


Open Opened 2 days ago by  **Rubén Montero**

Un diálogo con cuerpo

Resumen

- Crearemos un archivo de interfaz `change_status_dialog.xml`, que contendrá únicamente un `<EditText>`
- Lo *inflaremos* mediante un nuevo método privado en `StatusActivity`. De paso, guardaremos una referencia al `EditText`
- Lo mostraremos en el diálogo
- Añadiremos un `OnClickListener` al botón del diálogo

Descripción

Existen diferentes métodos de utilidad en `AlertDialog.Builder` para mostrar *contenido* en un `AlertDialog`:

- `setMessage`: Hace que muestre un *texto*.
- `setMultiChoiceItems`: Permite mostrar opciones de selección *múltiple*.
- `setSingleChoiceItems`: Permite mostrar opciones de selección *sencilla (exclusiva)*.
- `setTitle`: Permite cambiar el título.
- `setIcon`: Permite cambiar el icono.

Desgraciadamente, ningún método nos ayudará inmediatamente a mostrar un *Campo de texto* (`EditText`).

Vamos a hacerlo manualmente con:

- `setView`: Permite mostrar un contenido totalmente personalizado.

La tarea

Crea en `res/layout/` un nuevo fichero de interfaz `change_status_dialog.xml`:

Dentro, **añade** un `<EditText>` con las siguientes características:

```
<EditText
    android:id="@+id/edit_text_change_status"
    android:hint="Hey! I have changed my status"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="40dp"/>
```

Ahora, en `StatusActivity.java`, **crea** un nuevo método privado:

```
private View inflateDialogView() {
}
```

Como ves, devuelve un tipo `View`.

Este `View` será nuestro archivo de interfaz XML *inflado* manualmente. **Implementa** el método **así**:

```
private View inflateDialogView() {
    LayoutInflater inflater = getLayoutInflater();
    View inflatedView = inflater.inflate(R.layout.change_status_dialog, null);
    return inflatedView;
}
```

A continuación, vamos a guardar una referencia al `EditText` de la vista para poder acceder al texto tecleado por el usuario más tarde.

Añade un atributo tipo `EditText` a la clase e **inicialízalo** desde el método `inflateDialogView` como se muestra a continuación:

```
public class StatusActivity extends AppCompatActivity {
    /* ... */
    + private EditText editTextPutStatus;
```

```

/* ... */

private View inflateDialogView() {
    LayoutInflater inflater = getLayoutInflater();
    View inflatedView = inflater.inflate(R.layout.change_status_dialog, null);
+   editTextPutStatus = inflatedView.findViewById(R.id.edit_text_change_status);
    return inflatedView;
}

```

Con `inflateDialogView` completado, vuelve a `onClick` e **invoca** `.setView()` para que el *diálogo* muestre el XML creado al principio de la tarea, así:

```

@Override
public void onClick(View view) {
    AlertDialog.Builder myBuilder = new AlertDialog.Builder(context);
+   myBuilder.setView(inflateDialogView());
    myBuilder.setPositiveButton("Modificar", null); // Esto añade un botón al diálogo
    AlertDialog myDialog = myBuilder.create(); // Esta línea es como 'new AlertDialog'
    myDialog.show();
}

```

Para terminnar, *aún* no vamos a mandar la petición `PUT`. Pero, cuando el usuario pulse el botón, **muestra** un `Toast` que nos verifique el texto tecleado, así:

```

myBuilder.setPositiveButton("Modificar", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
        Toast.makeText(context, "Modificar a: " + editTextPutStatus.getText().toString(), Toast.LENGTH_LONG).
    }
}); // Esto añade un botón al diálogo

```

(Como ves, para ello necesitamos reemplazar el segundo parámetro `null` de `setPositiveButton` por un `DialogInterface.OnClickListener`).

¡Prueba la app! ¿Qué tal funciona?

Por último

Sube tus cambios al repositorio en un nuevo *commit*.



[Rubén Montero @ruben.montero](#) changed milestone to [%Sprint 3](#) 2 days ago