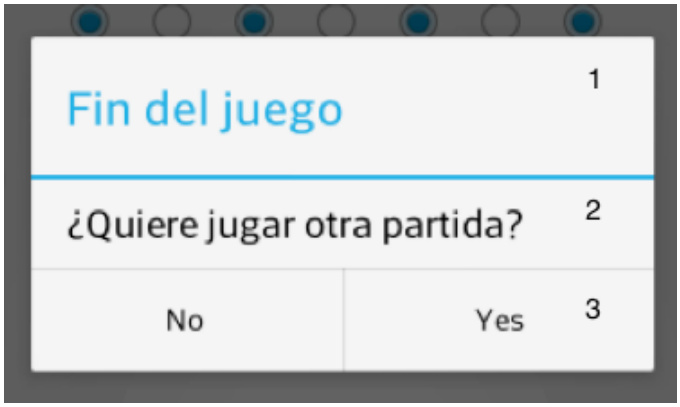
26. Diálogos

No solo podemos interactuar con los usuarios a través de actividades y menús con opciones. Cuando de lo que se trata es de pedir al usuario confirmación o de mostrar mensajes de alerta o error, los diálogos son preferibles pues se muestran como ventanas flotantes, más ligeras que una actividad. Mientras el diálogo está abierto, los usuarios solo pueden interactuar con sus botones, si bien la actividad que lo contiene sigue ejecutándose normalmente.

El SDK proporciona los siguientes tipos de diálogos:

- `Dialog`: clase base de todos los diálogos.
- `AlertDialog`: diálogo que muestra uno, dos o tres botones.
- `CharacterPickerDialog`: diálogo que permite seleccionar caracteres acentuados, por ejemplo.
- `DatePickerDialog`: diálogo que permite al usuario seleccionar y fijar una fecha.
- `TimePickerDialog`: diálogo que permite al usuario seleccionar y fijar una hora.
- `ProgressDialog`: diálogo que muestra una indicador del desarrollo de una actividad.

Un diálogo de alerta está compuesto por tres regiones como se muestra en la siguiente figura:



1. El **título** (1 en la figura) es opcional y solo se debe poner cuando el área de contenido, marcada como 2 en la figura, contiene un mensaje detallado, una lista u otras vistas añadidas por el programador. Cuando solo necesitamos mostrar un mensaje, basta con una alerta sin título.
2. El **área de contenido** (2 en la figura) puede contener un mensaje, una lista u otras vistas añadidas por el programador.
3. Los **botones de acción** (3 en la figura), que no pueden ser más de tres.

En el proyecto `CCC26` vamos a añadir a nuestro solitario un diálogo de alerta al final de cada partida para preguntar al jugador si desea echar otra o no. Para ello extenderemos la clase `DialogFragment`, que es un fragmento que muestra una ventana de diálogo flotando sobre la ventana de la actividad ligada al fragmento. Este fragmento contiene un objeto de tipo `Dialog`, cuya gestión debe hacerse a través del fragmento en lugar de hacer llamadas directas al diálogo. El código de la clase `AlertDialogFragment` es el siguiente:

`/src/AlertDialogFragment.java`

```
package es.uam.eps.android.CCC26;

import android.app.AlertDialog;
import android.app.Dialog;
import android.app.DialogFragment;
import android.content.DialogInterface;
import android.os.Bundle;

public class AlertDialogFragment extends DialogFragment{
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        final MainActivity main = (MainActivity) getActivity();

        AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(getActivity());
        alertDialogBuilder.setTitle(R.string.gameOverTitle);
        alertDialogBuilder.setMessage(R.string.gameOverMessage);
        alertDialogBuilder.setPositiveButton("Yes",
            new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int which) {
                   main.game.restart();
                   main.setFigureFromGrid();
                    dialog.dismiss();
                }
        });
        alertDialogBuilder.setNegativeButton("No",
            new DialogInterface.OnClickListener()     {
                public void onClick(DialogInterface dialog, int which) {
                    main.finish();
                    dialog.dismiss();
                }
        });
        return alertDialogBuilder.create();
    }
}
```

Es frecuente sobrescribir el método `onCreateDialog()` cuando se trata de generar diálogos adaptados por el programador. Dentro de este método callback, la forma más sencilla de generar el diálogo es instanciar un objeto de tipo `AlertDialog.Builder` y llamar a sus métodos `set` como sigue:

- `setTitle()` asigna el título de la cabecera de la ventana: `Fin del juego`.
- `setMessage()` asigna el mensaje debajo del título en el área de contenido: `¿Quiere jugar otra partida?`
- `setPositiveButton()` asigna un escuchador cuyo método `onClick()` se ejecuta cuando se pulsa el botón positivo del diálogo. En nuestro proyecto, `onClick()` llama a `restart()`de la clase `Game`, que reinicia el `grid`, y a `setFigureFromGrid()` de `MainActivity` para redibujar el tablero. Finalmente se elimina el diálogo.
- `setNegativeButton()` asigna un escuchador cuyo método `onClick()` se ejecuta cuando se pulsa el botón negativo del diálogo. En nuestro proyecto,

`onClick()` termina la actividad y elimina el diálogo, con lo que volvemos a la actividad `Initial`.

La visibilidad de `setFigureFromGrid()` ha de ser `public` para que este código funcione. Los recursos de `strings.xml` nos van a permitir más adelante adaptar los textos de los diálogos al idioma especificado en el dispositivo:

`/res/values/strings.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

<string name="app_name">CCC26</string>
    <string name="action_settings">Ajustes</string>
    ...
    <string name="gameOverTitle">Fin del juego</string>
    <string name="gameOverMessage">¿Quiere jugar otra partida?</string>
</resources>
```

La actividad `MainActivity` se encarga de instanciar y mostrar el fragmento donde antes se mostraba el mensaje mediante la clase `Toast`. Concretamente, si el juego ha terminado se invoca el método `show()`:

`/src/MainActivity.java`

```
...
public void onClick (View v){
    int id = ((RadioButton) v).getId();

    for (int i=0; i<SIZE; i++)
        for (int j=0; j<SIZE; j++)
            if (ids[i][j] == id) {
                game.play(i, j);
                break;
            }

    setFigureFromGrid();
    if (game.isGameFinished()){
        new AlertDialogFragment().show(getFragmentManager(), "ALERT DIALOG");
    }
} ...
```

El método `show()` tiene dos argumentos: el `FragmentManager` al que añadimos el fragmento y el TAG con el que se añade. Aunque no es obvio, en realidad, lo que la actividad está haciendo es una transacción como las explicadas en la unidad 24 (que no se añade al back stack, por cierto). Este es el resultado al final de una partida: