
Universidad
Tecmilenio®

Universidad Tecmilenio

Desarrollo de aplicaciones en plataforma Android

Introducción al ambiente de Android

Introducción

Click to add text

El panorama de los dispositivos móviles se ha convertido en un monopolio donde el mercado se maneja entre iOS de Apple y el Android de Google.

El sistema operativo Android es el líder del mercado a nivel mundial, superando al poderoso iOS de Apple. A enero del 2021, Android tiene una cuota de mercado del 71,93% de acuerdo con Mena(2021).

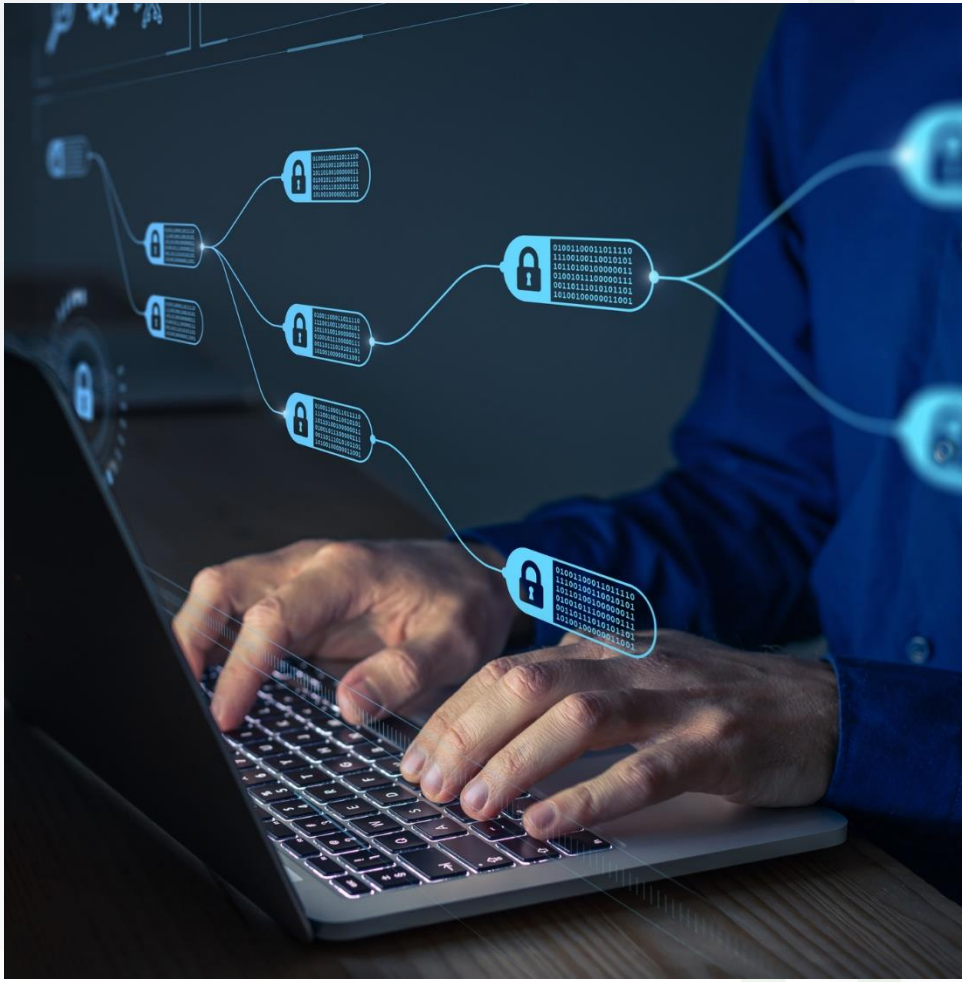
Android Studio representa el entorno de desarrollo integrado (IDE) oficial para el desarrollo de apps para Android.

Mena, M (2021). *Android e iOS dominan el mercado de los smartphones. Recuperado de https://es.statista.com/grafico/18920/cuota-de-mercado-mundial-de-smartphones-por-sistema-operativo*

Explicación

Para la instalación de Android, sigue los pasos indicados en la explicación del tema.

El asistente de configuración de Android Studio te guiará por el resto de la configuración, lo que incluye la descarga de los componentes del SDK de Android que se necesiten para el desarrollo.



Explicación

Una vez que se complete la instalación, inicia abriendo Android Studio.

Android Studio muestra la pantalla de bienvenida, donde puedes hacer clic en Start a new Android Studio project para crear un proyecto.

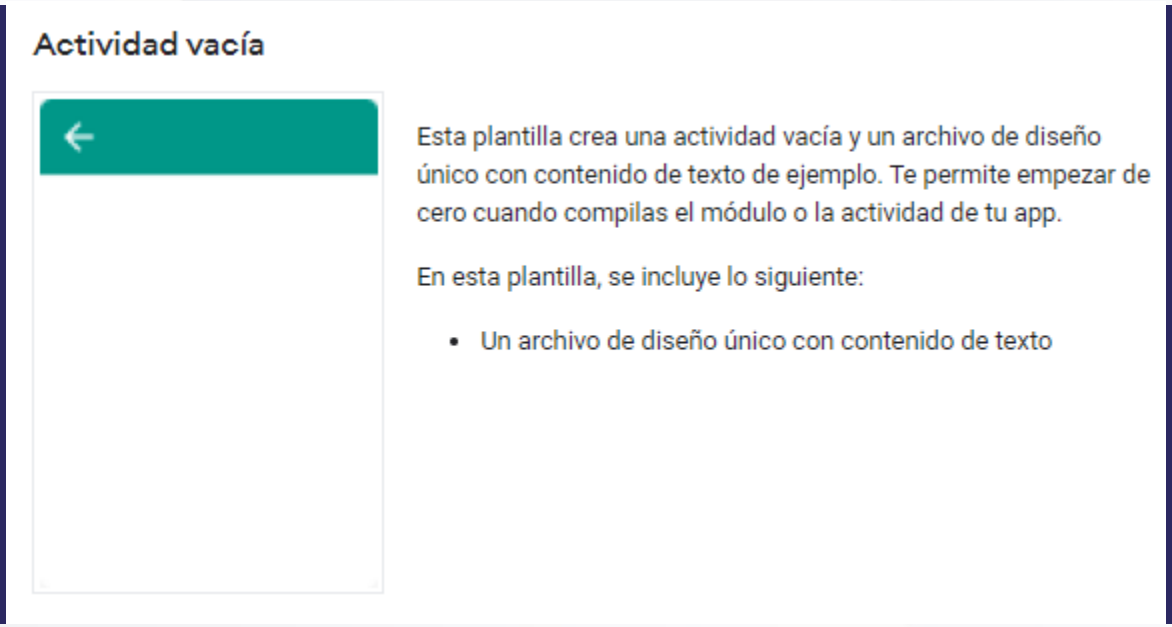
Si tienes un proyecto abierto, selecciona File > New > New Project en el menú principal para comenzar a crear un proyecto nuevo. Recuerda:

- Especificar el nombre del proyecto en Name.
- Especificar el nombre del paquete en Package name, el nombre del paquete también se usa como ID de aplicación, aunque es posible cambiarlo más adelante.
- Especificar en Save location la ruta local donde vas a guardar el proyecto.

  Seleccionar el nivel mínimo de API que admitirá la app en Minimum API level. Ya que si se elije más bajo, es posible que la app confíe en menos API de Android modernas.

Explicación

Recuerda que en una gran parte del curso utilizarás la plantilla conocida como Empty activity:



Explicación

Anatomía de una aplicación

Al crear tus aplicaciones, sobre cuando se está conociendo el entorno, es importante tomarse un tiempo para revisar los archivos más importantes.

1. Asegúrarse de que la ventana Project esté abierta (selecciona View > Tool Windows > Project) y que la vista Android esté seleccionada en la lista desplegable de la parte superior de la ventana. Donde se podrán ver los siguientes archivos:

app > java > com.example.nombredetuApp > MainActivity

Es la actividad principal. Es el punto de entrada de tu app. Cuando compilas y ejecutas la app, el sistema inicia una instancia del elemento Activity y carga su diseño.

Explicación

app > res > layout > activity_main.xml

Es el **archivo** en formato **XML** define el diseño de la interfaz de usuario (IU) de la actividad. Contiene elementos como botones, TextView, EditText, layouts, etc. En el caso de la aplicación de Hola Mundo, en este archivo verás el texto.

app > manifests > AndroidManifest.xml

En el archivo de manifiesto, se describen las características fundamentales de la app, y es donde se definen los permisos para usar aplicaciones que requieren acceso a mapas o internet.

Actividad

- Al elegir una plantilla, ¿cuál es la diferencia de la Empty activity entre las demás?
- De acuerdo a lo que haz practicado con una primera aplicación, ¿cuándo cambian los estados de la aplicación?
- ¿Para qué te servirá comprender los estados de una aplicación?

Cierre

Como puedes ver, hacer una aplicación en Android requiere práctica más que la dificultad que parece tener.

Cada vez más y más desarrolladores se unen a este ambiente de desarrollo debido a que es más barato ser desarrollador para subir aplicaciones, ya que cada año, el mercado de aplicaciones crece tremendamente.

Definitivamente es una excelente oportunidad de adentrarte en el ambiente de desarrollo de aplicaciones móviles.

Universidad Tecmilenio

Desarrollo de aplicaciones en plataforma Android

Interfaz del usuario

Introducción

Este tema te introduce al uso de layouts (distribuciones) y definición de los elementos de interfaz gráfica.

Aquí trabajarás comprenderás el uso de botones, gráficos, iconos y fondos de pantalla que dan la apariencia visual a la aplicación.

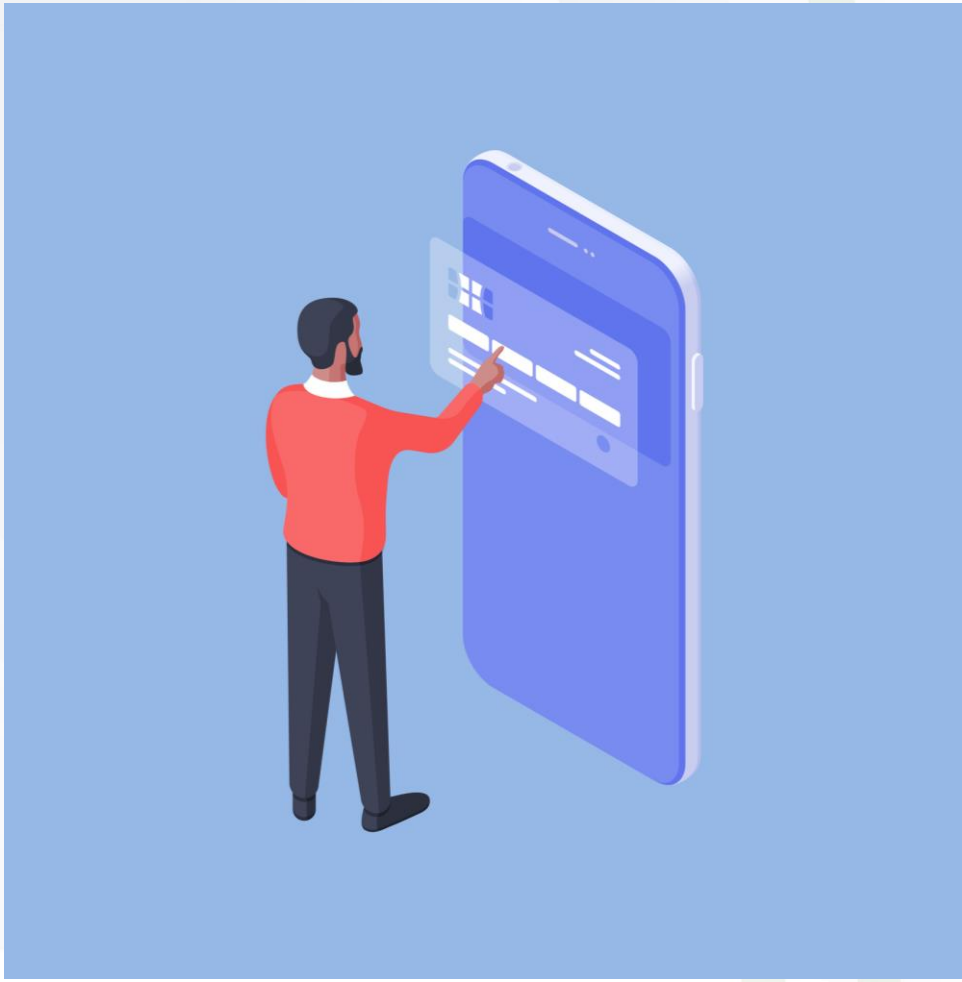
Explicación

Vistas de Android

Un layout es un objeto que representa el espacio contenedor de todas las vistas (Views) dentro de la actividad.

Es lo que define define la estructura y el orden de los elementos para que el usuario pueda interactuar con la interfaz.

En la parte lógica se representan mediante subclases Java que heredan de la clase ViewGroup.



Explicación

Diseñar la interfaz.

Hay dos formas para crear las interfaces en las aplicaciones de Android.

La primera se recomienda para cuando hay una carga de la pantalla, donde se realizan diseños y se muestran los componentes de manera estática, ya son definidos en la aplicación. En esta opción se define la interfaz en el archivo XML ubicado en src / layouts del proyecto.

Este archivo se importa hacia la actividad a través de la construcción setContentView(nombre_del_archivo_layout) y luego las vistas se asignan a objetos Java a través de la instrucción findViewById(R.id.nombre de la vista)

```
<ListView
    android:id="@+id/listView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"> </ListView>
```

Explicación

Recuerda que la vista texto (anterior) se encuentra en el archivo activity_main.xml que es donde se ha definido el id para la interfaz y se manda llamar en la parte lógica a través de MainActivity.java de la siguiente forma:

```
findViewById(R.id.listView);
```

Recuerda que todos los elementos de diseño, en este caso el ListView también tiene otros atributos como layout_width y el layout_height que son obligatorios para todas las vistas.

De esta forma, añadiendo y combinando los elementos es posible formar las interfaces visuales en Android Studio.

Explicación

Una segunda forma de interactuar con el diseño es a través del código, para lo que se usan los objetos instanciados que permiten modificar los atributos de las vistas en tiempo real: que van desde el tamaño, color de fondo, diseño de los textos, etc.

Importante acerca del ancho y alto (layout_width y layout_height):

- Valor exacto (se utilizan los píxeles de dimensión) en dp: 25dp.
- Igual que el padre: match_parent. Esto tomará el tamaño del contenedor.
- Toma el espacio justo que necesites: wrap_content. El elemento medirá el alto o ancho justo para su contenido.

Explicación

2.4 Uso de la distribución de malla (GridLayout)

Recuerda que GridLayout es un ViewGroup que alinea sus elementos en una cuadrícula o grid y su uso principal uso es para evitar anidar linear layouts al crear diseños complejos.

Su funcionamiento se basa en un sistema de índices con inicio en cero. Es decir, la primera columna (o fila) tiene asignado el índice 0, la segunda el 1, la tercera el 2, etc.

Explicación

GridLayout cuenta con los siguientes atributos importantes:

- columnCount: Cantidad de columnas que tendrá el grid.
- rowCount: Cantidad de filas del grid.
- useDefaultMargins: Si asignas el valor de true para establecer márgenes predeterminadas entre los ítems.

El siguiente ejemplo de código muestra un TextView para crear 2 columnas y 3 filas.

```
<TextView
    android:id="@+id/celda_1"
    android:layout_columnSpan="2"
    android:layout_rowSpan="3"
    android:text="Celda 1" />
```

Actividad

- ¿Cuál es la diferencia entre los principales Layouts?
- ¿Qué pasaría con las aplicaciones si no se usan correctamente los Layouts?
- ¿Cómo diferenciarías su uso para aplicarlo de manera correcta de acuerdo a las necesidades de la aplicación?

Cierre

Realizar el diseño de una interfaz gráfica en una aplicación puede sentirse como algo complicado al inicio, sin embargo al practicar se volverá más sencillo de entenderlo y aplicarlo.

Lo interesante es que los Layouts es posible definirlos a través del archivo diseño o bien integrarlos como parte del código (o parte lógica).

Universidad Tecmilenio

Desarrollo de aplicaciones en plataforma Android

Eventos de Toque

Introducción

Al utilizar cualquier dispositivo móvil generalmente accedemos a las aplicaciones haciendo movimientos con los dedos ya sea hacia arriba o abajo o a un lado o hacia el otro.

Estos eventos son conocidos como eventos de toque o gestos.

Explicación

Un "gesto táctil" se produce cuando un usuario toca la pantalla táctil de un dispositivo móvil con uno o más dedos, es entonces que la aplicación interpreta ese patrón de toques como un gesto específico.

La detección de gestos consta de dos partes:

Recopilación de datos sobre eventos táctiles:

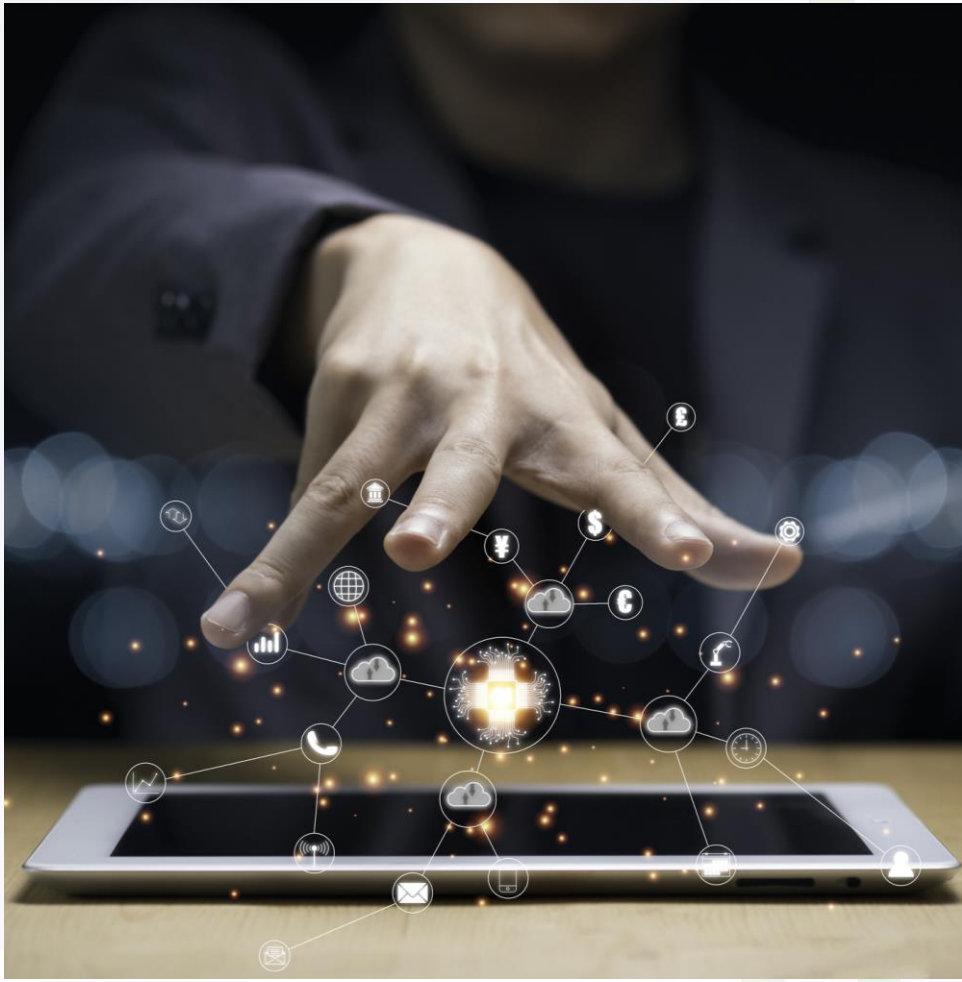
- Que es la interpretación de los datos para ver si se cumple con los criterios de cualquiera de los gestos que están disponibles en la aplicación.

Explicación

La segunda parte ocurre cuando un usuario coloca uno o más dedos en la pantalla, y se activa la devolución de llamada **onTouchEvent().**

- El gesto comienza cuando el usuario toca la pantalla por primera vez, continúa mientras el sistema busca la posición de los dedos del usuario y finaliza cuando captura el evento final de los dedos del usuario al salir de la pantalla.

A lo largo de esta interacción, el MotionEvent entregado a onTouchEvent() proporcióna los detalles de esa interacción que se da.



Explicación

¿Qué sucede cuando una aplicación usa gestos comunes?

Por ejemplo, al presionar dos veces, al mantener presionado un evento sobre la pantalla, al arrastrar y soltar con los dedos, entre otros, entra en acción la clase GestureDetector. GestureDetector que es la que facilita la detección de los gestos táctiles más comunes.

Ya vimos que al colocar los dedos sobre la pantalla se inicia un evento y entra en acción onTouchEvent(), pues como alternativa a ello se puede adjuntar un objeto View.OnTouchListener a cualquier objeto View, ya que con el método setOnTouchListener(). Esto permite escuchar eventos táctiles sin subclasificar un View existente.

```
View myView = findViewById(R.id.my_view);
myView.setOnTouchListener(new OnTouchListener() {
    public boolean onTouch(View v, MotionEvent event) {
        // ... Respond to touch events
        return true;
    }
});
```

Explicación

Para detectar los gestos mas comunes, Android proporciona la clase GestureDetector.
Entre los gestos más comunes que admite Android se encuentran: onDown(), onLongPress(), onFling(), principalmente.

Recuerda: se puede usar GestureDetector junto con el método onTouchEvent().

Al crear una instancia de un objeto GestureDetectorCompat, uno de los parámetros que toma es una clase que implementa la interfaz GestureDetector.OnGestureListener.
GestureDetector.OnGestureListener se encarga de notificar a los usuarios que ha ocurrido un evento táctil en específico.

Para permitir que un objeto GestureDetector reciba eventos, anula el método onTouchEvent() de la Vista o Activity y lo que hace es pasar todos los eventos observados a la instancia del detector.

Actividad

- ¿Cuáles son los gestos mas comunes que se pueden usar en aplicaciones de Android?
- ¿Cuáles usos crees que tengan esos eventos de toques en las aplicaciones?
- ¿Cuál es el método que permite detectar o escuchar cuándo se produce un toque?

Cierre

Realizar el diseño de una interfaz gráfica en una aplicación puede sentirse como algo complicado al inicio, sin embargo al practicar se volverá más sencillo de entenderlo y aplicarlo.

Lo interesante es que los Layouts es posible definirlos a través del archivo diseño o bien integrarlos como parte del código (o parte lógica).

Universidad Tecmilenio

Desarrollo de aplicaciones en plataforma Android

Eventos Fragmentos, flujo maestro-detalle y menú

Introducción

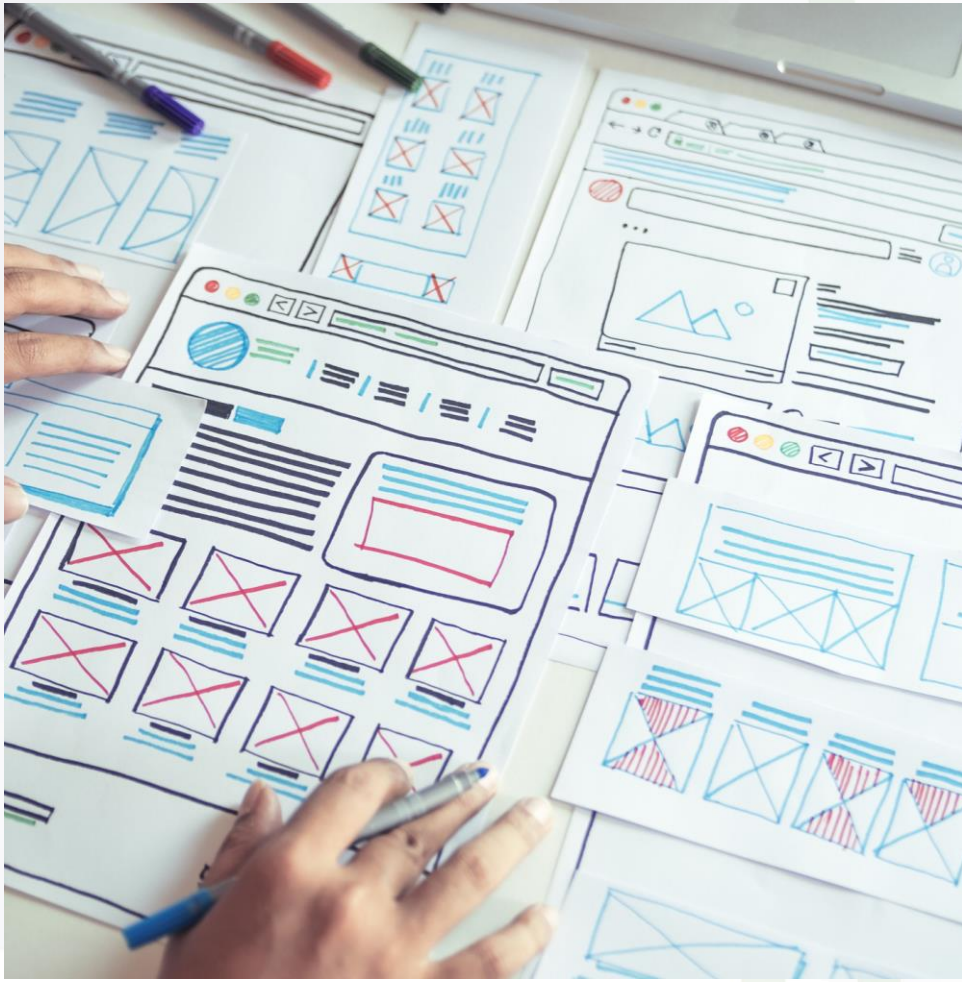
La historia cuenta que conforme fueron apareciendo los dispositivos con pantalla más amplia como la Tablet, hubo necesidad de que Android solucionara la situación de adaptar la interfaz gráfica a las aplicaciones con esa nueva pantalla.

Ya que, como te imaginas, la interfaz gráfica de un teléfono móvil no se adaptaba con facilidad a la pantalla de la Tablet por su tamaño. Para ello se pensó en una nueva forma de adaptarlo , logrando un nuevo componente que es el Fragment.

Explicación

Un *fragment* se define como una porción de la interfaz de usuario la cual permite agregarse o eliminarse de la interfaz de forma independiente al resto de los elementos de la actividad, y que también permite reutilizarse para otras actividades.

Su ventaja es que permite dividir la interfaz en varias porciones de forma que se pueda diseñar diversas configuraciones de pantalla, dependiendo de su tamaño y orientación, y lo más interesante es que es sin tener que duplicar código, sino a través de usar o no los distintos fragmentos de acuerdo con lo configurado por el programador.



Explicación

Recuerda que el *fragment* representa un comportamiento o una parte de la interfaz de usuario en la FragmenActivity.

También es importante destacar que un fragmento siempre debe estar alojado en una actividad y el ciclo de vida del fragmento se ve afectado directamente por el ciclo de vida de la actividad anfitriona.

Explicación

Para crear un fragmento, es necesario crear una subclase ***Fragment***, o subclase dependiente de ella).

La clase *Fragment* cuenta con un código muy semejante a una Activity. Contiene métodos de devolución de llamada similares a la Activity como son: onCreate(), onStart(), onPause() y onStop().

* Recomendación si debe convertir una aplicación de Android existente para utilizar fragmentos, lo único que se requiere es simplemente trasladar código de los métodos de devolución de llamada de tu actividad a los métodos respectivos del fragmento que estés creando.

Explicación

Entre las principales subclases más destacadas de la clase *Fragment* están:

DialogFragment. Es una subclase resulta buena alternativa cuando se quieren usar métodos del asistente de diálogos de la Activity, ya que le permite al usuario volver a un fragmet que haya descartado.

ListFragment. Lo interesante es que proporciona varios métodos para administrar una vista de lista, como la devolución de llamada onListItemClick() para manipular eventos al dar clic.

Actividad

Supón la siguiente situación:

Una aplicación de correo electrónico que debe mostrar la lista de correos disponibles, con sus campos: De y Asunto, y mostrar el contenido completo del correo seleccionado.

En un celular lo habitual es tener la pantalla que muestre el listado de correos, y al seleccionar uno por el usuario, se pueda navegar a una nueva actividad que muestre el contenido del correo.

Pero una Tablet cuenta con espacio suficiente para tener ambas partes de la interfaz en la misma pantalla, por ejemplo en una tablet en posición horizontal podríamos tener una columna a la izquierda con el listado de correos y dedicar la zona derecha a mostrar el detalle del correo seleccionado, todo ello sin tener que cambiar de actividad.

- Explica: ¿cómo se podría diseñar para una Tablet en posición horizontal?
- ¿Dónde ubicarías el listado de correos y mostrar el contenido del correo?
- Como programador ¿cómo implementarías los fragments?

Cierre

Ahora conoces los secretos para implementar fragments en una aplicación y sobre todo que comprendes por qué y para qué fueron creados. Así que si se te presenta una aplicación ya podrás decidir si hacerla en una actividad o unir esa actividad a través de diversos fragmentos con diseño que además serán visualmente más atractivos al usuario.