

Usando Cuadros de Diálogo

En esta práctica vamos a tener otra vez una única ACTIVITY, pero vamos a usar un tipo de objeto nuevo como son los CUADROS de DIÁLOGO, que es una forma de comenzar a utilizar los FRAGMENTS.

Además como se trata de que el usuario elija la gama de colores con la que funciona la app, para ello vamos a profundizar en el uso de los estilos y como cambiar el tema de nuestra app de forma dinámica.

También vamos a aprovechar para utilizar los ficheros de PREFERENCES, así cuando volvamos a usar la app, entrara con la gama de colores que se escogió la última vez, por tanto tendremos que guardar las elecciones del usuario en algún sitio no??.

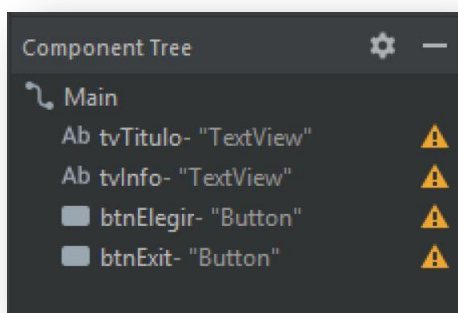
En resumen, nuestra interfaz va a ser muy simple pero vamos a aprovechar para profundizar en lo que nos permiten hacer los ficheros drawables, los estilos, colores y fuentes para personalizar y dinamizar nuestras apps.

FASE 1

Lo primero como siempre ver como organizamos la interfaz:

- TEXTVIEW → para el título.
- TEXTVIEW→ para darle información al usuario de lo que tiene que hacer
- BUTTON → llamaremos a la lista de gamas para poder elegir
- BUTTON → Va a ser el botón de salir de la APP

Cuando tenemos los elementos en el layout más o menos distribuidos, les vamos asignando los identificadores que se ven a la abajo.





FASE 2

Ahora vamos a dar algo de contenido y forma a nuestra app. Daremos algo de efecto a la pulsación de los botones y utilizaremos una fuente propia.

Además vamos a hacer que aparezca un icono (el mismo que vamos a usar como icono de la app y un título en la ActionBar.

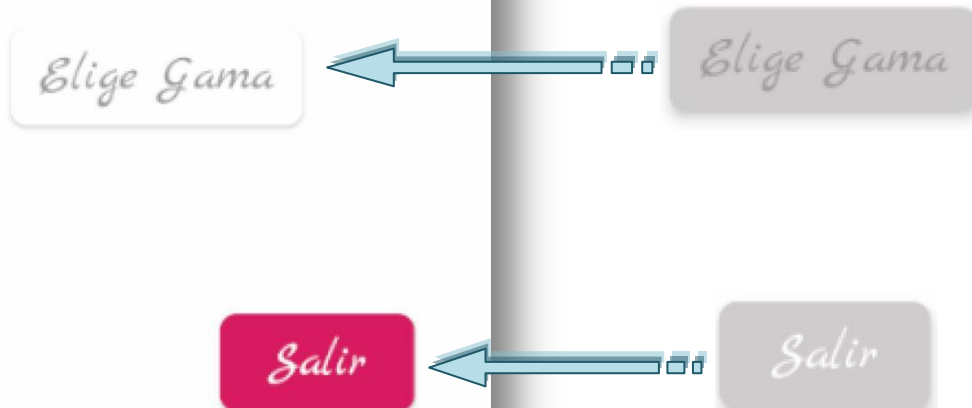


EN EL ACTIONBAR HEMOS PUESTO
NUESTRO ICONO Y UN TÍTULO



ESTO LO TENEMOS QUE HACER
MEDIANTE CÓDIGO JAVA

AL PULSAR LOS BOTONES TIENEN UN
ASPECTO DISTINTO AL NORMAL



USAMOS LOS FICHEROS XML EN
DRAWABLE.

Estos 3 detalles los dejamos para el final, para la Fase 4.

Usando Cuadros de Diálogo

FASE 3

Bueno pues vamos a por el código que hay faena:

En nuestra clase principal, MainActivity, empezamos con:

```
public class MainActivity extends AppCompatActivity {

    private int gama = 10;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main_lay);
        // Queremos que en la ActionBar se vean un icono y un titulo
        // para ello usamos el siguiente método
        ABconIconoTitulo();
    } //fin onCreate
```

Tenemos una única variable de la clase que es el número entero que contendrá la gama que vamos a aplicar, la inicializamos a 10 ya que solo vamos a contar con 5 gamas. En el onCreate solamente tenemos el método que nos va a permitir mostrar nuestro Icono y Titulo en la ActionBar, su contenido lo veremos en la Fase 4:

En este caso tenemos que controlar solamente dos botones vamos a hacerlo como en la práctica anterior, es decir, vamos a usar la propiedad onClick de los botones, donde les indicamos que nuestro método PULSACIONES se va a ocupar de ellos cuando el usuario los pulse:

- Boton Elegir – Tenemos que invocar al cuadro de dialogo con las gamas, para que el usuario seleccione.
- Boton Salir – acabamos la activity con un mensaje de despedida (damos un tiempo para que se vea antes de cerrar.

```
public void Pulsaciones(View boton)
{
    if(boton.getId()==R.id.btnElegir)
    {
    }
    else
    {
    }
} //fin Pulsaciones
```

Truco: si Pulsaciones aparece en gris, es que no lo usáis en ningún botón

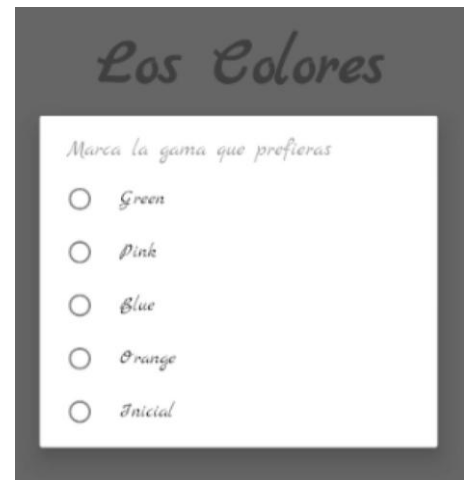


GAMAS

Antes de implementar el código para los botones como sabemos que desde el botón Elegir Gama necesitamos llamar a un cuadro de diálogo vamos a ver como implementarlo.

El Cuadro de Diálogo (DialogFragment) que vamos a usar va a mostrarle al usuario las gamas disponibles como un radioGroup y el usuario sólo podrá elegir una.

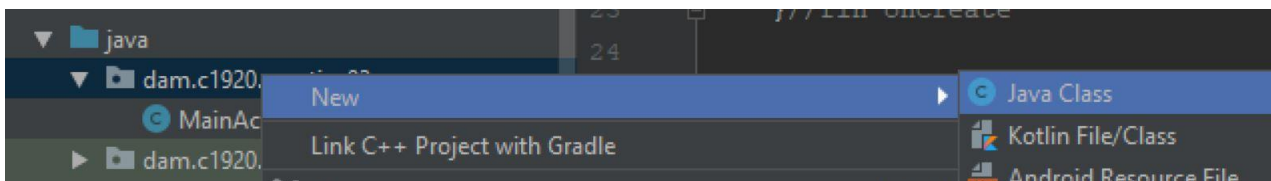
En ejecución se verá más o menos así →



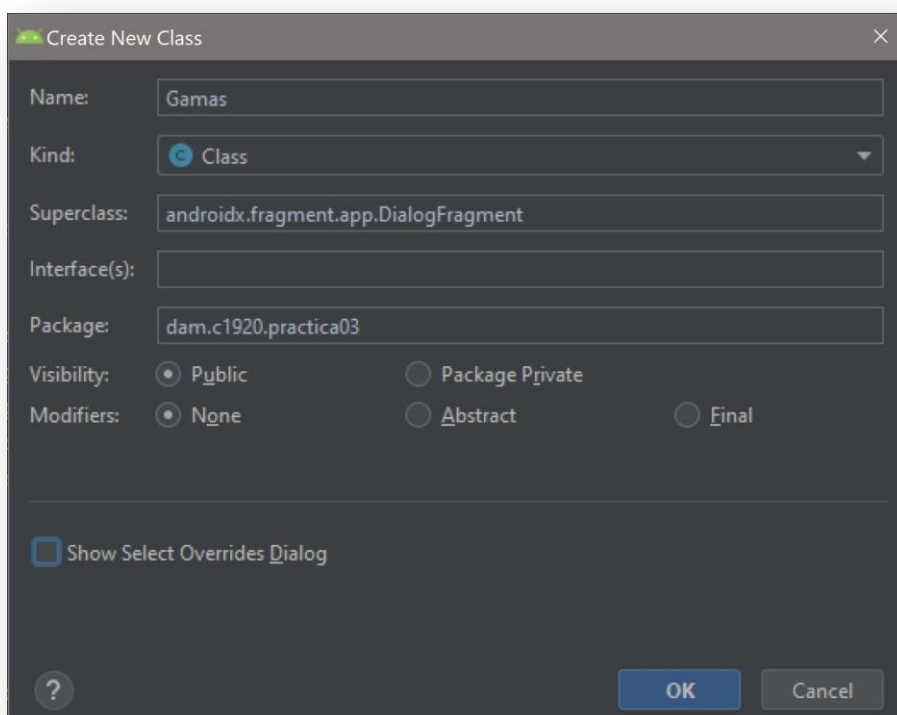
Ahora que ya sabemos lo que queremos para nuestra app vamos a implementarlo.

Lo primero que hacemos es crear una clase de java que llamaremos Gamas y que heredará de la clase android DialogFragment.

Nos situamos con el ratón sobre el nombre de nuestro proyecto java, desplegamos el menú contextual y elegimos New / Java Class, tal y como se ve abajo:



A continuación nos aparece la siguiente ventana:



Le damos nombre, y decimos que es una clase de java.

La Superclass es la clase de la que heredamos en nuestro caso:

DIALOGFRAGMENT

Comprobamos que está en el Package que corresponde, en este caso el único que tenemos en el proyecto

PULSAMOS OK

Usando Cuadros de Diálogo

Ya tenemos creada nuestra nueva clase, que aparece bastante vacía por cierto, así que le añadiremos los métodos que necesitamos para trabajar:

```
public class Gamas extends DialogFragment {

    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        return super.onCreateDialog(savedInstanceState);
    }

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
    }

    public interface Responder
    {
        void onRespuesta(int opcion);
    }

} //FIN Gamas
```

Sobreescribimos los métodos:

- onCreateDialog (es con el que creamos e inflamamos el cuadro de diálogo)
- onAttach (lo usamos cuando nos anclamos a una Activity),

Por último nos creamos una INTERFAZ que nos permitirá comunicarnos con la clase que active nuestro cuadro de diálogo y decirle el resultado, es decir, la opción elegida.

Además vamos a necesitar definir dos variables de la clase:

```
public class Gamas extends DialogFragment {

    private Context entorno;
    Responder eleccion;
```

- **Context entorno** – necesitamos saber en qué contexto vamos a desplegar el diálogo, y lo obtenemos a través del método onAttach.
- **Responder eleccion**– creamos una variable del tipo de la interfaz que hemos creado, la necesitamos para poder pasar el valor de la opción.(pub o priv?)



Vamos a implementar los métodos.

```
public Dialog onCreateDialog(Bundle savedInstanceState)
```

Necesitamos hacer varias cosas:

- 1º.- Debemos tener un Array de Strings con los nombres de las gamas que queremos mostrar. Como tenemos uno definido en arrays.xml lo cogemos igual que hicimos en la práctica anterior con las listas de libros, música y películas.
- 2º.- La base de nuestro cuadro es un objeto que llamamos AlertDialog, pues utilizaremos una variable del tipo AlertDialog.Builder le reservaremos memoria y la asociaremos al contexto del Activity que nos ha invocado.
- 3º.- Con esta variable ya podemos elegir como va a ser nuestro cuadro de diálogo:
 - Le damos un Título.
 - Le decimos que queremos que sea del tipo de una única opción y le damos los siguientes parámetros:
 - o La lista de las gamas
 - o Cual va a ser el ítem marcado por defecto o -1 si no hay ninguno.
 - o El Listener que va a reaccionar con el onClick y que va a enviar la información a través de la interface, para después destruir el cuadro de diálogo.
 - Devolvemos el resultado de crear este elemento.

Ahora veamos cómo queda el código que nos permite hacer todo esto:

```
@Override
public Dialog onCreateDialog(Bundle savedInstanceState) {
    final String[] colores = getResources().getStringArray(R.array.gamas);
    AlertDialog.Builder elCuadro = new AlertDialog.Builder(entorno);
    elCuadro.setTitle(getResources().getString(R.string.dfGama));
    elCuadro.setSingleChoiceItems(colores, checkedItem: -1,
        new DialogInterface.OnClickListener()
        {
            @Override
            public void onClick(DialogInterface dialog, int opc) {
                eleccion.onRespuesta(opc);
                dialog.dismiss();
            }
        });
    return elCuadro.create();
} //fin onCreateDialog
```




Usando Cuadros de Diálogo

```
public void onAttach(Context context)
```

Este método entra en acción cuando se invoca a nuestro cuadro de diálogo desde una Activity y básicamente lo que hace es pasarnos el contexto de la misma para que podamos crear el Fragment, ya que estos no tienen contexto, y para poder hacerse visibles necesitan estar "conectados" o "anclados" a una Activity.

Así que necesitamos hacer dos cosas:

- 1º.- Tenemos que guardar el context que nos llega en nuestra variable "entorno", para que esté disponible cuando se ejecuta el onCreateDialog.
- 2º.- Tenemos que asociar a nuestra variable de tipo interfaz el contexto, pero tendremos que castear el context para poder asignarlo a una interfaz.

El código queda así:

```
@Override
public void onAttach(Context context) {
    super.onAttach(context);
    entorno = context;
    eleccion = (Responder) context;
}
```

```
public interface Responder
```

Ya está hecha, no necesitamos hacer nada más.

MAINACTIVITY

Pues en la clase Gamas ya está todo preparado para que interactuemos con ella desde nuestra clase principal MainActivity.

Nos habíamos quedado en el método "Pulsaciones", en concreto íbamos a implementar el código necesario para invocar al Cuadro de diálogo, al pulsar el botón Elegir Gama. Esto es lo que tenemos que hacer:

- 1º.- Crear una variable del tipo "Gamas", es decir, crear un cuadro de diálogo y construirlo, dándole su espacio para existir.
- 2º.- Hay que hacerlo visible en pantalla para que el usuario pueda seleccionar la opción.

```
Gamas pregunta = new Gamas(); //creamos el cuadro de diálogo
pregunta.show(getSupportFragmentManager(), tag: "dialogo");
```



YA ESTÁ, PERO



¿¿ALGUIEN HA VISTO ALGO SOBRE LA OPCIÓN
QUE HA SELECCIONADO EL USUARIO??

Bueno pues para eso hemos creado una Interfaz, lo que pasa es que en MainActivity no la hemos cogido todavía, tenemos que implementarla y utilizar el método que tiene para poder acceder a la información que nos da.

Por tanto tenemos que cambiar la cabecera de MainActivity:

```
public class MainActivity extends AppCompatActivity  
    implements Gamas.Responder{
```

Y ahora sobrescribimos el método onRespuesta que lleva la interfaz, para que según la opción que nos llega cambiemos el tema de la App con uno de los estilos (styles) que tenemos definidos:

```
@Override  
public void onRespuesta(int opcion) {  
    gama = opcion; //variable de la clase  
    switch(gama)  
    {  
        case 0: this.setTheme(R.style.miGreen);    break;  
        case 1: this.setTheme(R.style.miPink);    break;  
        case 2: this.setTheme(R.style.miBlue);    break;  
        case 3: this.setTheme(R.style.miOrange);  break;  
        default: this.setTheme(R.style.AppTheme); break;  
    }  
    this.setContentView(R.layout.main_lay);  
} // fin del método onRespuesta
```

Mediante **this.setTheme(id style)** establecemos el nuevo tema de la app, y mediante **this.setContentView** hacemos lo mismo que en onCreate, que se establezca el layout así conseguimos que se refresque con el nuevo tema.

Siguiendo con el método Pulsaciones hemos visto que hacer si el usuario pulsa el botón Elegir Gama, pero qué ocurre si pulsa el botón Salir, esto es lo que vamos a hacer,

Usando Cuadros de Diálogo




en el mismo TextView que mostramos la información de funcionamiento vamos a mostrar un mensaje de despedida, y claro para que el usuario se dé cuenta del detalle, lo cambiaremos de color, queremos que se escriba con el color Accent que es el mismo que tiene el botón Salir en cada una de las gamas, así que según en qué gama se vaya el usuario el mensaje saldrá en un color distinto.

El código sería este:

```
else
{
    TextView despedida = findViewById(R.id.tvInfo);
    despedida.setTextColor(getColor(R.color.despedida));
    despedida.setText(getString(R.string.tvDespedida));
    Handler control = new Handler();
    Runnable va = new Runnable() {
        @Override
        public void run() {
            finish();
        }
    };
    control.postDelayed(va, delayMillis: 1000);
} //fin de Salir
```

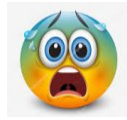
Claro que **R.color.despedida** no es un color cualquiera ...

Vamos a ver cómo sería más o menos una ejecución:

PANTALLA INICIAL	GAMA AZUL	GAMA ROSA	SALIR EN GAMA VERDE
			



PROBLEMA



SI CERRAMOS LA APP ADIÓS ELECCIÓN DEL USUARIO

Necesitamos poder guardar el número de la gama que ha elegido el usuario antes de que la APP se cierre, y poder recuperarlo antes de que la app se abra aparezca con la elección del usuario la última vez. Para hacerlo tenemos que conocer cómo funciona el CICLO DE VIDA DE UNA ACTIVITY, y en que estados es conveniente guardar y recuperar la información:

- **onPause** – es el estado al que pasa la Activity cuando deja de estar visible y todavía no está destruida, así que es el mejor momento para guardar la información que queremos que se conserve aunque la app se cierre
- **onResume** – es el estado en el que entramos justo antes de que se cargue la interfaz y la app sea visible para el usuario, así que es el momento ideal para recuperar la información que necesitamos.

Para guardar esta información vamos a usar un tipo de ficheros llamados SharedPreferences, solamente podemos almacenar datos de tipos básicos (entero, booleano, char, etc) son muy útiles y se guardan en el espacio de ejecución de la app.

```
@Override
protected void onPause()
{
    super.onPause();
    SharedPreferences seguridad = PreferenceManager.getDefaultSharedPreferences(context: this);
    SharedPreferences.Editor bloc = seguridad.edit();
    bloc.putInt(s: "gama", gama);
    bloc.apply();
}

@Override
protected void onResume()
{
    super.onResume();
    // antes de que se haga visible la activity recuperamos el color que ha elegido el usuario
    // si es la primera vez se reactivará el layout inicial
    SharedPreferences info = PreferenceManager.getDefaultSharedPreferences(context: this);
    gama = info.getInt(s: "gama", i: 10);
    switch (gama)
    {
        case 0: this.setTheme(R.style.miGreen); break;
        case 1: this.setTheme(R.style.miPink); break;
        case 2: this.setTheme(R.style.miBlue); break;
        case 3: this.setTheme(R.style.miOrange); break;
        default: this.setTheme(R.style.AppTheme); break;
    }
    this setContentView(R.layout.main_lay);
}
```

Usando Cuadros de Diálogo

FASE 4

Vamos a ultimar los detalles que nos faltan

- 1º - ¿Cómo conseguimos que aparezca el icono y el título en la ActionBar?
- 2º - ¿Cómo hacer que el mensaje de despedida que aparece cuando pulsamos el botón de salir aparezca con el color Accent de la gama en activo?
- 3º - ¿Cómo conseguir que los botones tengan el aspecto apagado cuando los pulsa el usuario?, y sobre todo ¿cómo conseguir que cambien el color del botón y del texto cuando cambiamos la gama?

1º

Desde el principio de la práctica, en el onCreate tenemos el método que nos va a permitir mostrar nuestro Icono y Titulo en la ActionBar, pues bien el contenido de este método son estas líneas, ya podeis implementarlo y ver el resultado:

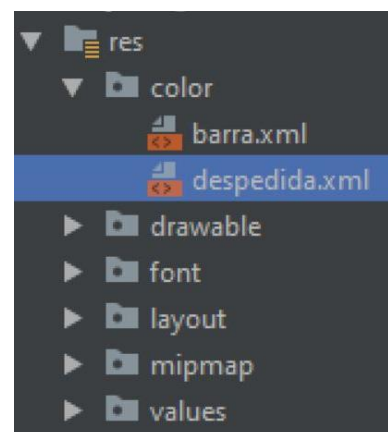
```
getSupportActionBar().setIcon(R.drawable.ic_colores); // OJO drawable no mipmap
getSupportActionBar().setTitle(" Las Gamas");
getSupportActionBar().setDisplayShowTitleEnabled(true);
getSupportActionBar().setDisplayHomeAsUpEnabled(true);
```

2º

Ya sabemos que dentro de res podemos crear carpetas para determinados tipos de información y ficheros xml.

Creamos una carpeta color, y dentro el fichero despedida.xml.

El código está en la imagen inferior y como podéis observar es del mismo tipo que los de fondos, lo que hacemos es definir un color como variable.



```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:color="?android:attr/colorAccent"/>
</selector>
```



3º

Para los fondos de los botones, los ficheros drawable nos permiten establecer distintos aspectos según el estado en el que se encuentre el botón, veamos Elegir gama:

```
<item android:state_pressed="true">
    <shape android:shape="rectangle">
        <corners android:radius="@dimen/radio2"/>
        <stroke android:color="?android:attr/colorPrimaryDark" />
        <solid android:color="@color/gris"/>
        <size android:width="@dimen/lado3"
            android:height="@dimen/lado1"/>
    </shape>
</item>
<item>
    <shape android:shape="rectangle">
        <corners android:radius="@dimen/radio2"/>
        <stroke android:color="?android:attr/colorPrimaryDark" />
        <solid android:color="?android:attr/windowBackground" />
        <size android:width="@dimen/lado3"
            android:height="@dimen/lado1"/>
    </shape>
</item>
```

Como podemos ver en el primer Item describimos el aspecto del botón cuando el usuario lo presiona, en el segundo Item no establecemos ningún estado, por tanto es el aspecto del botón en reposo lo que se describe.

Observamos que en varias líneas al indicar el color usamos como un comodín:

"?android:attr/nombre de un color"

Lo hacemos así porque nuestra idea es ir cambiando las gamas de la app y necesitamos que estos elementos cojan en cada momento el color que tiene ese elemento y no un color estático.