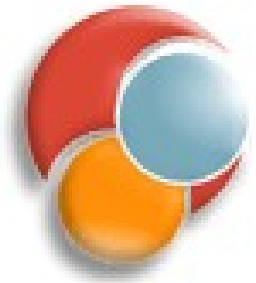


Desarrollo de Aplicaciones para Android

Sesión 2: Interfaz de usuario

Puntos a tratar

- Views
- Layouts
- Eventos
- Activities e Intents
- Menús y preferencias

Interfaces de usuario

- API que proporciona componentes de alto nivel
  - Views: botones, etiquetas, campos de texto, de fecha, hora, barras de progreso, etc.
  - Layouts
  - Eventos
  - Actividades
  - Menús y preferencias
- API para dibujar y capturar eventos a bajo nivel

Views, Layouts, Actividades

- Las actividades muestran un layout.
- Los layouts se pueden anidar y muestran views (botones, etiquetas, etc).
- Se usan los atributos `layout_width` y `layout_height` para controlar el tamaño del layout.

Layout

- Mostrar un Layout desde una actividad:

```
public class Interfaces extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ((TextView)findViewByID(R.id.TextView01)).setText("Hola Android");
        }
}
```

- `res/layouts/main.xml`:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="wrap_content">
<TextView
        android:text="Hola Android"
        android:id="@+id/TextView01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        />
</LinearLayout>
```

Layouts

- Layouts básicos de Android
  - **FrameLayout**: coloca todos los elementos en la esquina superior izquierda
  - **LinearLayout**: dispone todos los elementos en una única fila o columna
  - **TableLayout**: organiza los elementos en forma de tabla con varias filas y columnas
  - **RelativeLayout**: layout flexible para la colocación de elementos en posiciones relativas a otros
  - **Gallery**: lista horizontal de vistas con scroll
- *http://developer.android.com/guide/topics/ui/layout-objects.html*

Vistas

- Asignar una interfaz gráfica a una actividad

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    setContentView(R.layout.milayout);
    TextView miTexto = (TextView)findViewById(R.id.texto);
}
```

  - */res/layout/milayout.xml*
  - *setContentView()*: asignación de la interfaz a la actividad
    - Parámetro: identificador de recurso u objeto *View*
  - *findViewById()*: acceso a las vistas desde el código
    - Parámetro: el atributo *android:id* de la vista en el

Vistas

- Asignar una interfaz gráfica a una actividad mediante código

```
@Override
public void onCreate(Bundle sagedInstaceState) {
    super.onCreate(savedInstanceState);

    textView texto = new TextView(this);
    setContentView(texto);

    texto.setText("Hola Mundo!");
```

- En lugar de un layout, la actividad muestra un TextView

Spinner

- Cuadro de selección múltiple
- Pasos:
  - Añadir objeto *Spinner* al layout
  - Recursos: *prompt* y array de opciones
  - Asignar array de opciones al *Spinner*

Spinner

- Añadir objeto *Spinner* al layout

```
<Spinner
        android:id="@+id/spinner"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:drawSelectorOnTop="true"
        android:prompt="@string/eligeopcion"
    />
```

- Cadena para el *prompt* en */res/values/strings.xml*

```
<string name="eligeopcion">Elige!</string>
```

Spinner

- Array de opciones en */res/values/arrays.xml*

```
<resources>
    <string-array name="opciones">
        <item>Mensual</item>
        <item>Trimestral</item>
        <item>Semestral</item>
        <item>Anual</item>
    </string-array>
</resources>
```

```
Spinner s = (Spinner) findViewById(R.id.spinner);
ArrayAdapter adaptador = ArrayAdapter.createFromResource(this,
R.array.opciones, android.R.layout.simple_spinner_item);
adaptador.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_
item);
s.setAdapter(adaptador);
```

Spinner

- Eventos
  - No es posible asignar eventos a elemenos individuales

```
spinner.setOnItemSelectedListener(new OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView parentView, View
selectedItemView, int position, long id) {
        // Código
    }

    @Override
    public void onNothingSelected(AdapterView parentView) {
        // Código
    }

});
```

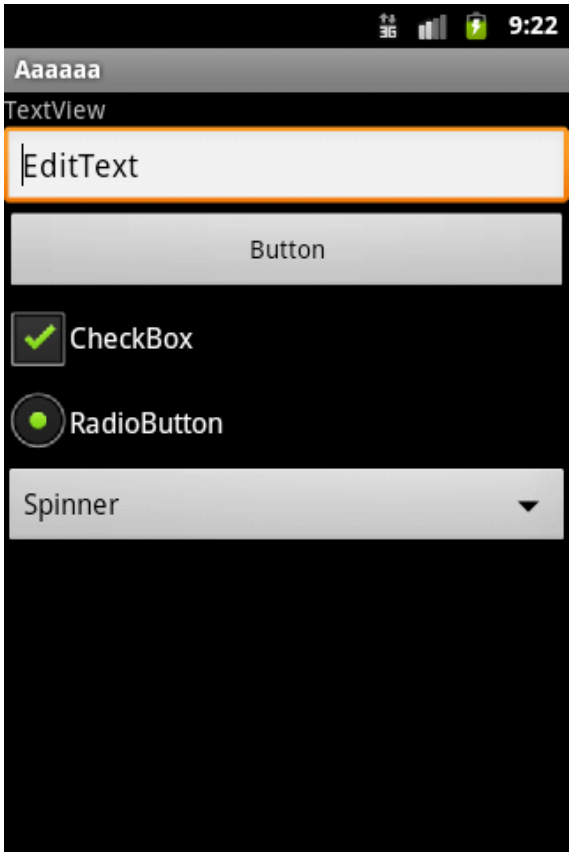
Vistas básicas

- `TextView`, etiqueta de texto.
- `EditText`, campo de texto.
- `Button`, botón pulsable con etiqueta de texto.
- `ListView`, grupo de views que los visualiza en forma de lista vertical.
- `Spinner`, lista desplegable, internamente es una composición de TextView y de List View.
- `CheckBox`, casilla marcable de dos estados.
- `RadioButton`, casilla seleccionable de dos estados, donde un grupo de RadioButtons sólo permitiría seleccionar uno de ellos al mismo tiempo.

Vistas básicas

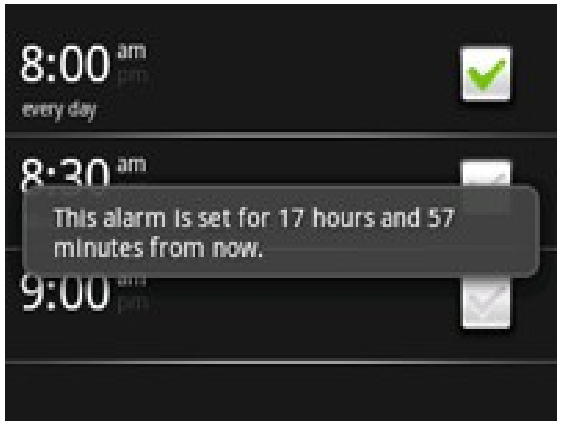
- `ViewFlipper`, un grupo de Views que nos permite seleccionar qué view visualizar en este momento.
- `ScrollView`, permite usar barras de desplazamiento. Sólo puede contener un elemento, que puede ser un Layout (con otros muchos elementos dentro).
- `DatePicker`, permite escoger una fecha.
- `TimePicker`, permite escoger una hora.
- Otros más avanzados como `MapView` (vista de Google Maps) y `WebView` (vista de navegador web), etc.
- Más información en http://developer.android.com/resources/tutorials/views/index.html

Vistas básicas



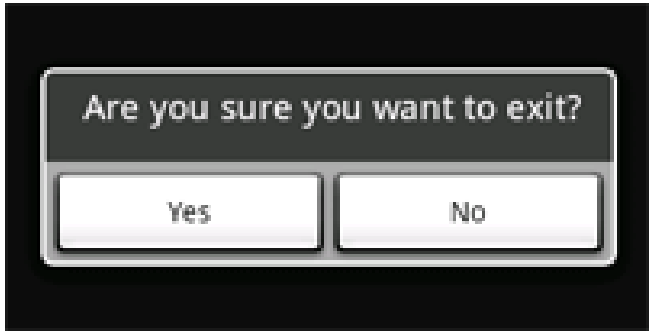
Tostadas

- Muestran información de poca importancia, durante unos instantes, de manera poco intrusiva.




```
Toast.makeText(MiActividad.this,
               "Preferencia de validación actualizada",
               Toast.LENGTH_SHORT).show();
```

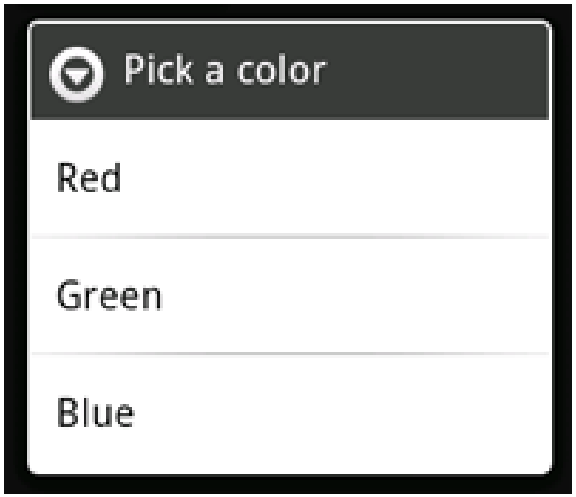
Alert Dialog




```
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setMessage("Are you sure you want to exit?")
       .setCancelable(false)
       .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
           public void onClick(DialogInterface dialog, int id) {
                MyActivity.this.finish();
           }
       })
       .setNegativeButton("No", new DialogInterface.OnClickListener() {
           public void onClick(DialogInterface dialog, int id) {
                dialog.cancel();
           }
       });
AlertDialog alert = builder.create();
```

Alert Dialog

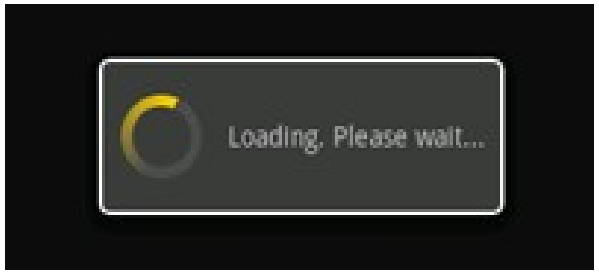
- Seleccionar de una lista:




```
final CharSequence[] items = {"Red", "Green", "Blue"};
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setTitle("Pick a color");
builder.setItems(items, new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int item) {
        Toast.makeText(getApplicationContext(), items[item],
                                        Toast.LENGTH_SHORT).show();
    }
});
AlertDialog alert = builder.create();
```

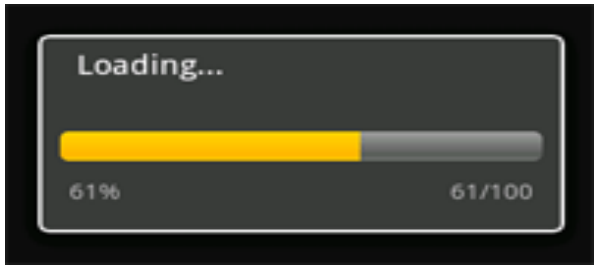
Progess Dialog

- Progeso indefinido




```
ProgressDialog dialog = ProgressDialog.show(MyActivity.this, "",
                "Loading. Please wait...", true);
```

- Progreso definido




```
ProgressDialog progressDialog;
progressDialog = new ProgressDialog(mContext);
progressDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
progressDialog.setMessage("Loading...");
progressDialog.setCancelable(false);
```

InputFilter para los EditText

- Filtrar o modificar el contenido de un `EditText` conforme se va introduciendo el texto.
- Filtros predefinidos:
  - `LengthFilter`: limita el número de caracteres
  - `AllCaps`: transforma los caracteres a mayúsculas
- Filtros propios:
  - `new InputFilter( ) { ... }`

InputFilter para los EditText

- Asignar los filtros a un campo de texto

```
EditText editText =
(EditText)findViewById(R.id.EditText01);
InputFilter[] filters = new InputFilter[3];
filters[0] = new InputFilter.LengthFilter(9);
filters[1] = new InputFilter.AllCaps();
filters[2] = new InputFilter( ) { … } // Filtro propio
dniEditText.setFilters(filters);
```

InputFilter para los EditText

```
filters[2] = new InputFilter() {
    public CharSequence filter(CharSequence source, int start, int end,
        Spanned dest, int dstart, int dend) {
    if (end > start) {
        String destTxt = dest.toString();
        String resultingTxt = destTxt.substring(0, dstart) +
                source.subSequence(start, end) + destTxt.substring(dend);
        if (!resultingTxt.matches("^[A-F0-9]*$")) {
         if (source instanceof Spanned) {
             SpannableString sp = new SpannableString("");
             return sp;
         } else {
             return "";
         }
        }
    }
    return null;
    }
};
```

Layouts

- `LinearLayout`, dispone los elementos uno después del otro.
- `FrameLayout`, dispone cualquier elemento en la esquina superior izquierda.
- `RelativeLayout`, dispone los elementos en posiciones relativas con respecto a otros, y con respecto a las fronteras del layout.
- `TableLayout`, dispone los elementos en forma de filas y columnas.
- `Gallery`, dispone los elementos en una única fila desplazable.

Eventos

```
ImageButton imageButton = (ImageButton)findViewById(R.id.ImageButton01);
imageButton.setOnClickListener(new OnClickListener() {

                @Override
                public void onClick(View v) {
                        Toast.makeText(getApplicationContext(),
                          "Gracias por pulsar.",
                          Toast.LENGTH_SHORT).show();

                }
        });
```

Actividades

- `Activity`: tarea destinada a mostrar una interfaz gráfica al usuario.
- Sólo podemos ver en pantalla una actividad a la vez.
- Una aplicación suele estructurarse en un conjunto de actividades.
- Una aplicación puede mostrar actividades de otras aplicaciones o actividades nativas del sistema (por ejemplo, la de enviar SMS).

Ciclo de vida de las actividades

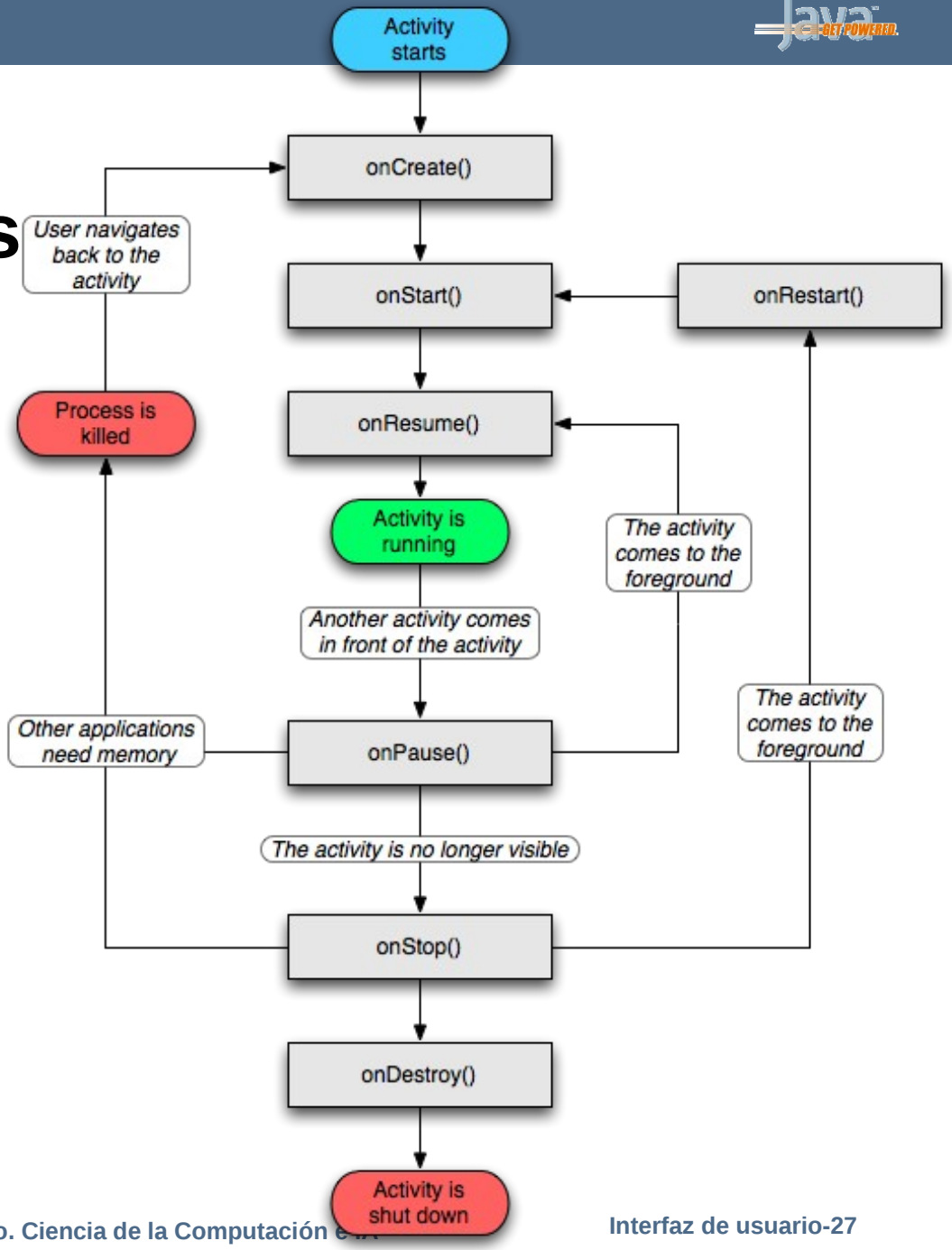
- Se define sobrecargando los siguientes métodos:

```
public class Activity extends ApplicationContext {
    protected void onCreate(Bundle savedInstanceState);
    protected void onStart();
    protected void onRestart();
    protected void onResume();
    protected void onPause();
    protected void onStop();
    protected void onDestroy();

}
```

Ciclo de vida de las actividades

- Ciclo de vida entre
  - `onCreate( )`
  - `onDestroy( )`
- Tiemo de vida visible entre
  - `onStart( )`
  - `onStop( )`
- Tiempo de vida en primer plano entre
  - `onResume( )`
  - `onPause( )`
  - ¡puede ser terminada!



Intents

- `Intent`, *propósito*. Es una descripción abstracta de una operación a realizar.
- Los utilizaremos para pasar de una actividad a otra.
- Usos:
  - Con `startActivity` para lanzar una actividad
  - Con `broadcastIntent` para enviarse a cualquier componente receptor `BroadcastReceiver`.
  - Con `startService` o `bindService` para comunicar con un servicio (`Service`) que corre en segundo plano.

Intents: lanzar actividades

- Lanzar una actividad propia:

```
Intent intent = new Intent(this, MiActividad.class);
startActivity(intent);
```

- Lanzar una actividad del sistema:

```
Intent intent = new Intent(Intent.ACTION_DIAL, Uri.parse("tel:965903400"));
startActivity(intent);
```

Intents

- Otras acciones del sistema
  - ACTION_ANSWER
  - ACTION_CALL
  - ACTION_DELETE
  - ACTION_DIAL
  - ACTION_EDIT
  - ACTION_INSERT
  - ACTION_PICK
  - ACTION_SEARCH
  - ACTION_SENDTO
  - ACTION_VIEW
  - ACTION_WEB_SEARCH

Ejemplo: abrir un visor de Google Maps

- Lanzar un intent:

```
Intent intent = new Intent(android.content.Intent.ACTION_VIEW,
Uri.parse("http://maps.google.com/maps?
saddr=20.344,34.34&daddr=20.5666,45.345"));
startActivity(intent);
```

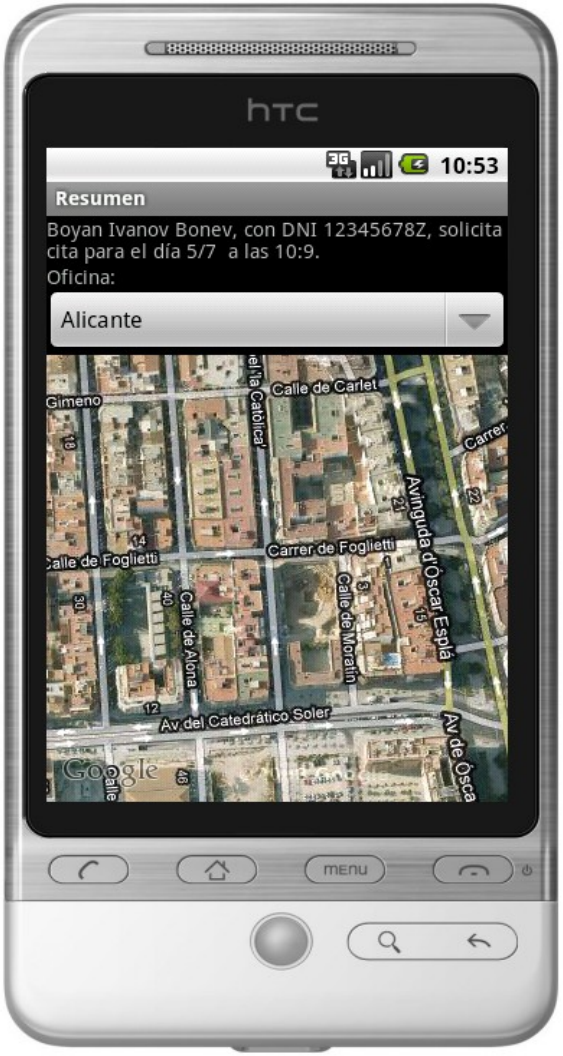
- Crear una actividad con un MapView
  - Requiere clave de API

Obtener clave de API de depuración

- Clave a partir de la huella digital MD5 del certificado digital que firma nuestras aplicaciones.
- Para el desarrollo será suficiente con que utilicemos el certificado debug.keystore, cuya ruta está indicada en Eclipse en las preferencias de Android Build.
- keytool -list -keystore debug.keystore
- http://code.google.com/android/add-ons/google-apis/maps-api-signup.html nos muestra la clave de la API, la copiamos la guardamos.

MapView

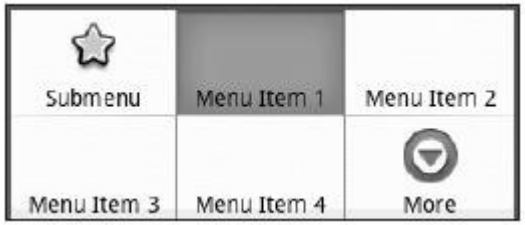
- En el layot, en el componente MapView, introducimos la clave
- También se debe añadir al AndroidManifest.xml la librería de google maps y permisos para Internet.
- En el método onCreate() se puede crear un MapController a partir del MapView para poder moverlo, hacer zoom, etc.



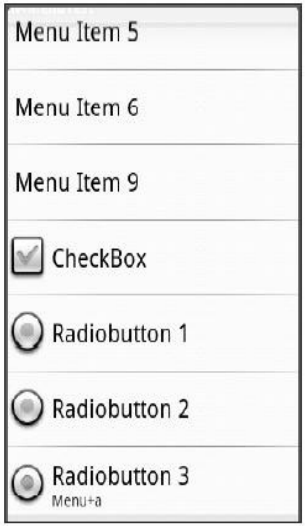
Menús

- Tipos de menús
  - Icon menus: parte inferior de la pantalla
  - Menús expandidos: al pulsar “más”
  - Submenús: ventanas flotantes
  - Menús contextuales: tras pulsación larga

Menús



- Tres etapas
  - **Menú de iconos**
    - Iconos y texto
    - Seis opciones como máximo (más opciones producirá un menú extendido)
    - No muestra checkboxes, botones de radio c
  - **Menú extendido**
    - Listado vertical con barra de scroll
    - Opciones que no caben en el menú de icon
    - No muestra iconos, pero sí checkboxes, rad atajos
  - **Submenú**
    - Ventana flotante
    - No se permiten submenús anidados



Icon menu

- res/menu/menu.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<menu
  xmlns:android="http://schemas.android.com/apk/res/android">
<item android:title="Preferencias" android:id="@+id/item01"></item>
<item android:title="Acerca de..." android:id="@+id/item02"></item>
</menu>
```

- Para que el menú se abra:

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

         }
```

Respuesta a eventos del menú

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
        switch(item.getItemId()){
        case R.id.item01:
                break;
        case R.id.item02:
                break;
        }
        return true;
}
```

Menús contextuales

- En el View al que se refiere el menú contextual:

```
@Override
public void onCreateContextMenu(ContextMenu m){
        super.onCreateContextMenu(m);
        m.add("ContextMenuItem1");
}
```

- Alternativa: registrar sólo en esta actividad, para que no se despliegue en todas

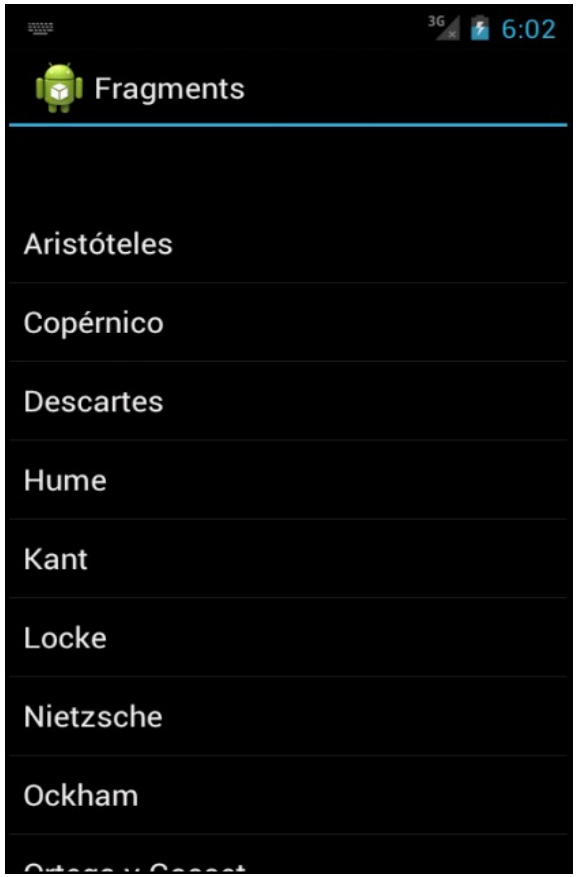
```
registerForContextMenu(view)
```

Fragmets

- A partir de Android 3.0 / 4.0
- Intención de aprovechar la misma aplicación para tablets y para pantallas pequeñas, tanto en apaisado como en vertical.
- Se declara a través de los layout XML y permite mostrar varias actividades en una misma pantalla.

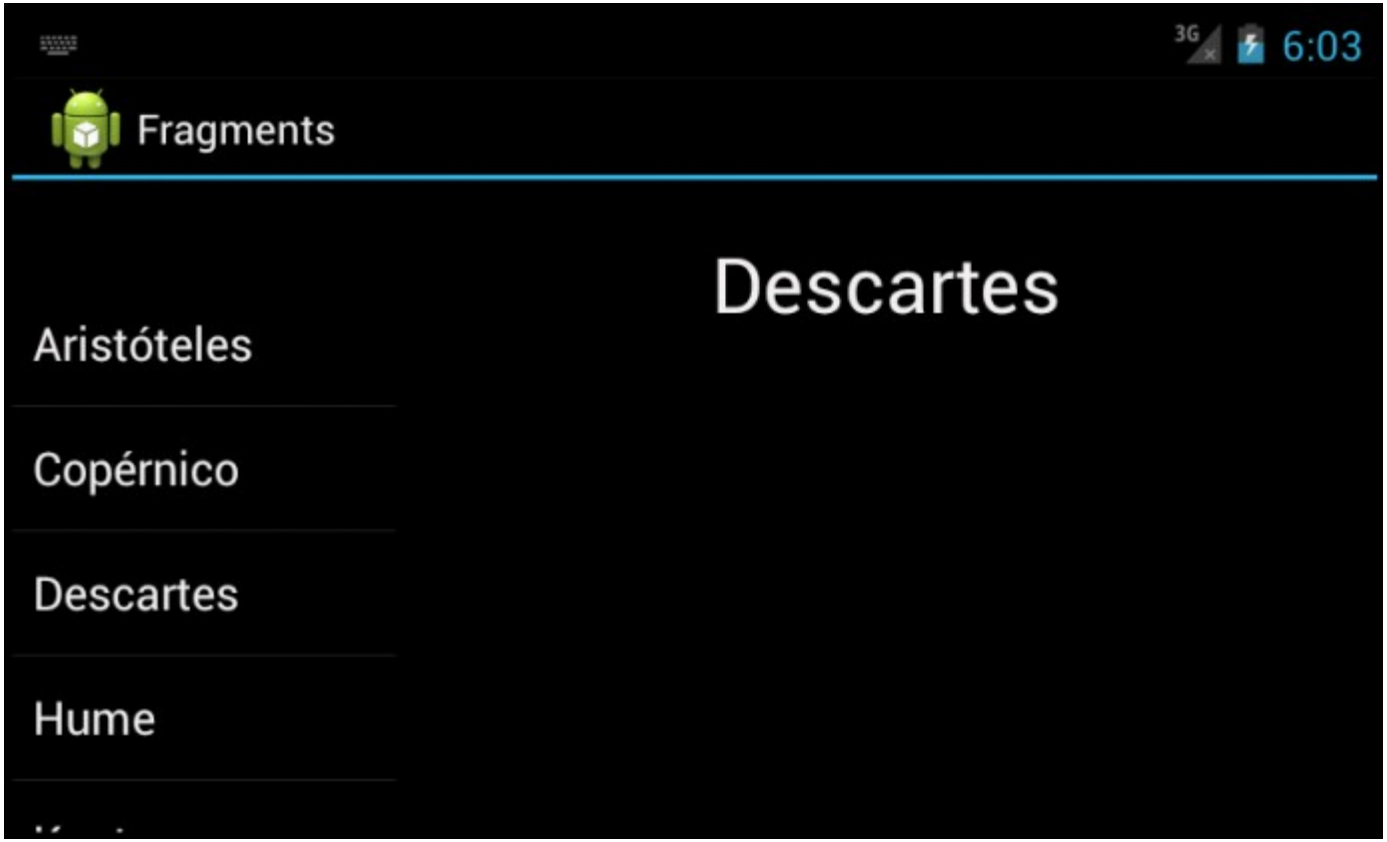
Fragments

- Vertical

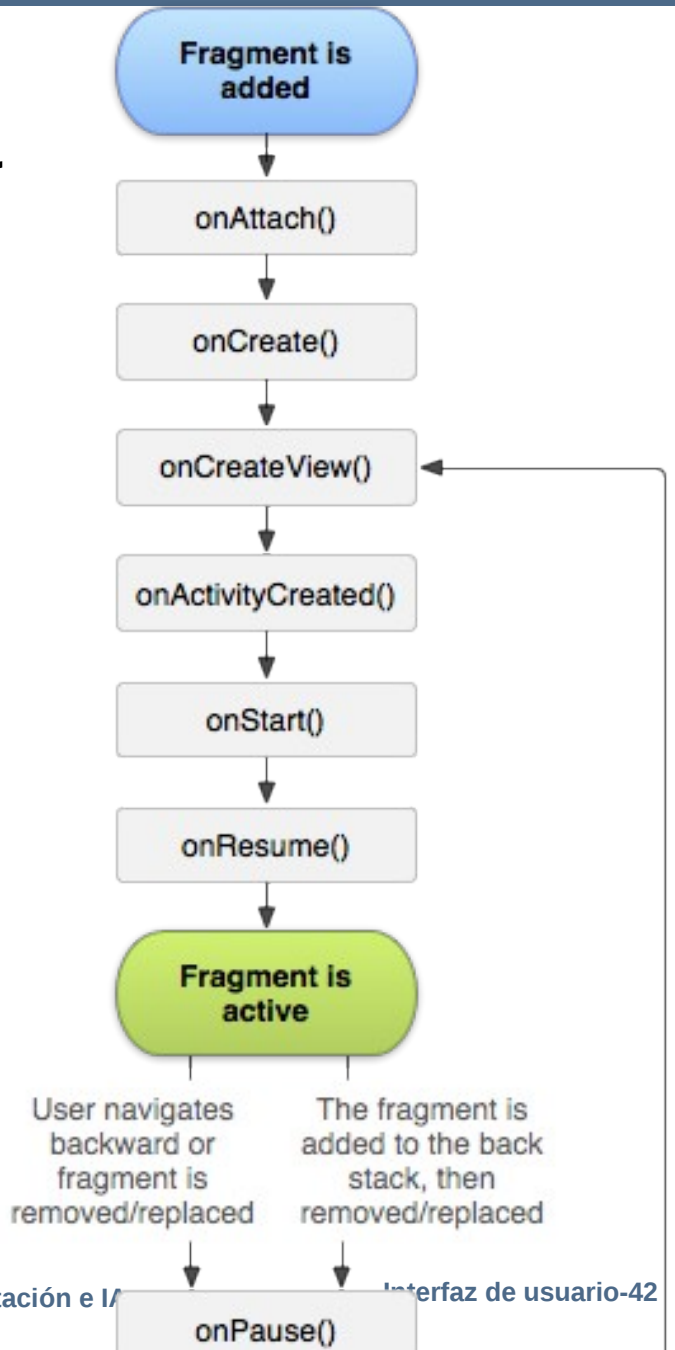


Fragments

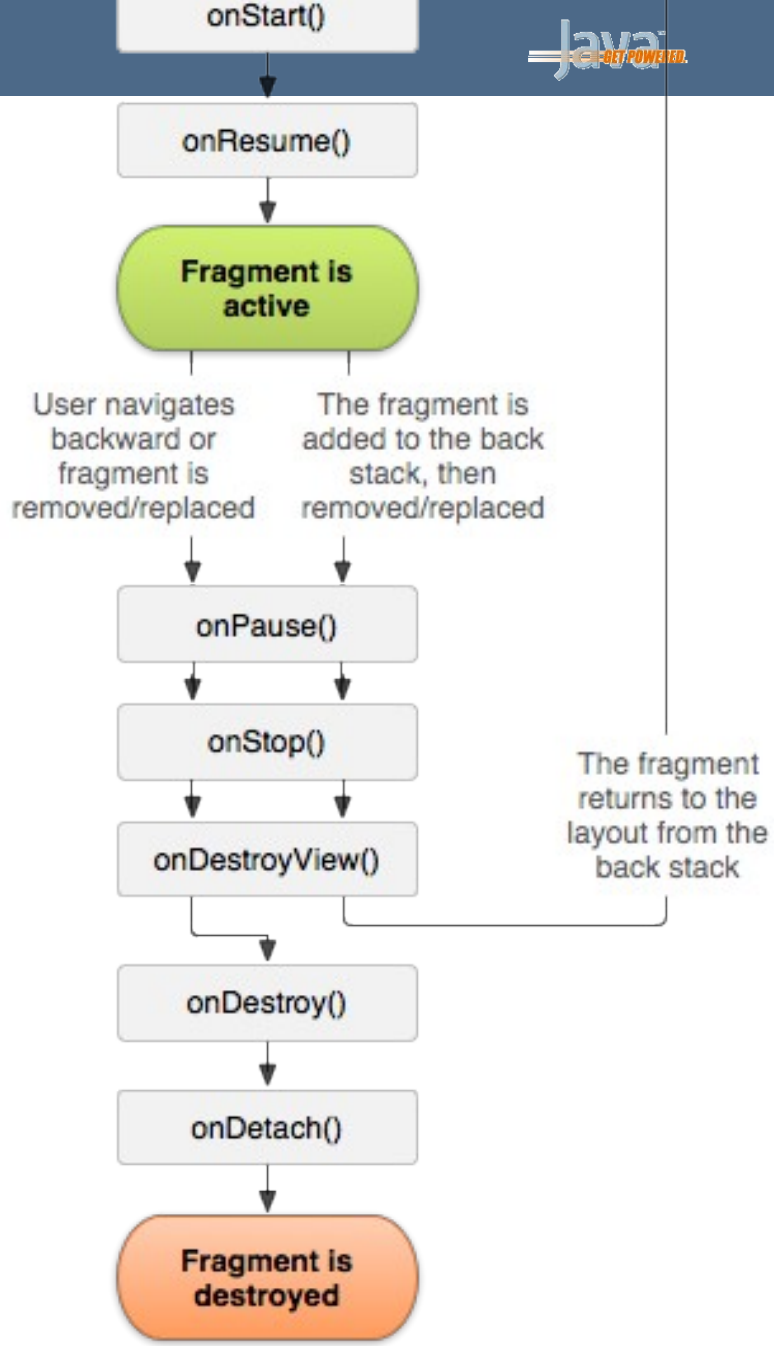
- Apaisado

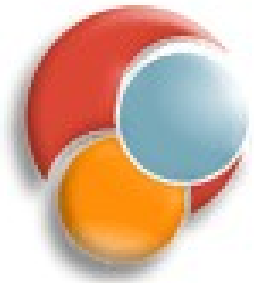


Fragments: ciclo de vida



Fragments: ciclo de vida





¿Preguntas...?