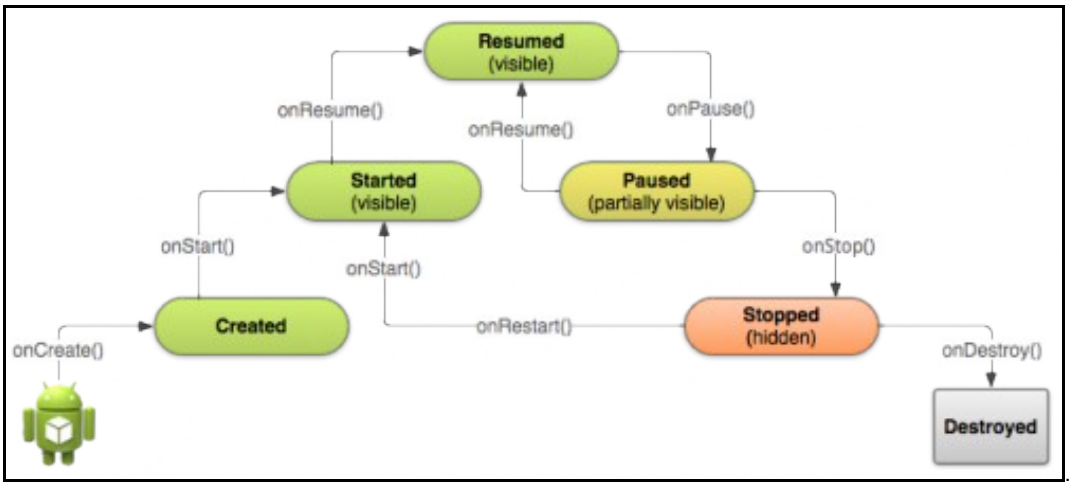
Ciclo de vida dunha aplicación

Sumario

Introdución

- Unha **Activity (Actividade)** é un elemento de Android que amosa unha pantalla con elementos (Vistas: botóns, textos, imaxes, etc) cos que os usuarios poden interactuar: chamar por teléfono, navegar por internet, realizar cálculos nunha calculadora.
- Unha actividade, xeralmente, ocupa toda a pantalla, aínda que pode ser menor ca esta ou flotar sobre outras ventás.
- Xeralmente, defínese unha actividade como **principal** no **AndroidManifest.xml**, pola cal se inicia a aplicación. (Como a función/método main() noutras linguaxes de programación).
- Cada aplicación pode ter varias actividades.
- Ademais, dende unha Actividade dunha aplicación pódese saltar á actividade doutra aplicación (P.E. dende WhatsApp podemos saltar a consultar os datos dun contacto da aplicación contactos).
- Cando iniciamos unha actividade, esta pasa ao **primeiro plano** (**Visible**) e a actividade anterior detense e envíase xusto para detrás da actual na Pila (**Back stack**).
- Esta pila usa o mecanismo de colas LIFO. Cando prememos a tecla de retroceso no móbil destrúese a actividade actual e recárgase á actividade que está no top da pila.

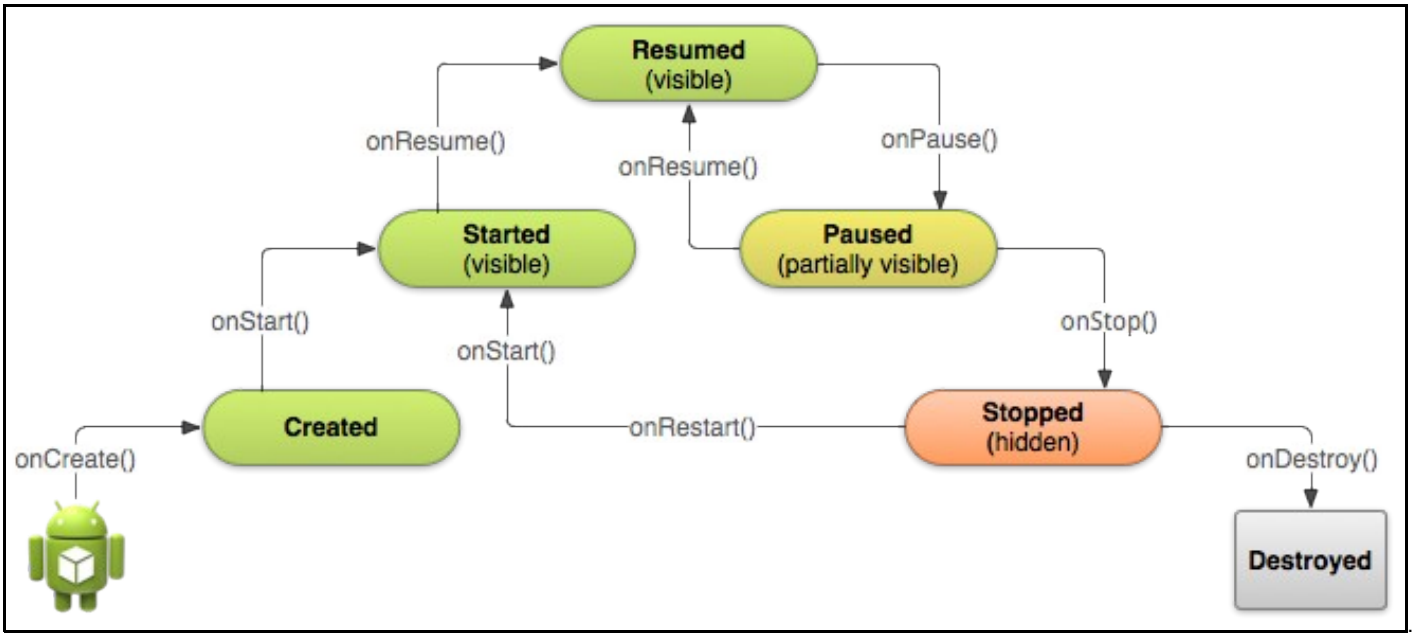
- Dende que iniciamos unha actividade até que saímos dela, esta pasa por distintos estados: **O ciclo de vida dunha actividade**:



- **Referencias**:
    - Actividades: http://developer.android.com/guide/components/activities.html
    - O ciclo de vida dunha actividade: http://developer.android.com/training/basics/activity-lifecycle/index.html

Ciclo de vida dunha actividade

- Cando unha actividade cambia de estado (porque se preme unha tecla do teclado, porque se preme un botón, etc) este cambio é notificado á actividade a través dos métodos **callback**.
- Todos estes métodos **callback** capturan os cambios de estado que se van producindo na actividade e poden ser sobreescritos para que realicen as operacións que desexemos.



- Observar a imaxe, observar como ten forma de pirámide, dende o estado no que se **lanza** unha actividade até que chega ao estado **Resumed (Running)**, esta pasa polos estados **Created** e **Started**.

- Cada un deses cambios de estado da actividade ten asociado un método *callback* que será chamado no momento de producirse o cambio. Por orde: **onCreate()**, **onStart()**, **onResume()**

- Unha actividade estará no estado **Paused** se esta está **semi-visible** porque hai outra actividade en primeiro plano por enriba da primeira que non ocupa toda a pantalla.
    - Neste cambio chamarase ao método **onPause()**.
    - De este estado pódese pasar á **Running** e o método **onResume()** será o chamado nese cambio de estado.

- Unha actividade pasa ao estado **Stopped** cando pasa a segundo plano, cando non está visible, ben porque se abriu unha nova actividade ou porque se premeu o botón **Home** do móbil. Se pasamos do estado **Resumed** a **Stopped** estes cambios son capturados polos métodos **onPause()** e **onStop()**.
    - Unha actividade pode pasar de **Stopped** a **Resumend**, pasando por **Started**, porque se volve a pasar a primeiro plano a actividade que antes estaba oculta na pila de actividades, neste caso dous métodos capturan os cambios: **onRestart()** e de novo **onStart()** e **onResume()**.
    - Mentres unha actividade está no estado **Stopped** retéñense todas as súas variables, información de estado e recursos que está usando.

- Unha actividade pasa ao estado **Destroyed** nos seguintes casos:
    - A actividade está en primeiro plano (Resumed) e prémese o botón **Back**, neste caso pasamos de **Resumed** a **Destroyed**, pasando por **Paused** e **Stopped** chamándose a tódolos métodos que hai polo camiño.
        - Se había unha actividade en segundo plano (que agora estará no estado Stopped) antes de destruír a actual, esa será traída a primeiro plano pasando de **Stopped** a **Resumed**.
    - A Actividade ten programado no seu código que se destrúa explicitamente co método **finish()**, por exemplo ao premer un botón de saír.
    - A Actividade está no fondo da pila de actividades (Stopped) e o sistema precisa os seus recursos para poder asignarllos a unha nova actividade que o usuario quere abrir. Neste caso, o sistema destrúe esa actividade e por tanto esta perde todo aquelo que non fose gardado antes de pasar ao estado de **Stopped**.
    - Dende o **administrador de aplicacións**.

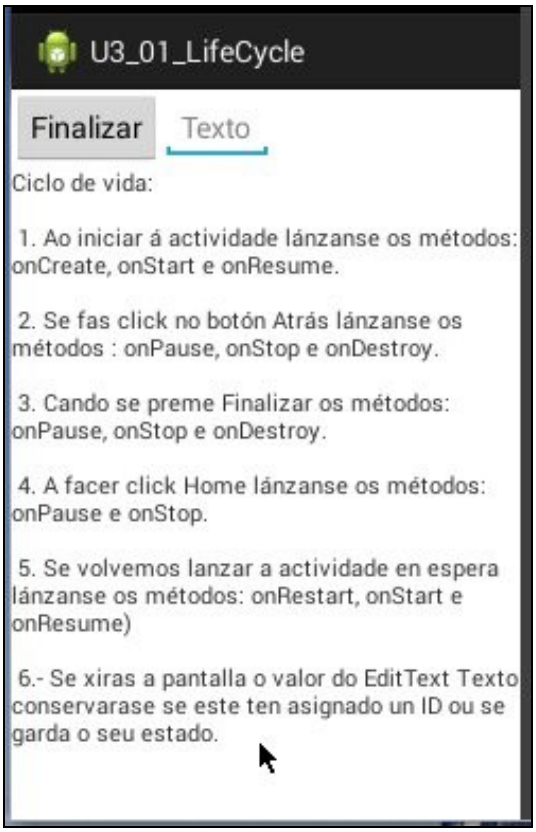
- **IMPORTANTE**: cando se **xira o dispositivo** de vertical a horizontal (e viceversa) e hai unha actividade en primeiro plano esta destrúese e vólvese a lanzar, con todo o que iso implica: Perderanse os valores das vistas salvo que estas teñan creado un ID: **android:id="@+id/...**

- Dependendo do que desexemos realizar é probable que non desexemos implementar todos os métodos do ciclo de vida.
- Coñecendo o seu funcionamento poderemos evitar:
    - Un colgue da aplicación se por exemplo recibimos unha chamada.
    - Non realizar un consumo excesivo de recursos se a actividade non está activa.
    - Non perder datos de usuario se este sae da aplicación e logo volve a ela.
    - Non perder datos de usuario se este cambia a orientación do dispositivo.

Caso práctico

- Comezamos creando un novo proxecto: **U3_01_LifeCycle**
    - Este proxecto está baseado nun proxecto do curso de Android da Aula Mentor (http://www.mentor.mec.es/)
- Nesta ocasión vanse implantar os 7 métodos anteriores e realizar tarefas no dispositivo: premer un botón, premer as teclas Atrás e Home, volver a lanzar a Actividade, xirar o dispositivo etc.
- Entre outras cousas imos ver se perde información ou non mentres a actividade está en segundo plano na pila.
- Cada método terá un Toast que indicará que foi chamado ese método callback.

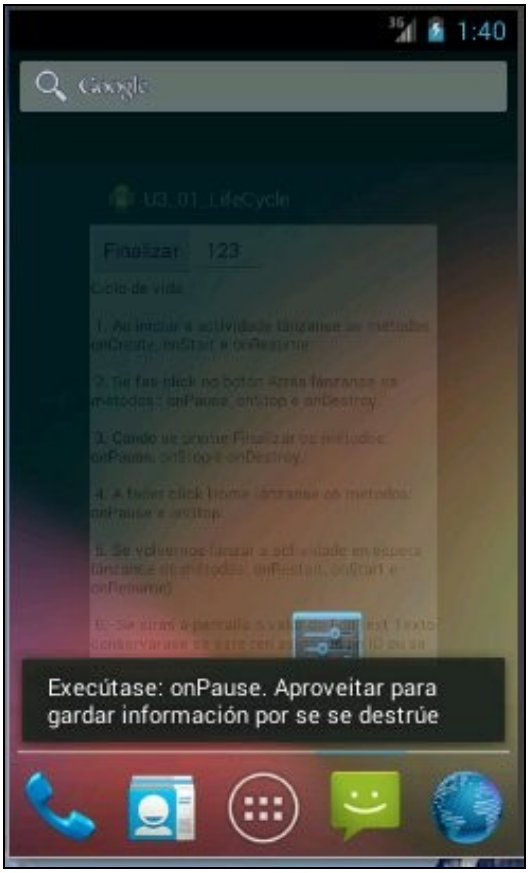
- A Actividade so ten:
    - Un botón para finalizar a actividade.
    - Unha caixa de texto no que poder escribir e comprobar no futuro se o escrito se conserva.
    - Unha etiqueta explicando o que fai a actividade.



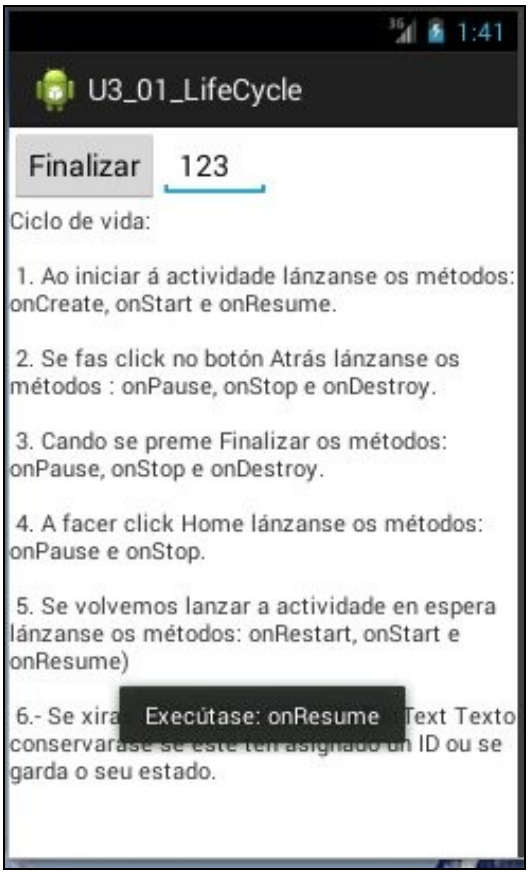
- Escribimos algo na caixa de texto.

- Se prememos, por exemplo, na tecla "Home" vemos cales son os estados polos que pasa a actividade e consecuentemente os métodos callback que son chamados.

- A imaxe amosa como a actividade está indo para o segundo plano, non está visible, está oculta.
- Os métodos que foron chamados nese proceso son: onPause(), onStop()
- A actividade estará no estado **Stopped**



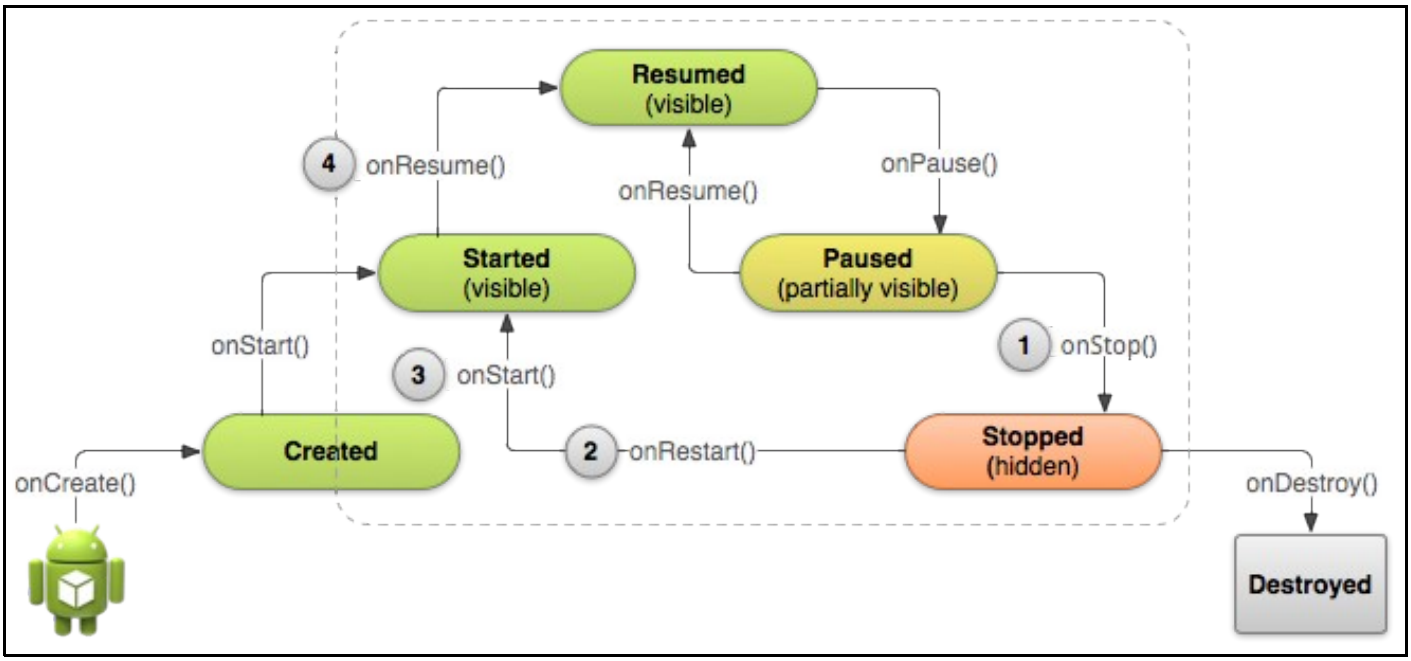
- Agora se volvemos lanzar a actividade, esta pasara do estado **Stopped** a **Resumed**.
- Polo tanto, vanse chamar os métodos: onRestart(), onStart(), onResume()
- Finalmente a actividade non perdeu información.



- O usuario pode probar agora a premer a tecla **Atras**, o botón **Finalizar** e xirar o dispositivo e comprobar se se conserva ou non a información: (Logo o resolveremos)

Parando e reiniciando unha aplicación

- O proceso que se recolle no exemplo anterior recóllese no seguinte esquema:



Ficheiros XML: Layout e strings

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:onClick="onFinalizarClick"
            android:text="Finalizar" />

        <EditText
            android:id="@+id/et"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:hint="Texto" />
    </LinearLayout>

    <ScrollView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/instrucions" />
    </ScrollView>

</LinearLayout>
```

- **Liña 18**: Grazas a que se crea un ID no EditText, o seu contido non se perderá cando cando se xire o dispositivo.
- Probar a eliminar ese atributo (id) do XML e volver compilar a aplicación. Escribir no EditText e xirar a pantalla. Que pasa?.

- **O Ficheiro /res/values/strings.xml**

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

```
    <string name="app_name">U3_01_LifeCycle</string>
    <string name="action_settings">Settings</string>
    <string name="instrucions">Ciclo de vida:\n\n
                1. Ao iniciar á actividade lánzanse os métodos: onCreate, onStart e
                onResume.\n\n

                2. Se fas click no botón Atrás lánzanse os métodos
                : onPause, onStop e onDestroy.\n\n

                3. Cando se preme Finalizar os métodos: onPause, onStop e
                onDestroy.\n\n

                4. A facer click Home lánzanse os métodos: onPause e onStop.\n\n

                5. Se volvemos lanzar a actividade en espera lánzanse os métodos:
                onRestart, onStart e onResume)\n\n

                6.- Se xiras a pantalla o valor do EditText Texto conservarase
                se este ten asignado un ID ou se garda o seu estado.
</string>
</resources>
```

O código Java

```
package com.example.u3_01_lifecycle;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.View;
import android.widget.Toast;

public class U3_01_LifeCycle extends Activity {

Bundle dato = new Bundle();

public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_u3_01__life_cycle);
Toast.makeText(this, "Execútase: onCreate. Aproveitar para recuperar info da última sesión", Toast.LENGTH_SHORT).show();
}

@Override
protected void onStart() {
super.onStart();
Toast.makeText(this, "Execútase: onStart", Toast.LENGTH_SHORT).show();
}

@Override
protected void onResume() {
super.onResume();
Toast.makeText(this, "Execútase: onResume", Toast.LENGTH_SHORT).show();
}

@Override
protected void onPause() {
super.onPause();
Toast.makeText(this, "Execútase: onPause. Aproveitar para gardar información por se se destrúe", Toast.LENGTH_SHORT).show();
}

@Override
protected void onRestart() {
super.onRestart();
Toast.makeText(this, "Execútase: onRestart", Toast.LENGTH_SHORT).show();
}

@Override
protected void onStop() {
super.onStop();
Toast.makeText(this, "Execútase: onStop", Toast.LENGTH_SHORT).show();
}
```

```
@Override
protected void onDestroy() {
super.onDestroy();
Toast.makeText(this, "Execútase: onDestroy. Saímos", Toast.LENGTH_SHORT).show();
}

public void onFinalizarClick(View v) {
finish();
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
// Inflate the menu; this adds items to the action bar if it is present.
getMenuInflater().inflate(R.menu.u3_01__life_cycle, menu);
return true;
}

}
```

- Os métodos callback están sobreescritos e chaman antes de nada aos métodos do pai.
- **Liñas 13,14**: Recíbese un **Bundle** (Conxunto de pares <parámetro, valor> ou null) e cárganse os seus valores nas vistas correspondente. Logo verase máis a fondo.
- **Liñas 55,56**: Ao executar o método **finish()** vaise pechar a actividade. Esta pasará polos estados correspondentes dende **Resumed** até **Destroyed**.

EXERCICIOS:

- Tiñamos unha cuestión pendente:*O usuario pode probar agora a premer a tecla **Atras**, o botón **Finalizar** e **xirar o dispositivo** e comprobar se se conserva ou non a información*:
    - ♦ A conclusión debera ser que nos dous primeiros casos non se conserva a información da caixa de Texto, porque se destrúe a aplicación, pero ...
    - ♦ Ao xirar o dispositivo tamén se destrúe a actividade e iniciase dende cero.
        - ◊ Pero o sistema garda automaticamente nun Bundle (Parámetro, Valor), para cada vista da Actividade, que teña creado un identificador "@+id/", o seu estado.
        - ◊ Ese Bundle é recuperado no método onCreate cando se inicia de novo a actividade.
        - ◊ Probar a eliminar a liña 18 do ficheiro XML do layout e realizar as probas oportunas. Que pasa co contido do EditText?

Estado Paused

- Para afondar no anterior e presentar o estado **Paused** o usuario debe baixar este proxecto de Android: http://developer.android.com/shareables/training/ActivityLifecycle.zip e importalo.
- Comprobar os distintos estados polos que pode pasar/quedar unha actividade. Entre eles que unha actividade estea parcialmente visible, isto é **Paused**.
- A aplicación ten 3 Activities (A, B, C) que se verá despois como se crean varias Actividades nunha aplicación e como se lanzan. A aplicación tamén ten un cadro de diálogo, para poder ver como unha Activity está **parcialmente-visible**.

- Algunhas probas coa aplicación

- 

Ao lanzar a aplicación a primeira Activity que se lanza é a **A**. Observar que está no estatus **Resumed**, pero antes pasou por **Created** e **Started**

- 

Se lanzamos a activity **B**, premendo no botón **Start B**, observar que a activity A, pasa polos estados **Paused** e **Stopped**, e, entre medias, lánzase a activity B. Observar que a activity A quédase no estado **Stopped**

- 

Se prememos no botón **Dialog** observar que se lanza un cadro de diálogo e por detrás queda a Activity B **parcialmente visible** e por tanto no estado **Paused**.

- 

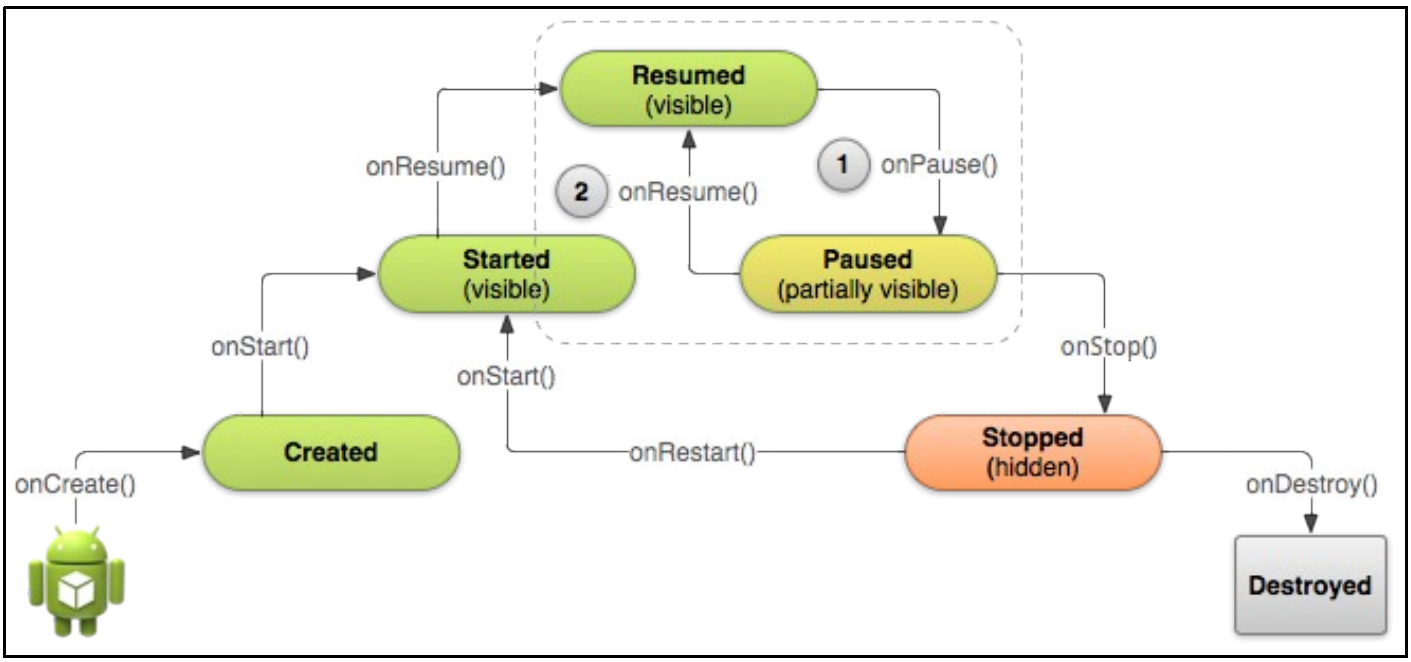
Ao pechar o cadro de diálogo pódese observar que a Activity B pasou de **Paused** a **Resumed**.

- 

Se pechamos a activity B, volvemos á activity que estaba na cola LIFO: a activity A. Vemos que agora a activity A está no estado **Resumed** pero antes pasou por **Restarted** e **Started**.
Observar, como paralelamente, a activity B, pasou do estado **Resumed** a **Destroyed**, pasando por **Paused** e **Stopped**.

- O usuario pode experimentar máis con esta aplicación para familiarizarse co ciclo de vida dunha activity, ollo dunha activity, non dunha aplicación. Esta última está composta por varias activities, como mínimo unha.

- O seguinte diagrama recolle esta situación:



- Cando se está neste estado (**Paused**) é cando se deben pasar os datos á BBDD, gardar as preferencias, etc, porque a partir de aquí o sistema non garante que sempre poida pasar polos estados **Stopped** e **Destroyed**.

Recrear unha actividade: gardar e recuperar o seu estado

- Existen varias casos nos que unha actividade pode ser destruída:
    - Premendo o botón Atrás,
    - Chamando a finish()
    - Xirando o dispositivo
    - Estando en segundo plano e o sistema precise a súa memoria e a destrúe

- Os 2 primeiros casos entendese que foron propiciados polo desexo do usuario.
- Os 2 últimos casos, poden ser accidentais (podese xirar o dispositivo sen querer ou precísase a memoria do dispositivo). Neste caso cando se relance de novo a actividade (**Recrear a actividade**) gustaríanos que a aplicación conservara o seu estado.
- Nestes casos, se o usuario quere volver á actividade, o sistema pode recuperar a información de estado dunha instancia anterior.
- Para iso o sistema antes de destruír a actividade garda o estado da aplicación: **instance state (Estado de instancia)**
- Ese estado está gardado nun obxecto da clase **Bundle** que garda pares **clave-valor**
- Unha vez que se lanza de novo a actividade (recrea), esta recupera o seu estado de instancia.

- Cando o sistema comeza a parar a actividade chama ao método **onSaveInstanceState()** (Paso 1)
    - ◊ Podemos sobreescribir o método e aproveitar para gardar a información que desexemos nun obxecto Bundle.
- Cando se recrea a actividade o sistema páselle o Bundle a 2 métodos: onCreate() e **onRestoreInstanceState()** e podemos aproveitar para recuperar a información previamente gardada.

- **Referencias**:
    - http://developer.android.com/training/basics/activity-lifecycle/recreating.html

Caso práctico

- Neste caso imos modificar a aplicación anterior para implementar os métodos anteriores.
- Imos deshabilitar que o sistema garde, de modo automático, o valor do EditText ao xirar a pantalla e imos codificar o necesario para que ao xirar a pantalla se siga conservando o valor do EditText.

O XML do Layout

- Antes de ver o código Java, observar a **Liña 22** (**android:saveEnabled="false"**) que lle indica ao sistema que non garde de modo automático o seu estado en caso de que, o sistema, teña que destruír a actividade.
- Realizar ese cambio no Layout da actividade e executar a actividade, introducir un valor no EditText e xirar o dispositivo. O EditTExt debera perder o seu valor.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:onClick="onFinalizarClick"
            android:text="Finalizar" />

        <EditText
            android:id="@+id/et"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:hint="Texto"
            android:saveEnabled="false" />
    </LinearLayout>

    <ScrollView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/instrucions" />
    </ScrollView>

</LinearLayout>
```

O código Java

- Agora imos resolver o problema anterior:

```
package com.example.u3_01_lifecycle;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

public class U3_01_LifeCycle extends Activity {

Bundle dato = new Bundle();

public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_u3_01__life_cycle);
Toast.makeText(this, "Execútase: onCreate. Aproveitar para recuperar info da última sesión", Toast.LENGTH_SHORT).show();

/*if (savedInstanceState != null) {
                        EditText et = (EditText) findViewById(R.id.et);
                        et.setText(savedInstanceState.getString("VALOR_EDIT_TEXT"));
                        Toast.makeText(this, "Recreando", Toast.LENGTH_SHORT).show();
                }*/
}

@Override
protected void onSaveInstanceState(Bundle estado) {
EditText et = (EditText) findViewById(R.id.et);
estado.putString("VALOR_EDIT_TEXT", et.getText().toString());
super.onSaveInstanceState(estado);

Toast.makeText(this, "Gardado estado: "+et.getText(), Toast.LENGTH_SHORT).show();
}
```

```
@Override
protected void onRestoreInstanceState(Bundle savedInstanceState) {
super.onRestoreInstanceState(savedInstanceState);
EditText et = (EditText) findViewById(R.id.et);
et.setText(savedInstanceState.getString("VALOR_EDIT_TEXT"));
Toast.makeText(this, "Recreando", Toast.LENGTH_SHORT).show();
}

@Override
protected void onStart() {
super.onStart();
Toast.makeText(this, "Execútase: onStart", Toast.LENGTH_SHORT).show();
}

@Override
protected void onResume() {
super.onResume();
Toast.makeText(this, "Execútase: onResume", Toast.LENGTH_SHORT).show();
}

@Override
protected void onPause() {
super.onPause();
Toast.makeText(this, "Execútase: onPause. Aproveitar para gardar información por se se destrúe", Toast.LENGTH_SHORT).show();
}

@Override
protected void onRestart() {
super.onRestart();
Toast.makeText(this, "Execútase: onRestart", Toast.LENGTH_SHORT).show();
}

@Override
protected void onStop() {
super.onStop();
Toast.makeText(this, "Execútase: onStop", Toast.LENGTH_SHORT).show();
}

@Override
protected void onDestroy() {
super.onDestroy();
Toast.makeText(this, "Execútase: onDestroy. Saímos", Toast.LENGTH_SHORT).show();
}

public void onFinalizarClick(View v) {
finish();
}


@Override
public boolean onCreateOptionsMenu(Menu menu) {
// Inflate the menu; this adds items to the action bar if it is present.
getMenuInflater().inflate(R.menu.u3_01__life_cycle, menu);
return true;
}

}
```

- **Liñas 27-32**:
    - Se xiramos a pantalla ou se o sistema ten que destruír a actividade vaise chamar ao método: **onSaveInstanceState(Bundle)**
    - Neste caso sobrescribimos o método da clase View. Neste caso ao obxecto Bundle (estado) ímoslle engadindo parellas de **CHAVE-VALOR** (**Liña 29**):**estado.putString("NOME_CONSTANTE","Texto que asignamos a NOME_CONSTANTE")**
    - Ao obxecto Bundle podemos engadirlle os pares que desexemos e do tipo de datos que precisemos. Por exemplo para enteiros: **estado.putInt("CANCION",n_cancion)**.
    - Finalmente chamamos ao método onSaveInstanceState() da superclase, para que garde o estado do resto dos compoñentes da actividade. Ollo que non se chama ao principio!!!, pois ao final pasámoslle o obxecto **estado**, cos pares CHAVE-VALOR.

- **Liñas 36-40**:

- Ao **Recrear** a Actividade lanzase automaticamente **onRestoreInstanceState(Bundle)**.
- Neste caso sobrescribimos o método da superclase.
- Comezamos chamando ao método do pai para que estableza o estado dos elementos da actividade.
- Na **liña 39** recupérase o valor dun dos pares almacenado no Bundle.

- **Liñas 19-23**:
  - No canto de sobrescribir o método **onRestoreInstanceState(Bundle)** podemos recuperar os datos do Bundle no método onCreate()
  - Pero o método onCreate() execútase cando se lanza a aplicación dende cero ou cando se "Recrea". No primeiro caso o Bundle vale null.
  - Por iso temos que controlar se o obxecto Bundle ten valores (liña 19).

-- Ángel D. Fernández González e Carlos Carrión Álvarez -- (2015).