MEDAC
Instituto Oficial de Formación Profesional

PROGRAMACIÓN MULTIMEDIA Y DISPOSITIVOS MÓVILES
TÉCNICO EN DESARROLLO DE APLICACIONES MULTIPLATAFORMA

Programación en Android I

03

ÍNDICE

OBJETIVOS

Conocer la nomenclatura de dimensiones de pantalla.

Conocer la localización de recursos en un proyecto Android.

Crear y aplicar estilos.

Internacionalizar una aplicación a otro idioma.

/ 1. Introducción y contextualización práctica

En esta unidad, estudiaremos cómo se **gestionan** y **organizan** los recursos dentro de una **aplicación Android**.

En primer lugar, veremos cómo se organizan y dividen las pantallas de los dispositivos Android según su tamaño.

Seguidamente estudiaremos cómo podemos organizar de forma correcta todos los recursos que necesita una aplicación Android estándar, como pueden ser imágenes, textos, colores, etc.

También veremos cómo podemos definir y aplicar estilos para cambiar el aspecto de una aplicación.

Por último, explicaremos cómo podemos internacionalizar una aplicación, es decir, cómo podemos traducir sus textos a otro idioma.

Todos estos conceptos los iremos ampliando y profundizando en los próximos temas, así que no te preocupes, ¡esto no ha hecho más que empezar!

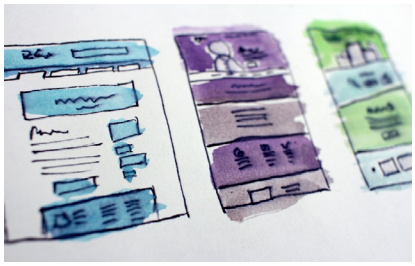

Fig. 1. Diseñando una aplicación.

Escucha el siguiente audio en el que planteamos el caso práctico que iremos resolviendo a lo largo de esta unidad:

Audio Intro. "Probando nuestras aplicaciones en nuestros móviles"

https://bit.ly/2Zpsd9y

/ 2. La pantalla

La **principal característica** de un smartphone es, indiscutiblemente, su **pantalla**, ya que gracias a ella podremos interactuar con todas nuestras aplicaciones.

En calidad de desarrolladores Android, nuestro objetivo principal es que cualquiera de nuestras aplicaciones se pueda **instalar en el máximo número de dispositivos posibles**, independientemente de que se trate de smartphones o tablets, para así tener la mayor cantidad de público.

La gran variedad de dispositivos móviles hace que estos tengan características diferentes. La más notable, como ya hemos indicado, es la pantalla. Esta va a condicionar el diseño de nuestras aplicaciones, ya que se trata de una pantalla pequeña y no podremos colocar mucho contenido.

Pero el tamaño de pantalla no es la única característica que vamos a tener que tener en cuenta, también tendremos que considerar:

- **Tamaño:** Se define como la longitud de la pantalla en diagonal y esta puede ser small, normal, large y extra large.
- **Densidad:** Es la cantidad de píxeles que tendrá la pantalla, y se mide en puntos por pulgada o DPI. Esta puede ser baja o LDPI, media o MDPI, alta o HDPI y extra alta o XHDPI.

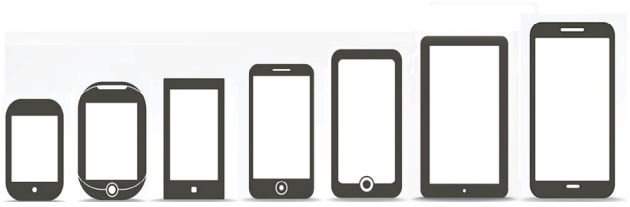

Fig. 2. Diferentes pantallas de smartphones.

- **Resolución:** Se define como la cantidad de píxeles de la pantalla tanto en horizontal como en vertical.

Según cual sea el **tamaño de la pantalla**, podremos **personalizar el diseño** de nuestras pantallas, es decir, que para una pantalla con una resolución pequeña podremos colocar una serie de elementos y, para una resolución de pantalla grande, otros.

Audio 1. "Probando pantallas"
https://bit.ly/3j0IxoH

/ 3. Recursos

Cualquier aplicación Android que se precie va a utilizar, en menor o en mayor medida, recursos para su creación.

Dentro de la categoría **recursos tenemos:**

- Textos.
- Imágenes.
- Colores.
- Definición de estilos.
- Sonidos.

Estos recursos son una parte fundamental y, siempre que sea posible, vamos a separarlos del código para conseguir una aplicación muchísimo más fácil de mantener.

Organizar los recursos de forma correcta hará que el propio sistema operativo Android se encargue de elegir por nosotros cada elemento dependiendo de la configuración del dispositivo, como el tamaño de pantalla, el idioma del dispositivo, etc.

Los recursos estarán localizados en un punto específico dependiendo del tipo de recurso. Como es lógico, los textos no estarán en el mismo sitio que los colores.

Todos y cada uno de los recursos, salvo las imágenes, van a estar dentro de **un fichero XML**, aprovechando así la simplicidad de este lenguaje de marcas para su configuración. Cada uno de ellos tiene, además, una sintaxis diferente.

Todos los recursos de nuestra aplicación estarán dentro de la **carpeta res del proyecto**, de la cual se desprenderán una serie de directorios para cada uno de ellos.

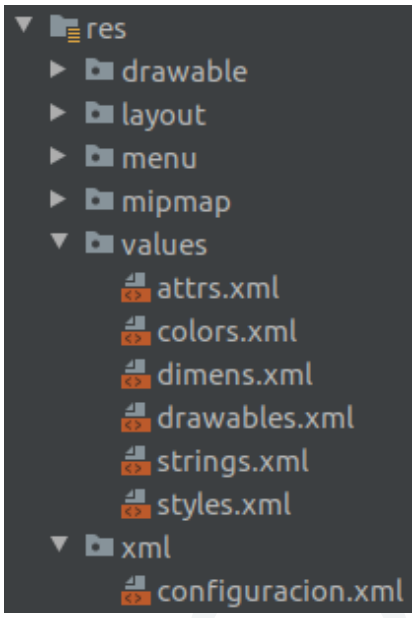

Fig. 3. Carpeta de recursos.

/ 4. Caso práctico 1: "Recursos y tamaño de la aplicación"

Planteamiento: Pilar y José están creando su primera aplicación. Esta es muy simple. Deberán mostrar por pantalla una imagen y cambiar el color de fondo de dicha aplicación. Nuestros amigos ya han encontrado una imagen que les gusta y la han colocado dentro de su proyecto para poder mostrarla en la interfaz gráfica. Pilar ha elegido una imagen muy simple, el logo de su marca de comida favorita, mientras que José ha elegido un fondo de pantalla de unas montañas nevadas de 25 MB en alta definición. «Esa imagen me gusta mucho, pero ¿no crees que ocupa mucho?», le comenta Pilar a José, a lo que este le responde que no pasa nada, que ese aspecto no influirá en nada.

Nudo: ¿Crees que José está en lo cierto? ¿Cómo crees que pueden influir el tamaño de las imágenes en nuestras aplicaciones al publicarlas?

Desenlace: Cuando estamos acostumbrados a crear programas de escritorio, no tenemos demasiado en cuenta el peso de las imágenes que hemos puesto ya que, en el ordenador, el tamaño no es tan restrictivo, pero ahora hemos cambiado de dispositivo, y no estamos en un escritorio. Nos encontramos en un smartphone.

Hay que tener mucho cuidado con los recursos, dado que estos poco a poco van añadiendo tamaño a nuestra aplicación. De esta forma, cuando nos dispongamos a publicarla, puede que nos llevemos una sorpresa porque esta pese demasiado, con los consecuentes retrasos y lentitud, los cuales ofrecerían una mala experiencia de navegación a muchos posibles usuarios.

Recuerda que el almacenamiento en un dispositivo móvil es reducido. Además, esas imágenes se tendrán que descargar para ser visualizadas y no todos los usuarios tendrán una conexión de datos ilimitada.

Por lo tanto, tenemos que empezar a cambiar el chip y ser muy selectivos con los recursos que integremos en nuestra aplicación, ya sean imágenes, sonidos, etc.

Fig. 4. Interfaz de usuario con imágenes.

/ 5. Imágenes y textos

Todas y cada una de las aplicaciones Android que desarrollemos van a tener textos en mayor o menor medida, desde un texto de «aceptar» en un botón a una lista desplegable.

Los textos los podremos poner de dos formas completamente opuestas.

La primera será directamente en la configuración de texto del elemento que estemos creando, lo cual no es aconsejable ya que, si queremos cambiar un texto en concreto, tendremos que ir a dicha pantalla y cambiarlo. Además, ¿y si hubiera que cambiarlo en 50 sitios? Imaginemos que queremos cambiar el texto de un botón de OK a ACEPTAR, puede haber muchísimos elementos en donde cambiarlo, y alguno se nos puede olvidar.

La forma correcta de almacenar los textos es en un fichero de recursos separado. Ese fichero se llama strings.xml y su ruta completa dentro del proyecto es **res/values/strings.xml.**

El fichero strings.xml tendrá un contenido similar a este:

```
<resources>
    <!-- Esto es un comentario -->
    <string name="app_name">Mi aplicación</string>
    <string name="saludo">Hola mundo</string>
</resources>
```

Fig. 5. Fichero de textos.

Como vemos en la figura anterior, cada texto tendrá dos propiedades:

- **Name:** El identificador del recurso de texto que usaremos para acceder a él.
- El propio **valor** del texto, que podrá contener todo tipo de caracteres.

Es posible poner comentarios en nuestros ficheros de recursos, ya que son ficheros XML que nos ayudarán a identificar a qué pantalla pertenecen dichos textos. Por ejemplo, para poder acceder al texto «saludo» pondremos @string/saludo.

Otro recurso que vamos a utilizar habitualmente son las imágenes. Para ello tenemos varias carpetas dentro del proyecto, como res/drawable, res/mipmap-hdpi, res/mipmap-xhdpi y similares. Los sufijos que siguen a mipmap se refieren a la densidad de la pantalla. Cada dispositivo Android tiene una densidad de píxeles por pantalla que se agrupa en varios grupos:

- **ldpi** (pequeña, x0,75)
- **mdpi** (media x1)
- **hdpi** (grande x1,5)
- **xhdpi** (extra grande x2)
- **xxhdpi** (extra extra grande x3)

En la carpeta drawable se ubicarán las imágenes que no cambiarán según el tamaño de la pantalla del dispositivo, es decir, las que serán «fijas».

/ 6. Colores

Al igual que con los textos, dentro de un proyecto de Android Studio podremos definir una serie de colores que utilizaremos en nuestras interfaces gráficas.

Estos colores se pueden colocar directamente sobre la configuración de los elementos gráficos. No obstante, imaginemos que queremos cambiar el color de fondo de nuestras pantallas de color. Si tenemos veinte pantallas en nuestra aplicación (que no son muchas, de hecho), en este caso, tendremos que ir pantalla por pantalla cambiando manualmente el fondo, pero ¿y si tenemos muchas más pantallas?

La forma correcta de almacenar los colores es en un fichero de recursos separado. Ese fichero se llama colors.xml y su ruta completa dentro del proyecto es res/values/colors.xml.

El fichero colors.xml tendrá un contenido similar a este:

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#009688</color>
    <color name="colorPrimaryDark">#00796B</color>
    <color name="colorAccent">#4CAF50</color>

    <color name="naranja">#d67601</color>
    <color name="blanco">#ffffff</color>
</resources>
```

Fig. 6. Fichero de colores.

Como vemos, los colores los definimos mediante su **código RGB** y se muestra una previsualización a la izquierda del código.

Hay varios colores que ya están predefinidos en la configuración de las pantallas y que podemos sobrescribir.

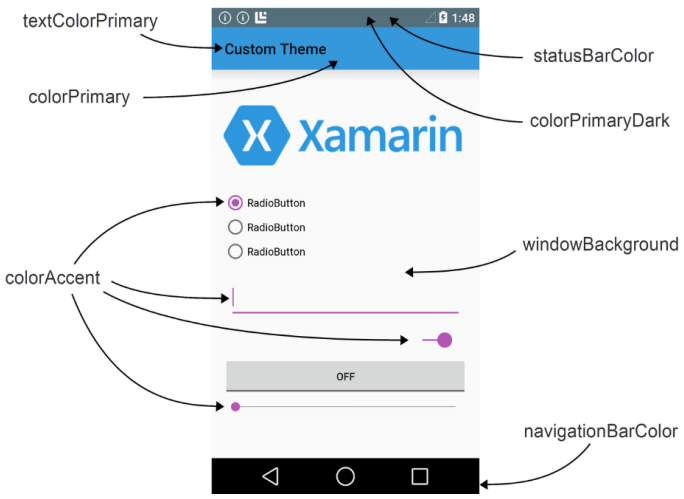

Fig. 7. Colores ya predefinidos.

/ 7. Definición de estilos

Podemos definir los estilos como una **colección de propiedades que indican la apariencia y el formato** de una pantalla. Un estilo puede especificar propiedades como la altura, el relleno, el color de la fuente utilizada en los textos, el tamaño de la fuente, el color de fondo y muchas más cosas.

En Android, los estilos comparten una filosofía similar a las de las **hojas de estilo CSS** del diseño web: permiten separar el diseño del contenido.

La forma correcta de almacenar los estilos es en un fichero de recursos separado. Ese fichero se llama styles.xml, y su ruta completa dentro del proyecto es **res/values/styles.xml.**

El fichero **styles.xml** tendrá un contenido similar a este:

Podremos crear tantos estilos como queramos dentro de nuestro proyecto, usando, también, otros recursos como pueden ser los colores.

En la siguiente figura podemos ver un ejemplo de estilo.

En este ejemplo estamos definiendo un estilo para un texto donde el color será el definido como OK, que es verde, y estará dentro del fichero de colores. Su tamaño será de 20 puntos y estará en negrita.

```
<resources>
    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>
</resources>
```

Fig. 8. Fichero de estilos.

```
    <style name="EstiloTextoOK" parent="@android:style/Theme.Dialog">
        <item name="android:textColor">@color/ok</item>
        <item name="android:textSize">20sp</item>
        <item name="android:textStyle">bold</item>
    </style>
```

Fig. 9. Estilo de texto.

Para cambiar el estilo de un elemento gráfico lo haremos a través de su propiedad style.

Vídeo 1. "Cambiando el estilo de una aplicación"
https://bit.ly/3flaR30

/ 8. Caso práctico 2: "Colores y textos innecesarios"

Planteamiento: Nuestros amigos siguen realizando la aplicación que tienen que entregar. Ahora quieren cambiar el color de fondo y poner algún texto acompañando a la imagen. Ya saben que tienen que tener mucho cuidado, ya que con cada recurso que se agrega, el tamaño de la aplicación aumenta, y tanto los colores como los textos son recursos. Pilar está indecisa con el color de fondo de su aplicación, así que tiene varios colores dentro de sus recursos. «¿Crees que esto aumentará mucho el tamaño de mi aplicación?», le pregunta a José.

Nudo: ¿Qué piensas al respecto? ¿Crees que tener unos colores o unos textos definidos dentro de nuestros recursos aumentará mucho el tamaño de las aplicaciones? ¿Influye que tengamos colores y textos declarados, pero que no usemos?

Desenlace: Ya sabemos que algo que debemos tener muy en cuenta es el aumento innecesario de tamaño de nuestras aplicaciones. Habría que evitarlo al máximo posible.

Fig. 10. Testeando apps.

En este sentido, ya sabemos que tanto colores como textos se incluyen dentro de un fichero XML de configuración, por lo que se puede deducir que unos cuantos colores y textos de más no van a suponer un gran aumento de tamaño de la aplicación, es más, quizás ni lo notemos. De hecho, lo más normal en el desarrollo de las aplicaciones es tener un fichero de colores y estilos bastante grande para así tener gran variedad a la hora de configurar las interfaces gráficas, de forma que no supone un gran problema de aumento de tamaño.

/ 9. Internacionalización de una aplicación

Uno de los **mayores problemas** a los que nos vamos a enfrentar cuando **desarrollamos aplicaciones Android** está relacionado con que estas pueden ser accedidas desde diferentes puntos del mundo, por lo que tendremos que prepararlas para soportar diferentes idiomas.

Fig. 11 Internacionalización.

Cuando **empezamos** a desarrollar aplicaciones, lo normal es poner todos los **textos en nuestro idioma nativo**, pero ¿qué ocurre cuando una aplicación lo suficientemente grande se puede descargar desde cualquier punto del planeta?

Aquí es donde entra en juego **la internacionalización de las aplicaciones**. Con esto vamos a conseguir traducir nuestra aplicación a cualquier lengua, siempre y cuando dominemos los idiomas a traducir o contratemos a alguien para que traduzca los textos necesarios.

Un requisito fundamental para conseguirlo se basa en **colocar todos los recursos en su fichero correspondiente**, como hemos visto en puntos anteriores. Si no hacemos esto, no podremos traducir nuestra aplicación.

Si queremos que nuestros textos estén disponibles en varios idiomas, tendremos que añadir una carpeta diferente para cada copia del fichero strings.xml. De este modo, si queremos traducir nuestra aplicación al inglés y al francés, tendríamos estos tres directorios:

- **res/values** (valores por defecto)
- **res/values-en** (valores para el inglés)
- **res/values-fr** (valores para el francés)

En la siguiente web podemos encontrar todos los códigos de idioma admitidos:

http://utils.mucattu.com/iso_639-1.html.

Es decir, a la carpeta se le añade un sufijo con el **código ISO del idioma concreto**, y dejaremos los textos por defecto en la carpeta sin sufijo. Cuando queramos un recurso del tipo que sea, Android buscará el que mejor encaje y, si no encuentra ninguno, podemos encontrarnos con cierres inesperados de la aplicación.

Una vez hecho esto, bastará con cambiar el idioma del dispositivo para que los textos aparezcan en dicho idioma.

Vídeo 2. "Traduciendo una aplicación"

https://bit.ly/3fsZiqs

/ 10. Resumen y resolución del caso práctico de la unidad

Esta unidad se ha centrado en todo lo que respecta a la **configuración de recursos** de una aplicación Android estándar.

Hemos visto las diferentes nomenclaturas para los diferentes tamaños de las pantallas de los dispositivos, así como que su configuración cambia según el tamaño.

Hemos estudiado también cómo se **organizan los diferentes recursos** dentro de un proyecto Android para que se pueda acceder a ellos de forma correcta. Los recursos más comunes son las imágenes, los textos, los colores, etc.

También hemos incidido en **los estilos**, los cuales nos van a servir para cambiar el aspecto de nuestra aplicación de una forma rápida y sencilla.

Por último, hemos comprobado cómo podemos internacionalizar una aplicación de forma muy sencilla, siempre que almacenemos los recursos en un lugar adecuado. Con esto podremos cambiar el idioma de nuestra aplicación sin problema.

Resolución del caso práctico de la unidad

Ya hemos estudiado en otras asignaturas que, si queremos una aplicación en varios dispositivos o sistemas operativos, tendremos que volver a compilarla, a no ser que el lenguaje sea multiplataforma.

Esto se puede extrapolar a las aplicaciones móviles ya que, como sabemos, existen dos grandes sistemas operativos móviles en la actualidad: **Android** e **iOS.**

Podemos llegar a pensar inicialmente que, al **desarrollar aplicaciones móviles**, estas servirán para todos los sistemas operativos de cualquier dispositivo, pero nada más lejos de la realidad. Cuando desarrollemos aplicaciones Android, por ejemplo, estas solo servirán en dispositivos Android. Por lo tanto, no las podremos probar en un móvil con iOS.

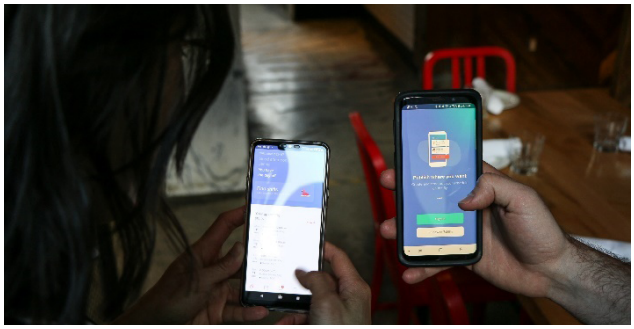

Fig. 12. Diferentes modelos de smartphones.

/ 11. Bibliografía

App resources overview |. (2019, 27 diciembre). Android Developers. https://developer.android.com/guide/topics/resources/providing-resources

Estilos y temas | Desarrolladores de Android |. (2019, 27 diciembre). Android Developers. https://developer.android.com/guide/topics/ui/themes.html?hl=es-419