Desarrollo Avanzado de Software - Curso 2023 / 2024

Laboratorio 1
Primera aplicación en Android

Contenido:

Objetivos: Trabajar con los elementos gráficos de Android desde el código para familiarizarnos con ellos y la forma de programar.

Desarrollo Avanzado de Software - Curso 2023 / 2024

1 Familiarizándonos con la interfaz de Android

En Android se define el **código** por un lado y la **interfaz gráfica** por otro (Modelo – Vista – Controlador). La unión entre la vista y el controlador se hace mediante la operación `setContentView()`

Sin embargo, por ahora, vamos a ignorar esa posibilidad y a mezclar ambos conceptos (Vista y Controlador).

- Cread una aplicación nueva y en la actividad principal comentad la línea de código correspondiente a la instrucción `setContentView(R.layout.activity_main)`

Vamos a construir una interfaz básica sencilla a través del código. Para ello, todos los elementos tienen que descender de la clase *View* y todos ellos necesitan en el constructor como parámetro el *contexto* de ejecución. Para ello, usaremos el contexto de la propia actividad mediante el uso de la variable *this*.

Para colocar un texto en nuestra aplicación, usaremos la clase *TextView*, que se encuentra en el paquete *android.widget*

- Cread una etiqueta de texto mediante la clase *TextView*. Para asignar el texto usad la operación `setText (CharSequence)`.

Para conseguir que la etiqueta se muestre por pantalla al ejecutar la aplicación, tendremos que pasársela como parámetro a la operación `setContentView`.

```
TextView etiqueta = new TextView(this);
etiqueta.setText("El texto que queramos mostrar…");
setContentView(etiqueta);
```

Con los pasos anteriores hemos definido la etiqueta *TextView* como la vista principal de nuestra aplicación, pero lo normal será que queramos que la vista de nuestra aplicación tenga más de un elemento. En estas ocasiones lo que definiremos será un elemento de tipo *layout* (que también desciende de la clase *View*) y será ese layout el que pasaremos como parámetro a la operación `setContentView`. Un layout es el equivalente a un contenedor de elementos.

Existen varios tipos de layout (ver apuntes del tema 02), cuya diferencia es la forma de colocar los elementos que se añadan al mismo. Por ejemplo, el *LinearLayout*, añade los elementos de manera lineal uno detrás de otro. Por defecto lo hace de manera horizontal. Se puede elegir la orientación con la operación `setOrientation()`, que acepta como parámetro `LinearLayout.VERTICAL` y `LinearLayout.HORIZONTAL`.

Para añadir elementos a un *layout* se usa la operación `addView` que recibe como parámetro un elemento de alguna clase que descienda de *View*. Para eliminar elementos a un *layout* se usa la operación `removeView` indicándole el elemento de tipo *View* que se desea eliminar de la interfaz.

```
LinearLayout ellayout = new LinearLayout(this);
ellayout.setOrientation(LinearLayout.HORIZONTAL);
ellayout.addView(etiqueta);
ellayout.removeView(etiqueta);
```

- Cread varias etiquetas de texto y mostradlas todas usando un *LinearLayout*. Probad con otros tipos de *layout*.

Desarrollo Avanzado de Software - Curso 2023 / 2024

A continuación, vamos a añadir otro tipo de elemento a nuestra interfaz, un botón. Los botones pertenecen a la clase *Button*, y también dispone de la operación `setText` para indicar el texto que tiene que mostrar el botón.

Para poder capturar el evento que se produce cuando se pulsa el botón debemos añadir un *listener*. Para ello, es necesario que nuestra clase además de heredar de la clase *Activity* (o alguna derivada como *AppCompatActivity*) *implemente* el método `OnClickListener` de la clase *View*.

```
public class MainActivity extends AppCompatActivity implements
View.OnClickListener {
```

Al botón habrá que decirle que el *listener* que va a recoger las interacciones está en la propia clase

```
elboton.setOnClickListener(this);
```

Además, en la clase habrá que implementar el método `onClick` que se ejecutará cada vez que se pulse el botón.

```
public void onClick(View v) {
```

Si tenemos más de un botón en la misma actividad, es mejor que cada uno defina su propio método `onClick`. Para ello definimos el método `onClick` dentro del propio método `setOnClickListener`.

```
elboton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

    }
});
```

Si no lo hacemos así, el método `onClick` sería el mismo para todos y deberíamos distinguir dentro qué botón lo ha ejecutado para saber qué hacer.

- Añadid un botón y una etiqueta con un contador que se vaya incrementando cada vez que se pulse el botón.
- Haced que cada vez que pulséis el botón cambie el color de fondo del *layout* usando la operación `setBackgroundColor`. Si importáis la clase *android.graphics.Color* podréis usar como valores *Color.YELLOW*, *Color.BLUE*, etc.
- Cread una aplicación con un *layout* con varios elementos y botones que al pulsarlos muestren u oculten parte de los elementos que tengáis en el *layout*
- Cread una aplicación que tenga dos *layouts* distintos que se puedan intercambiar pulsando un botón.

La clase *EditText* permite añadir campos de texto para que el usuario introduzca valores. Mediante las operaciones `getText` y `toString` se obtiene el texto que el usuario ha introducido.

```
String texto= lacaja.getText().toString();
```

- Añadid en el primer *layout* un campo *EditText* para que el usuario escriba su nombre y que al pulsar un botón se muestre un segundo *layout* donde le aparezca un mensaje personalizado con su nombre.