Apuntes

Resumen B4 Material Design. Patrones de diseño básicos.

Descargar estos apuntes

Índice

Patrones MD

Gestión de colores que ofrece Material Design

Descargar la configuración de colores elegida.

Layouts

LinearLayout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="48dp">

</LinearLayout>
```

📌 Aclaraciones:

- Para crear un diseño lineal en el que cada campo secundario use la misma cantidad de espacio en la pantalla, define el **`android:layout_height`** de cada vista en **`0dp`** (para un diseño vertical) o el **`android:layout_width`** de cada vista en **`0dp`** (para un diseño horizontal). Luego, fija el **`android:layout_weight`** de cada vista en 1. Si das valores distintos a 1 el tamaño de layout se distribuirá siguiendo esta distribución.

- **`android:layout_gravity`**
- **`android:hint`**
- **`android:ems`** determinar el tamaño de un carácter
- **`android:inputType`** determinar el tipo de caracteres y tipo de teclado virtual
- **`android:gravity`** tipo de alineamiento

CardView

CardView Material Design. Elemento visual en forma de tarjetas de información.

Está incluido en la librería **material**, por lo que hay que agregar esta dependencia a nuestro proyecto:

```
dependencies {
     implementation 'com.google.android.material:material:1.2.1'
}
```

El archivo XML correspondiente al diseño sería:

```
<?xml version="1.0" encoding="utf-8"?>
<com.google.android.material.card.MaterialCardView
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    card_view:cardCornerRadius="8dp"
    card_view:cardElevation="12dp">

</com.google.android.material.card.MaterialCardView>
```

Constraint Layout

```
<?xml version="1.0" encoding="utf-8"? >
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ImageView
        android:id="@+id/imagen"
        android:layout_width="match_parent"
        android:layout_height="720px"
        android:layout_marginTop="0dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/sanfrancisco" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

> 📌 Aclaraciones:
> Necesitamos poner al menos dos anclajes: **`constraintStart_toStartOf`** , **`constraintTop_toTopOf:`** , etc.
> **`app:srcCompat:`** cargar la imagen en la view

> **RatingBar** Con la propiedad `android:numStars:` elegimos el número de estrellas que deseamos que aparezcan. Con la propiedad `android:stepSize: 0.5` nos permitiría seleccionar media estrella(probar vosotros)

CoordinatorLayout

`CoordinatorLayout` básicamente nos permitía gestionar interacciones entre sus elementos hijos.

```
<androidx.coordinatorlayout.widget.CoordinatorLayout
     xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
```

FrameLayout

Nos permite solapar cosas unas encima de otras, por ejemplo una imagen de un carriro, con un texto contando productos.

Temas y estilos

La forma más sencilla de aplicar un tema es editar el fichero `AndroidManifest.xml` y buscar la propiedad `android:theme` y asignar un estilo concreto. Si esta propiedad se aplica a la etiqueta `application` la aplicación entera adopta el tema seleccionado. Si se aplica a la etiqueta `activity` únicamente esa actividad aplicará el tema.

Temas y estilos.

Hay que ir ahora al archivo `AndroidManifest.xml` a modificarlo.

Aunque se definen ambos con la etiqueta `styles` hay que tener en cuenta que son dos cosas distintas. Los temas se aplican a una jerarquía de vistas, mientras que los estilos se definen sobre una vista concreta.

Podríamos cambiar la apariencia de nuestros `EditText` añadiendo un estilo creado por nosotros. Para ello creamos un archivo llamado `styles` en la carpeta `values`:

```
<resources>
    <style name="textViewStyle">
        <item name="android:layout_width">match_parent</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:textColor">#FFAB91</item>
    </style>
</resources>
```

Este estilo puede ser asignado:

```
<EditText
        android:id="@+id/input_usuario"
        style="@style/textViewStyle"
        android:hint="Correo"/>
```

Añadir ToolBar a nuestra aplicación

Es usual que nuestro proyecto haga uso del `ActionBar` por defecto. Como nuestra intención es usar el `Toolbar`, entonces debemos deshabilitar el `ActionBar`.

Vamos a nuestro archivo **themes.xml** y asegurarnos de asignar las siguientes propiedades:

```
    <item name="windowNoTitle">true</item>
    <item name="windowActionBar">false</item>
```

`CoordinatorLayout` básicamente nos permitía gestionar interacciones entre sus elementos hijos.Dentro de este elemento incluiremos la `ToolBar`.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout
     xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <com.google.android.material.appbar.MaterialToolbar
        android:id="@+id/topAppBar"
        android:layout_width="match_parent"
        app:title="@string/app_name"
        style="@style/Widget.MaterialComponents.Toolbar.Primary" />
</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

Toolbar en Material Design.

Aplicar Scrolling a la ToolBar

En muchas aplicaciones cuando hacemos un scroll sobre la vista principal interesa ocultar la ToolBar, a este efecto se le llama **scrolling**.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:fitsSystemWindows = "true">

    <com.google.android.material.appbar.AppBarLayout
        android:id="@+id/app_bar_layout"
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        android:gravity="center"
        android:theme="@style/Widget.MaterialComponents.Toolbar.Primary">

        <com.google.android.material.appbar.MaterialToolbar
            android:id="@+id/topAppBar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            app:title="@string/app_name"
            style="@style/Widget.MaterialComponents.Toolbar.Primary"
            app:layout_scrollFlags="scroll|enterAlways"/>
    </com.google.android.material.appbar.AppBarLayout>

    <androidx.core.widget.NestedScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_behavior="@string/appbar_scrolling_view_behavior">

        <TextView
            android:text="@string/hello_world"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
             android:fitsSystemWindows = "true" />
    </androidx.core.widget.NestedScrollView>
</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

- `android:fitsSystemWindows` este atributo a `true` permite ajustar la ventana de la activity por debajo de la ToolBar.
- `AppBarLayout` siempre debe envolver a la `Toolbar` y los demás componentes pertenecientes a ella (como pestañas, imágenes, etc).

- La forma en que se afectan los hijos de `AppBar` se determina en `app:layout_scrollFlags`.
- Hacer desaparecer la `AppBar` por scrolling requiere que haya un objeto con scroll marcado con el atributo `app:layout_behavior="@string/appbar_scrolling_view_behavior"`. Este debe declararse por debajo de la `AppBar`. En nuestro ejemplo será un `NestedScrollView` que además incluye un `TextView`

> ✋Los valores que puede tomar `app:layout_scrollFlags` y que controlan el tipo de comportamiento con que desaparece la `Toolbar` son:
>
> - `scroll` : indica que un view desaparecerá al desplazar el contenido.
> - `enterAlways` :vuelve visible al view ante cualquier signo de scrolling.
> - `enterAlwaysCollapsed` : vuelve visible el view solo si se mantiene el scroll en la parte superior del contenido.
> - `exitUntilCollapsed` : desaparece el view hasta que sus dimensiones superen la altura mínima.

Aplicar Collapsing a la ToolBar

Para controlar las reacciones de expansión y contracción de los elementos de vista que se encuentran dentro de un `AppBarLayout` (una imagen, pestañas o cualquier otro elemento) es necesario utilizar un layout especial que envuelva la `ToolBar`, este layout se llama `CollapsingToolbarLayout`.

```
...
    <com.google.android.material.appbar.AppBarLayout
        android:id="@+id/app_bar_layout"
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        android:gravity="center"
        android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar">

        <com.google.android.material.appbar.CollapsingToolbarLayout
            android:id="@+id/collapsing_toolbar"
            android:layout_height="match_parent"
            android:layout_width="match_parent"
            app:contentScrim="?attr/colorPrimary"
            app:layout_scrollFlags="scroll|exitUntilCollapsed">

            <ImageView
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:scaleType="centerCrop"
                app:srcCompat="@drawable/image"
                app:layout_collapseMode="parallax" />

            <com.google.android.material.appbar.MaterialToolbar
                android:id="@+id/topAppBar"
                android:layout_width="match_parent"
                android:layout_height="?attr/actionBarSize"
                app:title="@string/app_name" />
        </com.google.android.material.appbar.CollapsingToolbarLayout>>
...
</androidx.coordinatorlayout.widget.CoordinatorLayout>
```