### 

|  |
| --- |
| NOMBRE Y ACTIVIDAD A REALIZAR |
| **Erick [1,2,3,\_,5]**  **Diego 7** |

### 

|  |
| --- |
| REFERENCIAS ( AQUÍ ) |
| 1-https://developer.android.com/guide/components/intents-filters?hl=ES  2 -https://developer.android.com/guide/components/intents-filters?hl=ES  3-https://developer.android.com/guide/components/intents-filters?hl=ES  4-https://developer.android.com/guide/components/intents-filters?hl=ES  5-<https://developer.android.com/guide/components/intents-filters?hl=ES>  6.<https://developer.android.com/guide/components/intents-filters?hl=ES>  7.- <http://www.sgoliver.net/blog/notificaciones-en-android-iii-dialogos/>  9.- <http://www.sgoliver.net/blog/notificaciones-en-android-iii-dialogos/>  10. - <http://www.sgoliver.net/blog/notificaciones-en-android-iii-dialogos/>  11 <https://developer.android.com/guide/topics/ui/dialogs?hl=ES>  12  13 <https://developer.android.com/guide/topics/ui/menus?hl=es-419>  14 <https://developer.android.com/guide/topics/ui/menus?hl=es-419>  15 <https://developer.android.com/guide/topics/ui/menus?hl=es-419>  16 <https://androidstudiofaqs.com/tutoriales/como-cambiar-la-orientacion-de-una-aplicacion-en-android>  17  18  19 <https://developer.android.com/studio/projects/templates?hl=es>  20<https://codelabs.developers.google.com/codelabs/material-design-style-sp/index.html?index=..%2F..%2Findex#0>  21<https://codelabs.developers.google.com/codelabs/material-design-style-sp/index.html?index=..%2F..%2Findex#0> |

-

### 

### 1. ¿Qué es un Intent en Android?

Una [Intent](https://developer.android.com/reference/android/content/Intent.html?hl=ES) es un objeto de acción que puedes usar para solicitar una acción de otro [componente de la aplicación](https://developer.android.com/guide/components/fundamentals.html?hl=ES#Components).

### 2. ¿Qué tipos de Intents existen?

Existen dos tipos de intents:

* Intents explícitas
* Intents implícitas

### 3. Describe detalladamente cada tipo de Intent.

* Intents explícitas: especifican qué componente se debe iniciar mediante su nombre (el nombre de clase completamente calificado). Usualmente, el usuario usa una intent explícita para iniciar un componente en su propia aplicación porque conoce el nombre de clase de la actividad o el servicio que desea iniciar. Por ejemplo, puede utilizarla para iniciar una actividad nueva en respuesta a una acción del usuario o iniciar un servicio para descargar un archivo en segundo plano.
* Intents implícitas: no se nombra el componente específicamente; pero, en cambio, se declara una acción general para realizar, lo que permite que un componente de otra aplicación la maneje. Por ejemplo, si desea mostrar al usuario una ubicación en un mapa, puede usar una intent implícita para solicitar que otra aplicación capaz muestre una ubicación específica en un mapa.

### 4. ¿Cuál es la diferencia entre los tipos de Intents?

Un intent explícito es más comúnmente utilizado en el [**lanzamiento de una actividad**](http://www.codigojavalibre.com/2015/11/abrir-otro-activity-usando-intent-en-android-studio.html), dentro de la misma aplicación.

Un intent implícito especifica una acción que puede invocar a cualquier aplicación en el dispositivo capaz de realizar la acción. Usando un intent implícito es útil cuando su aplicación no puede realizar la acción, pero otras aplicaciones, probablemente, puede y desea que el usuario elija qué aplicación utilizar para realizar esta acción.

http://www.codigojavalibre.com/2016/01/explicacion-de-los-intents-en-android.html

### 5. ¿En qué ocasiones usarías un Intent?

casos de uso fundamentales:

* Para comenzar una actividad:

Una RSS esZsz representa una única pantalla en una aplicación. Puedes iniciar una nueva instancia de una [Activity](https://developer.android.com/reference/android/app/Activity.html?hl=ES) pasando una [Intent](https://developer.android.com/reference/android/content/Intent.html?hl=ES) a [startActivity()](https://developer.android.com/reference/android/content/Context.html?hl=ES#startActivity(android.content.Intent)). La [Intent](https://developer.android.com/reference/android/content/Intent.html?hl=ES) describe la actividad que se debe iniciar y contiene los datos necesarios para ello.

Si deseas recibir un resultado de la actividad cuando finalice, llama a [startActivityForResult()](https://developer.android.com/reference/android/app/Activity.html?hl=ES#startActivityForResult(android.content.Intent,%20int)). La actividad recibe el resultado como un objeto [Intent](https://developer.android.com/reference/android/content/Intent.html?hl=ES) separado en el callback de [onActivityResult()](https://developer.android.com/reference/android/app/Activity.html?hl=ES#onActivityResult(int,%20int,%20android.content.Intent)) de la actividad. Para obtener más información, consulta la guía [Actividades](https://developer.android.com/guide/components/activities.html?hl=ES).

* Para iniciar un servicio:

Un [Service](https://developer.android.com/reference/android/app/Service.html?hl=ES) es un componente que realiza operaciones en segundo plano sin una interfaz de usuario. Puede iniciar un servicio para realizar una operación única (como descargar un archivo) pasando una [Intent](https://developer.android.com/reference/android/content/Intent.html?hl=ES) a [startService()](https://developer.android.com/reference/android/content/Context.html?hl=ES#startService(android.content.Intent)). La [Intent](https://developer.android.com/reference/android/content/Intent.html?hl=ES) describe el servicio que se debe iniciar y contiene los datos necesarios para ello.

Si el servicio está diseñado con una interfaz cliente-servidor, puedes establecer un enlace con el servicio de otro componente pasando una [Intent](https://developer.android.com/reference/android/content/Intent.html?hl=ES) a [bindService()](https://developer.android.com/reference/android/content/Context.html?hl=ES#bindService(android.content.Intent,%20android.content.ServiceConnection,%20int)). Para obtener más información, consulte la guía [Servicios](https://developer.android.com/guide/components/services.html?hl=ES).

* Para entregar un mensaje:

Un mensaje es un aviso que cualquier aplicación puede recibir. El sistema entrega varios mensajes de eventos del sistema, como cuando el sistema arranca o el dispositivo comienza a cargarse. Puedes enviar un mensaje a otras apps pasando una [Intent](https://developer.android.com/reference/android/content/Intent.html?hl=ES) a [sendBroadcast()](https://developer.android.com/reference/android/content/Context.html?hl=ES#sendBroadcast(android.content.Intent)), [sendOrderedBroadcast()](https://developer.android.com/reference/android/content/Context.html?hl=ES#sendOrderedBroadcast(android.content.Intent,%20java.lang.String)) o [sendStickyBroadcast()](https://developer.android.com/reference/android/content/Context.html?hl=ES#sendStickyBroadcast(android.content.Intent)).

### 6. ¿Cómo creas un objeto del tipo Intent?

<https://developer.android.com/guide/components/intents-filters?hl=ES>

La información principal que contiene una Intent es la siguiente:

· Nombre del componente: El nombre del componente que se debe iniciar.

· Acción: Una string que especifica la acción genérica que se debe realizar (como ver o elegir).

· Datos: El URI que hace referencia a los datos en los que se debe realizar la acción o el tipo de MIME de esos datos.

· Categoría: Una string que contiene información adicional sobre el tipo de componente que la intent debe manejar.

Ejemplo:

Intent downloadIntent = new Intent(this, DownloadService.class);

downloadIntent.setData([**Uri.parse**](https://developer.android.com/reference/android/net/Uri.html?hl=ES#parse(java.lang.String))(fileUrl));

startService(downloadIntent);

### 7. ¿Qué es una alerta en Android?

Es un diálogo que se limita a mostrar un mensaje sencillo al usuario, y un único botón de OK para confirmar su lectura.

Extra:

Lo construiremos mediante la clase AlertDialog, y más concretamente su subclase AlertDialog.Builder,

### 8. ¿Cuáles son los tipos de alertas en Android?

<http://www.hermosaprogramacion.com/2015/06/como-crear-dialogos-en-android/>

O no tiene que ver con esto?  
<https://www.nosinmiubuntu.com/alertas-y-notificaciones-en-android/>

### 

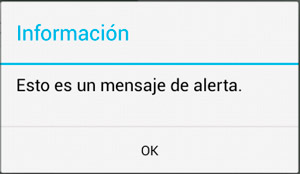
### 9. Describe detalladamente cada tipo de alertas:

**Diálogo de Alerta**

Se limita a mostrar un mensaje sencillo al usuario con un único botón de OK para confirmar su lectura. Se construye mediante la clase AlertDialog.Builder. Al crear el objeto se establecen las propiedades del diálogo mediante sus métodos correspondientes: título [setTitle()], mensaje [setMessage()] y el texto y comportamiento del botón [setPositiveButton()].

//Ejemplo:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  1 | public class DialogoAlerta extends DialogFragment {  @Override  public Dialog onCreateDialog(Bundle savedInstanceState) {    AlertDialog.Builder builder =  new AlertDialog.Builder(getActivity());    builder.setMessage("Esto es un mensaje de alerta.")  .setTitle("Información")  .setPositiveButton("OK", new DialogInterface.OnClickListener() {  public void onClick(DialogInterface dialog, int id) {  dialog.cancel();  }  });    return builder.create();  }  } |

****

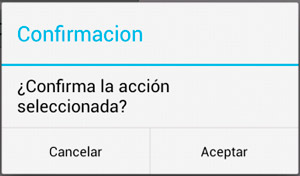
**Diálogo de Confirmación**

Es muy similar al anterior, pero es posibles dar respuestas del tipo Sí/No.

Su implementación es prácticamente igual a la anterior, salvo que añadiremos dos botones, uno para la respuesta afirmativa (setPositiveButton()), y otro para la respuesta negativa (setNegativeButton()).

//Ejemplo:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | public class DialogoConfirmacion extends DialogFragment {  @Override  public Dialog onCreateDialog(Bundle savedInstanceState) {    AlertDialog.Builder builder =  new AlertDialog.Builder(getActivity());    builder.setMessage("¿Confirma la acción seleccionada?")  .setTitle("Confirmacion")  .setPositiveButton("Aceptar", new DialogInterface.OnClickListener() {  public void onClick(DialogInterface dialog, int id) {  Log.i("Dialogos", "Confirmacion Aceptada.");  dialog.cancel();  }  })  .setNegativeButton("Cancelar", new DialogInterface.OnClickListener() {  public void onClick(DialogInterface dialog, int id) {  Log.i("Dialogos", "Confirmacion Cancelada.");  dialog.cancel();  }  });    return builder.create();  }  } |



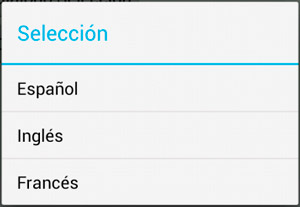
**Diálogo de Selección**

Se usa para mostrar una lista de opciones entre las que el usuario pueda elegir.

También se utiliza la clase AlertDialog, pero ya no se definen mensajes ni acciones a realizar por cada botón; directamente se indica la lista de opciones a mostrar (mediante el método setItems()) y se proporciona la implementación del evento onClick() sobre dicha lista (mediante un listener de tipo DialogInterface.OnClickListener), evento en el que se realizan las acciones según la opción. La lista de opciones se define como un array tradicional.

//Ejemplo:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | public class DialogoSeleccion extends DialogFragment {  @Override  public Dialog onCreateDialog(Bundle savedInstanceState) {    final String[] items = {"Español", "Inglés", "Francés"};    AlertDialog.Builder builder =  new AlertDialog.Builder(getActivity());    builder.setTitle("Selección")  .setItems(items, new DialogInterface.OnClickListener() {  public void onClick(DialogInterface dialog, int item) {  Log.i("Dialogos", "Opción elegida: " + items[item]);  }  });    return builder.create();  }  } |



Para visualizar en pantalla la opción u opciones seleccionadas por el usuario podemos utilizar los métodos setSingleChoiceItems() o setMultiChiceItems(), en vez de el setItems() utilizado anteriormente. La diferencia entre ambos métodos, tal como se puede suponer por su nombre, es que el primero de ellos permitirá una selección simple y el segundo una selección múltiple, es decir, de varias opciones al mismo tiempo, mediante controles CheckBox.

Para utilizarlos, en el caso de setSingleChoiceItems(), sólo se diferencia de setItems() en que recibe un segundo parámetro adicional que indica el índice de la opción marcada por defecto. Si no se quiere tener ninguna marcadas inicialmente se pasa el valor -1.

//Ejemplo:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | builder.setTitle("Selección")  .setSingleChoiceItems(items, -1,  new DialogInterface.OnClickListener() {  public void onClick(DialogInterface dialog, int item) {  Log.i("Dialogos", "Opción elegida: " + items[item]);  }  }); |

Se conseguiría un diálogo como el de la siguiente imagen:



Si se opta por la selección múltiple, se tendrá que implementar un listener del tipo DialogInterface.OnMultiChoiceClickListener. En el evento onClick se recibe tanto la opción seleccionada (item) como el estado en el que ha quedado (isChecked). El segundo parámetro adicional que indica el estado por defecto de las opciones ya no será un simple número entero, sino que tendrá que ser un array de booleanos. En caso de no querer ninguna opción seleccionada por defecto pasaremos el valor null.

//Ejemplo:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | builder.setTitle("Selección")  .setMultiChoiceItems(items, null,  new DialogInterface.OnMultiChoiceClickListener() {  public void onClick(DialogInterface dialog, int item, boolean isChecked) {  Log.i("Dialogos", "Opción elegida: " + items[item]);  }  }); |

Y el diálogo nos quedaría de la siguiente forma:



Tanto si utilizamos la opción de selección simple como la de selección múltiple, para salir del diálogo tendremos que pulsar la tecla “*Atrás*” de nuestro dispositivo.

http://www.sgoliver.net/blog/notificaciones-en-android-iii-dialogos/

### 

### 10. ¿Qué elementos se le pueden agregar a una alerta y cómo?

**Diálogos Personalizados**

Para establecer completamente el aspecto de un cuadro de diálogo es como si estuviéramos definiendo la interfaz de una actividad, es decir, definiremos un layout XML con los elementos a mostrar en el diálogo. Por ejemplo dialog\_personal.xml que se colocará en la carpeta res/layout. Contendrá por ejemplo una imagen a la izquierda y dos líneas de texto a la derecha.

//Ejemplo

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | <LinearLayout xmlns:android="<http://schemas.android.com/apk/res/android>"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:orientation="horizontal"  android:padding="3dp" >    <ImageView  android:id="@+id/imageView1"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:src="@drawable/ic\_launcher" />    <LinearLayout  android:layout\_width="wrap\_content"  android:layout\_height="match\_parent"  android:orientation="vertical"  android:padding="3dp">    <TextView  android:id="@+id/textView1"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="@string/dialogo\_linea\_1" />    <TextView  android:id="@+id/textView2"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="@string/dialogo\_linea\_2" />    </LinearLayout>    </LinearLayout> |

Por su parte, en el método onCreateDialog() correspondiente utilizaremos el método setView() del builder para asociarle nuestro layout personalizado, que previamente tendremos que inflar como ya hemos comentado otras veces utilizando el método inflate(). Finalmente podremos incluir botones tal como vimos para los diálogos de alerta o confirmación. En este caso de ejemplo incluiremos un botón de *Aceptar*.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | public class DialogoPersonalizado extends DialogFragment {  @Override  public Dialog onCreateDialog(Bundle savedInstanceState) {    AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());  LayoutInflater inflater = getActivity().getLayoutInflater();    builder.setView(inflater.inflate(R.layout.dialog\_personal, null))  .setPositiveButton("Aceptar", new DialogInterface.OnClickListener() {  public void onClick(DialogInterface dialog, int id) {  dialog.cancel();  }  });    return builder.create();  }  } |

Así, al ejecutar la aplicación de ejemplo y lanzar el diálogo personalizado se verá lo siguiente:



<http://www.sgoliver.net/blog/notificaciones-en-android-iii-dialogos/>

### 

### 11. ¿Qué es un diálogo en Android?

### Un diálogo es una ventana pequeña que le indica al usuario que debe tomar una decisión o ingresar información adicional.

<https://developer.android.com/guide/topics/ui/dialogs?hl=ES>

### 12. ¿En qué maneras puede interactuar un diálogo con el usuario?

### 

### 13. ¿Qué es un menú en Android?

### presentar al usuario acciones y otras opciones en las actividades.

<https://developer.android.com/guide/topics/ui/menus?hl=es-419>

### 14. ¿Qué tipos de menús hay en Android?

### Menú de opciones y barra de app

### El [menú de opciones](https://developer.android.com/guide/topics/ui/menus?hl=es-419#options-menu) es la colección principal de elementos de menú para una actividad. Es donde debes colocar las acciones que tienen un impacto global en la app, como “Buscar”, “Redactar correo electrónico” y “Ajustes”.

### Menú contextual y modo de acción contextual

### Un menú contextual es un [menú flotante](https://developer.android.com/guide/topics/ui/menus?hl=es-419#FloatingContextMenu) que aparece cuando el usuario hace un clic largo en un elemento. Proporciona acciones que afectan el contenido seleccionado o el marco contextual.

### Menú emergente

### Un menú emergente muestra una lista de elementos en una lista vertical que está anclada a la vista que invocó el menú. Es adecuado para proporcionar una ampliación de acciones relacionadas con contenido específico o para proporcionar opciones para una segunda parte de un comando.

<https://developer.android.com/guide/topics/ui/menus?hl=es-419>

### 15. Estructura o componentes de un menú.

### 

Menu, items, grupo

<https://developer.android.com/guide/topics/ui/menus?hl=es-419>

<menu>

Define un [Menu](https://developer.android.com/reference/android/view/Menu.html?hl=es-419), que es un contenedor para elementos del menú. Un elemento <menu> debe ser el nodo raíz del archivo y puede tener uno o más elementos <item> y <group>.

<item>

Crea un [MenuItem](https://developer.android.com/reference/android/view/MenuItem.html?hl=es-419), que representa un único elemento en un menú. Este elemento puede contener un elemento <menu> anidado para crear un submenú.

<group>

Contenedor opcional e invisible para elementos <item>. Te permite categorizar los elementos del menú para que compartan propiedades, como el estado de una actividad o visibilidad. Para obtener más información, consulta la sección [Creación de grupos del menú](https://developer.android.com/guide/topics/ui/menus?hl=es-419#groups).

### 16. ¿Qué se necesita para dar orientación a la pantalla de la aplicación en Android?

Android Manifest

android:screenOrientation="portrait"

android:screenOrientation="landscape">

<https://androidstudiofaqs.com/tutoriales/como-cambiar-la-orientacion-de-una-aplicacion-en-android>

### 

### 17. ¿Qué se hace en los casos en el que se necesita portabilidad entre equipos?

### 

### 18. ¿Se puede dar un diseño ya formado en Android Studio?

### 

### Si, existen plantillas

### 19. ¿Cómo se puede usar una plantilla en Android Studio?

<https://developer.android.com/studio/projects/templates?hl=es>

### 

### 20. ¿Qué es Material Design y cómo puede usarse en Android?

<https://codelabs.developers.google.com/codelabs/material-design-style-sp/index.html?index=..%2F..%2Findex#0>

[Material Design en la aplicación Google I/O](https://www.youtube.com/watch?v=XOcCOBe8PTc&hl=es)

Material Design es una guía integral para el diseño visual, de movimientos y de interacción en distintas plataformas y dispositivos. Android ahora es compatible con las aplicaciones de Material Design. Para usar Material Design en tus aplicaciones de Android, sigue las pautas descritas en la [especificación de Material Design](http://www.google.com/design/spec?hl=es) y usa los nuevos componentes y funcionalidades disponibles en Android 5.0 (nivel de API 21) y versiones posteriores.

En Android, se proporcionan los siguientes elementos que te permitirán crear aplicaciones en Material Design:

* un tema nuevo;
* nuevos widgets para vistas complejas;
* nuevas API (interfaces de programación de aplicaciones) para sombras y animaciones personalizadas.

Para obtener más información sobre la implementación de Material Design en Android, consulta [Creación de aplicaciones con Material Design](https://developer.android.com/training/material/index.html?hl=es).

### 

### 21. ¿Qué se necesita para usar Material Design en Android?

<https://codelabs.developers.google.com/codelabs/material-design-style-sp/index.html?index=..%2F..%2Findex#0>

### Requerimientos

* B[iblioteca Android Design Support](http://android-developers.blogspot.co.ke/2015/05/android-design-support-library.html).
* [Android Studio](https://developer.android.com/sdk/installing/studio.html) versión 1.4+ y [JDK 7+](http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html)
* Experiencia desarrollando apps Android
* Un dispositivo de prueba con Android 4.1+\*
* Un cable USB micro a USB.