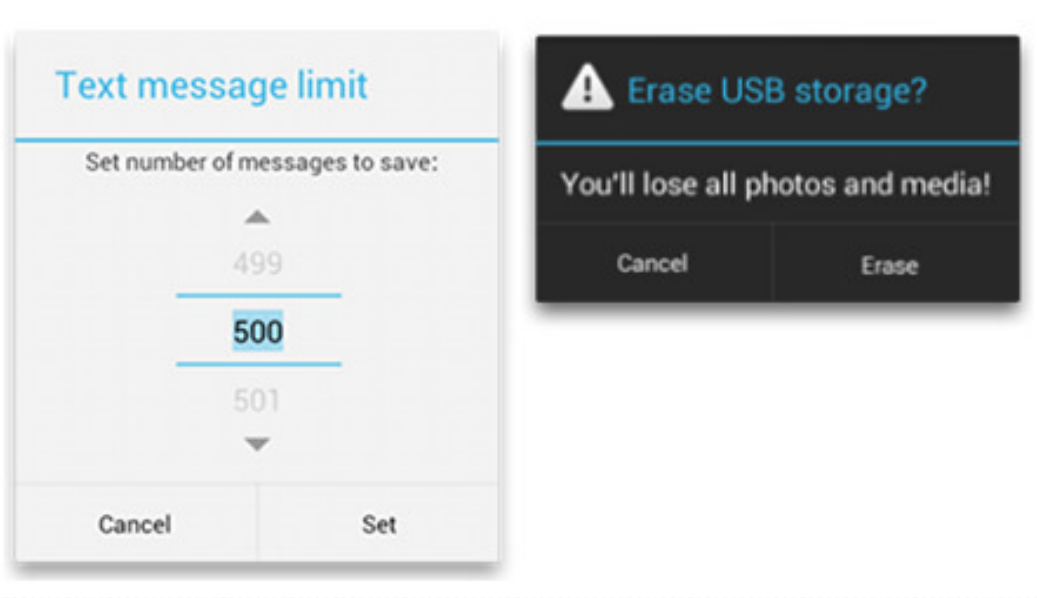
Dialogs

Dialog

- En esta parte seguimos el tutorial de Android https://developer.android.com/guide/topics/ui/dialogs.html

Dialog

- Un dialog (diálogo) es una pequeña ventana que pide al usuario tomar una decisión o dar información extra.



Importante

- Los diálogos son asíncronos (no bloqueantes).
- La siguiente instrucción a la apertura del diálogo se ejecuta sin esperar que se cierre el diálogo.
- La comunicación con un dialog se realiza mediante callbacks.

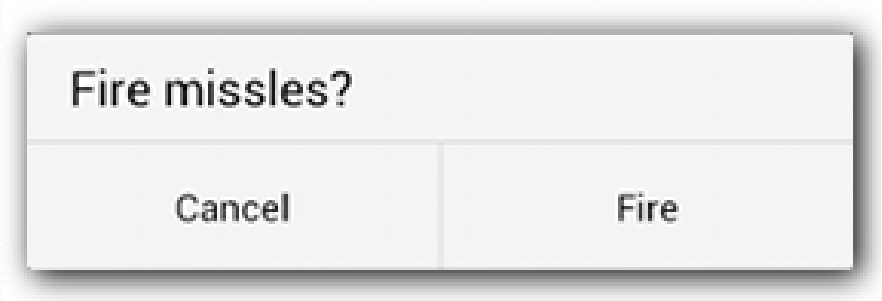
Tipos de dialogs

- AlertDialog
- DatePickerDialog
- TimePickerDialog

AlertDialog

- Se recomienda usar DialogFragment como contenedor de AlertDialog:
- Maneja correctamente el ciclo de vida.
- Permite reusar la UI del Dialog.

Ejemplo

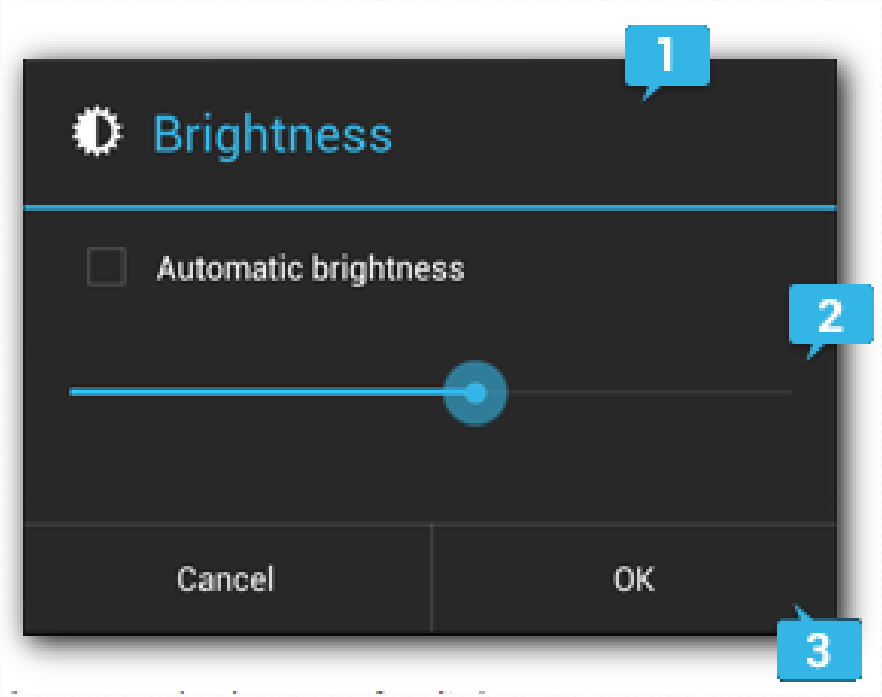


Ejemplo

```
public class FireMissilesDialogFragment extends DialogFragment {
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        // Use the Builder class for convenient dialog construction
        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
        builder.setMessage(R.string.dialog_fire_missiles)
               .setPositiveButton(R.string.fire, new DialogInterface.OnClickListener() {
                   public void onClick(DialogInterface dialog, int id) {
                       // FIRE ZE MISSILES!
                   }
               })
               .setNegativeButton(R.string.cancel, new DialogInterface.OnClickListener() {
                   public void onClick(DialogInterface dialog, int id) {
                       // User cancelled the dialog
                   }
               });
        // Create the AlertDialog object and return it
        return builder.create();
    }
}
```

Partes de un Dialog

1. Título
2. Contenido
3. Acciones



Pasos para crear un Dialog

1. Llamar al constructor:

   AlertDialogBuilder builder = AlertDialog.Builder(getActivity())

2. Configurar el Dialog usando *setMessage*(), *setTitle*(), etc.
3. Definir los callbacks de los botones
   1. Botón positivo: “OK”, “Aceptar”, “Si”
   2. Botón negativo: “Cancelar”, “No”
   3. Botón neutral: “Recordar más tarde”

Pasos para crear un Dialog

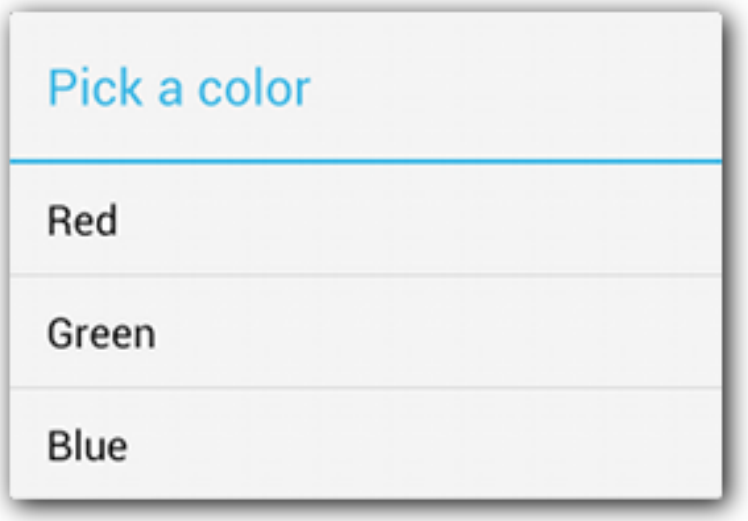
4. Obtener el AlertDialog a partir del objeto builder.

   AlertDialog dialog = builder.create();

Dialogs con listas

- Lista tradicional (una selección)
- Lista con botones de radio (una selección)
- Lista con checkboxes (multiselección)

Lista tradicional

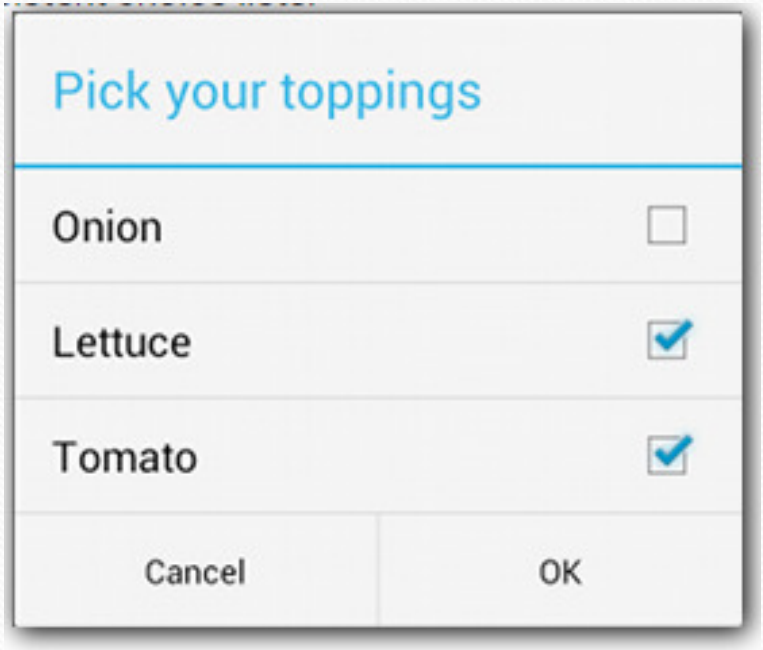


Lista tradicional

```
@Override
public Dialog onCreateDialog(Bundle savedInstanceState) {
    AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
    builder.setTitle(R.string.pick_color)
           .setItems(R.array.colors_array, new DialogInterface.OnClickListener() {
               public void onClick(DialogInterface dialog, int which) {
               // The 'which' argument contains the index position
               // of the selected item
           }
    });
    return builder.create();
}
```

Lista de una o más selecciones

- Usar *setMultiChoiceItems*() o *setSingleChoiceItems*() según sea el caso.



Ejemplo

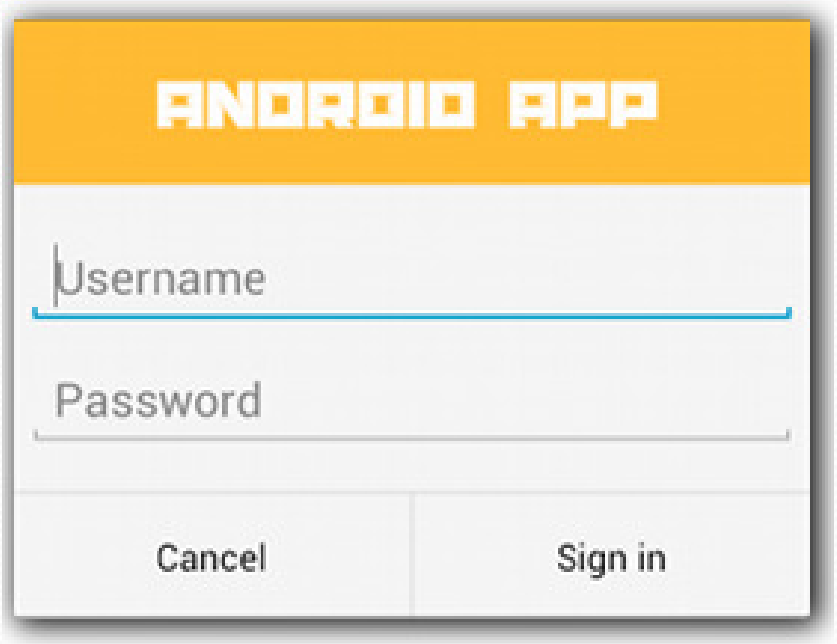
```
@Override
public Dialog onCreateDialog(Bundle savedInstanceState) {
    mSelectedItems = new ArrayList();  // Where we track the selected items
    AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
    // Set the dialog title
    builder.setTitle(R.string.pick_toppings)
    // Specify the list array, the items to be selected by default (null for none),
    // and the listener through which to receive callbacks when items are selected
           .setMultiChoiceItems(R.array.toppings, null,
                      new DialogInterface.OnMultiChoiceClickListener() {
               @Override
               public void onClick(DialogInterface dialog, int which,
                       boolean isChecked) {
                   if (isChecked) {
                       // If the user checked the item, add it to the selected items
                       mSelectedItems.add(which);
                   } else if (mSelectedItems.contains(which)) {
                       // Else, if the item is already in the array, remove it
                       mSelectedItems.remove(Integer.valueOf(which));
                   }
               }
           })
```

Ejemplo

```
    // Set the action buttons
            .setPositiveButton(R.string.ok, new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int id) {
                    // User clicked OK, so save the mSelectedItems results somewhere
                    // or return them to the component that opened the dialog
                    ...
                }
            })
            .setNegativeButton(R.string.cancel, new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int id) {
                    ...
                }
            });

    return builder.create();
}
```

Dialog con layout propio



res/layout/dialog_signin.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
    <ImageView
        android:src="@drawable/header_logo"
        android:layout_width="match_parent"
        android:layout_height="64dp"
        android:scaleType="center"
        android:background="#FFFFBB33"
        android:contentDescription="@string/app_name" />
    <EditText
        android:id="@+id/username"
        android:inputType="textEmailAddress"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:layout_marginLeft="4dp"
        android:layout_marginRight="4dp"
        android:layout_marginBottom="4dp"
        android:hint="@string/username" />
```

res/layout/dialog_signin.xml

```
    <EditText
        android:id="@+id/password"
        android:inputType="textPassword"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="4dp"
        android:layout_marginLeft="4dp"
        android:layout_marginRight="4dp"
        android:layout_marginBottom="16dp"
        android:fontFamily="sans-serif"
        android:hint="@string/password"/>
</LinearLayout>
```

Dialog

```
@Override
public Dialog onCreateDialog(Bundle savedInstanceState) {
    AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
    // Get the layout inflater
    LayoutInflater inflater = getActivity().getLayoutInflater();

    // Inflate and set the layout for the dialog
    // Pass null as the parent view because its going in the dialog layout
    builder.setView(inflater.inflate(R.layout.dialog_signin, null))
```

Dialog

```
    // Add action buttons
           .setPositiveButton(R.string.signin, new DialogInterface.OnClickListener() {
               @Override
               public void onClick(DialogInterface dialog, int id) {
                   // sign in the user ...
               }
           })
           .setNegativeButton(R.string.cancel, new DialogInterface.OnClickListener() {
               public void onClick(DialogInterface dialog, int id) {
                   LoginDialogFragment.this.getDialog().cancel();
               }
           });
    return builder.create();
}
```

Enviando eventos

- Escenario:
  - Una actividad abre un dialog para solicitar datos.
  - ¿Cómo le envía el dialog los datos a la actividad?
- Solución:
  - El dialog define una interface con métodos para el botones positivo y negativo.
  - La actividad implementa la interface y los métodos.
  - En el callback de los botones, el dialog invoca los métodos de la actividad.

Dialog

```
public class NoticeDialogFragment extends DialogFragment {

    /* The activity that creates an instance of this dialog fragment must
     * implement this interface in order to receive event callbacks.
     * Each method passes the DialogFragment in case the host needs to query it. */
    public interface NoticeDialogListener {
        public void onDialogPositiveClick(DialogFragment dialog);
        public void onDialogNegativeClick(DialogFragment dialog);
    }
```

Dialog

```
    // Use this instance of the interface to deliver action events
    NoticeDialogListener mListener;

    // Override the Fragment.onAttach() method to instantiate the NoticeDialogListener
    @Override
    public void onAttach(Activity activity) {
        super.onAttach(activity);
        // Verify that the host activity implements the callback interface
        try {
            // Instantiate the NoticeDialogListener so we can send events to the host
            mListener = (NoticeDialogListener) activity;
        } catch (ClassCastException e) {
            // The activity doesn't implement the interface, throw exception
            throw new ClassCastException(activity.toString()
                    + " must implement NoticeDialogListener");
        }
    }
    ...
}
```

Actividad

```
public class MainActivity extends FragmentActivity
                          implements NoticeDialogFragment.NoticeDialogListener{
    ...

    public void showNoticeDialog() {
        // Create an instance of the dialog fragment and show it
        DialogFragment dialog = new NoticeDialogFragment();
        dialog.show(getSupportFragmentManager(), "NoticeDialogFragment");
    }

    // The dialog fragment receives a reference to this Activity through the
    // Fragment.onAttach() callback, which it uses to call the following methods
    // defined by the NoticeDialogFragment.NoticeDialogListener interface
    @Override
    public void onDialogPositiveClick(DialogFragment dialog) {
        // User touched the dialog's positive button
        ...
    }

    @Override
    public void onDialogNegativeClick(DialogFragment dialog) {
        // User touched the dialog's negative button
        ...
    }
}
```

Dialog

```
public class NoticeDialogFragment extends DialogFragment {
    ...

    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        // Build the dialog and set up the button click handlers
        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
        builder.setMessage(R.string.dialog_fire_missiles)
               .setPositiveButton(R.string.fire, new DialogInterface.OnClickListener() {
                   public void onClick(DialogInterface dialog, int id) {
                       // Send the positive button event back to the host activity
                       mListener.onDialogPositiveClick(NoticeDialogFragment.this);
                   }
               })
               .setNegativeButton(R.string.cancel, new DialogInterface.OnClickListener() {
                   public void onClick(DialogInterface dialog, int id) {
                       // Send the negative button event back to the host activity
                       mListener.onDialogNegativeClick(NoticeDialogFragment.this);
                   }
               });
        return builder.create();
    }
}
```

Cerrar un dialog

- Es automático al escoger una opción en una lista o al oprimir un botón.
- Manualmente: *dismiss*();
- Más información: https://developer.android.com/guide/topics/ui/dialogs.html