Desarrollo de aplicaciones móviles I

Tema Nº6:Creando clon de Instagram.

Indicador de logro Nº6:Aplica los principales componentes de Material Design en una aplicación móvil que integra las métricas de diseño más comunes a este paradigma.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº6:**

Creando nuestro entorno de desarrollo.

**Subtema 6.1:**

Configurando nuestro tema Material

Material Design es una guía completa para el diseño visual, interactivo y de movimiento en plataformas y dispositivos. A fin de usar material design en tus apps para Android, sigue las pautas definidas en su especificación y usa los nuevos componentes y estilos disponibles en la biblioteca de compatibilidad de material design.

Android ofrece las siguientes funciones para ayudarte a crear apps de material design:

* Un tema de app de material design para diseñar todos tus widgets de IU
* Widgets para vistas complejas, como listas y tarjetas
* Nuevas API para sombras y animaciones personalizadas

Para aprovechar las características de los materiales, como el estilo de los widgets de IU estándar, y a fin de optimizar la definición de estilo de tu app, aplica un tema basado en material a tu app.

Pantalla de un teléfono celular

Descripción generada automáticamente Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Para ofrecer a tus usuarios una experiencia familiar, usa los patrones de UX más comunes del material:

* Promueve la acción principal de tu IU con un botón de acción flotante (FAB).
* Muestra tu marca, navegación, búsqueda y otras acciones con la barra de la app.
* Muestra y oculta la navegación de tu app con el panel lateral de navegación.
* Usa uno de los muchos otros componentes materiales para el diseño y la navegación de tu app, como barras de herramientas que se contraen, pestañas, una barra de navegación inferior y mucho más.

Puedes crear un tema de la misma manera en que creas estilos. La diferencia es cómo lo aplicas: en vez de aplicar un estilo con el atributo style en una vista, aplica un tema con el atributo android:theme en la etiqueta <application> o en una etiqueta <activity>, en el archivo AndroidManifest.xml.

**Ejemplos:**

Por ejemplo, aquí se muestra cómo aplicar el tema "oscuro" de material design de la biblioteca de compatibilidad de Android a toda la app:

Texto

Descripción generada automáticamente

Aquí se muestra cómo aplicar el tema "claro" a una sola actividad:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Ahora, cada vista de la app o actividad aplica los estilos definidos en el tema dado. Si una vista solo admite algunos de los atributos declarados en el estilo, solo aplica esos atributos e ignora los que no admite.

A partir de Android 5.0 (nivel de API 21) y la biblioteca de compatibilidad de Android v22.1, también puedes especificar el atributo android:theme para una vista del archivo de diseño. Esto modifica el tema para esa vista y cualquier vista secundaria, lo que es útil para alterar las paletas de colores del tema en una parte específica de tu interfaz.

Los ejemplos anteriores muestran cómo aplicar un tema como Theme.AppCompat que proporciona la biblioteca de compatibilidad de Android. Sin embargo, por lo general, querrás personalizar el tema para que se ajuste a la marca de tu app.

**Subtema 6.2:**

Widgets de Material Design

**Ejemplos:**

Creación de la aplicación Mis Lugares

Éste es el más simple de todos los layouts de Android. Un FrameLayout coloca todos sus controles hijos alineados con su esquina superior izquierda, de forma que cada control quedará oculto por el control siguiente (a menos que éste último tenga transparencia). Por ello, suele utilizarse para mostrar un único control en su interior, a modo de contenedor (placeholder) sencillo para un sólo elemento sustituible, por ejemplo, una imagen.

En este curso vamos a crear una aplicación de ejemplo. Tendrá por nombre Mis Lugares y permitirá que los usuarios guarden información relevante sobre los sitios que suelen visitar (restaurantes, tiendas, etc.). En este apartado comenza-remos creando la aplicación Android que solo contendrá una actividad con una barra de acciones extendida.

1. Crea un nuevo proyecto con los siguientes datos:

* Phone and Tablet / Basic Activity
* Name: Material Design
* Package name: com.example.materialdesign
* Language: Java ó Kotlin
* Minimum API level: API 19 Android 4.4 (KitKat)

Cuando creas un nuevo proyecto con Android Studio, si escoges una actividad del tipo Scrolling Activity (o Basic Activity), tendrá un diseño inicial basado en Material Design. Esta actividad, utilizará un tema que hereda de android:Theme.Material. Además, se incorporan varios widgets basados en este diseño, como: Toolbar o FloatingActionButton.

Con el tipo Scrolling Activity conseguimos además que el contenido principal pueda desplazarse, a la vez que la barra de acciones cambia de tamaño. Se conoce como barra de acciones extendida. Es un patrón de diseño propuesto en Material Design, que habrás observado en algunas aplicaciones. Al entrar en la actividad la barra de acciones se muestra en modo extendido. Al desplazar el contenido la barra de acciones pasará al tamaño habitual.

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

Para conseguir este efecto se utiliza el layout CollapsingToolbarLayout.

1. En el navegador de proyecto pulsa con el botón derecho sobre la clase ScrollingActivity y selecciona Refactor > Rename. Introduce como nuevo nombre MainActivity. En la carpeta res/layout renombra el fichero activity\_scrolling.xml por activity\_main.xml. En la misma carpeta renombra content\_scrolling.xml por content\_main.xml. Finalmente, en la carpeta res/menu renombra menu\_scrolling.xml por menu\_main.xml.
2. Ejecuta el proyecto y verifica su comportamiento:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Pasemos a estudiar algunos elementos del proyecto. En el explorador del proyecto abre el fichero Gradle Scripts > build.gradle (Module:app). Observa como en la sección dependencies se han incluido las siguientes librerías:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

La primera es la librería de compatibilidad v7, que incorpora las clases más importantes como: AppCompatActivity, Toolbar o CardView. La segunda es para incluir la librería de compatibilidad de Material Desig. Incorpora otras clases como: FloatingActionButton, AppBarLayout, TabLayout, NavigationView o Snackbar. Gracias al uso de estas librerías podremos utilizar estas clases con un nivel mínimo de API 7, a pesar de que la mayoría han sido introducidas en la versión 5.0 de Android.

Hay una diferencia entre estas librerías: Las clases definidas en la librería de compatibilidad son del API de Android. Es decir, esa misma clase aparece en algún API de An-droid posterior. Cuando en una aplicación la versión mínima de API sea mayor o igual a la de la aparición de una clase, ya no tiene sentido usarla de la librería de compatibilidad. Por el contrario, las clases definidas en la librería de diseño son solo de esta librería. Has de lincarla siempre que necesites una de sus clases.

1. Pasemos a estudiar los layouts creados en el proyecto. En el explorador del proyecto abre el fichero app > res > layout > activity\_main.xml. En la vista Desig se mostrará una previsualización del layout. Selecciona la vista Code pulsando sobre la lengüeta de la parte superior.
2. Estudia la estructura de este layout. Observa como las etiquetas de las vistas son muy largas. Esto es debido a que no están definidas en el API de Android si no en una librería (en concreto a las design). La función de cada una de vistas se estudia en el siguiente apartado. Puede ser un buen momento para leerlo.
3. Abre el fichero app > res > layout > content\_main.xml. Selecciona la vista Code pulsando sobre la lengüeta de la parte inferior. Se trata de un layout muy simple: un <NestedScrollView> con un <TextView> dentro.
4. Abre ahora app > java > com.example.mislugares > MainActivity.java. En el método onCreate() se inicializan algunas de las vistas introducidas en el layout. Este código se explica en el siguiente apartado.
5. Si al ejecutar la aplicación pulsas en los tres puntos de la esquina superior derecha, se mostrará un menú con la opción Settings. Como se ha definido este menú y otras opciones sobre la barra de acciones se explicarán más adelante.

**Vistas de Material Design: CoordinationLayout, AppBarLayout, FloatingActionButtony SnackBar**

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. Observa como al ejecutar el proyecto anterior, cuando pulsas el botón flotante se mueve hacia arriba mientras aparece un texto.

Para poder realizar esta animación, y alguna otra más, se utiliza CoordinationLayout. Es un descendiente de LinearLayout que nos proporciona un nivel adicional de control sobre las vistas que contiene para controlar su comportamiento y permite crear interdependencias. Este contenedor se suele utilizar como raíz del layout.

Para entender cómo se utiliza abre el fichero activity\_main.xml. Observa cómo tras eliminar los atributos la estructura resultante es similar al siguiente:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

El CoordinatorLayout actúa de contenedor y va a permitir que los elementos que contiene puedan realizar animaciones coordinadas. Mediante el atributo fitsSystemWindows podemos conseguir modificar la barra de estado del sistema. En concreto aplicando a esta barra el color colorPrimaryDark (azul oscuro en el ejemplo anterior), para que combine con el color de la aplicación colorPrimary (azul más claro en el ejemplo anterior). El uno de colores en Material Design se explica en el siguiente apartado.

El AppBarLayout también es un LinearLayout vertical y ha sido diseñado para usarlo dentro de un CoordinatorLayout. A su vez es habitual que contenga un elemento de tipo Toolbar y opcionalmente pestañas (TabLayout). Su finalidad es contener los elementos de la barra de acciones de la aplicación (conocida según la nueva nomenclatura de Material Design como *AppBar*). Gracias al atributo layout\_scrollFlags podemos conseguir que los elementos de dentro de esta barra puedan desplazarse hasta que sean ocultados.

Dentro de AppBarLayout tenemos un Toolbar. Como veremos en breve es un nuevo widget que tiene la misma función que un ActionBar.

El contenido del layout que ha de visualizar la aplicación se incluye en un fichero aparte content\_main. De esta forma queda más estructurado y podemos crear contenidos específicos según el tipo de pantalla, versión, … Observa como en el proyecto se ha creado un content\_main que simplemente muestra un texto.

Tras el <include> se ha insertado un FloatingActionButton. Se trata de un tipo de botón característico introducido en *Material Design* para las acciones principales.

Otro aspecto que no hay que olvidar es que ahora hemos de inicializar desde código algunos elementos. Esto se realiza en el método onCreate() de la actividad:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Este código resultará más sencillo de entender, tras realizar el apartado *Uso práctico de Vistas*. Las dos primeras líneas han sido explicadas en la unidad anterior. El método setSupportActionBar() es utilizado para indicar a la actividad la barra de acciones que ha de utilizar. Esta barra de acciones es almacenada en el objeto toolbar, que es creado en el layout XML.

**En Java** para encontrar el objeto se utiliza findViewById(R.id.toolbar). **En Kotlin** este proceso es realizado automáticamente. Lo único que tienes que hacer es asegurarte que se ha importado el paquete kotlinx.android.synthetic.main.activity\_main.\*.

Con respecto al botón flotante simplemente se carga en el objeto fab y se le asigna un escuchador onClick. Más adelante se describe el uso de escuchadores para personalizar la acción asociada. Actualmente, cuando se pulsa muestra un mensaje de texto utilizando un Snackbar. Se trata de una nueva forma de mostrar cuadros de dialogo, introducida en Material Desing, similar a Toast o AlertDialog. La diferencia es que se muestran en la parte inferior de la pantalla y que se visualizan con una animación de desplazamiento.

**Actividad:**

1. CUESTIONARIO TÉCNICO

* Determine cuál es la diferencia entre un AppBarLayout y un LinearLayout
* Detalle la diferencia entre in TableLayout y un GridLayout
* ¿Qué permite configurar la propiedad atributo fitsSystemWindows?
* ¿Cuál es la utilidad del componente Floating Action Button?
* ¿Cuál es su opinión respecto al hecho de que Google determine métricas de diseño y considere en mejores posiciones en los resultados de búsqueda en las tiendas de aplicaciones aquellas que las integren?

1. CONCLUSIONES DE LA EXPERIENCIA

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_