Desarrollo Avanzado de Software

Laboratorio 5.- *Mail sender* con corrección ortográfica

Contenido:

Objetivos: Aprender a trabajar con menús, preferencias, ficheros e intents implícitos.

1 Definición de menús en la barra de herramientas

Al generar una nueva aplicación a través de Android Studio, las actividades incorporan por defecto en su interfaz gráfica una barra en la parte superior donde se muestra el nombre de la aplicación. Esta barra es conocida como ***Action Bar*** o barra de acciones.

En ocasiones querremos utilizar dicha barra (mejor dicho, el espacio que ocupa esa barra) para ofrecer distintas opciones al usuario. Esas opciones podrán tener forma de icono o estar recogidas a través de un menú flotante que mientras no sea activado estará oculto. Para estas situaciones, Android ofrece un tipo distinto de barra, la barra de herramientas o ***Toolbar***.

Para utilizar una barra de herramientas en nuestras aplicaciones, el primer paso es eliminar la barra de acciones. Para ello, haremos que el estilo de nuestra aplicación sea un estilo que no incluya ActionBar.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.laboratorio5">

    <application
        ...
        android:theme="@style/Theme.AppCompat.Light.NoActionBar">
        ...
    </application>
</manifest>
```

A continuación habrá que añadir un elemento Toolbar al layout de la actividad.

```
<androidx.appcompat.widget.Toolbar
    android:id="@+id/labarra"
    android:layout_width="match_parent"
    android:layout_height="?attr/actionBarSize"
    android:background="#64F353"
    android:elevation="4dp"
    ...
/>
```

Para cumplir las reglas de Material Design

Para definir los elementos que se deben mostrar en el menú, se utiliza un fichero xml que se almacena en la carpeta ***res/menu***.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:android="http://schemas.android.com/apk/res/android">

    <item
        android:id="@+id/opcion1"
        android:icon="@android:drawable/btn_star_big_on"
        android:title="Favorito"
        app:showAsAction="ifRoom|withText" />
...
</menu>
```

Cada elemento de menú

Cómo mostrar el elemento

En el fichero xml se usa el elemento *item* para definir cada una de las entradas del menú. Cada entrada tiene un id, puede tener un icono y un texto (título) a mostrar. El atributto showAsAction indica cómo se debe mostrar dicho elemento en el menú:

- ifRoom: muestra el elemento en la barra si hay espacio. Si no hay espacio lo muestra en el menú oculto.
- withText: muestra el texto (atributo título) del elemento al lado del icono
- never: nunca lo pone en la barra, siempre en el menú oculto

- always: siempre lo muestra en la barra. Cuidado al usarlo porque puede que no entren todos los elementos.

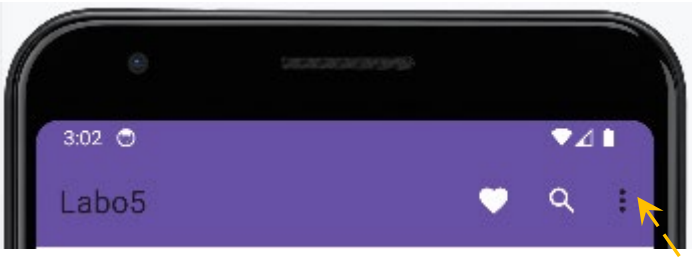

Fig. 1.- Menú en la barra de herramientas

Para asignar el fichero xml con la definición del menú a la ***Toolbar***, hay que sobreescribir el método `onCreateOptionsMenu` en la actividad:

Nombre del fichero xml con el menú

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.definicion_menu,menu);
    return true;

}
```

Para que la ***Toolbar*** incluida en el fichero layout de la actividad sea considerada como barra de acciones y poder configurarla acorde a nuestras necesidades, hay que cogerla de layout y asignarla como ***ActionBar*** :

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    …
    setSupportActionBar(findViewById(R.id.labarra);
    …
}
```

Para reaccionar ante la interacción del usuario con cualquiera de las opciones del menú, hay que sobreescribir en la actividad el método onOptionsItemSelected, donde obtendremos id correspondiente al elemento del menú seleccionado y trabajaremos sobre ello. Para usar el switch, los ids tienen que ser constant y hay que añadir `android.nonFinalResIds=false` en el fichero gradle.properties. La alternativa es usar if else if en vez de un la estructura switch:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id=item.getItemId();
    switch (id){
        case R.id.opcion1:{
            …
        }
        …
    }
    return super.onOptionsItemSelected(item);
}
```

> Si queremos que varias de las actividades compartan la misma Toolbar, podemos definir una actividad con la Toolbar y las acciones correspondientes y que el resto de actividades de nuestra aplicación hereden de esa actividad en vez de la clase AppCompatActivity.

2 Envío de emails con corrección ortográfica

2.1 Funcionamiento

La aplicación que se va a implementar va a permitir al usuario escribir un email, pero antes de enviarlo va a comprobar el texto introducido contra un diccionario personal, donde el usuario podrá ir añadiendo las palabras que desee.

El diccionario personal será un fichero de texto donde el usuario añadirá las palabras que quiere que sean consideradas correctas. Las palabras que no estén contenidas en dicho fichero serán consideradas erróneas.

- Implementad una aplicación que permita al usuario introducir un destinatario, un asunto y un texto para enviar un email. Cuando intente enviar el email, el texto se comprobará mediante su diccionario personal, y sólo se enviará si es completamente correcto. El usuario tendrá la opción de comprobar el texto previamente, y se le mostrará mediante un código de colores qué palabras no se encuentran en el diccionario. Seguid el prototipo de interfaz que se describe a continuación.

2.2 Prototipo de la interfaz

La pantalla principal (Fig. 2) tendrá 3 campos de texto, uno para el destinatario del correo electrónico, otro para el asunto y un tercero para introducir el texto.

El menú de la pantalla principal tendrá 4 opciones. Dos de ellas se mostrarán en forma de botones en la ***Toolbar*** y las otras dos se mostrarán en el menú oculto. El icono en forma de ojo (Fig. 3) se corresponde con la opción de "Comprobar". Esta opción marcará en el texto qué palabras son las que no se encuentran en el diccionario.

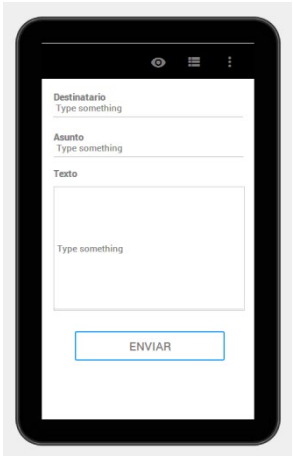

Fig. 2.- Interfaz principal

Fig. 3.- Menú de la pantalla principal

Desarrollo Avanzado de Software

En esta práctica, el menú tendrá dos opciones a la vista:

- Un icono con forma de ojo que se corresponde con el recurso ***android:drawable/ic_menu_view.***

  Se mirará si las palabras están en el diccionario personalizado, y se marcarán si no lo están. Los colores y formato de marcado se configurarán en la opción de menú ***Ajustes***.

- Un icono con forma de "listado" que se corresponde con el recurso ***Android:drawable/ic_menu_agenda.***

  El icono con forma de listado, se corresponde con la opción de "Corregir", que permitirá al usuario añadir aquellas palabras mostradas como incorrectas (porque no están en el diccionario) a su diccionario personal. Por cada palabra que no esté contemplada en el diccionario, saldrá un diálogo preguntando si se quiere añadir esa palabra al diccionario personal o ignorarla. Si se opta por añadirla al diccionario personal, habrá que incluirla en el fichero del diccionario personal.

El menú oculto (Fig. 3) tendrá, a su vez, otras 2 opciones:

- "Ajustes"→ Se abrirá una actividad de preferencias donde se pueda configurar el color y el formato (negrita, cursiva...) para marcar aquellas palabras que no están en el diccionario.
- Mostrar Diccionario Personal → Mostrará al usuario el listado con las palabras que se encuentran definidas en el fichero del diccionario personal del usuario. Además, pulsando sobre una de las palabras del diccionario personal, se podrá eliminar la palabra.

Para modificar el estilo de un texto se utilizan los objetos de tipo *Editable*. Mediante su operación `setSpan` se le asigna el estilo que se desea al texto situado entre las posiciones de comienzo y fin que se definan.

```
Editable stringconestilo = Editable.Factory.getInstance().newEditable(unstring);
stringconestilo.setSpan(new StyleSpan(Typeface.BOLD), posinicio, posfin,
                        Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
stringconestilo.setSpan(new ForegroundColorSpan(Color.RED), posinicio, posfin,
                        Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
```

Pondría el texto de "unstring" que se encuentre entre las posiciones "posinicio" y "posfin" en rojo y en negrita.
El texto con el formato estaría en "stringconestilo".

Para abrir un cliente de correo desde nuestra aplicación, usaremos un ***intent*** implícito con la acción ACTION_SENDTO:

```
String[] email = "iker.sobron@ehu.eus";
String subject = "DAS";
String body = "Hola Iker";
Intent emailIntent = new Intent(Intent.ACTION_SENDTO);
emailIntent.setdata(Uri.parse("mailto:"));
emailIntent.putExtra(Intent.EXTRA_EMAIL, email);
emailIntent.putExtra(Intent.EXTRA_SUBJECT, subject);
emailIntent.putExtra(Intent.EXTRA_TEXT, body);
startActivity(Intent.createChooser(emailIntent, "Enviar e-mail"));
```