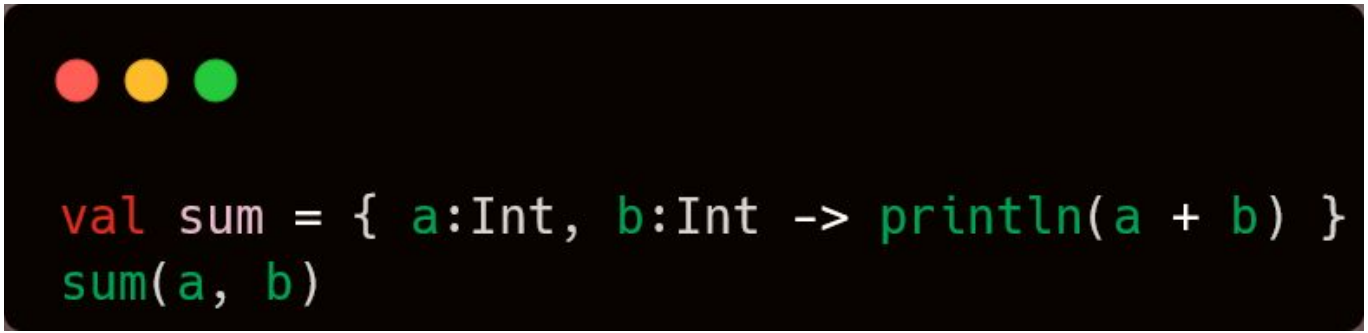
Interactuando con las vistas

Las aplicaciones normales, no tienen un árbol de jerarquía tan sencillo como el de la diapositiva anterior, por eso accedemos a nuestras Views a través de un **ID** que asignamos en el **layout** y lo llamamos en nuestro código con el método **findViewById**.

Una vez que logramos esto podemos modificar prácticamente cualquier atributo de nuestro View y utilizar los métodos de nuestros objetos.

Funciones Lambda

- Las expresiones Lambda son funciones que no tienen nombre
- Las expresiones Lambda y las funciones anónimas son “funciones literales”, esto quiere decir que son funciones que no están declaradas, pero se pasan directamente como una expresión
- Lambda está definida con llaves {} que toman variables como parámetros (si los tiene) y el cuerpo de la función



Resources (res)

Como vimos antes en esta carpeta se encuentran distintos tipos de recursos que utilizamos en nuestra aplicación.

En este caso utilizaremos imágenes de la carpeta **drawable** por lo que debemos mover los archivos descargados en esta carpeta.

Tarea

Lo que debemos lograr ahora es generar una app, que genere números al azar y coloque la imagen del dado correspondiente, luego podemos hacer lo mismo para 5 dados y lanzar en un Toast la jugada de nuestro juego Generala correspondiente.

Obs.: Las imágenes de los dados se encuentran en el drive.