Introducción a Android

Sesión 4

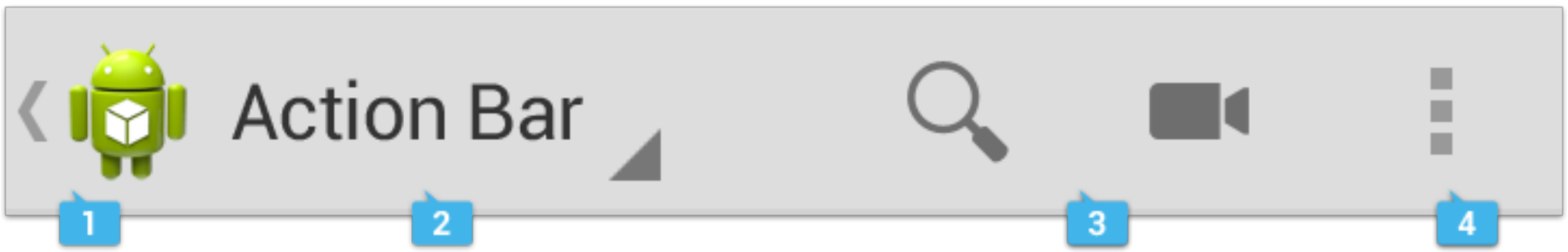
¿Qué vamos a ver hoy día?

- ActionBar
- Cuadros de diálogo

Action Bar

- Componente gráfico de una aplicación Android que brinda las siguientes facilidades:
    - Un espacio dedicado a la identidad de la aplicación y a indicar la ubicación del usuario en la misma
    - Un patrón de navegación y cambio de vistas consistente en todas las apps
    - Permite resaltar la funcionalidad más importante de la aplicación dándole fácil acceso, mientras que la funcionalidad menos accedida se oculta tras un menú

Action Bar: componentes



1. Icono: define la identidad de la aplicación.
   Puede indicar también que estamos uno o más niveles de navegación hacia adentro

2. Control de vistas: indica nuestra ubicación en la app.
   También puede permitirnos cambiar de vista en una misma pantalla

3. Botones de acción: muestran las acciones más importantes en la pantalla

4. Menú de acciones: agrupa las acciones menos utilizadas

Cómo implementarla

- La Action Bar está disponible desde la **versión 3.0 (API level 11)**.
  Para versiones anteriores, se debe utilizar la *Support Library*

- Para mostrar la Action Bar en Android 2.1 (API level 7) o superior:

  1. Extender la clase `ActionBarActivity`

  2. Usar uno de los temas `Theme.AppCompat`

- A partir de Android 3.0 (API level 11), el Action Bar viene incluido por defecto

Acceso desde código Java

- Android 2.1 (API level 7) o superior

```
import android.support.v7.app.ActionBar;
…
ActionBar actionBar = getSupportActionBar();
```

- Android 3.0 (API level 11) o superior

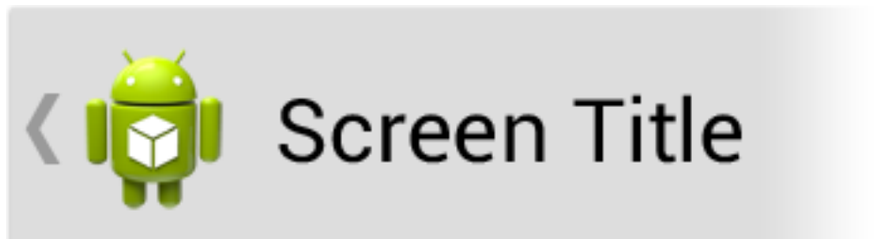
```
import android.app.ActionBar;
…
ActionBar actionBar = getActionBar();
```

Añadir Action Items

- Definir items y su aspecto: `res/menu/*.xml`:
    - Permite definir id, icono, título y si se muestra en el Action Bar o en el menú
- Asociar menú a un Activity: `onCreateOptionsMenu`
- Definir acciones de los ítems del menú: `onOptionsItemSelected`

Navegación

- Podemos utilizar el Action Bar para indicar los niveles de navegación de nuestra aplicación y permitir al usuario retroceder al nivel superior



- Diferente al botón Atrás:
  - Botón Atrás (en el dispositivo): retroceder entre las pantallas visitadas en orden cronológico
  - Navegación con Action Bar: jerarquía dentro de la aplicación, definida por nosotros (p.ej., listado y detalle)

Navegación - implementación

- Habilitar el botón de navegación: `ActionBar.setDisplayHomeAsUpEnabled()`

- Definir nivel superior: 2 formas:

1. En el archivo `AndroidManifest.xml`:

```
    <activity
        android:name="com.example.myfirstapp.DisplayMessageActivity"
        android:label="@string/title_activity_display_message"
        android:parentActivityName="com.example.myfirstapp.MainActivity" >
        <!-- Parent activity meta-data to support API level 7+ -->
        <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value="com.example.myfirstapp.MainActivity" />
    </activity>
</application>
```

Navegación - implementación

2. En código Java: `getSupportParentActivityIntent()` y `onCreateSupportNavigateUpTaskStack()`:

   - Aplica cuando el Activity padre se define dinámicamente

   - `getSupportParentActivityIntent()`: cuando la navegación es dentro de la app.
     Se crea el Intent que abrirá el Activity padre

   - `onCreateSupportNavigateUpTaskStack()`: cuando la navegación viene desde otra app.
     Se fabrica un `TaskStack` que replica la jerarquía de nuestra app

Action View

- Componente más elaborado que puede ir en lugar de un botón simple en el Action Bar (p.ej. un cuadro de búsqueda)

- Se define con el atributo `actionViewClass` en el XML donde se define el menú del Action Bar:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:yourapp="http://schemas.android.com/apk/res-auto" >
    <item android:id="@+id/action_search"
          android:title="@string/action_search"
          android:icon="@drawable/ic_action_search"
          yourapp:showAsAction="ifRoom|collapseActionView"
          yourapp:actionViewClass="android.support.v7.widget.SearchView" />
</menu>
```

Action View - código Java

- Desde `onCreateOptionsMenu` se puede obtener una referencia al ActionView como un objeto Java (p.ej. para añadir listeners)

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.main_activity_actions, menu);
    MenuItem searchItem = menu.findItem(R.id.action_search);
    SearchView searchView = (SearchView) MenuItemCompat.getActionView(searchItem);
    // Configure the search info and add any event listeners
    ...
    return super.onCreateOptionsMenu(menu);
}
```

- Android 3.0 (API level 11) en adelante: se puede emplear directamente `MenuItem.getActionView:`

```
SearchView searchView = (SearchView) menuItem.getActionView();
```

Cuadros de diálogo

- Se implementan con las clases `DialogFragment` y `AlertDialog`

- Pasos:

  - Extender `DialogFragment` y sobreescribir `onCreateDialog`

  - Configurar el diálogo con los métodos de `AlertDialog.Builder`

  - Llamar al método `show` del `DialogFragment`

Cuadros de diálogo

```
public class MensajeDialogFragment extends DialogFragment {
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        // Se usa la clase Builder para construir diálogos predefinidos
        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) {
                       // Código para cuando el usuario ha seleccionado OK
                   }
               })
               .setNegativeButton(R.string.cancel, new DialogInterface.OnClickListener() {
                   public void onClick(DialogInterface dialog, int id) {
                       // Código para cuando el usuario ha seleccionado Cancel
                   }
               });
        // Finalmente, llamamos al método create para crear el diálogo
        return builder.create();
    }
}
```

Cuadros de diálogo - interacción con el usuario

- 3 botones posibles para un AlertDialog: positivo, negativo y neutral
  - `AlertBuilder.setPositiveButton`, `.setNegativeButton` y `.setNeutralButton`
- Si queremos ofrecer una lista de opciones: `setItems` o `setSingleChoiceItems`
- Para implementar las acciones: `DialogInterface.OnClickListener`
  - `onClick` recibe el indice seleccionado en la lista o una constante que indica el botón presionado

Cuadros de diálogo

- Opciones de selección múltiple: `setMultiChoiceItems` y `DialogInterface.OnMultiChoiceClickListener`
  - Se dispara un evento cada vez que se marca o desmarca una opción

- Mostrar una interfaz unas veces como diálogo y otras como pantalla completa:
  - Usar `onCreateView`; no usar `AlertDialog`
  - Recordar que se puede utilizar como dialogo o como Fragment