



ANDROID PARA INGENIEROS

2ª EDICIÓN

2ª SESIÓN

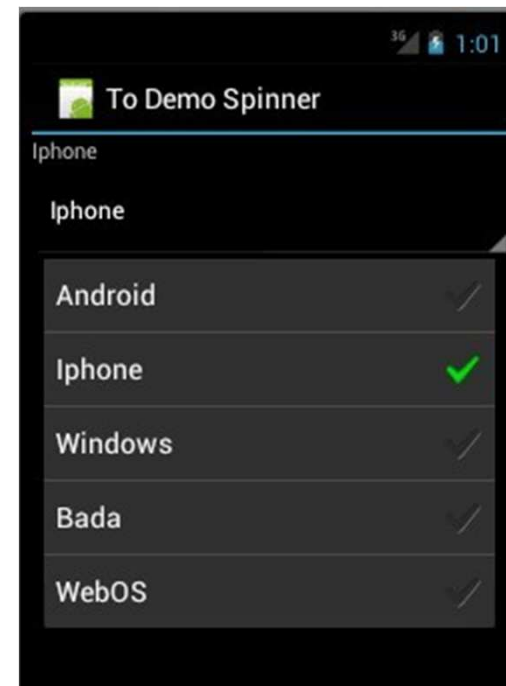
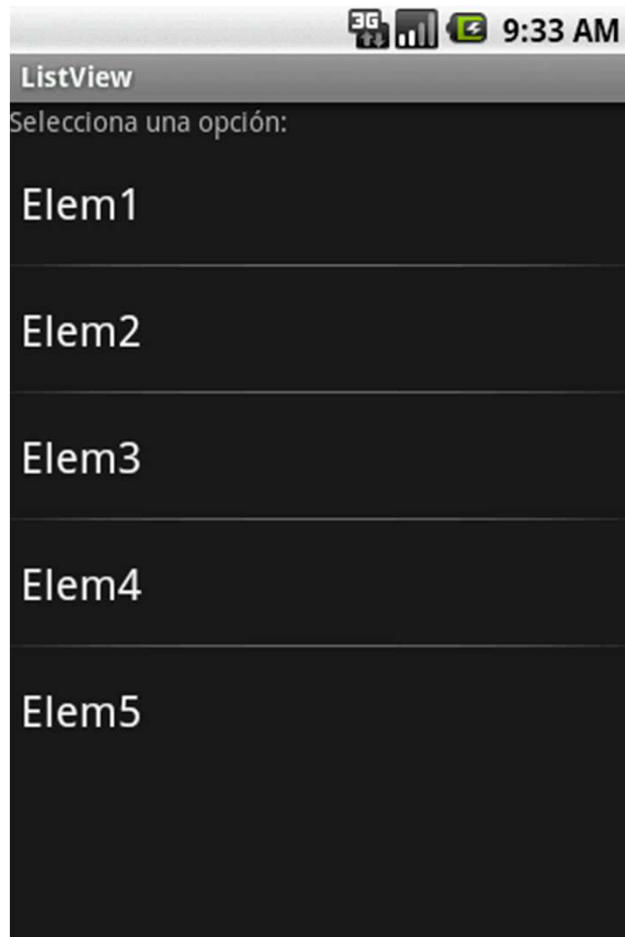
1

Curso 2015-2016

ÍNDICE

1. Listas
2. Radio Button
3. Múltiples ventanas
4. Notificaciones
5. Tiempo

1- LISTAS



1- LISTAS

- Hay dos tipos de listas importantes en Android que vamos a ver: «ListView» and «Spinner».
- El «ListView» es un componente que visualiza una lista deslizable verticalmente de varios elementos.
- Cada elementos de la lista puede ser definido por su propio «layout» o podemos usar algunos ya predefinidos.

1- LISTAS

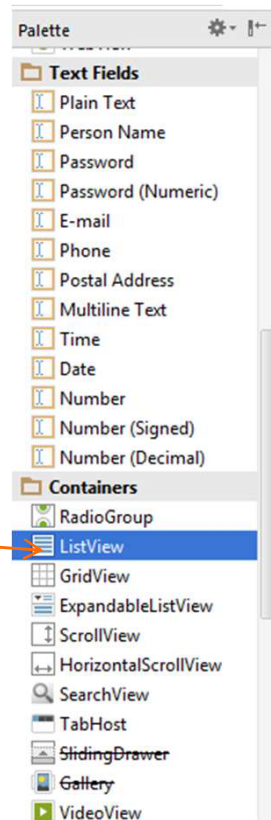
- Para simplificar un poco este componente vamos a usar los «layout» que viene predefinidos en Android.
- Si queremos una apariencia personalizada para los elementos de nuestra lista deberemos crear nuestros propios «layout».
- Este componente tiene un tamaño estático, es decir, no podemos aumentar o disminuir su tamaño mientras estamos ejecutando nuestra app.

1- LISTAS

- El componente «ListView» tiene cuatro partes importantes:
 - Crear un lista con los elementos que contendrá nuestro componente.
 - Crear un adaptador que conectará nuestro componente con la lista de elementos.
 - Configuramos nuestro componente
 - Finalmente creamos un escuchador para el componente.

1- LISTAS

- El componente «ListView» lo podemos encontrar aquí:



1- LISTAS

- Vamos a arrastrar el componente dentro de nuestra aplicación:



1- LISTAS

- Antes de crear la lista necesitamos declarar e inicializar este componente como hemos hecho previamente con otros componentes.

```
package com.example.isma.listas;

import ...

public class MainActivity extends AppCompatActivity {

    private ListView list_ejemplo;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        list_ejemplo = (ListView) findViewById(R.id.list_ejemplo);

        crearLista();
    }
}
```

1- LISTAS

- Vamos a ver las cuatro partes que conlleva crear una lista.
 - Crear una lista de «String» con los elementos que contendrá nuestro componente:

// 1- Lista de elementos

```
final List<String> misFrutas = new ArrayList<String>();  
misFrutas.add("Manzanas");  
misFrutas.add("Peras");  
misFrutas.add("Platanos");  
misFrutas.add("Melocotones");
```

1- LISTAS

- Vamos a crear un «Adapter» para controlar nuestro componente y el «layout» que va a usar para mostrar la lista:

// 2- Creamos un adapter

*ArrayAdapter<String> adapter = **new***

*ArrayAdapter<String>(**this**,*

*android.R.layout.**simple_list_item_1**, misFrutas);*

1- LISTAS

- Ahora configuramos nuestro componente lista para que use el «Adapter»:

// 3- Configuramos el ListView
list_ejemplo.setAdapter(adapter);

1- LISTAS

- Finalmente creamos un escuchador («Listener») para el componente lista que reaccione donde se haya pulsado y muestre la selección:

```
// 4- Creamos el escuchador para la list
list_ejemplo.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        String fruta = ((TextView) view).getText().toString();
        int seleccion = (int) id;

        Toast.makeText(MainActivity.this, "Fruta=" + fruta + ",id=" + seleccion,
        Toast.LENGTH_LONG).show();

    }
});
```

1- LISTAS

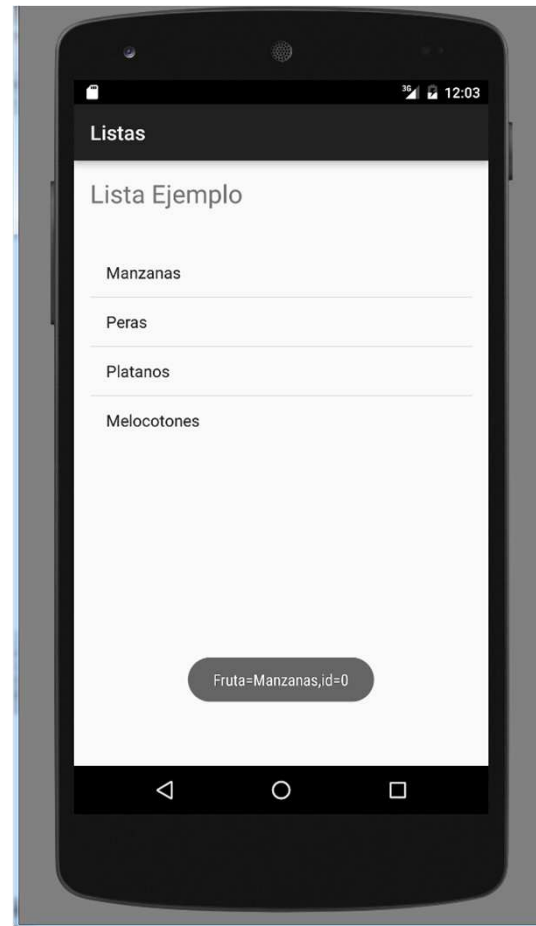
- Agrupamos estos cuatro pasos en una función: «crearLista()»

```
public void crearLista(){  
    // 1- Lista de elementos  
    final List<String> misFrutas = new ArrayList<String>();  
    misFrutas.add("Manzanas");  
    misFrutas.add("Peras");  
    misFrutas.add("Platanos");  
    misFrutas.add("Melocotones");  
  
    // 2- Creamos un adapter  
    ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, misFrutas);  
  
    // 3- Configuramos el ListView  
    list_ejemplo.setAdapter(adapter);  
  
    // 4- Creamos el escuchador para la list  
    list_ejemplo.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
        @Override  
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  
            String fruta = ((TextView) view).getText().toString();  
            int seleccion = (int) id;  
  
            Toast.makeText(MainActivity.this, "Fruta=" + fruta + ",id=" + seleccion, Toast.LENGTH_LONG).show();  
        }  
    });  
}
```

1- LISTAS

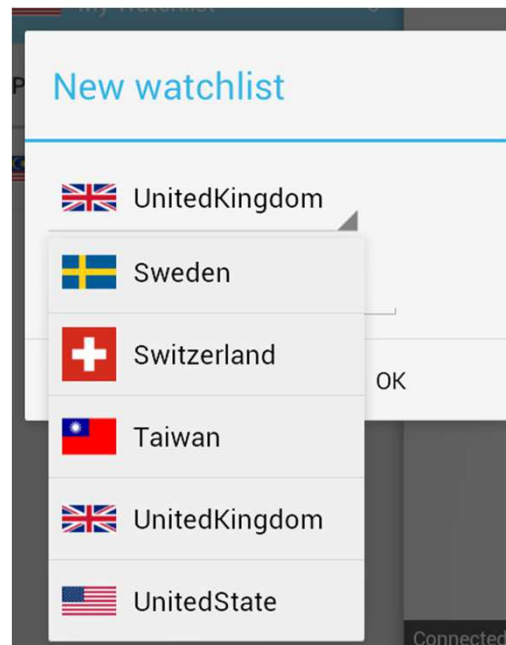
- En resumen, el método «crearLista()» realiza las siguientes acciones:
 - Crea una lista de «String» con cuatro nombres de frutas.
 - Crea un «Adapter» para nuestra lista de «String» diciéndole que tipo de «layout» utilizará.
 - Conecta el «Adapter» a nuestro componente.
 - Finalmente se crea un escuchador para mostrar el nombre de la fruta y el id que se ha seleccionado.

1- LISTAS



1- LISTAS

- Otro componente muy útil para crear lista es el componente «Spinner».

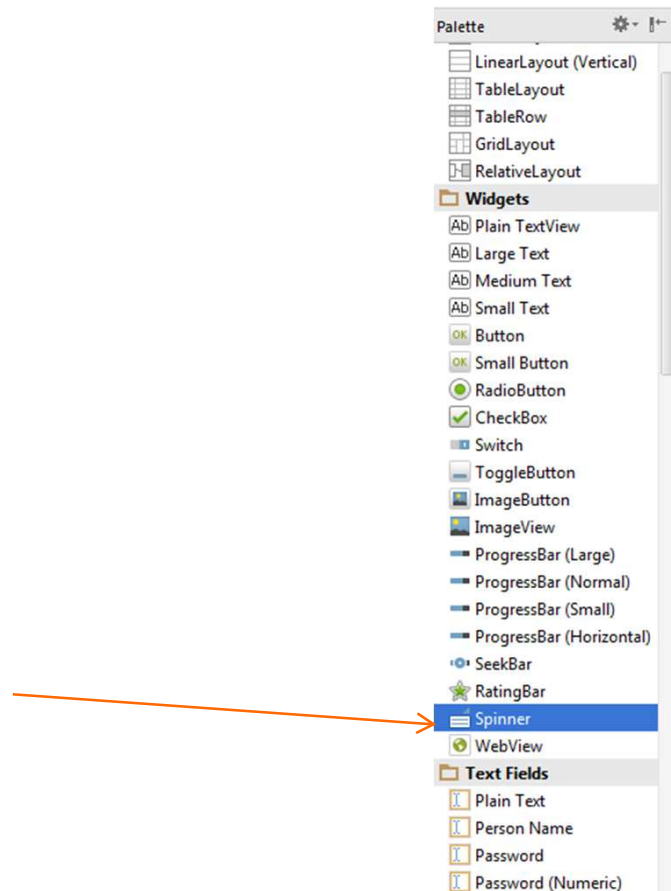


1- LISTAS

- A diferencia del «ListView», este componente se despliega con las opciones que tiene al pulsar sobre el.
- Una vez desplegado podemos deslizar sobre este componente y seleccionar el elemento que queramos.
- Este componente puede ser dinámico, es decir, se puede cambiar el contenido mientras ejecutamos nuestra app.

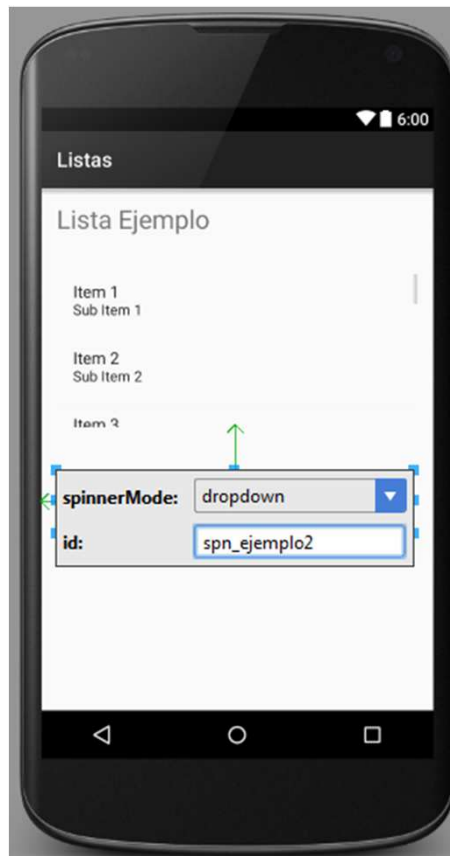
1- LISTAS

- El componente «Spinner» lo podemos encontrar aquí:



1- LISTAS

- Vamos a arrastrar el componente dentro de nuestra aplicación:



1- LISTAS

- Antes de crear la lista necesitamos declarar e inicializar este componente como hemos hecho previamente con otros componentes.

```
public class MainActivity extends AppCompatActivity {  
  
    private ListView list_ejemplo;  
    private Spinner spn_ejemplo2;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        list_ejemplo = (ListView) findViewById(R.id.list_ejemplo);  
        spn_ejemplo2 = (Spinner) findViewById(R.id.spn_ejemplo2);  
  
        crearLista();  
        crearSpinner();  
    }  
}
```

1- LISTAS

- Agrupamos estos cuatro pasos en una función: «crearSpinner()»

```
public void crearSpinner(){
    // 1- Lista de elementos
    final List<String> misFrutas2 = new ArrayList<String>();
    misFrutas2.add("Sandia");
    misFrutas2.add("Melon");
    misFrutas2.add("Uvas");
    misFrutas2.add("Manzana");

    // 2- Creamos un adapter
    ArrayAdapter<String> adapter2 = new ArrayAdapter<String>(this, android.R.layout.simple_spinner_dropdown_item, misFrutas2);

    // 3- Configuramos el ListView
    spn_ejemplo2.setAdapter(adapter2);

    // 4- Creamos el escuchador para la list
    spn_ejemplo2.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
            String fruta = ((TextView) view).getText().toString();
            int seleccion = (int) id;

            Toast.makeText(MainActivity.this, "Fruta=" + fruta + ",id=" + seleccion, Toast.LENGTH_LONG).show();
        }

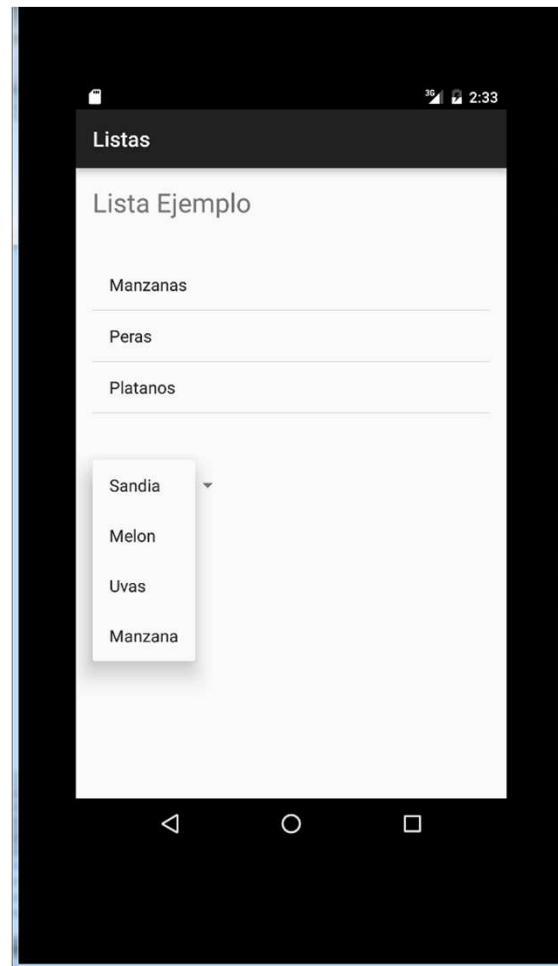
        @Override
        public void onNothingSelected(AdapterView<?> parent) {
        }
    });
}
```

1- LISTAS

- El procedimiento para generar y usar un «Spinner» es muy parecido al «ListView» salvo que el método escuchador es diferente:

```
spn_ejemplo2.setOnItemSelectedListener(new  
    AdapterView.OnItemSelectedListener() {  
        ...
```

1- LISTAS



EJERCICIO

- Modificar el ejercicio de conversor de unidades para añadir más conversiones. Añadir las conversiones horas → segundos y viceversa.
 - Para ello debemos eliminar el componente «CheckBox» y añadir una lista con las conversiones posibles.
 - Dependiendo del tipo de conversión cambie en nombre de las magnitudes de entrada y salida, y realice correctamente la operación que corresponda.

2- RADIO BUTTON



2- RADIO BUTTON

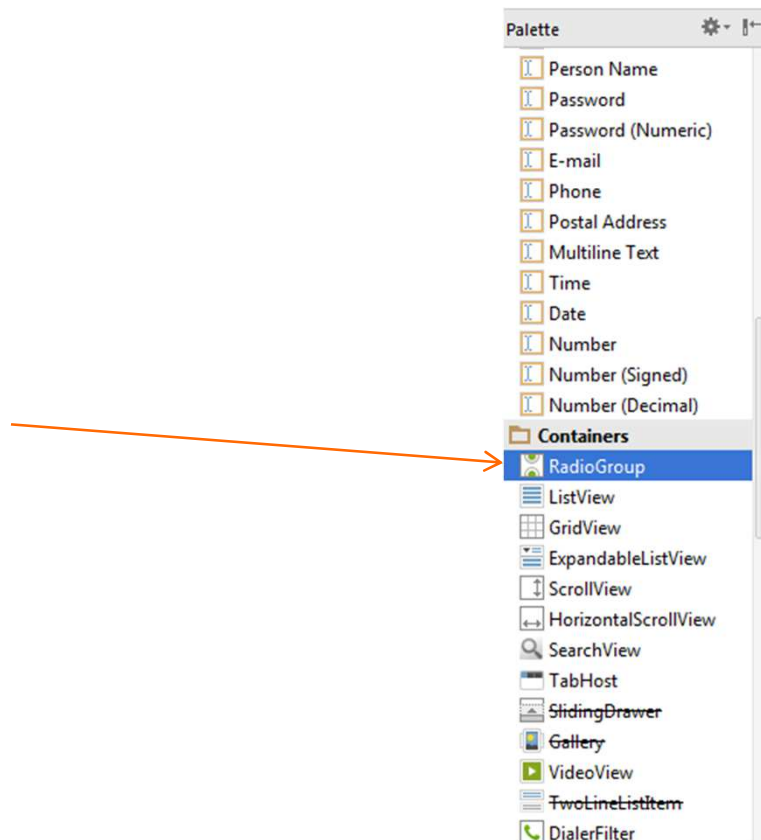
- El componente «RadioButton» permite al usuario seleccionar una opción entre varias.
- Es útil cuando queremos que el usuario elija solo una opción.
- Para usar este componente debemos crear antes el componente «RadioGroup»

2- RADIO BUTTON

- Todos los «RadioButton» que estén dentro del mismo «RadioButton» pertenecerán al mismo grupo donde el usuario solo podrá seleccionar una opción a la vez.
- Se puede detectar si el «RatioButton» está marcada con la función **isChecked()**.
- Además se puede crear un escuchador («Listener») para ejecutar código cuando es pulsado.

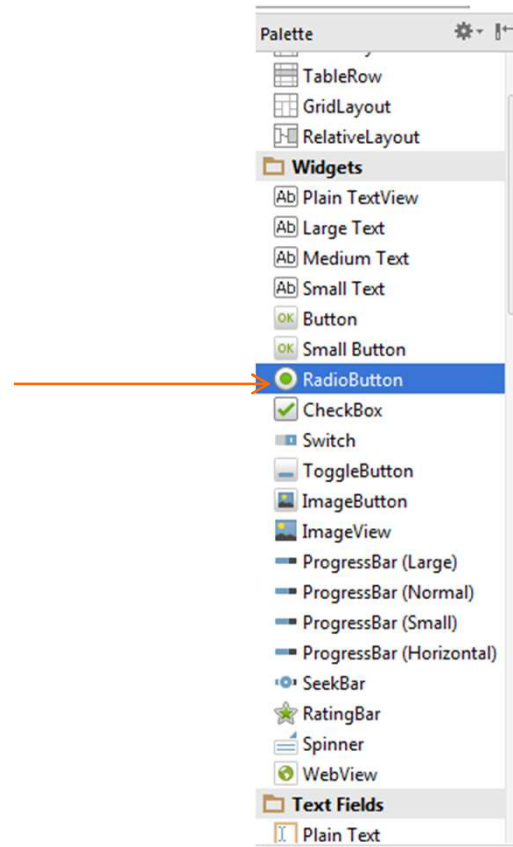
2- RADIO BUTTON

- El componente «RadioGroup» lo podemos encontrar aquí:



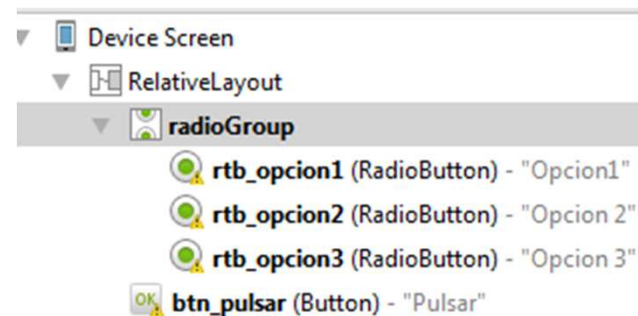
2- RADIO BUTTON

- El componente «RadioButton» lo podemos encontrar aquí:



2- RADIO BUTTON

- Vamos a arrastrar un «RadioGroup» y tres «RadioButton» dentro de nuestra aplicación:
- También hemos añadido un botón.



2- RADIO BUTTON

- Vamos a definir y declarar los tres «RadioButton», también el botón, como hemos hecho otras veces:

```
public class MainActivity extends AppCompatActivity {  
  
    private RadioButton rtb_opcion1, rtb_opcion2, rtb_opcion3;  
    private Button btn_pulsar;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        rtb_opcion1 = (RadioButton) findViewById(R.id.rtb_opcion1);  
        rtb_opcion2 = (RadioButton) findViewById(R.id.rtb_opcion2);  
        rtb_opcion3 = (RadioButton) findViewById(R.id.rtb_opcion3);  
        btn_pulsar = (Button) findViewById(R.id.btn_pulsar);  
    }  
}
```


2- RADIO BUTTON

- Vamos a ver la opción de crear un escuchador para el primer «RadioButton»:

```
rtb_opcion1.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        Toast.makeText(MainActivity.this, "Opción 1", Toast.LENGTH_LONG).show();  
    }  
});
```

2- RADIO BUTTON

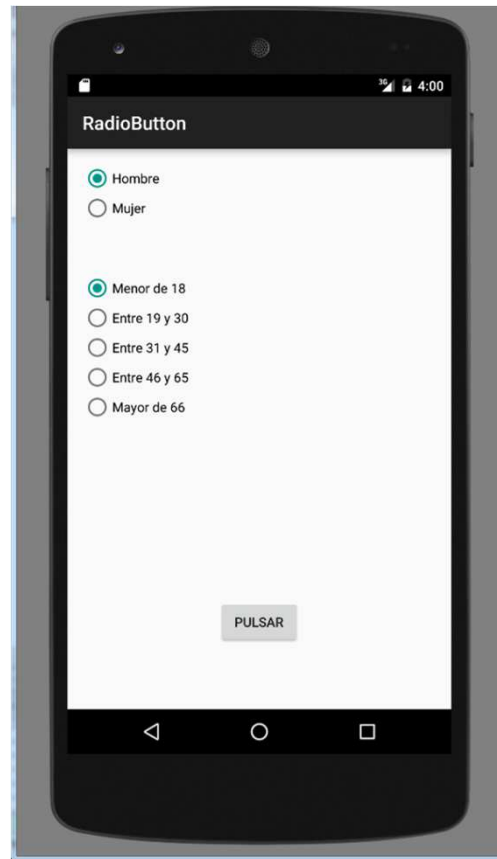
- Y finalmente usando el botón, comprobar que «RadioButton» está marcado y decirlo:

```
btn_pulsar.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        if (rtb_opcion1.isChecked())  
            Toast.makeText(MainActivity.this, "Esta pulsada la opción 1", Toast.LENGTH_LONG).show();  
  
        if (rtb_opcion2.isChecked())  
            Toast.makeText(MainActivity.this, "Esta pulsada la opción 2", Toast.LENGTH_LONG).show();  
  
        if (rtb_opcion3.isChecked())  
            Toast.makeText(MainActivity.this, "Esta pulsada la opción 3", Toast.LENGTH_LONG).show();  
    }  
});
```

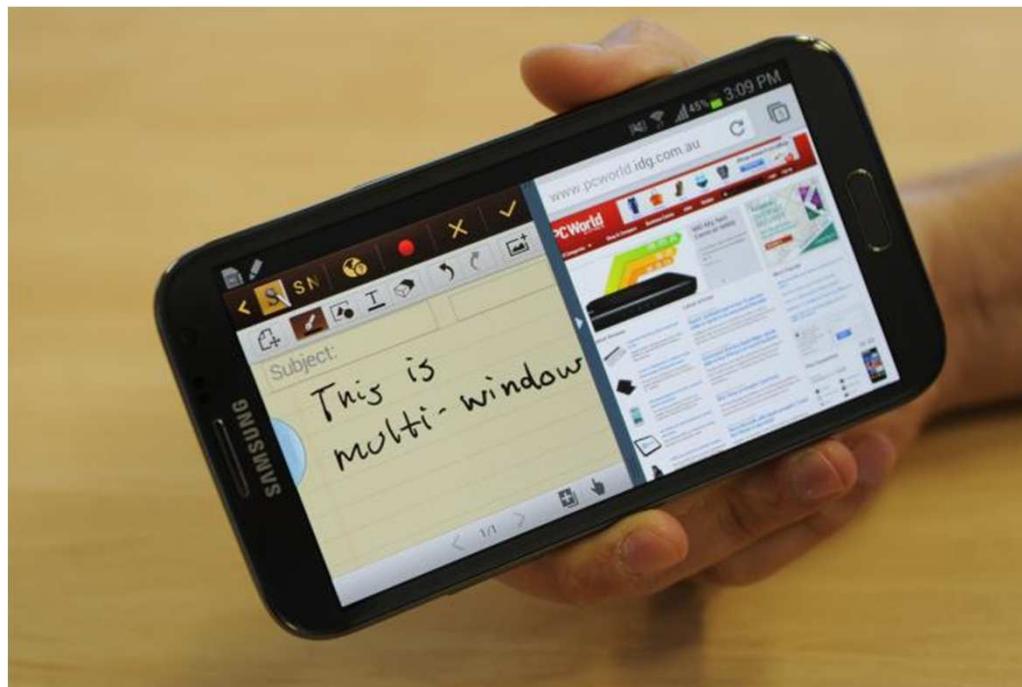
EJERCICIO

- Crear una aplicación que nos permite seleccionar entre hombre o mujer. Además nos permita seleccionar la edad entre estos cinco grupos: menor de 18, 19-30, 31-45, 46-65 y mayor de 66.
 - Usar «RadioButton» para seleccionar.
 - Utilizar un botón «Calcular» que nos muestre por la pantalla de nuestro dispositivo si es hombre/mujer y el grupo de edad a la que pertenece.
 - Ver un ejemplo de la interfaz (siguiente transparencia)

EJERCICIO



3- MÚLTIPLES VENTANAS



3- MÚLTIPLES VENTANAS

- En una aplicación Android es muy útil tener diferentes ventanas con diferente información. De esta forma podremos navegar de una a otra dependiendo de lo que quiera el usuario.
- Hasta ahora teníamos todo el contenido en una ventana.
- Esta útil y práctico clasificar el contenido de nuestra aplicación en diferentes ventanas.

3- MÚLTIPLES VENTANAS

- Desde una ventana se puede navegar a otra ventana si queremos.
- También es posible pasar información de una ventana a otra.
- Cada ventana (o interfaz) tendrá su archivo con el código «xml» que representa la apariencia. Además tendrá su funcionamiento en un archivo «java». Esto se le llama «Activity».

3- MÚLTIPLES VENTANAS

- La forma de lanzar otra ventana es usando los «intent».
- Los «intent» podrán tener o no información adicional. Esta es la forma de pasar información entre ventanas.
- !!!!!Todas las ventanas comparten el mismo espacio de nombre de los componentes, es decir, los componente deben tener nombres diferentes.

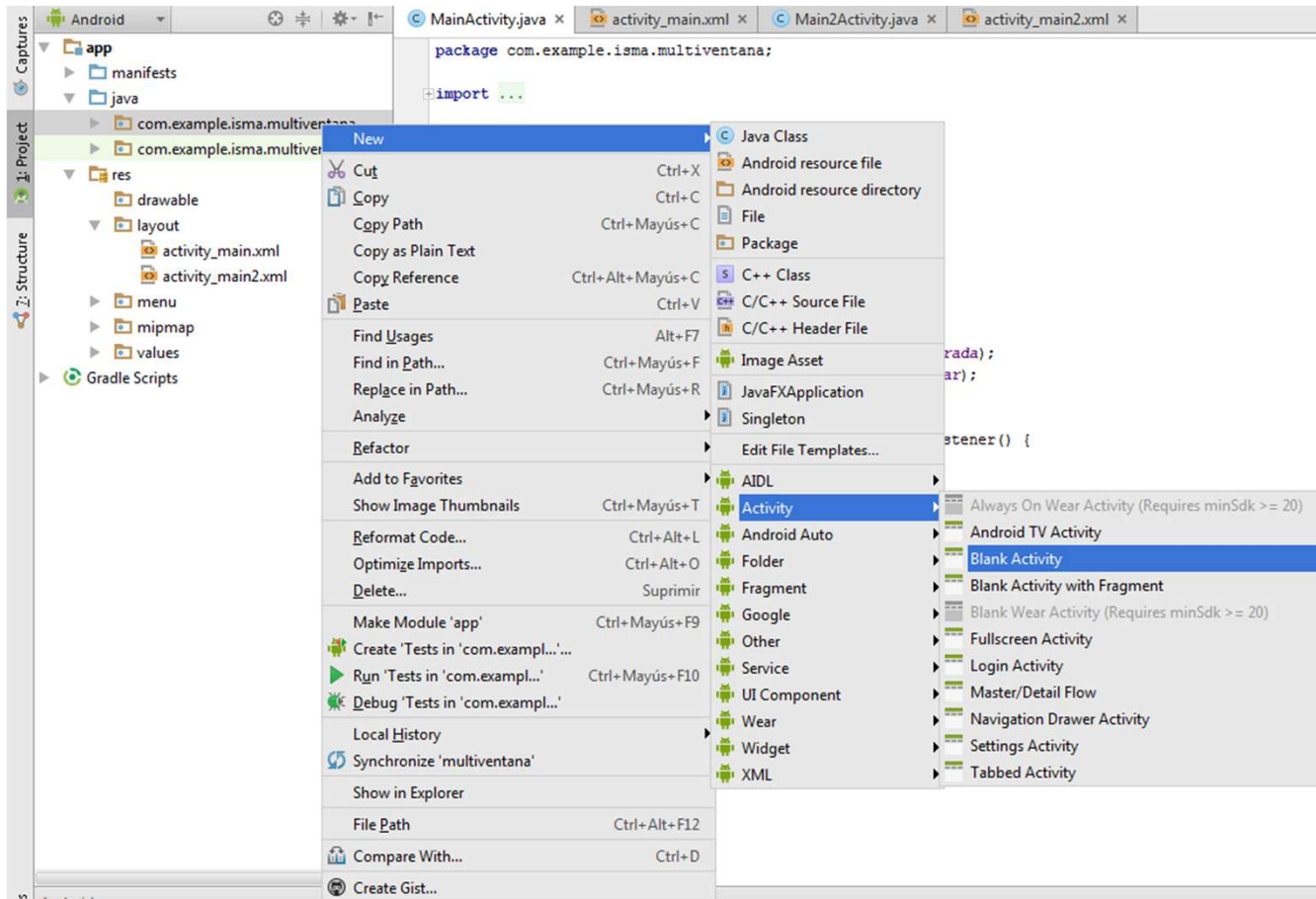
3- MÚLTIPLES VENTANAS

- Para entender como funciona la creación y manejo de múltiples ventanas vamos a ver un ejemplo muy sencillo:
 - Vamos a crear un ventana donde nos pida el nombre y que tenga un botón.
 - Otra ventana donde nos salude con el nombre que hemos introducido antes y un botón de volver.

3- MÚLTIPLES VENTANAS

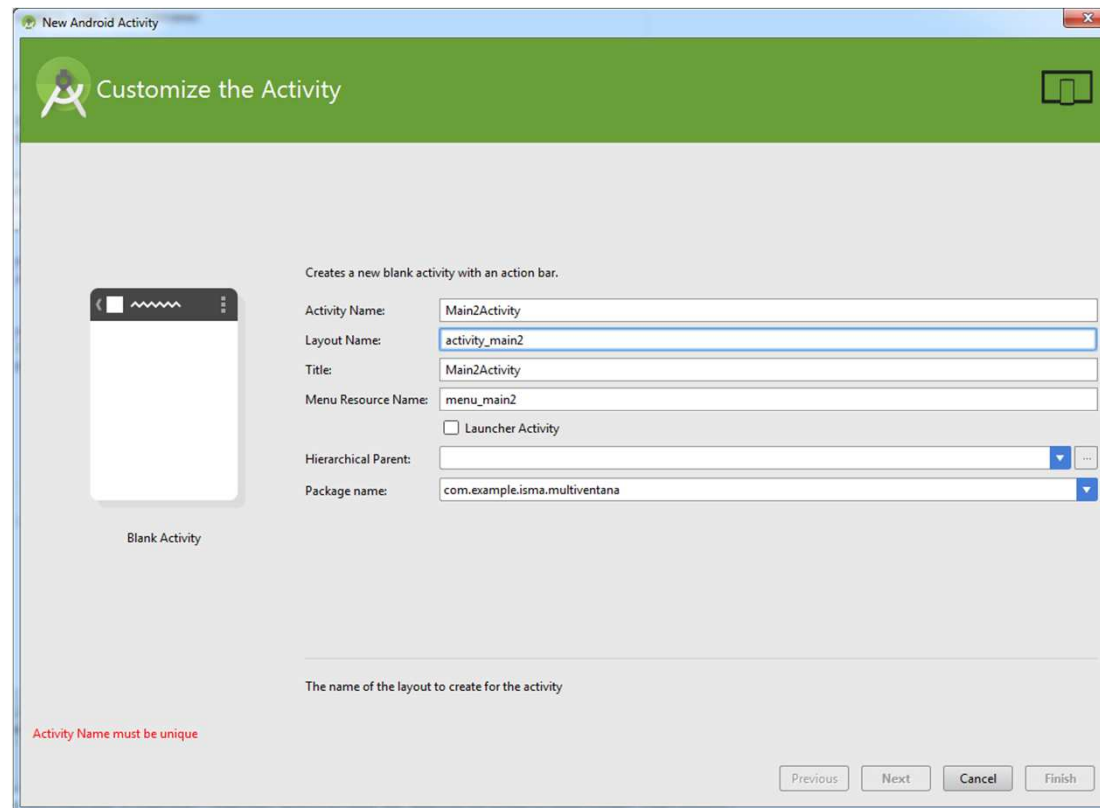
- En primer lugar vamos a crear otra ventana (archivos «xml» y «java»), es decir, vamos a crear una nueva «Activity».
- Vamos al árbol de nuestro proyecto:
 - Java → com.example... → (botón derecho) → new → Activity → Blank Activity

3- MÚLTIPLES VENTANAS



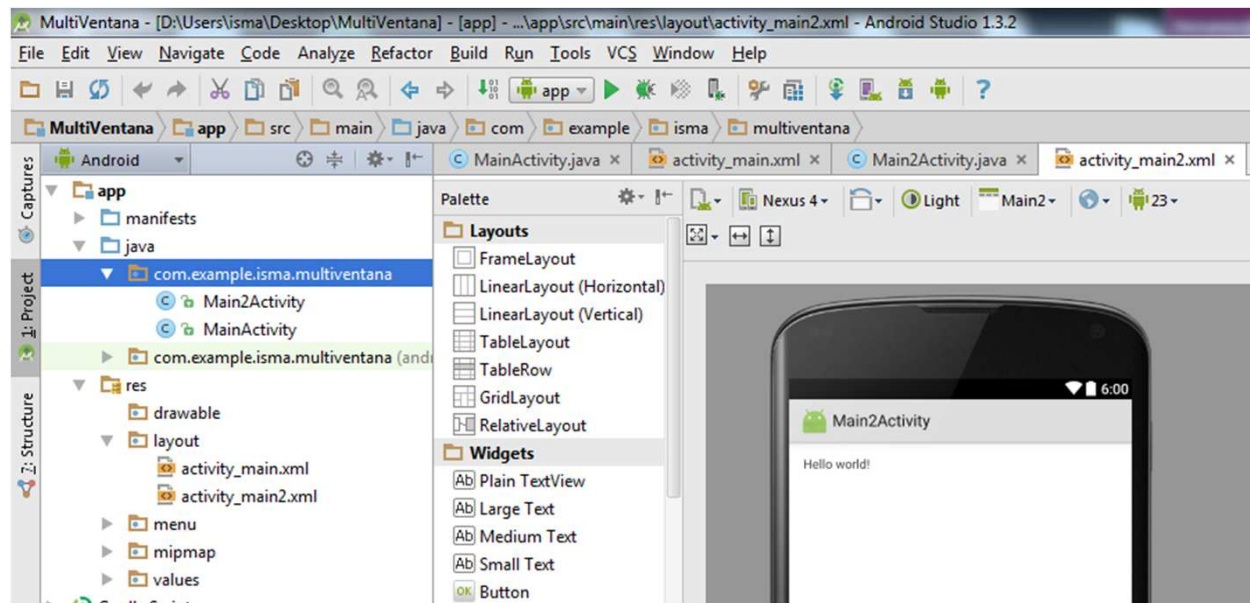
3- MÚLTIPLES VENTANAS

- Después nos pedirá el nombre de nuestro nuevo «Activity», «layout», «título», ...



3- MÚLTIPLES VENTANAS

- Una vez terminemos tendremos otro nuevo «layout» y otro archivo «java» para representar el nuevo «Activity» que hemos creado.



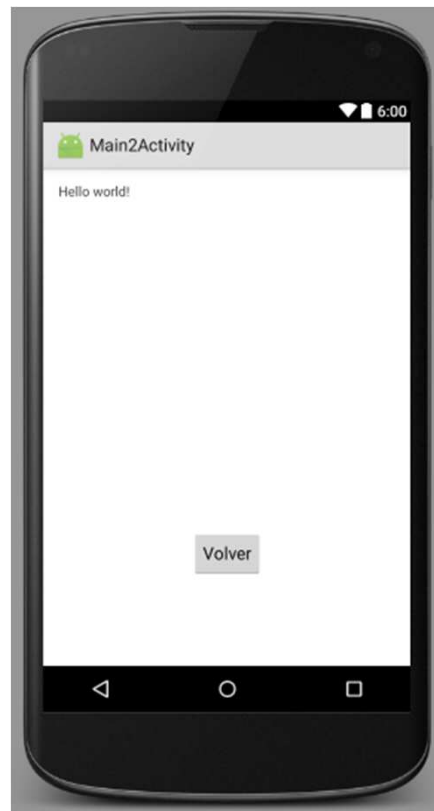
3- MÚLTIPLES VENTANAS

- Ahora vamos a crear el contenido de ambas ventanas. En el «Activity» principal vamos a crear algo así:



3- MÚLTIPLES VENTANAS

- Y en el nuevo «Activity» hemos creado algo así:



3- MÚLTIPLES VENTANAS

- La forma de lanzar una nueva ventana es usando los «Intent» de Android. Vamos a crear un «Intent» en el «Activity» principal cuando se pulse el botón.
- Además en los «Intent» se puede añadir información «Extras» para que el «Activity» receptor reciba información desde este.

3- MÚLTIPLES VENTANAS

```
public class MainActivity extends AppCompatActivity {  
  
    private EditText edt_entrada;  
    private Button btn_aceptar;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        edt_entrada = (EditText)findViewById(R.id.edt_entrada);  
        btn_aceptar = (Button)findViewById(R.id.btn_aceptar);  
  
        btn_aceptar.setOnClickListener(new View.OnClickListener() {  
            @Override  
            public void onClick(View v) {  
                String nombre = edt_entrada.getText().toString();  
  
                if(nombre.length() > 0 ) {  
                    Intent lanzarVentana = new Intent(getApplicationContext(), Main2Activity.class);  
                    lanzarVentana.putExtra("Nombre", nombre);  
                    startActivity(lanzarVentana);  
                    finish();  
                }  
            }  
        });  
    }  
}
```

Creamos un Intent

Cerramos la ventana actual (Opcional)

Iniciamos la ventana

Añadimos información (Opcional)

Ventana a la que vamos a llamar

3- MÚLTIPLES VENTANAS

- La última parte es recibir la información en el «Activity» que hemos creado.
- Y permitir volver al «Activity» principal con el botón volver.

3- MÚLTIPLES VENTANAS

```
public class Main2Activity extends AppCompatActivity {

    private TextView txt_titulo;
    private Button btn_volver;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);

        txt_titulo = (TextView) findViewById(R.id.txt_titulo);
        btn_volver = (Button) findViewById(R.id.btn_volver);

        // Obtiene los extras si los tiene
        Bundle extras = getIntent().getExtras();
        if(extras != null) {
            String dato = extras.getString("Nombre");

            if (dato != null)
                txt_titulo.setText("Hola " + dato);
        }

        btn_volver.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent lanzarVenPrincipal = new Intent(getApplicationContext(), MainActivity.class);
                startActivity( lanzarVenPrincipal );
                finish();
            }
        });
    }
}
```

Recibir la
información
desde la otra
ventana

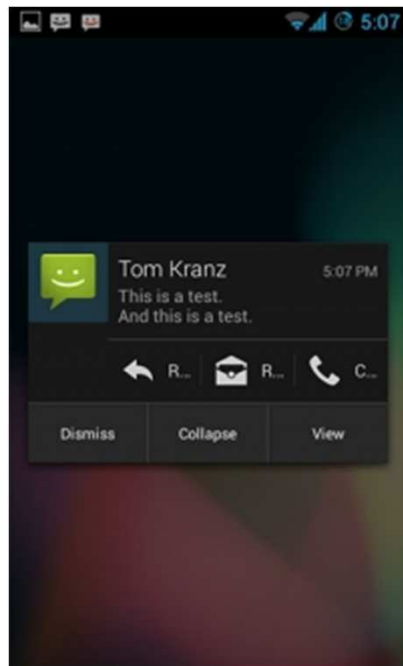
Comprobación
de que se ha
recibido la
información

Volver a la
ventana
principal con el
botón de
volver.

EJERCICIO

- Crear una aplicación para un dispositivo Android que tenga tres ventanas (tres «Activity»). Para gestionar las carreras que hay en Ciudad Real y Toledo disponibles.
 - Crear una ventana principal (UCLM) donde pida el nombre al usuario y permita ir a la ventana de Ciudad Real y Toledo con dos botones.
 - Mostrar en las ventanas de Ciudad Real «Hola + nombre» y mostrar las carreras disponibles. Poner un botón para ir a la otra ventana y otro para volver a la ventana principal.
 - Usar un «TextView» o una lista para mostrar las carreras de cada campus.

4- NOTIFICACIONES

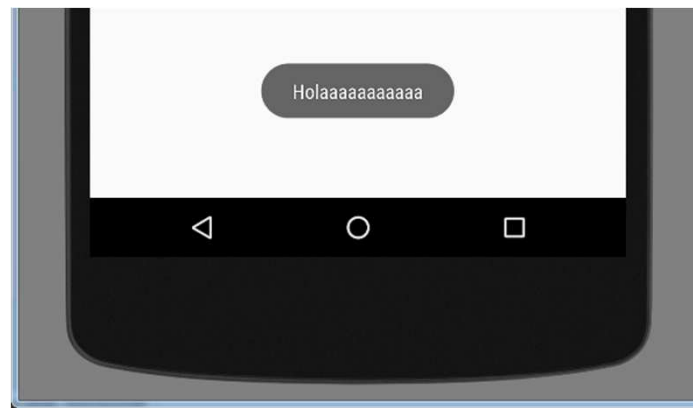


4- NOTIFICACIONES

- Las notificaciones es una herramienta útil de Android para mostrar información al usuario o para que responda ante un evento.
- Hay tres tipos importantes de notificaciones en Android que vamos a ver a continuación.
- Vamos a crear un app con tres botones, uno para cada tipo de notificación.

4- NOTIFICACIONES

- El primer tipo de notificaciones son los diálogos («dialog»), son usados para mostrar información por pantalla temporalmente.
- Ya los hemos usado antes.



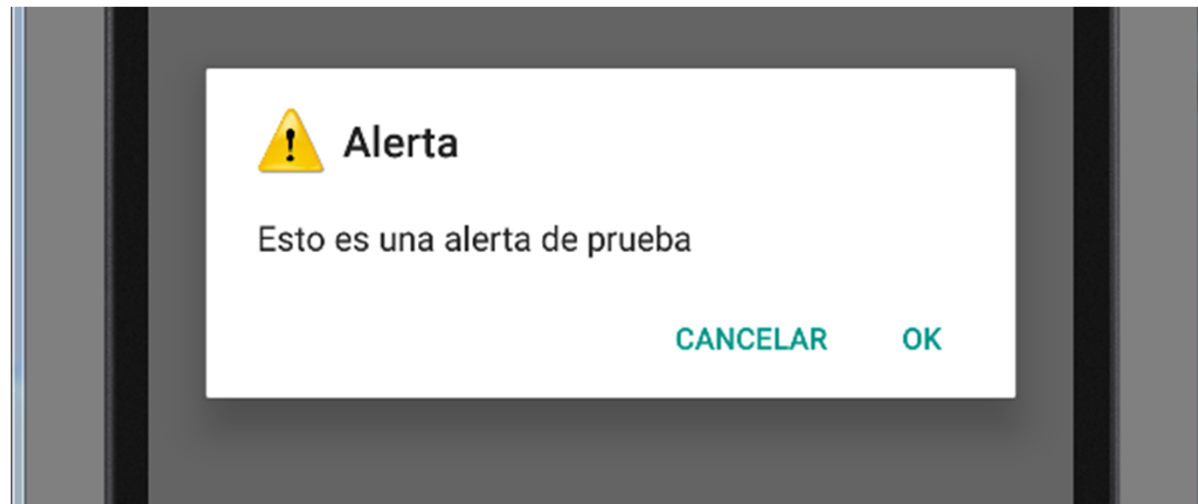
4- NOTIFICACIONES

- El código sería así:

```
// Dialog  
btn_dialog.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        Toast myDialog = Toast.makeText(getApplicationContext(),  
"Holaaaaaaaaaaaaa", Toast.LENGTH_LONG);  
        myDialog.show();  
    }  
});
```


4- NOTIFICACIONES

- El segundo tipo de notificaciones son las ventanas de diálogo («Alert Dialog»).



4- NOTIFICACIONES

- El código sería así:

```
// Alert dialog
btn_alert.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        final AlertDialog.Builder alerta = new AlertDialog.Builder(MainActivity.this);
        alerta.setTitle("Alerta");
        alerta.setMessage("Esto es una alerta de prueba");
        alerta.setIcon(R.drawable.alerta_image);

        alerta.setPositiveButton("OK", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                // Si pulsa OK ...
            }
        });

        alerta.setNegativeButton("Cancelar", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                // Si pulsa Cancelar ...
            }
        });

        alerta.create();
        alerta.show();
    }
});
```

4- NOTIFICACIONES

- El tercer tipo de notificaciones son las «notifications». Este tipo muestra la información en la barra de notificaciones de nuestro dispositivo (arriba).
- Al desplegar nuestra barra de notificaciones podemos pinchar sobre ella e ir a otro «Activity».
- El código sería así:

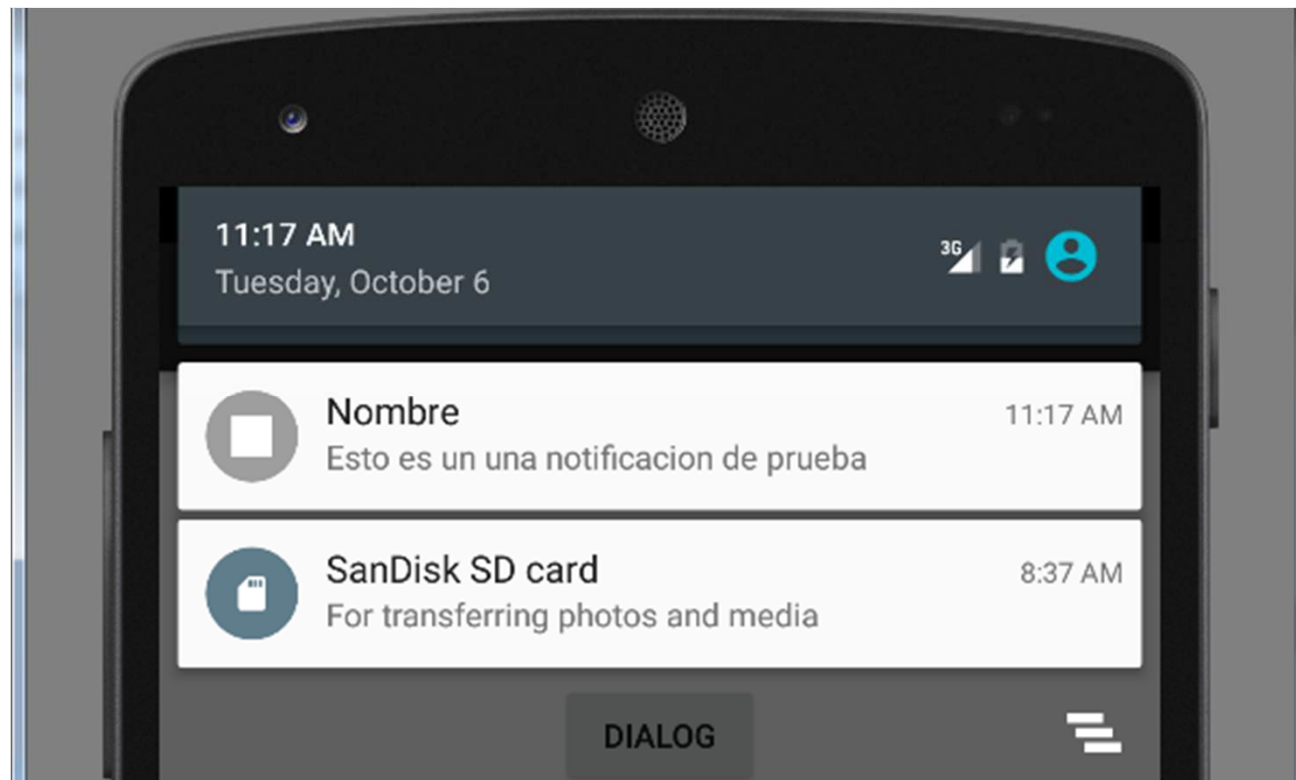
4- NOTIFICACIONES

```
// Notification
btn_notification.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Configurar la notificación
        NotificationCompat.Builder notificacion = new NotificationCompat.Builder(MainActivity.this);
        notificacion.setSmallIcon(R.drawable.notificacion_icono);
        notificacion.setTicker("Notificacion");
        notificacion.setWhen(System.currentTimeMillis());
        notificacion.setContentTitle("Nombre");
        notificacion.setContentText("Esto es un una notificacion de prueba");
        notificacion.setSound( RingtoneManager.getDefaultUri(Notification.DEFAULT_SOUND) );
        notificacion.setDefaults(Notification.DEFAULT_VIBRATE);

        // Llamar a un Activity al pulsar la notificación (Opcional)
        PendingIntent miPI;
        Intent miIntent = new Intent();
        Context miContext = getApplicationContext();
        miIntent.setClass(miContext, MainActivity.class);
        miIntent.putExtra("ID", 1);
        miPI = PendingIntent.getActivity(miContext, 0, miIntent, 0);
        notificacion.setContentIntent(miPI);

        // Crear la notificación
        Notification n = notificacion.build();
        NotificationManager nm = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
        nm.notify(1, n);
    }
});
```

4- NOTIFICACIONES



EJERCICIO

- Crear una app para Android que nos permita escribir mensajes y enviarlos.
- Una vez escribamos el mensaje y le demos a enviar, debe aparecer una notificación con el mensaje.
 - No es necesario llamar a otro «Activity» es este caso.
- Poner un botón salir que cuando se presione aparezca un «Alert Dialog» que nos pregunte si estamos seguros de salir.
 - Solo salir si pulsamos «Si».

5- TIEMPO



5- TIEMPO

- La medición del tiempo puede ser algo importante en nuestra aplicaciones.
- Tenemos tres aspectos importantes relacionados con el tiempo.
 - Obtener la hora y fecha actual del sistema.
 - Comparar horas y fechas diferentes.
 - Realizar procesos que le lanzan trascurrido un tiempo y se puedan repetir periódicamente.

5- TIEMPO

- Para obtener la hora y fecha en Android podemos usar «Date» (para API mayor que 1) para obtener la fecha y hora del sistema de esta forma:

```
Date fecha_actual= new Date();
```

```
fecha_actual.getYear();
```

```
fecha_actual.getMonth();
```

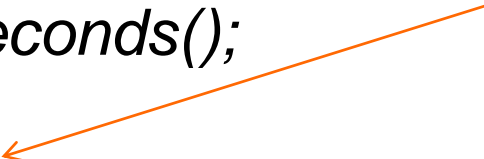
```
fecha_actual.getDay();
```

```
fecha_actual.getMinutes();
```

```
fecha_actual.getSeconds();
```

```
fecha_actual.getTime();
```

Obtiene la hora
actual en
milisegundos



5- TIEMPO

- Una forma sencilla de crear otras horas y fechas es usando los milisegundos actuales y sumando o restando lo que corresponda, vamos a ver un ejemplo:

```
Date fecha_nueva = new Date( );
```

```
int mins = 5;
```

```
long incremento = fecha_actual.getTime() + mins*1000*60;
```

```
fecha_nueva.setTime( incremento );
```

5- TIEMPO

- De esta forma es sencillo comparar horas y fechas diferentes.

```
If (fecha_actual.getTime() > fecha_nueva.getTime() ){  
    // Realizar acciones ...  
}
```

5- TIEMPO

- Ahora vamos a ver como se crear una tarea que se ejecute pasado un tiempo y se pueda repetir periódicamente.
- Esta nueva tarea se ejecutará en un hilo en paralelo. Funcionará a la vez que el resto de nuestra aplicación.
- Debemos crear una nueva clase dentro de nuestra clase «MainActivity» de esta forma:

5- TIEMPO

```
public class MiTarea extends TimerTask{
    @Override
    public void run() {
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                // Ejecutar...
                Toast.makeText(getApplicationContext(), "Holaaa",
                Toast.LENGTH_LONG).show();

            }
        });
    }
}
```

5- TIEMPO

- Debemos declarar dos variables justo al empezar nuestra «MainActivity»

```
private Timer timer;  
private MiTarea miTarea;
```

- Después usando dos botones vamos a activar y desactivar nuestra tarea.

5- TIEMPO

- Para activar nuestra tarea debemos hacer esto:

```
// Botón empezar
btn_empezar.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (timer != null)
            timer.cancel();

        timer = new Timer();
        miTarea = new MiTarea();
        timer.schedule(miTarea, 1000, 5000); // En milisegundos
    }
});
```

Tiempo de espera hasta que se ejecuta nuestra tarea

Tiempo de espera para la siguiente repetición.

5- TIEMPO

- Para desactivar nuestra tarea debemos hacer esto:

```
// Botón terminar  
btn_terminar.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        if (timer != null){  
            timer.cancel();  
            timer = null;  
        }  
    }  
});
```


EJERCICIO

- Crear una aplicación para nuestro dispositivo Android. El usuario podrá introducir un número de minutos. Cuando pase los minutos establecidos aparecerá un «dialog» diciendo «ALARMA».
 - Utilizar un «EditText» para introducir los minutos.
 - Utilizar un botón para activar la alarma.
 - Utilizar otro botón para desactivar la alarma.
 - Cuando hayan pasado los minutos mostrar un «dialog» que diga ALARMA cada 5 segundos.
 - Para leer el contenido del «EditText» y convertido a un número podemos usar:

```
long num = Integer.parseInt( edt_entrada.getText().toString( ) );
```
 - Ver un ejemplo de la interfaz a continuación.

EJERCICIO

