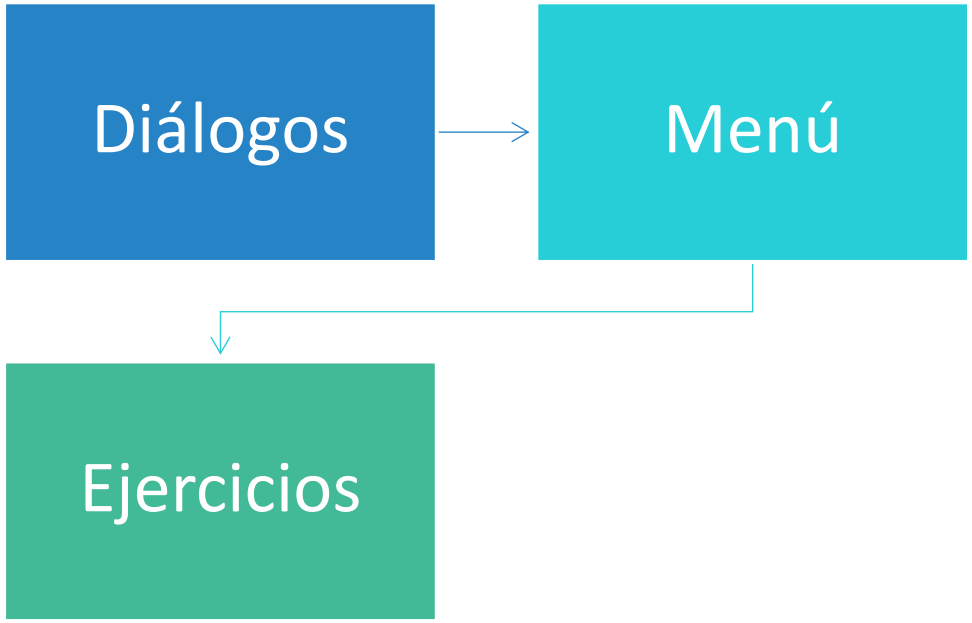
2º DAM

Programación Multimedia y Dispositivos Móviles

Programación Android

José A. Lara

Contenido



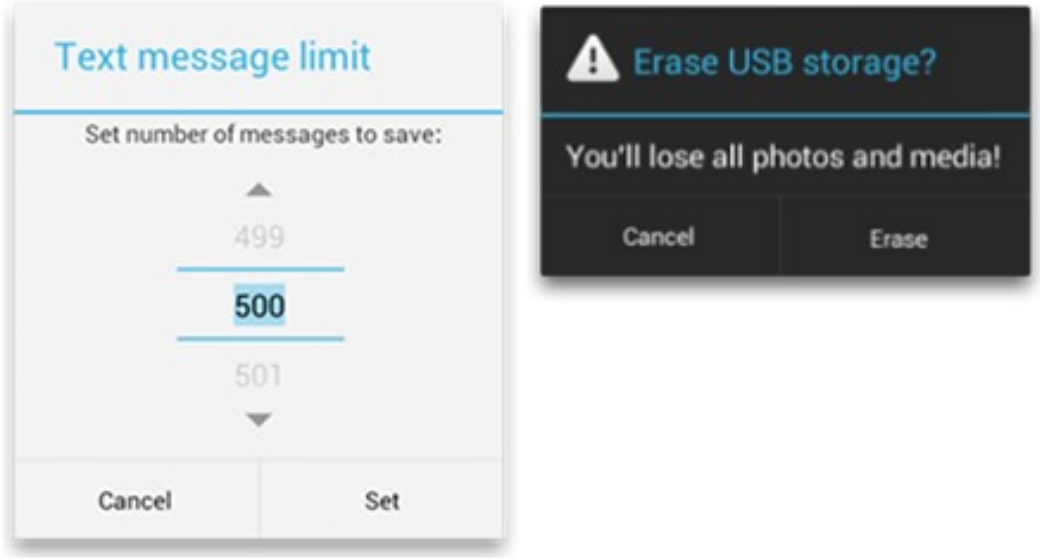
android

Diálogos

Programación Android

Diálogos

Un diálogo es una ventana pequeña que le indica al usuario que debe tomar una decisión o ingresar información adicional. Un diálogo no ocupa toda la pantalla y, generalmente, se usa para eventos modales que requieren que los usuarios realicen alguna acción para poder continuar.



android

Programación Android

Diálogos

Se puede crear un diálogo personalizado empleando directamente la clase Dialog, pero es habitual utilizar alguno de los diálogos ya existentes en esta tabla:

Evento	Descripción
AlertDialog	Ventana de alerta o confirmación
ProgressDialog	Ventana de progreso
DatePickerDialog	Ventana para seleccionar una fecha
TimePickerDialog	Ventana para seleccionar una hora

android

Diálogos - AlertDialog

Un diálogo que puede mostrar un **título**, hasta tres **botones**, una lista de elementos seleccionables o un diseño personalizado. Para crear un AlertDialog se utiliza la subclase **Builder** y se le van añadiendo elementos tales como el título, o los botones que va a llevar el diálogo. Por ejemplo, para mostrar un Alert con un mensaje simple y un botón para ocultarlo se puede emplear el código siguiente:

```
val dialogo:AlertDialog=AlertDialog.Builder( context: this)
    .setTitle("Título del diálogo")
    .setMessage("Esto es un diálogo")
    .setNeutralButton( text: "Cerrar", listener: null).create()
dialogo.show()
```

```
AlertDialog dialogo=new AlertDialog.Builder( context: this)
        .setTitle("Título del diálogo")
        .setMessage("Esto es un diálogo")
        .setNeutralButton( text: "Cerrar", listener: null).create();
dialogo.show();
```

android

Diálogos - AlertDialog

Existen tres tipos de botones, Positive, Negative y Neutral y no se puede crear más de uno de cada tipo, por lo que un AlertDialog poseerá a lo sumo 3 botones.

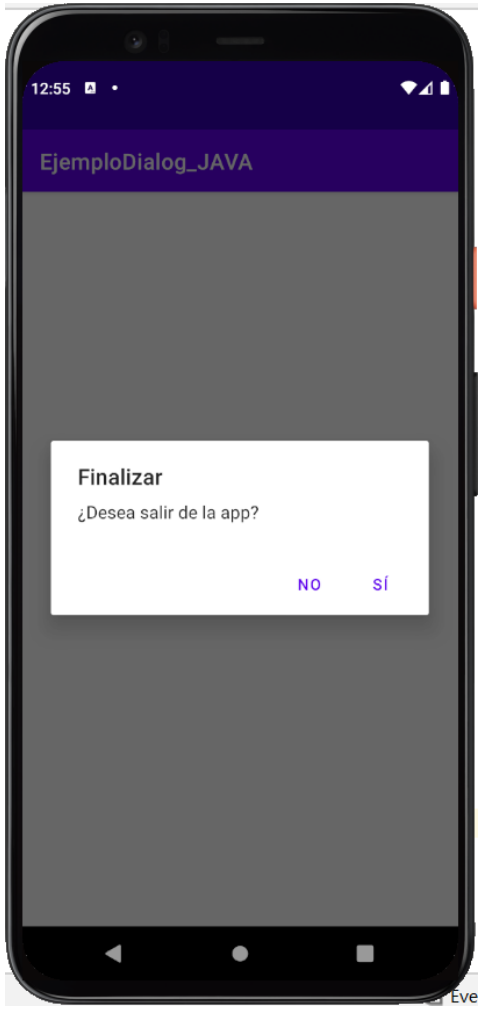
Independientemente del nombre de estos botones, la funcionalidad puede ser definida por el desarrollador en el momento de creación del diálogo, para lo cual es necesario definir los Listeners correspondientes que realicen las acciones asociadas.

Normalmente se crea un AlertDialog a través del método create() de AlertDialog.Builder una sola vez, y se muestra en las sucesivas ocasiones que sea necesario a través del método show().

Programación Android

Diálogos - AlertDialog

```
val dialogo = AlertDialog.Builder( context: this)
    .setTitle("Finalizar")
    .setMessage("¿Desea salir de la app?")
    .setPositiveButton(
        text: "Sí"
    ) { dialogInterface, i -> finish() }.setNegativeButton(
        text: "No"
    ) { dialogInterface, i -> dialogInterface.cancel() }.create()
dialogo.show()
```

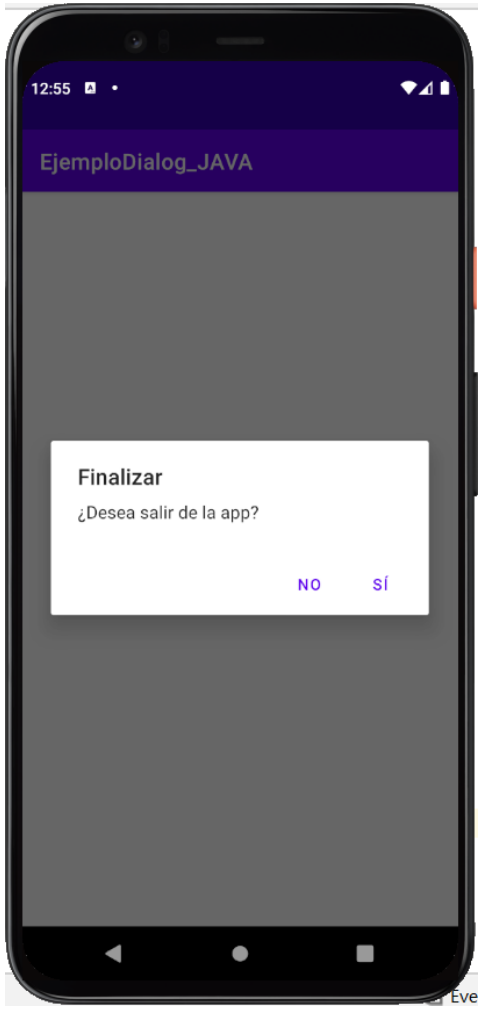


android

Programación Android

Diálogos - AlertDialog

```
AlertDialog dialogo=new AlertDialog.Builder( context: this)
        .setTitle("Finalizar")
        .setMessage("¿Desea salir de la app?")
        .setPositiveButton( text: "Sí", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                MainActivity.this.finish();
            }
        }).setNegativeButton( text: "No", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                dialogInterface.cancel();
            }
        }).create();
dialogo.show();
```



android

Diálogos - AlertDialog

Es posible usar un AlertDialog para crear menús contextuales, añadiendo una lista de elementos seleccionables a la ventana mostrada:

```
val items = arrayOf<CharSequence>("Rojo", "Verde", "Azul")
val builder = AlertDialog.Builder( context: this)
builder.setTitle("Elige un color")
builder.setItems(
    items
) { dialogInterface, i ->
    Toast.makeText(applicationContext, items[i], Toast.LENGTH_SHORT).show()
}
val dialog = builder.create()
dialog.show()
```

android

Diálogos - AlertDialog

Es posible usar un AlertDialog para crear menús contextuales, añadiendo una lista de elementos seleccionables a la ventana mostrada:

```
CharSequence[] items = {"Rojo", "Verde", "Azul"};
AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
builder.setTitle("Elige un color");
builder.setItems(items, new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
        Toast.makeText(getApplicationContext(),items[i], Toast.LENGTH_SHORT).show();
    }
});
AlertDialog dialog = builder.create();
dialog.show();
```

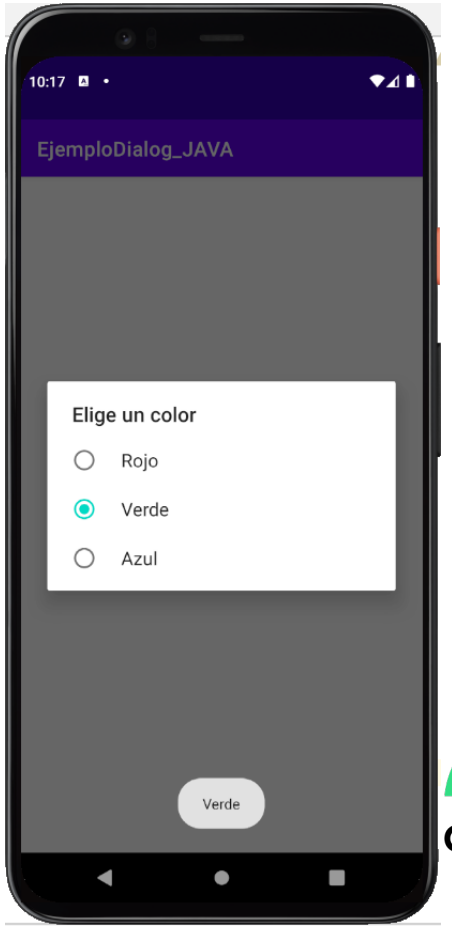
android

Programación Android

Diálogos - AlertDialog

Es posible también establecer listas de selección simple o múltiple utilizando el método **setSingleChoiceItems** o **setMultiChoiceItems** respectivamente.

```
val items = arrayOf<CharSequence>("Rojo", "Verde", "Azul")
val builder = AlertDialog.Builder( context: this)
builder.setTitle("Elige un color")
builder.setSingleChoiceItems(
    items, checkedItem: 1
) { dialogInterface, i ->
    Toast.makeText(applicationContext, items[i], Toast.LENGTH_SHORT).show()
}
val dialog = builder.create()
dialog.show()
```

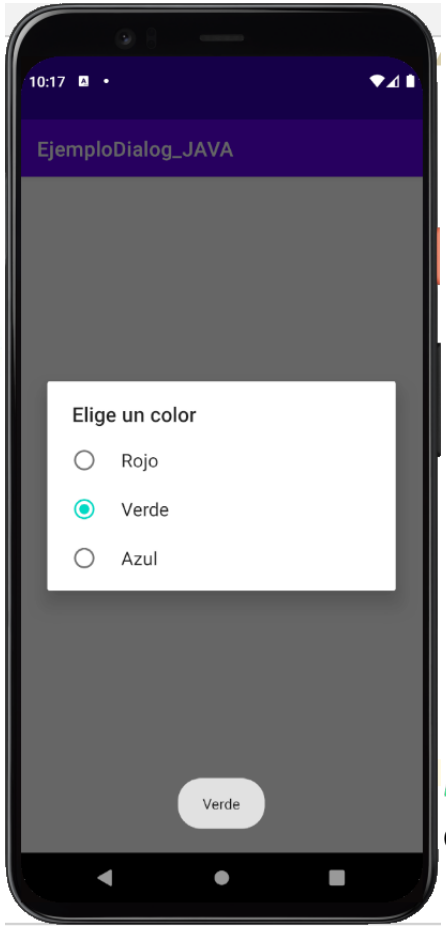


android

Diálogos - AlertDialog

Es posible también establecer listas de selección simple o múltiple utilizando el método **setSingleChoiceItems** o **setMultiChoiceItems** respectivamente.

```
CharSequence[] items = {"Rojo", "Verde", "Azul"};
AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
builder.setTitle("Elige un color");
builder.setSingleChoiceItems(items, checkedItem: -1, new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
        Toast.makeText(getApplicationContext(),items[i], Toast.LENGTH_SHORT).show();
    }
});
AlertDialog dialog = builder.create();
dialog.show();
```

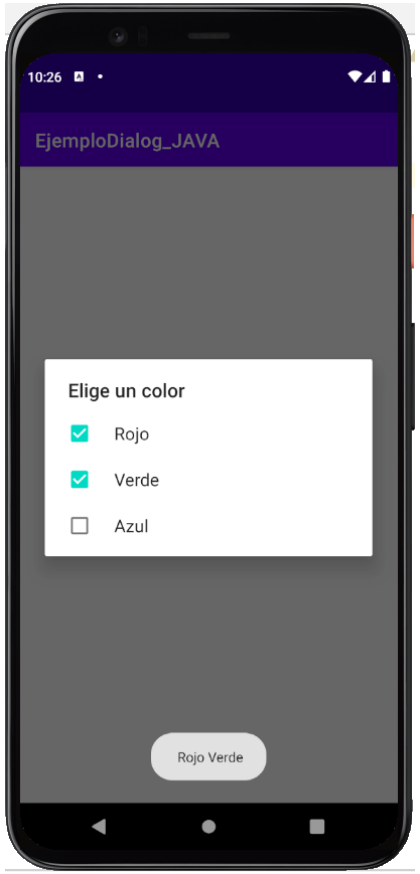


android

Programación Android

Diálogos - AlertDialog

```
val items = arrayOf<CharSequence>("Rojo", "Verde", "Azul")
val checkedItems = booleanArrayOf(false, false, false)
val builder = AlertDialog.Builder( context: this)
builder.setTitle("Elige un color")
builder.setMultiChoiceItems(
    items, checkedItems
) { dialogInterface, i, b ->
    var resultado = ""
    for (j in checkedItems.indices) {
        if (checkedItems[j]) resultado += " " + items[j]
    }
    Toast.makeText(applicationContext, resultado, Toast.LENGTH_SHORT).show()
}
val dialog = builder.create()
dialog.show()
```

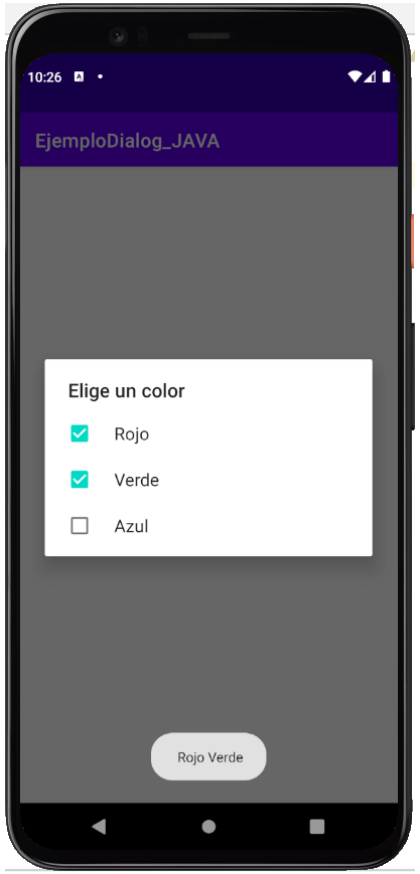


android

Programación Android

Diálogos - AlertDialog

```
CharSequence[] items = {"Rojo", "Verde", "Azul"};
boolean[] checkedItems = {false,false,false};
AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
builder.setTitle("Elige un color");
builder.setMultiChoiceItems(items, checkedItems, new DialogInterface.OnMultiChoiceClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i, boolean b) {
        String resultado="";
        for(int j=0;j<checkedItems.length;j++) {
            if (checkedItems[j])
                resultado+=" "+items[j];
        }
        Toast.makeText(getApplicationContext(), resultado, Toast.LENGTH_SHORT).show();
    }
});
AlertDialog dialog = builder.create();
dialog.show();
```



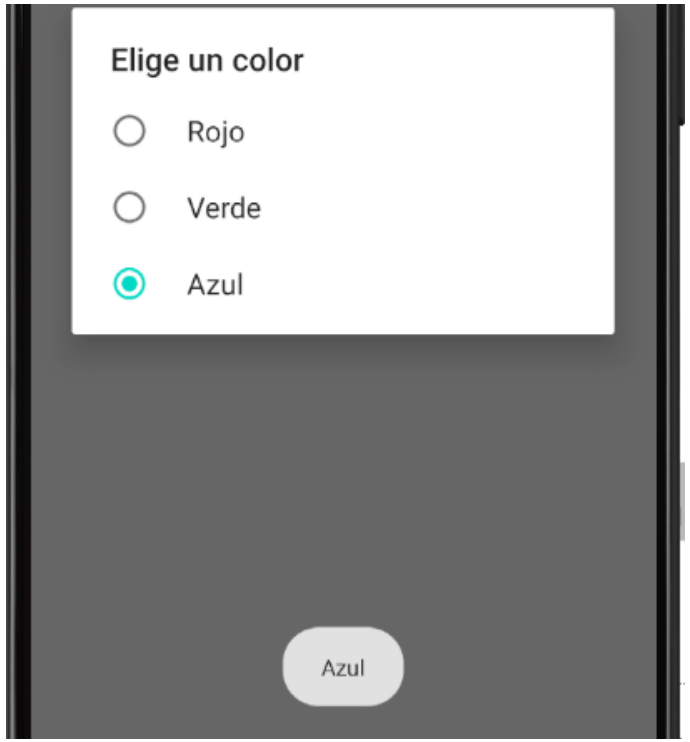
android

Programación Android

Diálogos - Toast

Existe un tipo especial de ventana para notificación que no es un Dialog denominada Toast.

Este tipo de ventana aparece durante unos segundos sobre la pantalla actual la cual se mantiene activa.



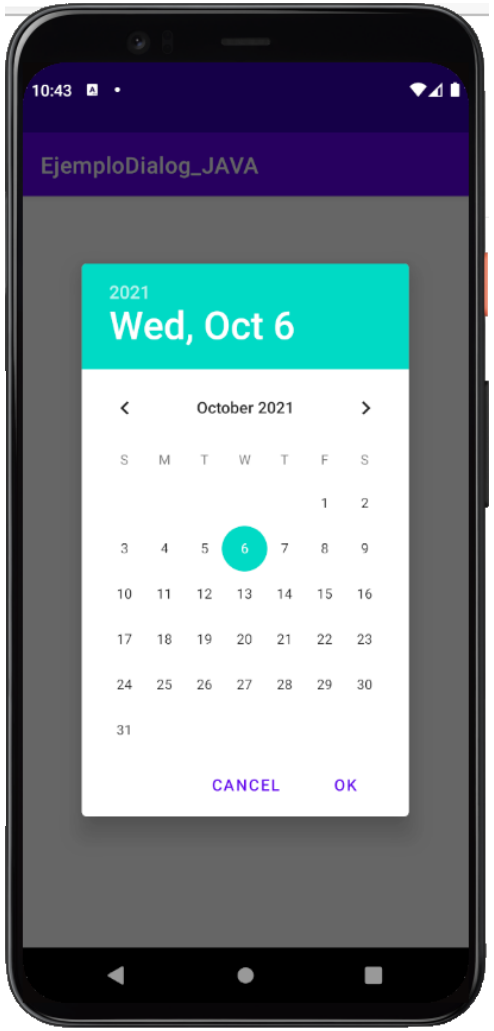
android

Programación Android

Diálogos - DatePickerDialog y TimePickerDialog

Son diálogos con una IU predefinida que le permite al usuario seleccionar una fecha o una hora.

```
val txtDate = findViewById<TextView>(R.id.txtDate)
val c = Calendar.getInstance()
val mYear = c[Calendar.YEAR]
val mMonth = c[Calendar.MONTH]
val mDay = c[Calendar.DAY_OF_MONTH]
val datePickerDialog = DatePickerDialog( context: this,
    { datePicker, i, i1, i2 -> txtDate.text = "$i2/$i1/$i" }, mYear, mMonth, mDay
)
datePickerDialog.show()
```



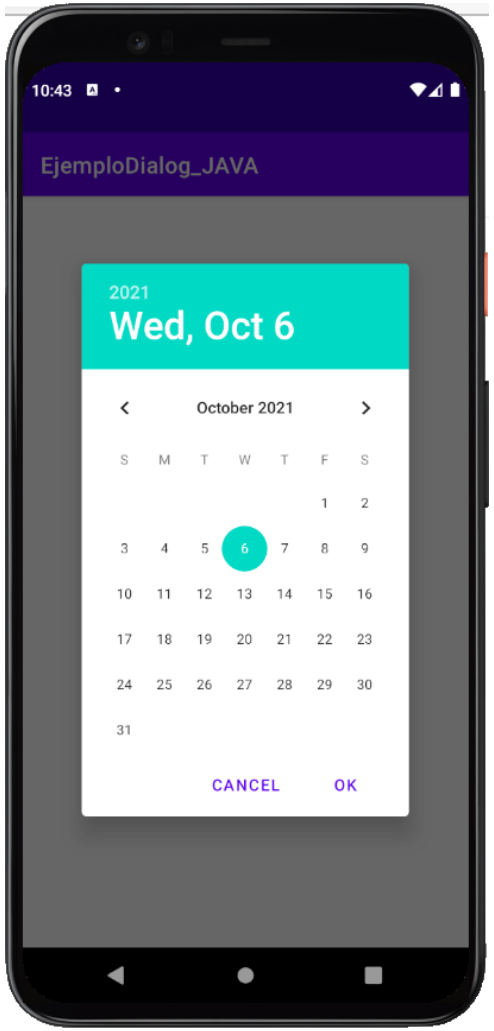
android

Programación Android

Diálogos - DatePickerDialog y TimePickerDialog

Son diálogos con una IU predefinida que le permite al usuario seleccionar una fecha o una hora.

```
TextView txtDate = findViewById(R.id.txtDate);
final Calendar c = Calendar.getInstance();
int mYear = c.get(Calendar.YEAR);
int mMonth = c.get(Calendar.MONTH);
int mDay = c.get(Calendar.DAY_OF_MONTH);
DatePickerDialog datePickerDialog = new DatePickerDialog( context: this,
        new DatePickerDialog.OnDateSetListener() {
    @Override
    public void onDateSet(DatePicker datePicker, int i, int i1, int i2) {
        txtDate.setText(i2+"/"+i1+"/"+i);
    }
},mYear,mMonth,mDay);
datePickerDialog.show();
```

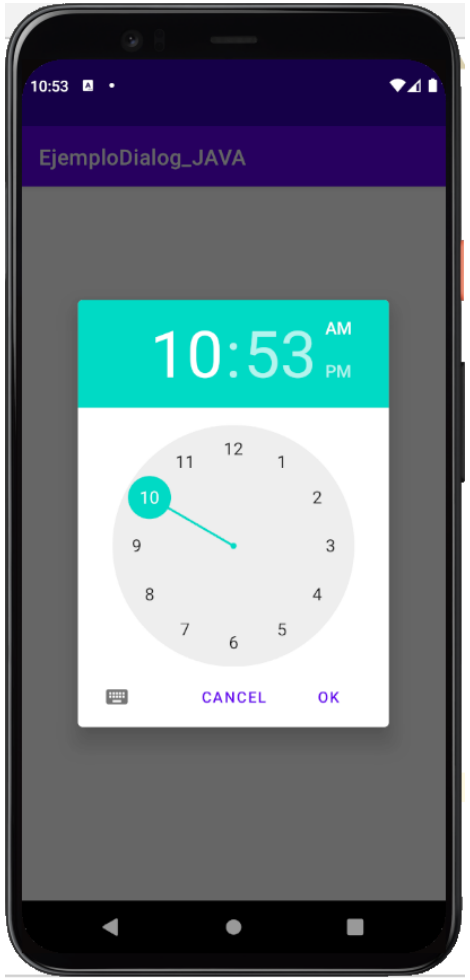


android

Diálogos - DatePickerDialog y TimePickerDialog

Son diálogos con una IU predefinida que le permite al usuario seleccionar una fecha o una hora.

```
val txtDate = findViewById<TextView>(R.id.txtDate)
val c = Calendar.getInstance()
val hour = c[Calendar.HOUR_OF_DAY]
val minute = c[Calendar.MINUTE]
val timePickerDialog = TimePickerDialog( context: this,
    { timePicker, i, i1 -> txtDate.text = "$hour:$minute" }
    , hour, minute, is24HourView: false
)
timePickerDialog.show()
```



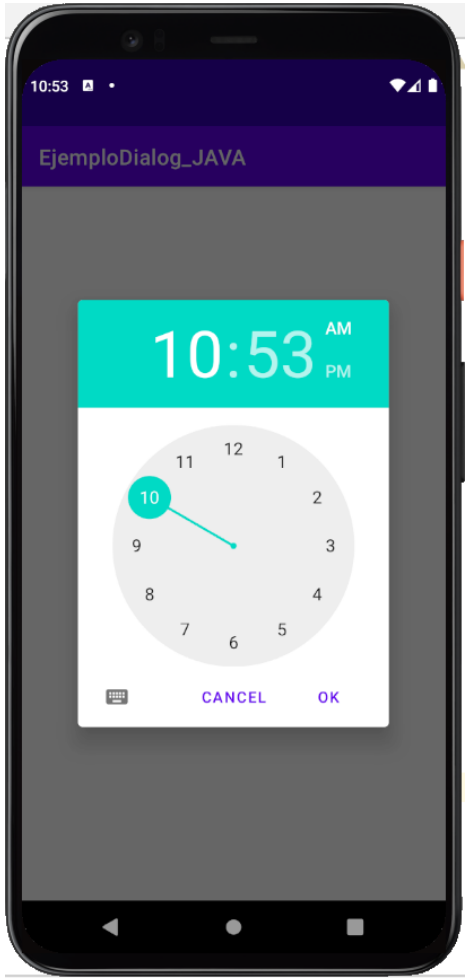
android

Programación Android

Diálogos - DatePickerDialog y TimePickerDialog

Son diálogos con una IU predefinida que le permite al usuario seleccionar una fecha o una hora.

```
TextView txtDate = findViewById(R.id.txtDate);
final Calendar c = Calendar.getInstance();
int hour = c.get(Calendar.HOUR_OF_DAY);
int minute = c.get(Calendar.MINUTE);
TimePickerDialog timePickerDialog = new TimePickerDialog( context: this,
        new TimePickerDialog.OnTimeSetListener() {
            @Override
            public void onTimeSet(TimePicker timePicker, int i, int i1) {
                txtDate.setText(hour+":"+minute);
            }
        },hour,minute, is24HourView: false);
timePickerDialog.show();
```



android

Menús

Programación Android

Menús

Los menús son elementos muy habituales en aplicaciones de interfaz gráfico para incorporar acciones adicionales.

Para definir un menú es necesario crear un fichero XML en la ubicación **/res/menu** indicando los elementos del menú. Cada elemento viene identificado por un ID (id), un título (title), y a menudo un icono (icon). Para cada uno de ellos se puede definir a su vez un submenú siguiendo la estructura XML.

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/Archivo"
        android:title="@string/archivo">
        <menu>
            <item android:id="@+id/crear"
                android:title="@string/crear"/>
            <item android:id="@+id/abrir"
                android:title="@string/abrir"/>
        </menu>
    </item>
</menu>
```

android

Menús

Para cargar el menú en la aplicación simplemente es necesario sobrescribir el método **onCreateOptionsMenu** de la Activity:

```kotlin
override fun onCreateOptionsMenu(menu: Menu?): Boolean {
    val inflater = menuInflater
    inflater.inflate(R.menu.menu, menu)
    return true
}
```

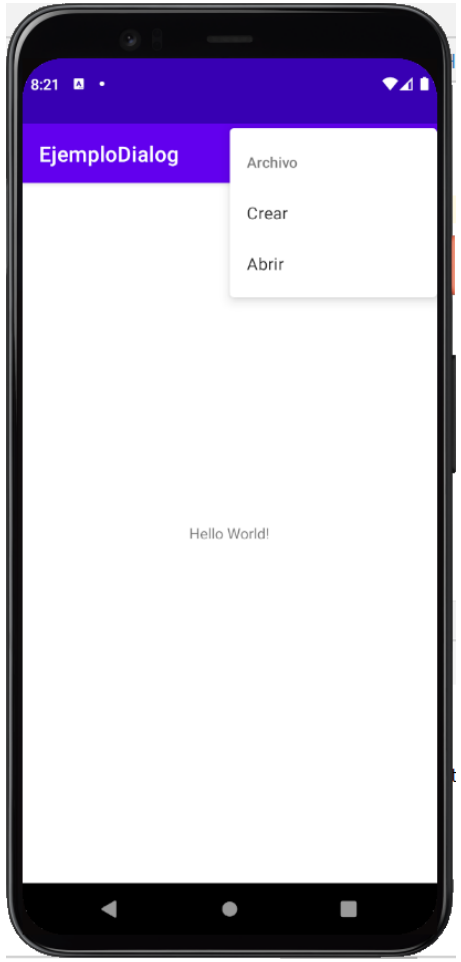
```java
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater=getMenuInflater();
    inflater.inflate(R.menu.menu,menu);
    return true;
}
```

Programación Android

Menús

Para gestionar los eventos de los elementos de menú es necesario sobrescribir un método de la Activity denominado **onOptionsItemSelected**.

```
override fun onOptionsItemSelected(item: MenuItem): Boolean {
    val toast: Toast
    when (item.itemId) {
        R.id.abrir -> {
            toast = Toast.makeText(
                applicationContext,
                text: "Abrir archivo", Toast.LENGTH_SHORT
            )
            toast.show()
        }
        R.id.crear -> {
            toast = Toast.makeText(
                applicationContext,
                text: "Crear archivo", Toast.LENGTH_SHORT
            )
            toast.show()
        }
    }
    return true
}
```



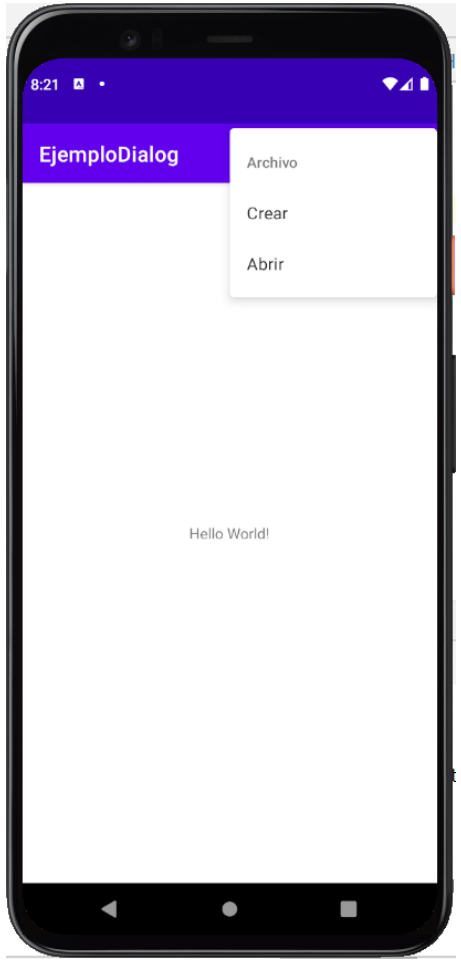
android

Programación Android

Menús

Para gestionar los eventos de los elementos de menú es necesario sobrescribir un método de la Activity denominado **onOptionsItemSelected**.

```
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    Toast toast;
    switch (item.getItemId()){
        case R.id.abrir:
            toast=Toast.makeText(getApplicationContext(),
                    text: "Abrir archivo",Toast.LENGTH_SHORT);
            toast.show();
            break;
        case R.id.crear:
            toast=Toast.makeText(getApplicationContext(),
                    text: "Crear archivo",Toast.LENGTH_SHORT);
            toast.show();
            break;
    }
    return true;
}
```

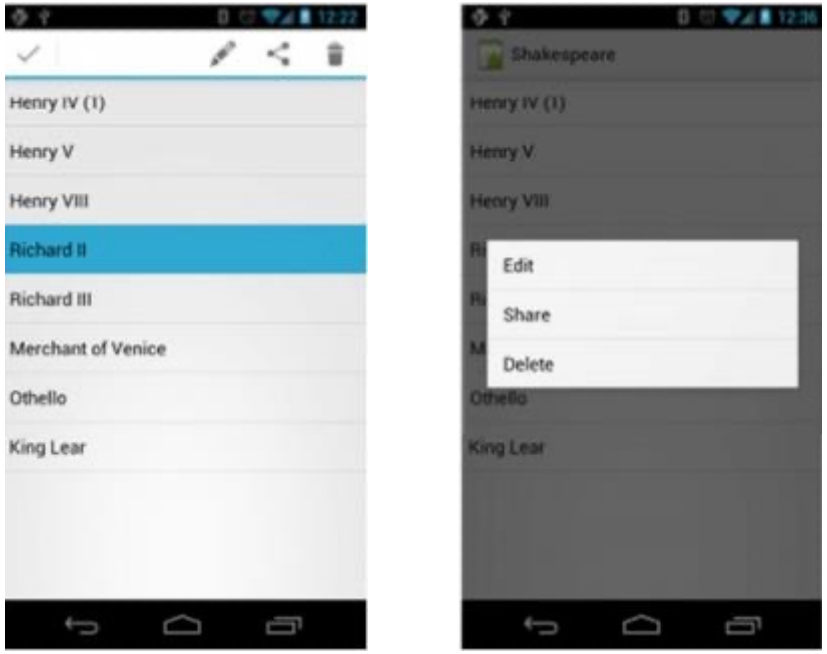


android

Menús

Un menú aparece cada vez que el usuario pulsa el icono de menú en la pantalla, cargando siempre los mismos elementos. Si se desea modificar en tiempo de ejecución el contenido del menú de la aplicación, se puede recurrir al método **onPrepareOptionsMenu**.

De forma similar es posible crear **menús contextuales** dependientes del elemento seleccionado, normalmente tras una pulsación larga sobre el mismo:



android

Menús

Para crear un menú de este tipo se siguen pasos análogos:

- Crear un archivo XML con el menú a utilizar.
- Cargar el menú a través de la sobreescritura del método **onCreateContextMenu**. Si es necesario un menú dependiente del elemento seleccionado se gestionará la carga del archivo de menú en este método, que recibe como parámetro el elemento View para el que va a ser cargado el menú.
- Responder al evento a través del método **onContextItemSelected**.
- Establecer que el componente concreto de vista muestre el menú contextual utilizando el método **registerForContextMenu**.

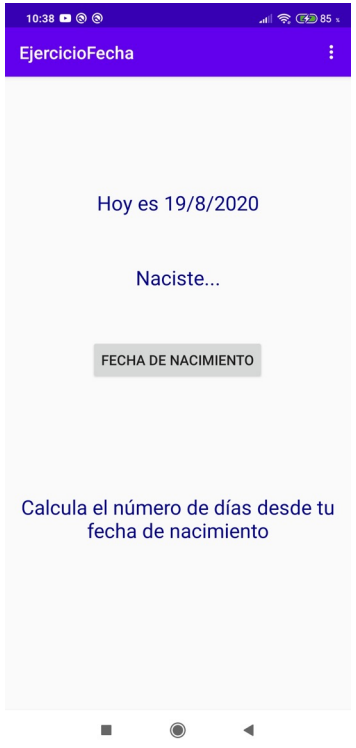
Más información: https://developer.android.com/guide/topics/ui/menus.html

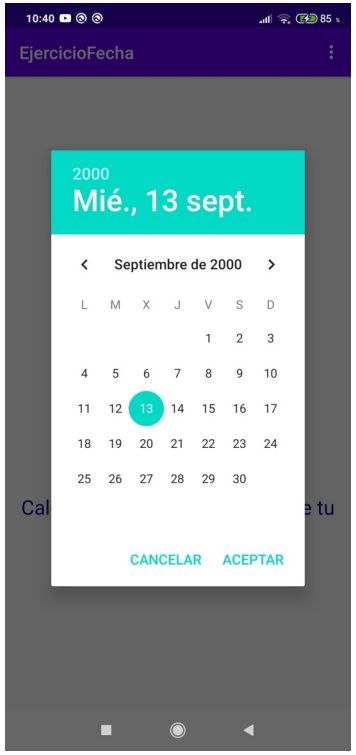
Ejercicios

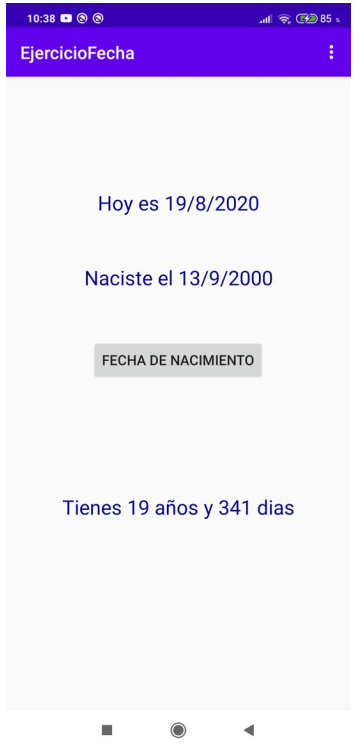
Programación Android

Ejercicio

Realizar una app Android que muestre un botón y 3 TextView. Al pulsar el botón debe aparecer un diálogo de fecha. Seleccionaremos la fecha de nacimiento y nos mostrará en el TextView los años y días que tenemos.







android

Programación Android

Ejercicio

Crear una aplicación Android que muestre una lista de comunidades y ciudades autónomas, tal que:

- Al pulsar sobre cada elemento de la lista muestre un **diálogo** “Yo soy de…” según la comunidad seleccionada.
- Existe un **menú general** con las opciones **Recargar** y **Limpiar** que recargará la lista con todos los datos o la eliminará por completo, respectivamente.
- Al dejar un ítem de la lista pulsado (**long click**) aparecerá un **menú contextual** con la opción Eliminar que mostrará un diálogo de confirmación. El menú contextual debe tener como título el nombre de la comunidad, y que la pregunta sea “¿Desea eliminar la Comunidad…” según la comunidad seleccionada

android

Programación Android

Ejercicio