|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Materia:** |  | Programación Móvil II | | |
| **Práctica:** |  | AudioLibros | | |
| **Alumno(s):** |  | Jesús García Páramo  Irving Nahum Paniagua Mora | **Fecha:** | 04/11/2019 |

**Objetivo**

**Temas del plan de estudios**

Tabla de contenido

[1.- Diseño avanzado de interfaces de usuario 2](#_Toc23790944)

[1.1.- Acceder a objetos globales de la aplicación 2](#_Toc23790945)

[1.1.1.- La clase Application 2](#_Toc23790946)

[1.1.2.- Uso del patrón Singleton 2](#_Toc23790947)

[1.2.- Material Design 2](#_Toc23790948)

[1.2.1.- Definición de la paleta de colores de la aplicación 2](#_Toc23790949)

[1.3.- RecyclerView 2](#_Toc23790950)

[1.4.- Fragments 3](#_Toc23790951)

[1.5.- Otros elementos de la interfaz de usuario 3](#_Toc23790952)

[1.5.1.- Menús contextuales 3](#_Toc23790953)

[1.5.1.1.- Barra de acciones con ActionBar y ToolBar 3](#_Toc23790954)

[1.5.1.2.- Configurar la barra de acciones desde un fragment 3](#_Toc23790955)

[1.5.2.- Vistas animadas: CoordinationLayout, AppBarLayout, FloatingActionButton 4](#_Toc23790956)

[1.5.3.- Pestañas con TabLayout 4](#_Toc23790957)

[1.6.- Navigation Drawer 4](#_Toc23790958)

[1.7.- Servicios de búsqueda 4](#_Toc23790959)

[1.8.- Animaciones 4](#_Toc23790960)

[1.9.- Otros aspectos introducidos en la versión 5.0 5](#_Toc23790961)

[1.9.1.- Extraer paletas de colores desde imágenes 5](#_Toc23790962)

[1.9.2.- Uso de gráficos vectoriales 5](#_Toc23790963)

**Material**

El Gran Libro de Android Avanzado 6ta Edición

Android Studio

**Marco Teórico**

**Desarrollo**

# 1.- Diseño avanzado de interfaces de usuario

La interfaz de usuario es una parte fundamental de las aplicaciones Android. Es el conjunto de elementos que permiten al usuario comunicarse e interaccionar con la app. Android ofrece la posibilidad de mostrar e interactuar con la información en distintos formatos: listas, tablas, checks, botones, listas desplegables y gallerías.

### 1.1.- Acceder a objetos globales de la aplicación

Cada uno de los componentes de una aplicación se escribe en una clase separada. Esto hace que en muchas ocasiones resulte complicado compartir objetos entre estos componentes.

Para poder acceder a una información global desde cualquier clase de nuestro proyecto, podemos utilizar el modificador static. De esta forma, no será necesario conocer la referencia a un objeto de la clase, solo con indicar el nombre de la clase podremos acceder a esta información.

### 1.1.1.- La clase Application

Esta clase ha sido creada para almacenar información global a toda la aplicación.

### 1.1.2.- Uso del patrón Singleton

Otra alternativa para almacenar información global a una aplicación es utilizar el patrón Singleton. Una clase definida con este patrón solo dispondrá de una instancia, a la que se podrá acceder desde cualquier clase utilizando un método estático.

### 1.2.- Material Design

Material Design se trata de una guía para el diseño visual de las aplicaciones, que Google no quiere aplicar exclusivamente a dispositivos móviles, sino que pretende utilizar material design en todo tipo de contenidos digitales (páginas Web, aplicaciones para ordenadores, vídeos…).

Se basa en diseños y colores planos. Uno de sus principios es dar peso o materialidad a los elementos del interfaz de usuario. Para ello va a tratar de darle volumen o profundidad utilizando sombras, capas y animaciones.

### 1.2.1.- Definición de la paleta de colores de la aplicación

Es uno de los principios que define Material Design, Google recomienda usar colores vivos y alegres. Cada aplicación ha de definir su propia paleta de colores que la diferencie del resto de aplicaciones.

### 1.3.- RecyclerView

Esta clase ofrece la misma funcionalidad que ListView o GridView, pero de forma más eficiente y flexible. RecyclerView resulta algo más compleja de manejar, pero tiene las siguientes ventajas:

* Reciclado de vistas (RecyclerView.ViewHolder).
* Distribución de vistas configurable (LayoutManager).
* Animaciones automáticas (ItemAnimator).
* Separadores de elementos (ItemDecoration).
* Trabaja conjuntamente con otros widgets introducidos en Material Design (CoordinationLayout).

RecyclerView muestra los elementos por medio de un LayoutManager. El sistema nos proporciona tres descendientes de LayoutManager, que son:

* LinearLayoutManager.
* GridLayoutManager.
* StaggeredGridLayoutManager.

### 1.4.- Fragments

Los fragments son bloques de interfaz de usuario que pueden utilizarse en diferentes sitios, simplificando así la composición de una interfaz de usuario.

Nos permiten diseñar y crear cada uno de los elementos de nuestra aplicación por separado. Luego, dependiendo del tamaño de pantalla disponible, mostraremos uno solo, o más de uno a la vez.

Cuando diseñemos un fragment, este ha de gestionarse a sí mismo, recibiendo eventos de entrada y modificando su vista sin necesidad de que la actividad que lo contiene intervenga. De esta forma, el fragment se podrá utilizar en diferentes actividades sin tener que modificar el código.

Cada fragment ha de implementarse en una clase diferente. Esta clase tiene una estructura similar a la de una actividad, pero con algunas diferencias. La primera es que esta clase tiene que extender Fragment. El ciclo de vida es muy parecido al de una actividad; sin embargo, dispone de unos cuantos eventos más, que le indican cambios en su estado con respecto a la actividad que lo contiene.

### 1.5.- Otros elementos de la interfaz de usuario

### 1.5.1.- Menús contextuales

Un menú contextual se utiliza cuando queremos seleccionar alguna acción de un elemento particular. Lo más frecuente es que aparezca cuando se realiza una pulsación larga sobre un elemento.

### 1.5.1.1.- Barra de acciones con ActionBar y ToolBar

Está situada en la parte superior de la pantalla, fue creada para que el usuario tuviera una experiencia unificada a través de las distintas aplicaciones.

Aglutina varios elementos: los más habituales son el nombre de la aplicación, los botones de acciones frecuentes y el botón <Overflow>.

### 1.5.1.2.- Configurar la barra de acciones desde un fragment

Cuando en la aplicación desarrollada se visualiza el fragment con el detalle de un libre, sin que aparezca el fragment de selección, algunas de las acciones que se muestran en la barra de acciones no tienen sentido. Para resolver este problema se asocia los ítems de menú desde un fragment. De esta forma, en la barra de acciones se visualizarán los ítems de menú introducidos por la actividad más los introducidos por los fragments.

### 1.5.2.- Vistas animadas: CoordinationLayout, AppBarLayout, FloatingActionButton

Las animaciones son muy importantes en Material Design. Google quiere que los elementos del interfaz de usuario se muevan de forma coordinada, de forma que, al moverse un elemento, este puede desplazar a otro.

### 1.5.3.- Pestañas con TabLayout

Android permite introducir pestañas, o tabs, mediante el uso de FragmentTabHost. Antes de la aparición de los fragments se utilizaba la clase TabHost, actualmente marcada como obsoleta. Otra alternativa que acaba de ser marcada como obsoleta; nos permitía incorporar las pestañas dentro de la Barra de Acciones. Tiene la ventaja de que, si caben, se integrarán dentro de la Barra de Acciones.

### 1.6.- Navigation Drawer

Al desplegarlo cubre la barra de acciones e incluso la barra de estado. Además, se recomienda introducir una cabecera con información de la aplicación y la cuenta de usuario. Para implementarlo se necesitan las siguientes clases:

* DrawerLayout: Layout proporcionado por la librería de compatibilidad, que usaremos como contenedor raíz del layout de la actividad.
* ActionBarDrawerToggle: hace de nexo entre el DrawerLayout y la barra de acciones.
* NavigationView: vista que podemos utilizar para diseñar el Navigation Drawer. Está compuesta por una cabecera y un menú. Se puede utilizar cualquier otro tipo de vista o layout.

### 1.7.- Servicios de búsqueda

Una de las características que ofrece Android es realizar búsquedas a través de la vista SearchView. Este elemento puede colocarse tanto dentro de un layout como en la barra de acciones. El framework de búsquedas nos ofrece las siguientes posibilidades:

* Proporcionar sugerencias de búsqueda basadas en consultas recientes.
* Proporcionar sugerencias de búsqueda personalizadas que responden a los datos disponibles en nuestra aplicación.
* Podemos habilitar la búsqueda por voz.

### 1.8.- Animaciones

Android nos proporciona 3 mecanismos para crear animaciones.

* Animaciones de Drawables: La clase AnimationDrawable permiten crear drawables que producen una animación fotograma a fotograma.
* Animacones de vistas: también conocidas como animaciones que permiten crear efectos de traslación, rotación, zum y alta a cualquier vista de nuestra aplicación.
* Animaciones de propiedades: nuevo mecanismo incorporado en Android permite animar cualquier propiedad de cualquier objeto, de una vista o no. Además, modifica el objeto en sí, no solamente cambia su representación en pantalla como ocurre en una animación tween.

### 1.9.- Otros aspectos introducidos en la versión 5.0

### 1.9.1.- Extraer paletas de colores desde imágenes

El uso del color es una caracteriza fundamental de material desing un ejemplo de esto lo comprobamos en la aplicación Topeka que recomendamos, para conseguir de una forma sencilla un uso adecuado del color, se ha añadido un API que nos permita extraer una paleta de colores desde una imagen.

### 1.9.2.- Uso de gráficos vectoriales

A partir de Android lollipop se incorpora la posibilidad de definir drawables de forma vectorial utilizando un formato basado en SVG. La gran ventaja de los gráficos vectoriales es que pueden ser reescalados sin perder definición. Solo necesitamos un pequeño fichero líneas y curvas, luego podrás representarlo en tamaño deseado resulta mucho mas sencillo que diseñar diferentes imágenes en mapas de bits para diferentes densidades.

**Resultado**

**Evaluación**

**Conclusiones**

**Bibliografía**