MEDAC
Instituto Oficial de Formación Profesional

DESARROLLO DE INTERFACES
TÉCNICO EN DESARROLLO DE APLICACIONES MULTIPLATAFORMA

Desarrollo de interfaces en Android (I)

13

ÍNDICE

OBJETIVOS

Crear menús que se ajusten a los estándares.

Crear menús contextuales cuya estructura y contenido sigan los estándares establecidos.

Distribuir acciones en menús, barras de herramientas, botones de comando, entre otros, siguiendo un criterio coherente.

Distribuir adecuadamente los controles en la interfaz de usuario.

Utilizar el tipo de control más apropiado en cada caso.

/ 1. Introducción y contextualización práctica

A lo largo de este módulo, hemos analizado ampliamente el diseño de interfaces para aplicaciones, centrándonos, sobre todo, en el desarrollo web. En este capítulo, adaptaremos algunos de los términos trabajados previamente y los completaremos con los elementos propios de las interfaces móviles.

En la actualidad, existen diferentes tecnologías para el desarrollo de aplicaciones móviles; en este módulo, analizaremos el caso de Android y de iOS, este último es el sistema operativo de los dispositivos iPhone o iPad para Apple, aunque puede ser fácilmente adaptado a otros dispositivos y sus respectivos sistemas operativos (watchOS o mac OS X, entre otros)

Por otra parte, como veremos en este tema, el diseño de interfaces de usuario centrada en el desarrollo de una aplicación móvil en Android se basa en un conjunto de objetos **contenedores** y de **objetos** contenidos. Será la combinación de estos elementos lo que permitirá la consecución de cualquier tipo de interfaz.

Fig. 1. Diseño previo de interfaces para Android.

Escucha el siguiente audio donde planteamos la contextualización práctica de este tema, encontrarás su resolución en el apartado Resumen y resolución del caso práctico.

Audio intro. "Los componentes de una interfaz gráfica"

https://bit.ly/3f7lo1c

/ 2. Android Studio. Repaso de conceptos

Existen diferentes entornos que permiten la integración de los componentes necesarios para el desarrollo de aplicaciones en dispositivos móviles y sus correspondientes interfaces. En los siguientes dos capítulos, veremos todo lo relativo a Android.

En este módulo, vamos a utilizar la herramienta Android Studio que, como ya sabemos de otras asignaturas, 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.

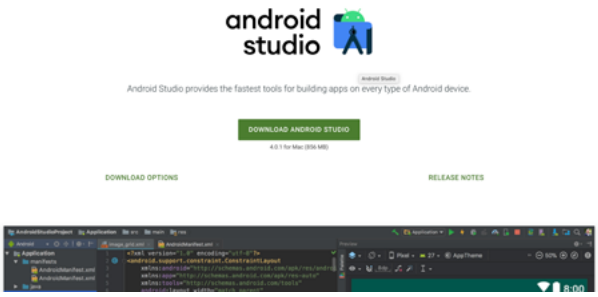

Fig. 2. Interfaz sitio oficial Android Studio.

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.

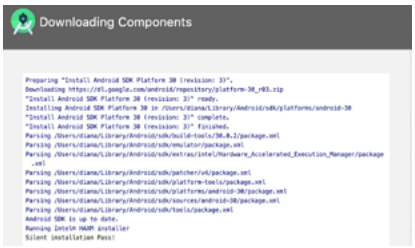

Fig. 3. Proceso de instalación. Última pantalla.

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.

2.1. 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.

Fig. 4. Logo Android Studio.

Para crear un nuevo proyecto desde cero, en la página de bienvenida se selecciona la opción "***Start a new Android Studio Project***", a continuación, desde el menú ***File***, accedemos a ***New Project***.

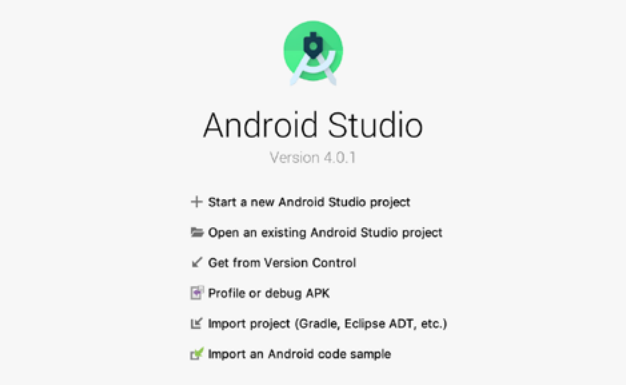

Fig. 5. Pantalla de creación Android Studio.

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.

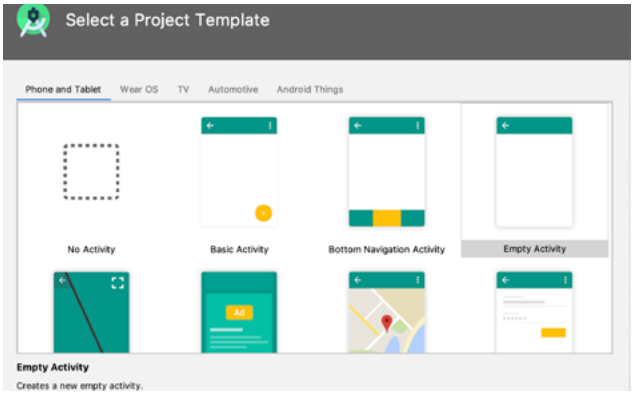

Fig. 6. Pantalla de selección de plantillas Android Studio.

Finalmente, tras escoger la plantilla, se realiza la configuración del proyecto.

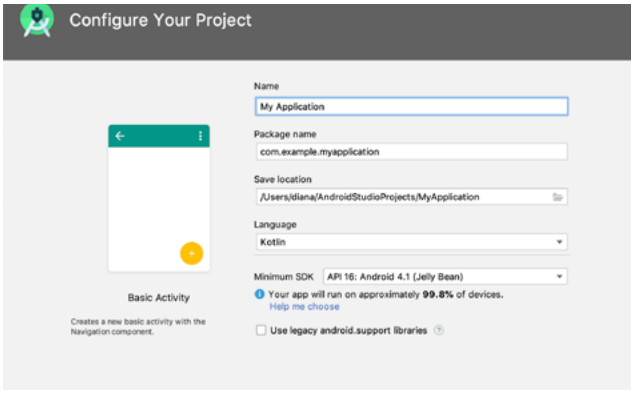

Fig. 7. Pantalla de configuración Android Studio.

2.2. 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**.

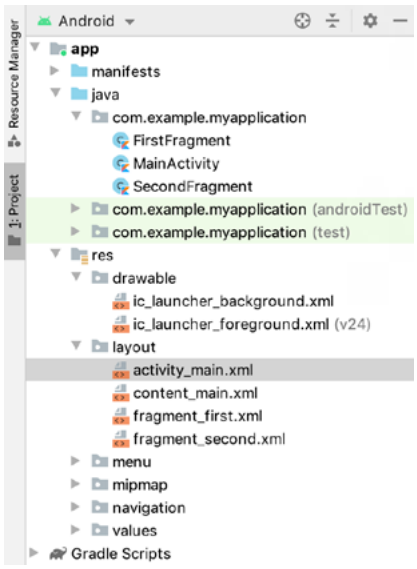

Fig. 8. Acceso a activity_main.xaml.

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) y ***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.

Como se puede observar en la siguiente imagen (Figura 9), 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).

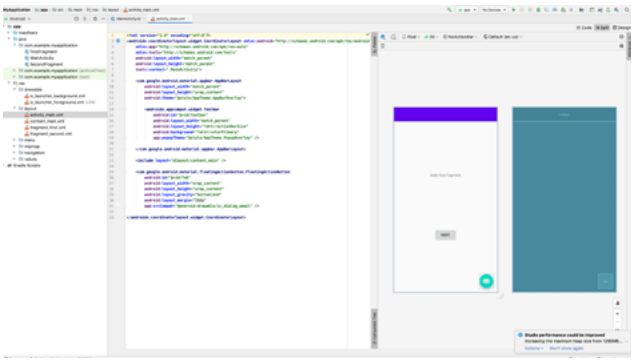

Fig. 9. Interfaz completa Android Studio.

Vídeo 1. "Creación de un proyecto con Android"
https://bit.ly/36YSqOT

/ 3. 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 el tamaño de los componentes y su posición de forma automática.

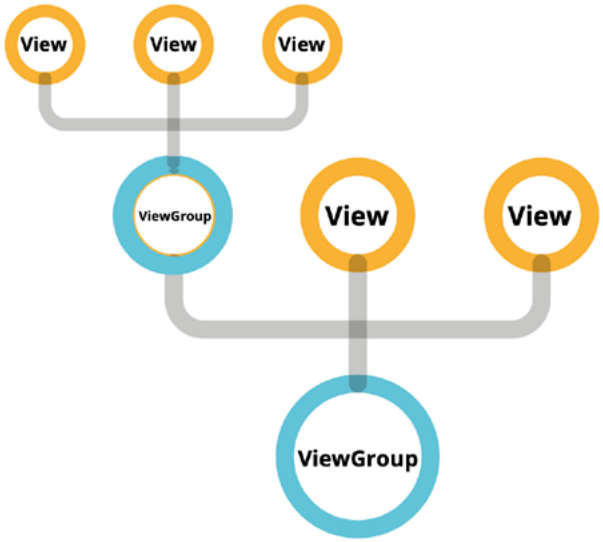

Fig. 10. Diagrama ViewGroup y View.

Audio 1."Layout vs widget"
https://bit.ly/2INjFUn

/ 4. Paleta de componentes

La paleta de componentes se encuentra disponible desde la isla de diseño *Design*, desplegando la pestaña ***Palette***. Existen dientes tipos de componentes, organizados en ocho grupos:

Common	En este bloque, aparecen los componentes cuyo uso es más habitual, como cajas de texto o botones.
Text	Elementos que permiten añadir a la interfaz cadenas de texto tales como números de teléfono, fechas...
Buttons	Elementos de tipo botón o un compartimento similar (checkbox, togglebutton, radiobutton...).
Widgets	Elementos "extra" que permiten personalizar la interfaz de desarrollo (calendarios, barras de progreso...).
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.
Containers	Aunque similares a los layouts, estos son un tipo de View utilizado para realizar ciertas distribuciones dinámicas.
Google	Elementos relativos a Google, como mapas o anuncios.

Tabla 1. Tipos de componentes.

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***.

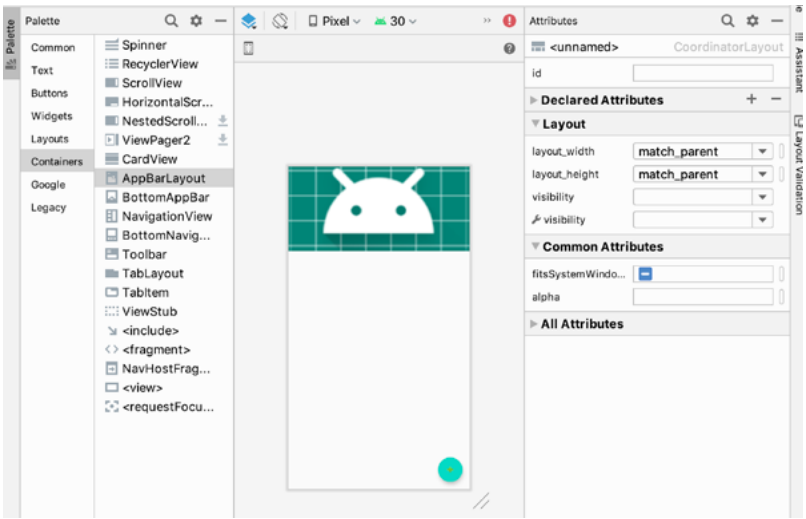

Fig. 11. Interfaz con pestaña de atributos.

4.1. Elementos I: *Button, TextField y TextView*

a. 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.

Fig. 12. Representación Button en Android Studio.

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

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

Código 1. Definición de un elemento Button.

b. 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>.

```
<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"
/>
```

Código 2. Definición de un elemento TextField.

c. 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.

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

Código 3. Definición de un elemento TextView.

4.2. Elementos II: *CheckBox, ToggleButton y Spinner*

a. 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*.

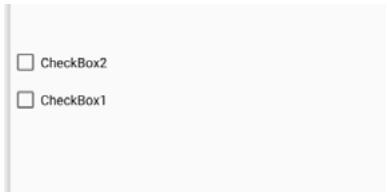

Fig. 13. Representación checkBox en Android Studio.

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

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

Código 4. Definición de un elemento CheckBox.

b. 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.

```
<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"
/>
```

Código 5. Definición de un elemento ToggleButtons.

c. Spinner

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

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

Código 6. Definición de un elemento Spinner..

/ 5. 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.

a. 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.

Fig. 14. Distribución RelativeLayout.

/ 6. Layouts: TableLayout y GridLayout

a. 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:

Fig. 15. Distribución 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>
```

Código 7. Definición TableLayout.

b. 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.

/ 7. Layouts: 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:

Fig. 16. Distribución LinearLayout horizontal.

En el menú *Layouts*, es posible encontrar dos elementos de tipo lineal, el primero de tipo **horizontal** en el que la distribución es como la mostrada en la figura 13, y otro de tipo **vertical** donde la distribución se hace en forma de filas.

Fig.17. Implementando Layouts.

Audio 2. "La herramienta AppBarLayout"
https://bit.ly/3nyuxEa

Vídeo 2. "Análisis de la herramienta e inserción de los primeros elementos"
https://bit.ly/3lIRRyD

/ 8. Caso práctico 1: "Creación de una aplicación con Android"

Planteamiento: En este ejercicio, se pide 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.

Nudo: 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.

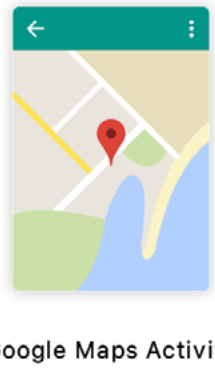

Fig.18. Google Maps Activity. Diseño.

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

```
<?xml version="1.0" encoding="utf-8"?>
    <fragment
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:map="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"
/>
```

Código 8. Código XML Google Mapos Activity.

/ 9. Caso práctico 2: "Diseño de ventanas de diálogo para interfaces móviles con Android"

Planteamiento: 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.

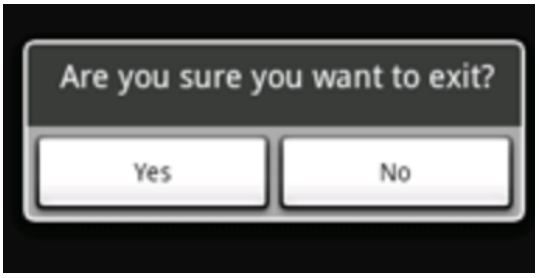

Fig.19. Cuadro de diálogo del caso práctico 2.

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

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

Código 9. Código Android de creación de un cuadro de diálogo.

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

El desarrollo de interfaces para aplicaciones Android requiere de unos pasos concretos para poder acceder a la herramienta de diseño. En primer lugar, será imprescindible 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. Esta interfaz nos muestra tres zonas clave para todo desarrollo: el explorador de proyectos y ficheros, la zona de desarrollo y la zona de lienzo y diseño.

Uno de los elementos clave para el desarrollo de interfaces en Android son los ***layout***. Este elemento permite adaptar 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*.

Existen múltiples opciones en cuanto al diseño del *layout*: ***RelativeLayout, LinearLayout, GridLayout o TableLayout***.

Es necesario prestar atención a los atributos característicos del diseño de layout: **android:layout_width, android:layout_height, android:layout_gravity, android:layout_*margin*** (*top, left, right*...).

Resolución del caso práctico inicial

El diseño propuesto para la aplicación móvil de entrenamiento deportivo consistiría en una pantalla principal dividida en dos partes claramente diferenciables:

- En la parte superior, se establecería una distribución de tipo ***LinearLayout*** con la información de los próximos entrenamientos de manera vertical en orden cronológico.

- En la parte inferior, se diseñarían dos elementos tipo ***Buttons***: uno para enlazar la ubicación del deportista en el momento actual y poder compartirla, y el otro para consultar los nuevos avisos generados en la *app*.

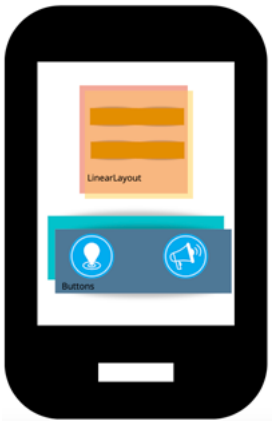

Fig. 20. Diseño del caso de introducción.

/ 11. Bibliografía

Fernández, A.; García-Miguel, B., y García-Miguel, D. (2020). *Desarrollo de Interfaces* (1.a ed.). Madrid, España: Síntesis.

11.1. Webgrafía

Android Developers. Consultado en: https://developer.android.com