Instituto Tecnológico Superior del Sur de Guanajuato

“Soporte para diferentes pantallas”

Programación Móvil I

Elaborado por:

Isabel Contreras Rico

No. Control

S17120218

Catedrático:

Ing. Gustavo Iván Vega Olvera

Fecha:

09/10/2020

Índice

Crear compatibilidad con diferentes tamaños de pantalla

El correcto diseño de una aplicación móvil debe ser flexible, ajustarse a los diferentes tamaños de pantalla que existen en el mercado, y así poder satisfacer a un mayor número de usuarios.

Diseño flexible

Para crear un diseño independiente del hardware, existen técnicas como las que se enlistan a continuación.

Uso de ConstraintLayout

ConstraintLayout permite especificar el tamaño y posición de los componentes de la aplicación en relación a otros componentes o a la pantalla misma. Así, al cambiar las dimensiones de la pantalla, los componentes se ajustan automáticamente.

Para emplear este tipo de diseño, es más sencillo emplear la vista de diseño que ofrece Android Studio, para arrastrar y ajustar los componentes.

Evitar tamaños de diseño hard-coded

Es preciso evitar los diseños hard-coded, es decir, establecidos estáticamente. Para el ancho y altura, debe usarse "**`wrap_content`**", que indica que debe ajustarse el contenido dentro de la vista; y "**`match_parent`**", que hace que se expanda lo más posible dentro de la vista principal.

Diseños alternativos

El diseño puede responder a diferentes tamaños de pantalla, sin embargo, la vista producida puede no resultar tan agradable para el usuario si se trata de un dispositivo muy grande. Para ello deben proporcionarse recursos de diseño alternativos.

Pueden crearse diseños específicos para determinados tamaños de pantalla, a través de crear directorios en **`res/layout/`** con calificadores de configuración de pantalla, como **`layout-w600dp`**, para pantallas de un ancho de 600dp.

Para crear este diseño duplicado es necesario seguir los siguientes pasos:

1. Diseño predeterminado (barra de herramientas) → **Orientation for Preview**
2. Elegir **Create Landscape Variant**, o **Create Other.**
3. **Create Other** → **Select Resource Directory** → seleccionar calificador de pantalla izquierdo → agregar a la lista **Chosen qualifiers**

Calificador de ancho mínimo

El calificador de pantalla de ancho mínimo proporciona diseños alternativos para pantallas con un ancho mínimo medido en pixeles independientes de la densidad. Especifica el **lado más pequeño** de la pantalla independiente de la orientación.

`res/layout/main_activity.xml` → pantallas menores a 600dp

`res/layout-sw600dp/main_activity.xml` → pantallas con mínimo 600dp

Calificador de ancho disponible

Puede cambiarse el diseño según el ancho o la altura disponibles actualmente. Es decir, varía de acuerdo a la posición de la pantalla.

`res/layout/main_activity.xml` → pantallas menores a 600dp

`res/layout-w600dp/main_activity.xml` → ancho disponible de 600dp

Calificadores de orientación

Es posible cambiar la experiencia del usuario cuando este alterne entre las orientaciones horizontal y vertical. Para ello se agregan los calificadores port o land precedidos por los otros calificadores de tamaño.

`res/layout/main_activity.xml` → pantallas menores a 600dp

`res/layout-land/main_activity.xml` → pantallas en posición horizontal

`res/layout-sw600dp/main_activity.xml` → pantallas con mínimo 600dp

`res/layout-sw600dp-land/main_activity.xml` → pantallas con mínimo 600dp en landscape

Compatibilidad con diferentes densidades de pixeles

Un detalle importante a la hora de diseñar aplicaciones para diferentes dispositivos, es que no todas las pantallas comparten pixeles del mismo tamaño, es decir, no es lo mismo una pantalla con 160 pixeles por pulgada cuadrada, que una de 480 en el mismo espacio, esta característica se conoce como **densidad de pixeles**.

Pixeles independientes de la densidad

Definir dimensiones con pixeles representa un problema, ya que la distinta densidad de pixeles puede generar imágenes escaladas y una mala experiencia para el usuario.

Para que el tamaño se mantenga visible a pesar de la densidad de pixeles, es necesario diseñar con unidad de medida ***(dp) pixeles independientes de la densidad***. Es una unidad de pixel virtual que equivale a 1 pixel en una pantalla de densidad media.

En el caso del texto, es preciso usar ***pixeles escalables (sp)*** como unidad de medida. sp de manera predeterminada tiene el mismo tamaño que dp, pero el primero se ajusta al tamaño de letra preferido por el usuario

```
<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/clickme"
android:layout_marginTop="20dp" />
```

```
<TextView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:textSize="20sp" />
```

Conversión de unidades dp a píxeles

Hay casos específicos, donde es necesario emplear el tamaño de pixeles reales del dispositivo. Por ejemplo, en un desplazamiento obligatorio de 16 pixeles, 16 dp en un celular serán aproximadamente 2.5cm, en cambio en una Tablet puede significar 1.7cm. Por ello, es importante especificar en el código como dp y posteriormente realizar una conversión desde Java o Kotlin.

```
// Gesto expresado en dp
    private static final float GESTURE_THRESHOLD_DP = 16.0f;
// Escala de la densidad de la pantalla
    final float scale = getResources().getDisplayMetrics().density;
// Convertir dp a pixeles basado en densidad
    mGestureThreshold = (int) (GESTURE_THRESHOLD_DP * scale + 0.5f);
// Use mGestureThreshold como distancia en pixeles
```

Valores de configuración escalados previamente

Tanto Java como Kotlin se apoyan de una clase llamada "***ViewConfiguration***" que provee distancias, velocidades y tiempos normales que usa el sistema Android. Los métodos de esta clase que empiezan con el prefijo "***getScaled***" aseguran que el valor en pixeles se mostrará apropiadamente, independiente de la densidad de pixeles actual.

```
private static final int GESTURE_THRESHOLD_DP =
      ViewConfiguration.get(myContext).getScaledTouchSlop();
```

Proporcionar mapas de bits alternativos

Para ofrecer gráficos de calidad, es necesario crear varias versiones de cada mapa de bits en la aplicación. De lo contrario, Android escalará los mapas de bits arrojando imágenes borrosas. Existen muchos intervalos de densidad disponibles. Para crear estos elementos se recomienda seguir el estándar de proporción de escalamiento **3:4:6:8:12:16**

3	0.75 x	densidad baja (ldpi)	36 x 36
4	1.0 x	densidad media (mdpi)	48 x 48
6	1.5 x	densidad alta (hdpi)	72 x 72
8	2.0 x	densidad muy alta (xhdpi)	96 x 96
12	3.0 x	densidad muy, muy alta (xxhdpi)	144 x 144
16	4.0 x	densidad extremadamente alta (xxxhdpi)	192 x 192

Los archivos degenerados se colocan en el directorio correspondiente en **res/,** al hacer referencia a **@drawable/awesomeimage** el sistema elegirá el correcto automáticamente.

res/
drawable-xxxhdpi/
awesome-image.png
drawable-xxhdpi/
awesome-image.png

Ejemplo ilustrativo

Para ilustrar las características que se mencionan a lo largo de este documento, se ha planteado un sencillo ejemplo, donde se tiene una plantilla de personal (6 agrupaciones de imagen y nombre) que debe ajustarse a los cambios de orientación de la pantalla.

Portrait

El diseño vertical de la pantalla se muestra a través de la siguiente imagen.

Ilustración 1. Diseño vertical (portrait)

Landscape

El diseño horizontal se ejemplifica a continuación

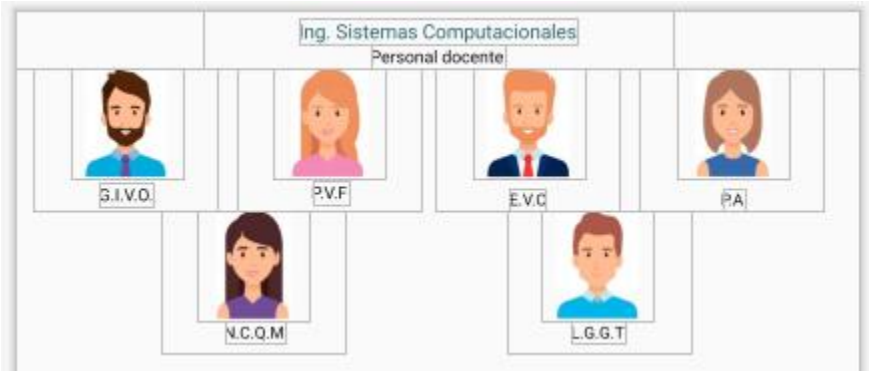

Ilustración 2. Diseño horizontal (landscape)

Código del activity_main.xml

El contenido de este archivo es muy extenso, y repetitivo, por ello, se ha tomado una muestra del código que define cada grupo de perfil, es decir, un ConstraintLayout que contiene dentro de sí una ImageView y un TextView.

En las ilustraciones 3, 4 y 5, se puede apreciar algunas de las recomendaciones del diseño responsivo. Como el uso de ***constraintLayout***, el manejo de ***dp*** como unidad para componentes, ***wrap parent*** para ajustarse al contenido en el caso de los contenedores, y los textos con ***sp*** como unidad de medida.

```
<androidx.constraintlayout.widget.ConstraintLayout
    android:id="@+id/constraintLayout4"
    android:layout_width="160dp"
    android:layout_height="123dp"
    android:layout_marginStart="24dp"
    android:layout_marginLeft="24dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.156"
    app:layout_constraintStart_toEndOf="@+id/constraintLayout2"
    app:layout_constraintTop_toTopOf="parent">
```

Ilustración 3. Definición del constraintLayout (agrupador perfil)

```
<ImageView
    android:id="@+id/imageView"
    android:layout_width="97dp"
    android:layout_height="96dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:srcCompat="@drawable/paty" />
```

Ilustración 4. Definición del ImageView (foto perfil)

```
    <TextView
        android:id="@+id/textView5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="P.V.F"
        android:textColor="#000"
        android:textSize="15sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/imageView"
        app:layout_constraintVertical_bias="0.229" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Ilustración 5. Definición del TextView (nombre perfil)

Contenido del proyecto

En el visualizador del contenido del proyecto, desde la vista de "***Android***" se puede apreciar la creación de layout distintos, para el diseño vertical y horizontal. Estos son seleccionados de manera automática en tiempo de ejecución, cuando el sistema detecta que el dispositivo se encuentra en alguno de estos modos.

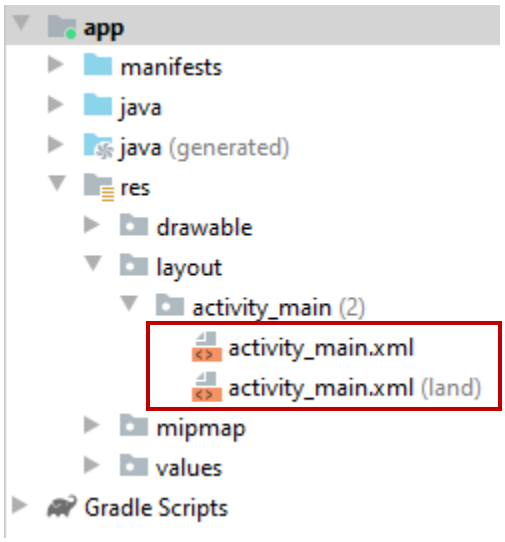

Ilustración 6. Contenido de la carpeta del proyecto.

Ejecución de app

Para ver la aplicación en funcionamiento, se ejecutó en un dispositivo común, y en las Ilustraciones 7 y 8 se muestran los resultados.

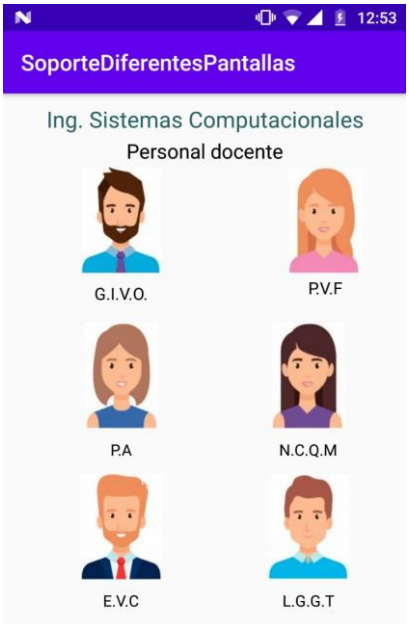

Ilustración 7. Visualización vertical

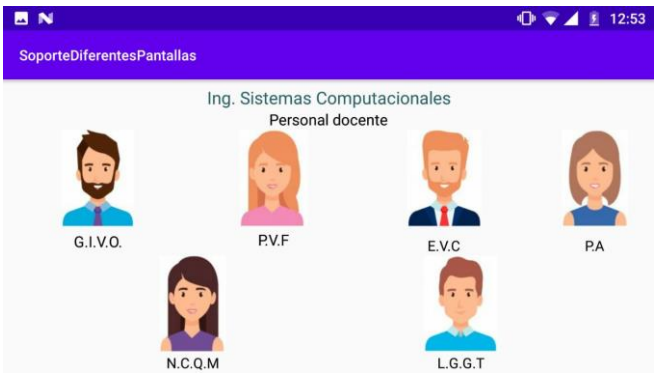

Ilustración 8. Visualización horizontal.