DESARROLLO DE INTERFACES

TÉCNICO EN DESARROLLO DE APLICACIONES MULTIPLATAFORMA

Desarrollo de interfaces en Android

Android Studio. Conceptos

Vamos a utilizar la herramienta Android Studio que es el entorno de desarrollo integrado oficial creado para la plataforma Android.

En primer lugar, vamos a realizar el proceso de descarga e instalación de la herramienta, que aunque es sencilla, es conveniente tener presentes las pautas esenciales de instalación.

1. Desde la página web oficial, se descarga la última versión para el sistema operativo en el que se está trabajando el desarrollo de aplicaciones e interfaces con Android.

2. Tras la descarga, se inicia la instalación del software en el equipo, **ejecutando el instalador** que se ha guardado (posiblemente en la carpeta de descargas).

3. Para completar el proceso, se escogen las diferentes **opciones de configuración** que se muestran al usuario, hasta completar el proceso pulsando el botón Finish. En función del sistema operativo, es posible que nos solicite la concesión de ciertos **permisos de seguridad**.

Tras completar el proceso de instalación, vamos a realizar un repaso sobre la **creación de aplicaciones Android utilizando Android Studio**. Al igual que para el caso del desarrollo de interfaces vistos en los capítulos iniciales de este módulo, es necesario conocer los componentes principales para optimizar su uso y, por tanto, el resultado final.

Creación del proyecto

En primer lugar, vamos a realizar un breve repaso sobre la creación de aplicaciones Android utilizando Android Studio. Tras completar el proceso de instalación, basta con ejecutar la aplicación.

Para crear un nuevo proyecto desde cero,

1. en la página de bienvenida se selecciona la opción "**Start a new Android Studio Project**",
2. a continuación, desde el menú File, accedemos a **New Project**.
3. Tras iniciar el nuevo proyecto, aparece la opción de **selección de la plantilla** para el proyecto. Será posible escoger desde la opción en blanco en la que se realiza el diseño desde cero hasta todo tipo de pantallas prácticamente completas que nos agilizarán mucho el diseño en algunas situaciones.
4. Finalmente, tras escoger la plantilla, se realiza la **configuración del proyecto**:

    Name:

    Package name:

    Save location:

    Language:

Minimum SDK:

Use legacy android.support libraries: (checkbox)

Entorno de diseño y desarrollo

Tras completar los pasos anteriores, ya se habría creado el nuevo proyecto. **Por defecto**, cuando se accede por primera vez, el fichero que se muestra al desarrollador es el llamado **MainActivity**.kt.

Será necesario acceder al **fichero activity_main.xml**, desde el cual estará disponible la **pantalla de diseño** y el código asociado a la creación de cada **nuevo componente**.

1. Desde el **menú Project** que aparece a la izquierda, se selecciona **app, res, layout** y, finalmente, **activity_main.xml**.

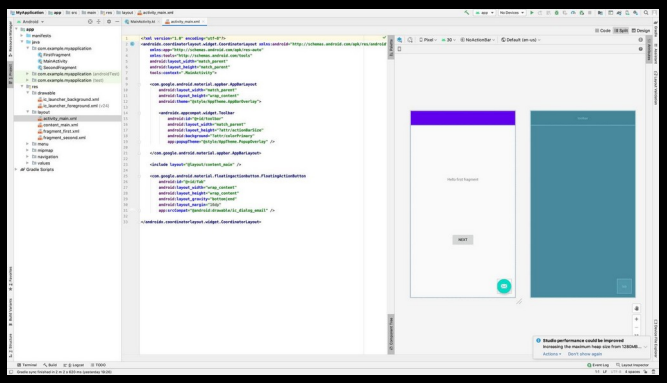
2. Como se ha visto en anteriores capítulos, es posible tener una vista del código generado o de la vista de diseño.

En este caso, será posible **visualizar** el entorno de desarrollo **de tres formas distintas**:

- Code (se muestra el código de desarrollo),
- Design (aparece el lienzo de diseño sobre el que se sitúan y distribuyen los elementos),
- Split (una combinación de las dos anteriores). La **conmutación** entre estas tres modalidades se lleva a cabo desde la **parte superior** de la pantalla a la derecha.

El entorno de diseño queda divido en tres zonas de interacción si se escoge la opción Split:

- **Explorador** de proyectos y ficheros (a la izquierda).
- **Zona de desarrollo** del código de implementación (zona **central**).
- **Zona de diseño** y lienzo donde se colocarán los elementos (a la **derecha**).



Creación de un proyecto con Android

En los temas 13 y 14, abordaremos el desarrollo de interfaces para aplicaciones en Android. Ahora recordaremos cómo crear un proyecto nuevo en Android Studio para implementar su interfaz. Es necesario tener descargado e instalado Android Studio; pueden acceder al enlace disponible aquí: https://developer.android.com/studio?hl=es-419

Una vez instalado, ejecutaremos la aplicación.

En caso de tener un proyecto abierto, podemos crear uno nuevo desde cero yendo a "**File**", seleccionando "**New**", y luego escogiendo "**New Project**". Android Studio ofrece varias opciones, como importar proyectos, crear módulos, importar módulos, entre otras. En este caso, nos enfocaremos en la creación de un nuevo proyecto.

¿Qué nos proporciona Android Studio? Nos permite elegir diferentes plantillas predefinidas que agilizan el desarrollo de aplicaciones y sus interfaces. Por ejemplo, si queremos desarrollar una aplicación para una tienda online con una interfaz que indique la posición exacta, podemos utilizar una plantilla que implemente la actividad con Google Maps y adaptarla según nuestras necesidades.

Android Studio ya incluye plantillas para la pantalla de inicio de sesión, facilitando el desarrollo de la aplicación y aumentando la satisfacción del usuario al almacenar datos de sesión. Al elegir una plantilla, como "Basic Activity", podemos personalizarla proporcionando datos como el nombre, el lenguaje de programación (por ejemplo, Java), el package name, y finalizar el proceso de creación del proyecto.

Navegando por las carpetas, encontramos "drawable" para crear animaciones o imágenes, "layout" que define la distribución de componentes de la aplicación, y otras secciones. Es fundamental tener en cuenta el "AndroidManifest" con datos sobre la aplicación y la carpeta "Java" que contiene diferentes paquetes. En capítulos posteriores, analizaremos en detalle los principales archivos necesarios y exploraremos las distintas partes de la interfaz.

Tipos de componentes

El diseño de interfaces de usuario centrada en el desarrollo de una aplicación móvil en Android es en un conjunto de **objetos contenedores y de objetos contenidos**. Es la combinación de estos elementos lo que permite la consecución de cualquier tipo de interfaz:

- ViewGroup:

  Estos objetos son los **contenedores** que permiten **controlar la posición y distribución** del resto de elementos utilizados para la construcción de la interfaz de usuario de la aplicación. Se trata del **elemento clave** para lograr cualquier tipo de diseño. Un elemento ViewGroup podrá contener otros elementos ViewGroup y componentes simples de tipo View.

- View:

  Reciben este nombre los **componentes** de la interfaz de usuario que implementan una **función concreta** sobre el diseño global de la aplicación.

  Por ejemplo, los elementos View pueden ser las cajas de texto, lienzos en blanco o botones. Este tipo de elementos **no puede contener nada**, es decir, no pueden contener "en su interior" otros elementos View ni ViewGroup.

Uno de los elementos clave para el desarrollo de interfaces en Android son los **layout**. Este elemento permite la **adaptación al esquema de distribución** que van a seguir los componentes dentro de un determinado contenedor, por lo tanto, se trata de un **elemento de tipo ViewGroup**. En concreto, permiten **definir las siguientes características de diseño**:

- Posición de los elementos.
- Dimensión de los elementos.
- Distribución de los elementos.

Sin los layout, los elementos ocuparían todo el contenedor. El uso de los layout nos **permite modificar** de forma **automática:**

- **el tamaño** de los componentes,
- y su **posición**.

Layout vs widget

Un layout define la **estructura visual** de una interfaz de usuario.

En Android se proporcionan mediante **lenguaje XML clases y subclases** que son muy útiles, como los widgets o los layouts.

Los **layouts** sirven para **controlar la posición de los diferentes widgets** secundarios que se puedan encontrar en el diseño. Además, sirven para **asegurar** que las **dimensiones** del widget principal son las **correctas** y que **cumple las restricciones** especificadas.

Si se crea un layout sin un widget principal, en cuanto el layout **se adjunte a un widget** con setLayout(), **se establecerá el tamaño** de ese widget principal.

Paleta de componentes

La paleta de componentes se encuentra disponible desde la isla de diseño **Design**, desplegando la **pestaña Palette**.

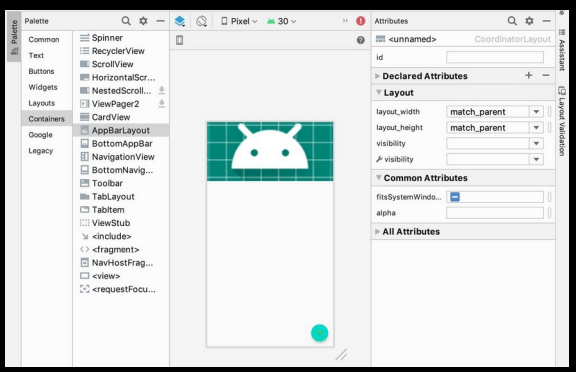
Existen diferentes tipos de componentes, organizados en **ocho grupos**:

Tipos de componentes

1. Common: En este bloque, aparecen los componentes cuyo **uso es más habitual**, como cajas de texto o botones.
2. Text: Elementos que permiten añadir a la interfaz cadenas de texto tales como números de teléfono, fechas...
3. Buttons: Elementos de tipo botón o un compartimento similar (checkbox, togglebutton, radiobutton...).
4. Widgets: Elementos "**extra**" que permiten personalizar la interfaz de desarrollo (calendarios, barras de progreso...).
5. Layouts: Elementos extremadamente importantes en el desarrollo de interfaces, puesto que permiten definir la posición del resto de elementos, su dimensión y la distribución de estos.
6. Containers: Aunque **similares a los layouts**, estos son un tipo de View utilizado para realizar ciertas **distribuciones dinámicas**.
7. Google: Elementos relativos a Google, como **mapas o anuncios**.
8. Legacy: (No viene explicado en el tema ni aparece listado, solo en la imagen...)

A continuación, vamos a analizar algunos de los elementos más importantes para el desarrollo de interfaces en dispositivos móviles, atendiendo a su definición y al código que generan.

Cada uno de los **elementos** contenidos en estos bloques permite **alterar el valor de sus atributos**. Para ello, tras colocar un elemento en la zona de diseño, será posible **consultar** sus atributos desplegando la pestaña que aparece en la derecha y pone Attributes.



Elementos: Button, TextField y TextView

Button

Se trata de elementos de tipo botón. Estos pueden aparecer en forma de botones de texto, imagen o icono.

Suelen presentar una **acción asociada** al ser pulsados.

El código de creación que se genera al colocar un elemento de este tipo sobre el lienzo de diseño es el siguiente:

Definición de un elemento Button

```
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_text" />
```

TextField

Los elementos Multiline Text permiten al usuario introducir cualquier tipo de texto de una línea o más en una caja. Suele utilizarse bastante en el diseño de **formularios**. La etiqueta utilizada para la creación de este tipo de elementos es <**EditText**>.

Definición de un elemento TextField

```
<EditText
    android:id="@+id/editTextTextMultiLine"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:ems="10"
    android:gravity="start|top"
    android:inputType="textMultiLine" />
```

TextView

Este elemento permite mostrar un mensaje en forma de cadena de texto o similar a un usuario. Se trata de una etiqueta de texto.

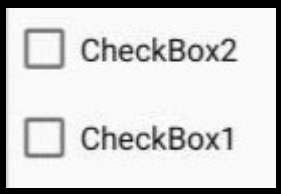
Definición de un elemento TextView

```
<TextView
   android:id="@+id/textView"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="@string/text_view" />
```

Otros elementos: CheckBox, ToggleButton y Spinner

CheckBox

Este tipo de elementos permiten a los usuarios **seleccionar una o más opciones** de entre las que son mostradas por la aplicación. Los elementos checkBox se encuentran en la Palette dentro del grupo denominado **Buttons**.



La sintaxis de cada uno de los elementos que forman un grupo de checkBox es la siguiente:

Definición de un elemento CheckBox

```
<CheckBox
      android:id="@+id/checkBox"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:onClick="onCheckClicked"
      android:text="CheckBox1" />
```

ToggleButton

Los elementos de tipo ToggleButton también se encuentran localizados en la carpeta de tipo Button. Este tipo de elementos permite al usuario **conmutar** entre dos estados.

Definición de un elemento ToggleButton

```
<ToggleButton
    android:id="@+id/toggleButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:checked="true"
    android:textOff="@string/text_off"
    android:textOn="@string/text_on" />
```

Spinner

Este elemento es utilizado para modelar los **menús desplegables** en una aplicación con Android. Se encuentra disponible en la carpeta de elementos **Containers**.

Definición de un elemento Spinner

```
<Spinner
    android:id="@+id/spinner2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
```

Layouts: RelativeLayout

En el menú de Layout de la paleta de elementos, podemos observar que existen múltiples opciones. Para añadir al diseño una de ellas, basta con seleccionar la que más se adecúa a las especificaciones del desarrollo y colocarla en el visor del lienzo central.

Existen diferentes opciones: **RelativeLayout, LinearLayout, GridLayout o TableLayout**, entre otras.

Cuando un elemento va a ser insertado en un layout, es necesario prestar atención a los **atributos característicos del diseño** de estas capas:

- android:layout_**width**: determina el valor del **ancho** del elemento.
- android:layout_**height**: determina el valor de la **altura** del elemento.
- android:layout_**gravity**: determina la **posición** en la que debe situarse un elemento **dentro del contenedor** en el que se encuentra.
- android:layout_**margin** (top, left, right...): determina la **distancia** con respecto a los **márgenes** del contenedor y la **alineación**.

Los atributos que determinan las **dimensiones del ancho y alto** de un elemento pueden tomar dos valores a través del menú de atributos (Attributes):

- match_parent: se asigna como dimensión el **tamaño** completo del **elemento padre**.
- wrap_content: se utiliza para indicar que la dimensión del elemento se **ajusta al contenido** del **mismo**.

RelativeLayout

La distribución de elementos RelativeLayout permite colocar los elementos de manera **relativa al elemento contenedor padre**. Este layout es el que permite **mayor libertad** en cuanto a la **distribución** de los elementos.

Más Layouts: TableLayout y GridLayout

TableLayout:

En la distribución de tipo TableLayout, los elementos quedan distribuidos en forma de tabla, por lo que se indicarán las filas y las columnas que lo componen. Las **filas** utilizan un objeto de tipo **TableRow para ser definidas**. Se trata de una de las distribuciones más utilizadas, puesto que de una forma relativamente sencilla permite realizar una distribución de los elementos de la aplicación que **facilita la navegación del usuario** por la interfaz.

Por ejemplo, en el código de creación de este tipo de distribución que se muestra a continuación, se estaría creando una tabla con tres filas:

Definición TableLayout

```
<TableLayout
   android:layout_width="match_parent"
   android:layout_height="match_parent">

   <TableRow
      android:layout_width="match_parent"
      android:layout_height="match_parent" />

   <TableRow
      android:layout_width="match_parent"
      android:layout_height="match_parent" />

   <TableRow
      android:layout_width="match_parent"
      android:layout_height="match_parent" />

</TableLayout>
```

GridLayout:

La distribución GridLayout crea una rejilla formada por un conjunto de líneas verticales y horizontales en cuyos “huecos” se colocarán los elementos. La **posición de inserción** queda referenciada a través de un **índice**.

Otro Layout: LinearLayout

El layout LinearLayout permite alinear los elementos uno tras otro en una dirección concreta (vertical u horizontal). Las **dimensiones** de cada uno de los elementos serán determinadas utilizando las propiedades destinadas a tal fin (layout_height y layout_width). El diseño genérico, si no se modifican las dimensiones, será el siguiente: Distribución LinearLayout horizontal.

En el menú Layouts, es posible encontrar dos elementos de tipo lineal, el primero de tipo horizontal, y otro de tipo vertical donde la distribución se hace en forma de filas.

La herramienta AppBarLayout

Una herramienta clave para la programación y personalización de cualquier aplicación de desarrollo para un dispositivo móvil como Android es AppBarLayout.

Se trata de una barra que se encuentra normalmente en la **parte superior** de las aplicaciones en Android y que suele contener elementos tales como el **nombre de la aplicación**, el acceso a la configuración —normalmente representado con **tres puntos verticales**, una **caja de búsqueda**, entre otras—, y cuando se desea insertar elementos de tipo widget, la etiqueta inicial de inserción será distinta, siendo necesario indicar la ruta exacta de acceso al elemento.

Análisis de la herramienta e inserción de los primeros elementos

Para comenzar con el desarrollo de nuestro proyecto, nos dirigiremos al archivo principal. En este archivo, podremos enfocarnos en la implementación de la interfaz de la aplicación. ¿Dónde podemos acceder a este archivo? Navegaremos hacia la carpeta 'App', que es la principal de nuestro proyecto. Luego, ingresaremos a la carpeta 'res', seguido de 'layout' y seleccionaremos el archivo 'Activity-Main.xml', que se desplegará a continuación.

Este archivo estará compuesto por diferentes elementos. Recordemos que previamente habíamos seleccionado una plantilla, lo que ha agregado varios elementos. Por ejemplo, elegimos una plantilla que incluye la AppBarLayout, la cual representa la zona superior con las características definidas para cada elemento, como el color de fondo, dimensiones en ancho y altura, identificador y otros atributos.

Para diseñar nuestra interfaz, añadimos elementos desde la paleta de herramientas. Al seleccionar cada tipo de elemento en la paleta, éstos aparecen, y podemos explorarlos para su inserción. Además, es crucial saber cómo visualizar tanto el código como el diseño de la interfaz simultáneamente. En la parte superior, podemos alternar entre estas vistas para trabajar de manera eficiente.

Si seleccionamos un elemento, como un botón, podemos observar cómo cambian sus atributos. Al cambiar el texto de un botón, por ejemplo, podemos ver la actualización tanto en el código como en el diseño visual.

Para personalizar aún más el diseño de nuestra interfaz, podemos especificar diferentes tipos de layout. Estos permiten distribuir los elementos en posiciones específicas. Por ejemplo, arrastramos un layout y colocamos en su interior los elementos deseados para organizarlos eficazmente.

En el código, a medida que colocamos elementos, se actualizan sus atributos, como ancho, alto y contenido. Existen diversas opciones para definir dimensiones, como usar la dimensión exacta o 'wrap-content' para ocupar solo el ancho del elemento actual.

En resumen, al comprender cómo utilizar las plantillas, paletas, layouts y visualizar simultáneamente el código y el diseño, podemos personalizar eficientemente la interfaz de nuestra aplicación de acuerdo con nuestras necesidades.

Implementación de fragment con Maps

Vamos a implementar una pantalla de aplicación que contenga un mapa de actividad de Google Maps, indicando el código necesario en XML para el desarrollo de la misma.

Al crear un nuevo proyecto, una de las plantillas existentes permite crear una versión previa de este tipo de acciones de forma directa.

Pero si queremos crearlo desde cero, es decir, en un documento en blanco, debemos tener claro que el componente esencial será MapView, que está situado en la carpeta de la paleta llamada Google.

El código que permite crear una pantalla para incorporar elementos de actividad a través de Google Maps es el siguiente:

Código XML Google Maps Activity

```
<fragment
    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:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MapsActivity" />
```

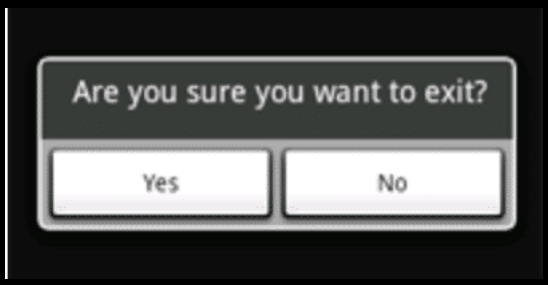
Diseño de ventanas de diálogo para interfaces móviles con Android

El uso de ventanas emergentes en el desarrollo de interfaces móviles puede resultar un elemento esencial para mantener una comunicación activa con los usuarios, y, por tanto, mejorar la interacción y aumentar el nivel de satisfacción de los clientes de una aplicación.

Por esta razón, en este caso práctico, se pide diseñar el código Android necesario para producir una **ventana emergente** de este tipo.

Nudo: El cuadro de diálogo mostrará dos opciones Yes o No, como el que se muestra en la siguiente imagen.

Desenlace: El resultado final de implementar el código es una **interfaz** en la que aparece un cuadro de diálogo como el mostrado en la imagen:



Código Android de creación de un cuadro de diálogo

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setMessage("Are you sure you want to exit?")
       .setCancelable(false)
       .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
          @Override
          public void onClick(DialogInterface dialog, int which) {
             MainActivity.this.finish();
           }
       }).setNegativeButton("No", new DialogInterface.OnClickListener() {
          @Override
          public void onClick(DialogInterface dialog, int which) {
             dialog.cancel();
           }
       });

AlertDialog alert = builder.create();
```

https://developer.android.com