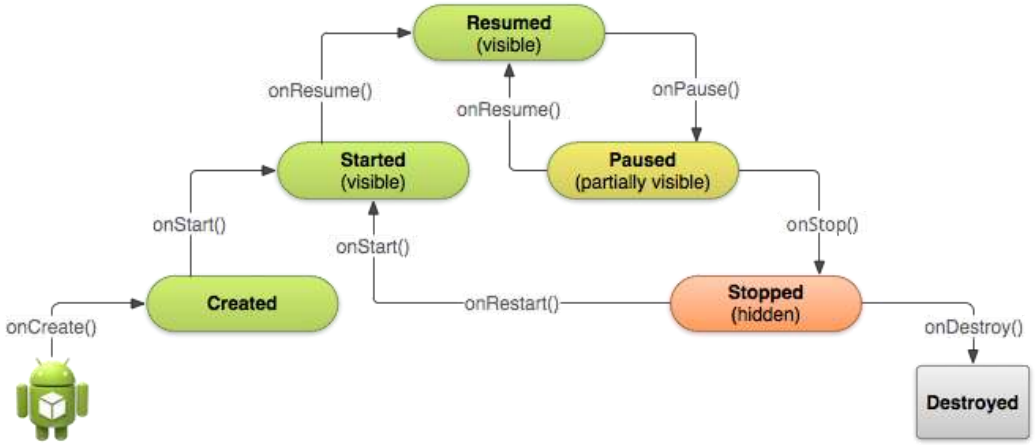
INDICE

1.1. Introducción

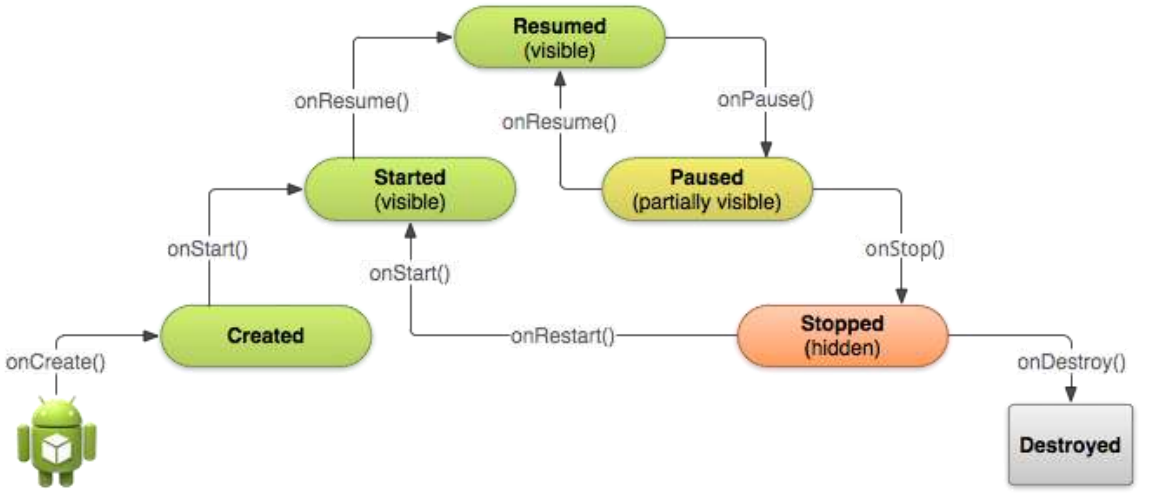
- ✓ Una **Activity (Actividad)** es un elemento de Android que muestra una pantalla con elementos (Vistas: botones, textos, imágenes, etc) con los que los usuarios pueden interactuar: llamar por teléfono, navegar por internet, realizar cálculos en una calculadora.
- ✓ Una actividad, geralmente, ocupa toda la pantalla, aunque que puede ser menor que esta o flotar sobre otras ventanas.
- ✓ Generalmente, se define una actividad como **principal** en el **AndroidManifest.xml**, por la cual se inicia la aplicación. (Como la función/método main() en otros lenguajes de programación).
- ✓ Cada aplicación puede tener varias actividades.
- ✓ Además, desde una Actividad de una aplicación se puede saltar a la actividad de otra aplicación (P.E. desde WhatsApp podemos saltar a consultar los datos de un contacto de la aplicación contactos).
- ✓ Cuando iniciamos una actividad, esta pasa al **primer plano (Visible)** y la actividad anterior se detiene y se envía justo para detrás de la actual en la Pila (**Back stack**).
- ✓ Esta pila usa el mecanismo de colas LIFO. Cuando pulsamos la tecla de retroceso en el móvil se destruye la actividad actual y se recarga la actividad que está en el top de la pila.
- ✓ Desde que iniciamos una actividad hasta que salimos de ella, esta pasa por distintos estados: **El ciclo de vida de una actividad:**



- ✓ **Referencias:**
  - Actividades: https://developer.android.com/guide/components/activities.html
  - El ciclo de vida de una actividad: developer.android.com/guide/components/activities

1.2. Ciclo de vida de una actividad

- ✓ Cuando una actividad cambia de estado (porque se pulsa una tecla del teclado, porque se pulsa un botón, etc) este cambio es notificado a la actividad a través de los métodos **callback**.
- ✓ Todos estos métodos **callback** capturan los cambios de estado que se van produciendo en la actividad y pueden ser sobrescritos para que realicen las operaciones que deseemos.



- ✓ Observar la imagen, observar como tiene forma de pirámide, desde el estado en el que se **lanza** una actividad hasta que llega al estado **Resumed (Running)**, esta pasa por los estados **Created** y **Started**.

- ✓ Cada uno de esos cambios de estado de la actividad tiene asociado un método callback que será llamado en el momento de producirse el cambio. Por orden: **onCreate(), onStart(), onResume().**
- ✓ Una actividad estará en el estado Paused si esta está semi-visible porque hay otra actividad en primer plano por arriba de la primera que no ocupa toda la pantalla.
  - o En este cambio se llama al método **onPause().**
  - o De este estado se puede pasar al **Running** y el método **onResume()** será llamado en ese cambio de estado.
- ✓ Una actividad pasa al estado **Stopped** cuando pasa a segundo plano, cuando no está visible, bien porque se abrió una nueva actividad o porque se pulso el botón **Home** del móvil. Se pasamos del estado **Resumed** a **Stopped** estos cambios son capturados por los métodos **onPause()** y **onStop().**
  - o Una actividad puede pasar de **Stopped** a **Resumend**, pasando por **Started**, porque si vuelve a pasar al primer plano la actividad que antes estaba oculta en la pila de actividades, en este caso dos métodos capturan los cambios: **onRestart()** y de nuevo **onStart()** y **onResume()**.
  - o Mientras una actividad está en el estado **Stopped** se retienen todas sus variables, información de estado y recursos que está usando.
- ✓ Una actividad pasa al estado **Destroyed** en los siguientes casos:
  - o La actividad está en primer plano (Resumed) y se pulsa el botón **Back**, en este caso pasamos de **Resumed** a **Destroyed**, pasando por **Paused** y **Stopped** llamándose a todos los métodos que hay por el camino.
    - Si había una actividad en segundo plano (que ahora estará en el estado Stopped) antes de destruir la actual, esa será traída al primer plano pasando de **Stopped** a **Resumed**.
- ✓ La Actividad tiene programado en su código que se destruya explícitamente con el método **finish()**, por ejemplo al pulsar un botón de salir.
- ✓ La Actividad está en el fondo de la pila de actividades (Stopped) y el sistema precisa sus recursos para poder asignarlos a una nueva actividad que el usuario quiere abrir. En este caso, el sistema destruye esa actividad y por tanto esta pierde todo aquello que no fuese guardado antes de pasar al estado de **Stopped**.
- ✓ Desde el **administrador de aplicaciones**.

- ✓ **IMPORTANTE:** cuando se **gira el dispositivo** de vertical a horizontal (y viceversa) y hay una actividad en primer plano esta se destruye y se vuelve a lanzar, con todo lo que eso implica: Se perderán los valores de las vistas salvo que estas tengan creado un ID: **android:id="@+id/...**

- ✓ Dependiendo de lo que deseemos realizar é probable que no deseemos implementar todos los métodos del ciclo de vida.
- ✓ Conociendo su funcionamiento podremos evitar:
  - o Un cuelgue de la aplicación si por ejemplo recibimos una chamada.
  - o No realizar un consumo excesivo de recursos si la actividad no está activa.
  - o No perder datos de usuario si este sale de la aplicación y luego vuelve a ella.

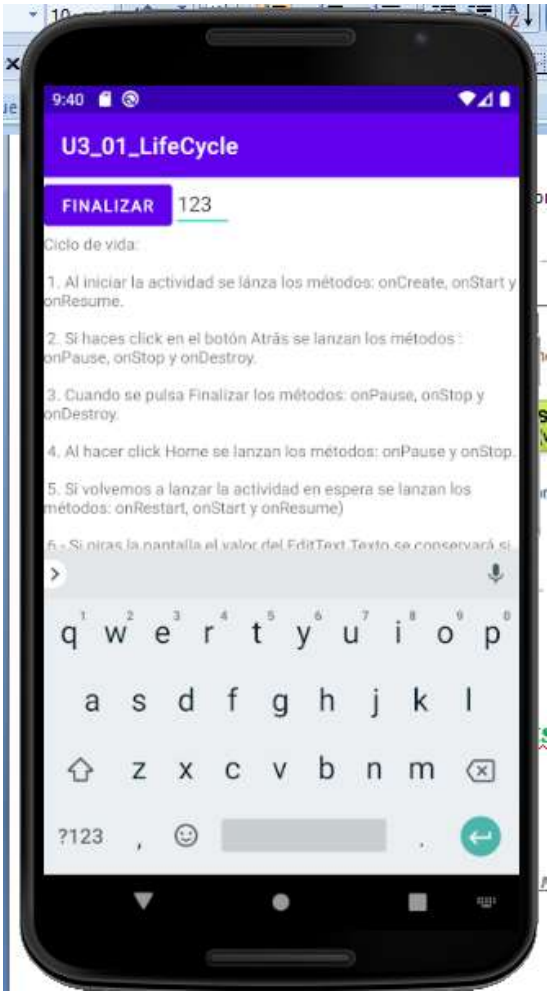
- No perder datos de usuario si este cambia la orientación del dispositivo.

1.2.1.Caso práctico 1

- ✓ Comenzamos creando un nuevo proyecto: **U4_01_LifeCycle**
  - Este proyecto está basado en un proyecto del curso de Android del Aula Mentor (http://www.mentor.mec.es/)
- ✓ En esta ocasión se van a implantar los 7 métodos anteriores y a realizar tareas en el dispositivo: pulsar un botón, pulsar las teclas Atrás y Home, volver a lanzar la Actividad, girar el dispositivo etc.
- ✓ Entre otras cosas vamos a ver si se pierde información o no mientras la actividad está en segundo plano en la pila.
- ✓ Cada método tendrá un Toast que indicará que fue llamado ese método callback.

- ✓ La actividad solo tiene:
  - Un botón para finalizar la actividad.
  - Una caja de texto en el poder escribir y comprobar en el futuro si lo escrito se conserva.
  - Una etiqueta explicando lo que hace la actividad.

- Escribimos algo en la caja de texto.



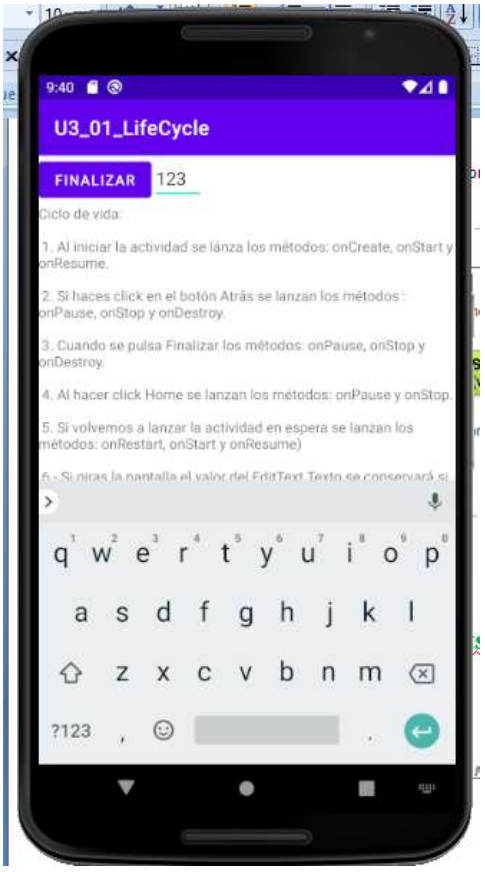
✓ Si pulsamos, por ejemplo, en la tecla "Home" vemos cuáles son los estados por los que pasa la actividad y consecuentemente los métodos callback que son llamados.

✓ La imagen muestra como la actividad está yendo para el segundo plano, no está visible, está oculta.

✓ Los métodos que fueron llamando en ese proceso son: onPause(), onStop().

✓ La actividad estará en estado **Stopped.**

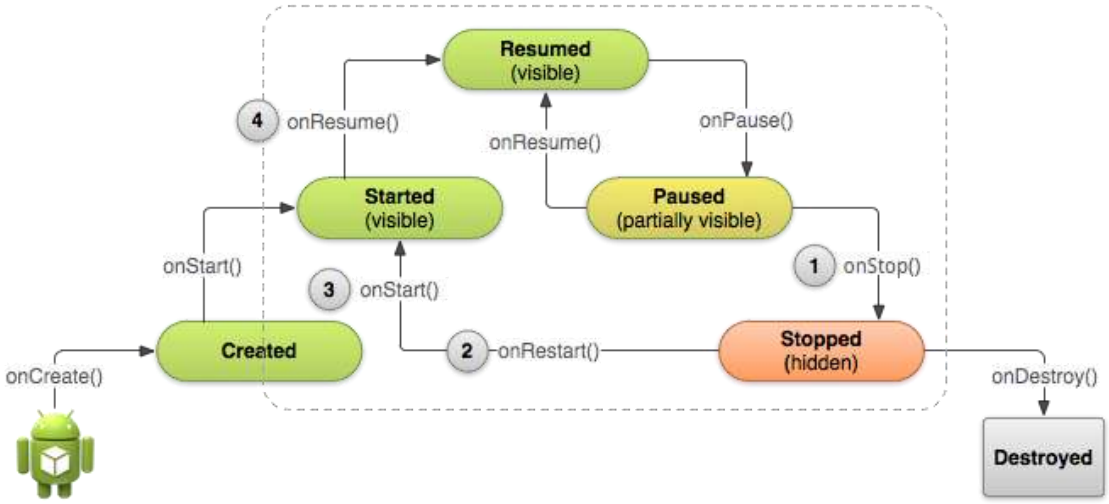
- ✓ Ahora si volvemos a lanzar la actividad, esta pasara del estado **Stopped** a **Resumed**.
- ✓ Por lo tanto, se van a llamar los métodos: onRestart(), onStart(), onResume()
- ✓ Finalmente la actividad no perdió información.



✓ El usuario puede probar ahora a pulsar la tecla **Atrás**, el botón **Finalizar** y girar el dispositivo y comprobar si se conserva o no la información: (Luego lo resolveremos)

1.2.1.1. Parando y reiniciando una aplicación

✓ El proceso que se recoge en el ejemplo anterior se recoge en el siguiente esquema:



1.2.2.Ficheros XML: layouts y strings

1.2.2.1. El Fichero /res/values/strings.xml

```
<resources>
    <string name="app_name">U3_01_LifeCycle</string>
    <string name="action_settings">Settings</string>
    <string name="instrucciones">Ciclo de vida:\n\n
      1. Al iniciar la actividad se lanzan los métodos: onCreate, onStart y
      onResume.\n\n

      2. Si haces click en el botón Atrás se lanzan los métodos
      : onPause, onStop y onDestroy.\n\n

      3. Cuando se pulsa Finalizar los métodos: onPause, onStop y
      onDestroy.\n\n

      4. Al hacer click Home se lanzan los métodos: onPause y onStop.\n\n

      5. Si volvemos a lanzar la actividad en espera se lanzan los métodos:
      onRestart, onStart y onResume)\n\n

      6.- Si giras la pantalla el valor del EditText Texto se conservará
      si este tiene asignado un ID o se guarda su estado.
   </string>
</resources>
```

1.2.2.2. El código XML

```
   <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:id="@+id/btnFinalizar_UD03_01_CicloVida"
               android:layout_width="wrap_content"
               android:layout_height="wrap_content"
               android:onClick="onFinalizarClick"
               android:text="Finalizar" />

           <EditText
               android:id="@+id/etDato_UD03_01_CicloVida"
               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/instrucciones" />
        </ScrollView>

    </LinearLayout>
```

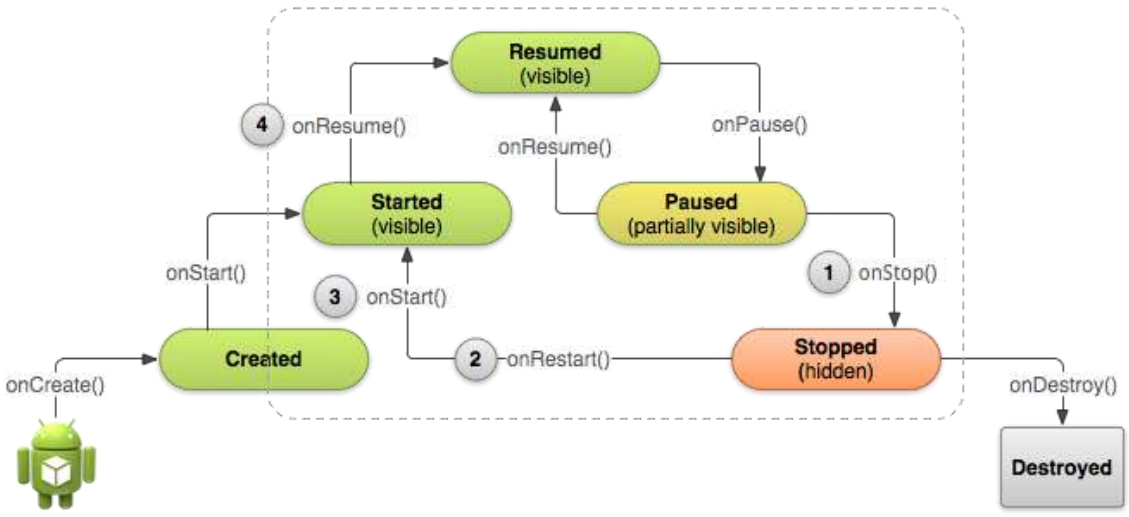
- ✓ **Línea 19:** Gracias a que se crea un ID en el EditText, su contenido no se perderá cuando se gire el dispositivo.
- ✓ Probar a eliminar ese atributo (id) del XML y volver a compilar la aplicación. Escribir en el EditText y girar la pantalla. ¿Qué pasa?.

1.2.2.3. El código Java

```
package com.example.u3_01_lifecycle;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    private void gestionarEventos(){

        Button btnFinalizar = findViewById(R.id.btnFinalizar_UD03_01_CicloVida);
        btnFinalizar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                finish();
            }
        });

    }

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

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

    @Override
    protected void onPause() {
        super.onPause();
```

```
        Toast.makeText(this, "Se ejecuta: onPause. Aprovechar para guardar información por
si se destruye", Toast.LENGTH_SHORT).show();
    }

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

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

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

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toast.makeText(this, "Se ejecuta: onCreate. Aprovechar para recuperar la info de la
última sesión", Toast.LENGTH_SHORT).show();
        gestionarEventos();

    }
}
```

- ✓ Los métodos callback están sobrescritos y se llaman antes de nada a los métodos del padre.
- ✓ **Líneas 13,14**: Se recibe un **Bundle** (Conjunto de pares <parámetro, valor> o null) y se cargarán sus valores en las vistas correspondientes. Luego se verá más a fondo.
- ✓ **Líneas 55,56**: Al ejecutar el método **finish()** se va a cerrar la actividad. Esta pasará por los estados correspondientes desde **Resumed** hasta **Destroyed**.



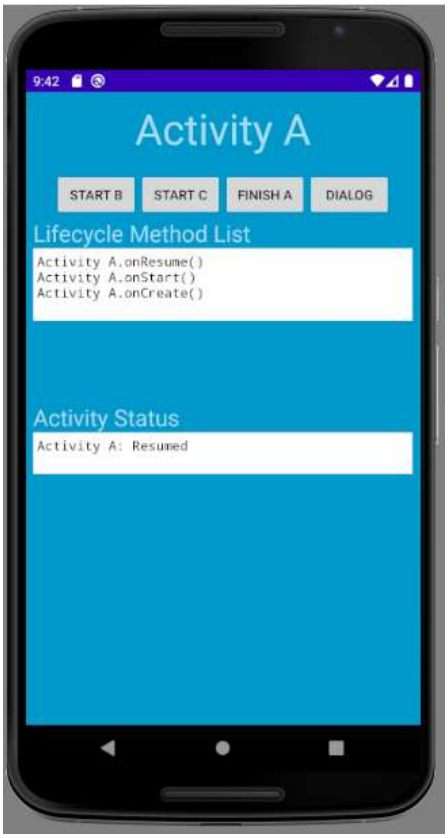
EJERCICIOS:

- ✓ Teníamos una cuestión pendiente: *El usuario puede probar ahora a pulsar la tecla **Atras**, el botón **Finalizar** y **girar el dispositivo** y comprobar si se conserva o no la información*:
  - La conclusión deberá ser que en los dos primeros casos no se conserva la información de la caja de Texto, porque se destruye la aplicación, pero ...
  - Al girar el dispositivo también se destruye la actividad y se inicia desde cero.
    - Pero el sistema guarda automáticamente en el Bundle (Parámetro, Valor), para cada vista de la Actividad, que tenga creado un identificador "@+id/", o su estado.
    - Ese Bundle es recuperado en el método onCreate cuando se inicia de nuevo la actividad.
    - Probar a eliminar a línea 18 del fichero XML del layout y realizar las pruebas oportunas. ¿Qué pasa con el contenido del EditText?

1.2.3.Estado Paused

- ✓ Para ahondar en lo anterior y presentar el estado **Paused** el estudiante debe bajar el proyecto de Android: http://developer.android.com/shareables/training/ActivityLifecycle.zip
- ✓ Comprobar los distintos estados por los que puede pasar/quedar una actividad. Entre ellos que una actividad este parcialmente visible, esto es **Paused**.
- ✓ La aplicación tiene 3 Activities (A, B, C) que se verá después como se crean varias Actividades en una aplicación y como se lanzan. La aplicación también tiene un cuadro de diálogo, para poder ver como una Activity está **parcialmente-visible**.

Algunas pruebas con la aplicación



Al lanzar la aplicación la primera Activity que se lanza es la **A**. Observar que está en el estatus **Resumed**, pero antes paso por **Created** y **Started.**

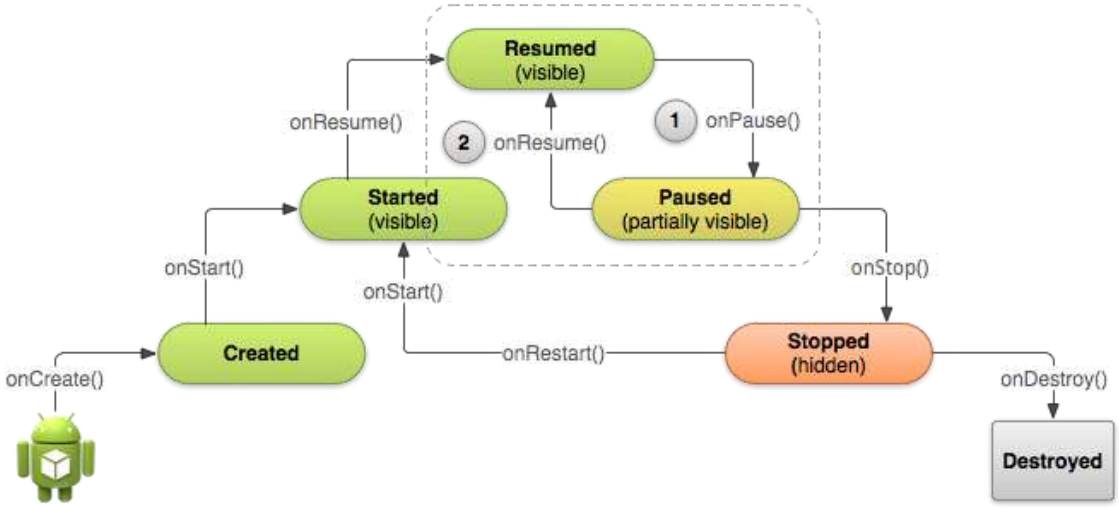
Si lanzamos la activity **B**, pulsando en el botón **Start B**, observar que la activity A, pasa por los estados **Paused** y **Stopped**, y, entre medias, se lanza la activity B. Observar que la activity A se queda en el estado **Stopped.**

Si pulsamos el botón **Dialog** observar que se lanza un cuadro de diálogo y por detrás queda la Activity B **parcialmente visible** aunque a mí no me sale en estos momentos y por tanto en el estado **Paused**.

Al cerrar el cuadro de diálogo se puede observar que la Activity B pasó de **Paused** a **Resumed**.

Al cerrar la activity B, volvemos a la activity que estaba en la cola LIFO: la activity A. Vemos que ahora la activity A está en el estado **Resumed** pero antes paso por **Restarted** y **Started**. Observar, como paralelamente, la activity B, paso del estado **Resumed** a **Destroyed**, pasando por **Paused** y **Stopped**.

- ✓ El estudiante puede experimentar más con esta aplicación para familiarizarse con el ciclo de vida de una activity, ojo de una activity, no de una aplicación. Esta última está composta por varias activities, como mínimo una.
- ✓ El siguiente diagrama recoge esta situación:



- ✓ Cuando se está en este estado (**Paused**) es cuando se deben pasar los datos a las BBDD, guardar las preferencias, etc, porque a partir de aquí el sistema no garantiza que siempre pueda pasar por los estados **Stopped** y **Destroyed**.

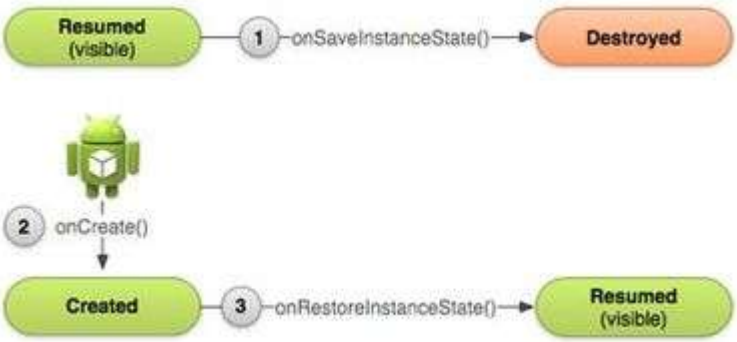
1.3. Recrear una actividad: guardar y recuperar su estado

- ✓ Existen varias casos nos que una actividad puede ser destruida:
  - Pulsando el botón Atrás.
  - Llamando a finish ().
  - Girando el dispositivo.
  - Estando en segundo plano y el sistema precise su memoria y la destruye.

- ✓ Los 2 primeros casos se entiende que fueron propiciados por lo el deseo del usuario.
- ✓ Los 2 últimos casos, pueden ser accidentales (se puede girar el dispositivo sin querer o se precisa la memoria del dispositivo). En este caso cuando se relance de nuevo la actividad (**Recrear la actividad**) nos gustaría que la aplicación conservara o su estado.
- ✓ En estos casos, si el usuario quiere volver a la actividad, el sistema puede recuperar la información de estado de una instancia anterior.
- ✓ Para esto el sistema antes de destruir la actividad guarda el estado de la aplicación: **instance state (Estado de instancia).**
- ✓ Ese estado está guardado en un objeto de la clase **Bundle** que guarda pares **clave-valor.**
- ✓ Una vez que se lanza de nuevo la actividad (recrea), esta recupera su estado de instancia.

- ✓ Cuando el sistema comienza a parar la actividad llama al método **onSaveInstanceState**() (Paso 1)
  - Podemos sobrescribir el método es aprovechar para guardar la información que deseamos en un objeto Bundle.
  - Cuando se recrea la actividad el sistema le pasa el Bundle a 2 métodos: onCreate() y onRestoreInstanceState() y podemos aprovechar para recuperar la información previamente guardada.
- ✓ Seguramente la aplicación tiene otra información aparte del estado de las View´s que es necesario recuperar, como las variables usadas para guardar el progreso del usuario o cualquiera otra información que como programadores queramos guardar.

  Para eso existe el método onSaveInstanceState que llama al sistema cuando la aplicación pasa a segundo plano o se destruye.



- ✓ A este método se le pasa el objeto de la clase Bundle donde está guardada toda la información (view´s) de la aplicación, por lo que el programador puede añadir más información para ser recuperada posteriormente si el sistema vuelve a llamar a la aplicación, ya que le vuelve a pasar el mismo objeto de la clase Bundle.
- ✓ El objeto de la clase Bundle funciona en base a pares **CLAVE-VALOR**. La clave y el identificador del valor que se le va a pasar. Viene a ser como un nombre de una variable en programación.

  Así tenemos:
  - método **putXXXXX(CLAVE,VALOR)** siendo XXXXX un tipo de dato, por ejemplo, putInt(), para añadir datos al objeto Bundle.

- método **getXXXXX(CLAVE)**' siendo XXXXX un tipo de dato, por ejemplo, getInt(), para recuperar el valor de un elemento clave guardado.

Por ejemplo, si quiero guardar una cadena en una 'clave' de nombre 'LOGIN', pondría:

objetoBundle.putString("LOGIN","CADENA A GUARDAR");

Si se quiere recuperar dicho valor, en el mismo objeto bundle:

String login = objetoBundle.getString("LOGIN");

Lógicamente el nombre de la clave tiene que ser el mismo cuando guardamos los datos y cuando lo introducimos.

✓ Cuando la aplicación se re-inicializa (esto es, cuando se rota el dispositivo o volvemos a la aplicación una vez que el S.O. liberó su memoria por necesidad) se llama al **método onRestoreInstanceState**, donde se recuperan los datos guardados.

Nota: Antes de llegar al estado Resumed pasa por el estado Started.

✓ **Ejemplo:**

```
1     @Override
2     protected void onSaveInstanceState(Bundle datosgardados){
3
4         datosgardados.putInt("POSICION",reproductor.getPosicionActualCancion());
5         datosgardados.putString("CANCION",reproductor.cancionactual);
6
7         // SEMPRE O CODIGO POSTO POR NOS ANTES DA CHAMADA A SUPER.
8         super.onSaveInstanceState(datosgardados);
9 }
```

Para restaurar el sistema usaremos el método onRestoreInstanceState.

```
1     @Override
2     protected void onRestoreInstanceState(Bundle datosgardados){
3
4             super.onRestoreInstanceState(datosgardados);
5
6             // CODIGO POSTO POR NOS PARA RECUPERAR OS DATOS, SEMPRE DESPOIS
7             // DA CHAMADA A SUPER
8             reproductor.cancionactual=datosgardados.getString(C_DATOCANCION_PECHAR);
9             reproductor.posicioncancion=datosgardados.getInt(C_DATOPOSCANCION_PECHAR);
10
11 }
```

Nota: Es importante llamar a los métodos super.XXXXX en la orden indicada en el ejemplo anterior.

Nota: Podemos hacer que Android no guarde el estado de una view poniendo como atributo: **android:saveEnabled="false"** en el archivo XML del layout o por programación.

✓ **Referencias:**

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

1.3.1.Caso práctico 2

✓ Comenzamos creando un nuevo proyecto: **U4_02_LifeCycle**

✓ Crear una nueva 'Empty Activity' de nombre: **U4_02_LifeCycle** de tipo Launcher.

✓ Cambiar el contenido del archivo del Layout XML por este:

✓
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <EditText
        android:id="@+id/etTexto_UD03_02_CicloVida"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="textPersonName"
        android:text="Name"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.472"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.228" />

    <Button
        android:id="@+id/btnBoton1_UD03_02_CicloVida"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="84dp"
        android:layout_marginEnd="8dp"
        android:text="Dar Valor"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.482" />

    <Button
        android:id="@+id/btnBoton2_UD03_02_CicloVida"
```

```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="48dp"
            android:text="Ver valor"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.498"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintVertical_bias="0.324" />

    </androidx.constraintlayout.widget.ConstraintLayout>
```

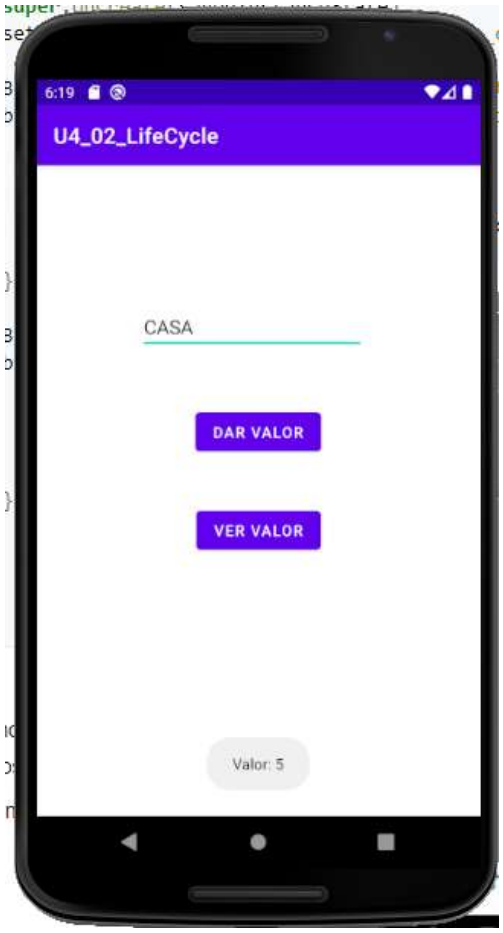
✓ Cambiar el contenido de la activity por este:

```
package com.example.u4_02_lifecycle;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    private int valor = 0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button btnBoton1 = findViewById(R.id.btnBoton1_UD03_02_CicloVida);
        btnBoton1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                valor = 5;
                Toast.makeText(getApplicationContext(), "Valor: " + valor, To-
ast.LENGTH_LONG).show();
            }
        });

        Button btnBoton2 = findViewById(R.id.btnBoton2_UD03_02_CicloVida);
        btnBoton2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(getApplicationContext(), "Valor: " + String.valueOf(valor), To-
ast.LENGTH_LONG).show();
            }
        });
    }
}
```

✓ Se ejecutamos esta aplicación podemos comprobar cómo cuando se inicia el valor de la variable vale 0.

Modificaremos el contenido de la caja de texto y pulsamos el botón.

En este momento a nivel de programación la variable vale 5.



✓ Se salimos de la aplicación y volvemos, podemos comprobar cómo el contenido de la caja de texto permanece y si pulsamos sobre el botón 'Ver Valor' también guardó el dato.

✓ En este caso no tendríamos que guardar el estado de nada para que la aplicación continuase (en otro caso, por ejemplo, en un reproductor de música podríamos guardar la posición de la canción que está siendo reproducida).

✓ Pero qué pasa si el S.O. necesita memoria y elimina dicha aplicación del Stack?

✓ También pasa lo mismo cuando rotamos el dispositivo. La activity se 'recrea' nuevamente.

- ✓ Los datos se perdieron.
- ✓ Vamos a modificar el código para que se guarden los datos introducidos:
- ✓

```
package com.example.u4_02_lifecycle;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    private int valor = 0;

    @Override
    protected void onSaveInstanceState(Bundle datosgardados){

        datosgardados.putInt("VALOR",valor);

        // SIEMPRE EL CODIGO PUESTO POR NOSOTROS ANTES DE LA LLAMADA A SUPER.
        super.onSaveInstanceState(datosgardados);
    }

    @Override
    protected void onRestoreInstanceState(Bundle datosgardados){
        super.onRestoreInstanceState(datosgardados);

        // CODIGO PUESTO PARA RECUPERAR LOS DATOS, SIEMPRE DESPUES DE
        // LA LLAMADA A SUPER
        valor   = datosgardados.getInt("VALOR");

    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button btnBoton1 = findViewById(R.id.btnBoton1_UD03_02_CicloVida);
        btnBoton1.setOnClickListener(new View.OnClickListener() {
```

```
            @Override
            public void onClick(View v) {
                valor = 5;
                Toast.makeText(getApplicationContext(), "Valor: " + valor, To-
ast.LENGTH_LONG).show();
            }
        });

        Button btnBoton2 = findViewById(R.id.btnBoton2_UD03_02_CicloVida);
        btnBoton2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(getApplicationContext(), "Valor: " + String.valueOf(valor),
Toast.LENGTH_LONG).show();
            }
        });
    }
}
```

- ✓ Podemos comprobar como ahora al rotar la pantalla los datos permanecen.

- ✓ En este caso el view es guardado por el propio android, ya que todos los view´s con un id puesto guardan su estado.
- ✓ De todas formas puede ser necesario guardar el estado de algún view, ya que puede ser inicializado en el método onStart.
- ✓ Por ejemplo, cargar un ListBox con un conjunto de datos de la base de datos, que el usuario elija uno de esos datos.
- ✓ Si en ese momento la actividad se recrea, al iniciarse, se volverían a cargar los datos da ListBox, pero la elección del usuario se perdería.

1.3.2.Caso práctico 3

1.3.2.1. El XML del Layout

- ✓ En este caso vamos a modificar la primera aplicación para implementar los métodos anteriores. Podemos crear una nueva activity copiando el código de la práctica o modificar dicha activity.
- ✓ Vamos a deshabilitar que el sistema guarde, de modo automático, el valor del EditText al girar la pantalla y vamos a codificar lo necesario para que al girar la pantalla se siga conservando el valor del EditText.

- ✓ Antes de ver el código Java, observar a **Línea 22 (android:saveEnabled="false")** que le indica al sistema que no guarde de modo automático su estado en caso de que, el sistema, tenga que destruir la actividad.
- ✓ Realizar ese cambio en el Layout de la actividad y ejecutar la actividad, introducir un valor en el EditText y girar el dispositivo. El EditTExt deberá perder su 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:id="@+id/btnFinalizar_UD03_01_CicloVida"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:onClick="onFinalizarClick"
            android:text="Finalizar" />

        <EditText
            android:id="@+id/etDato_UD03_01_CicloVida"
            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/instrucciones" />
    </ScrollView>
```

```
</LinearLayout>
```

1.3.2.2. O código Java

✓ Ahora vamos a resolver el problema anterior:

```
package com.example.u3_01_lifecycle;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    Bundle dato = new Bundle();

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toast.makeText(this, "Se ejecuta: onCreate. Aprovechar para recuperar la info de la
última sesión", Toast.LENGTH_SHORT).show();
        gestionarEventos();

    }

    @Override
    protected void onSaveInstanceState(Bundle estado) {
        EditText et = (EditText) findViewById(R.id.etDato_UD03_01_CicloVida);
        estado.putString("VALOR_EDIT_TEXT", et.getText().toString());
        super.onSaveInstanceState(estado);

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

    @Override
    protected void onRestoreInstanceState(Bundle savedInstanceState) {
        super.onRestoreInstanceState(savedInstanceState);
        EditText et = (EditText) findViewById(R.id.etDato_UD03_01_CicloVida);
        et.setText(savedInstanceState.getString("VALOR_EDIT_TEXT"));
        Toast.makeText(this, "Recreando", Toast.LENGTH_SHORT).show();
    }


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

    @Override
```

```
    protected void onResume() {
        super.onResume();
        Toast.makeText(this, "Se ejecuta: onResume", Toast.LENGTH_SHORT).show();
    }

    @Override
    protected void onPause() {
        super.onPause();
        Toast.makeText(this, "Se ejecuta: onPause. Aprovechar para guardar información por si
se destruye", Toast.LENGTH_SHORT).show();
    }

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

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

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

    private void gestionarEventos(){

        Button btnFinalizar = findViewById(R.id.btnFinalizar_UD03_01_CicloVida);
        btnFinalizar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                finish();
            }
        });

    }


}
```

- ✓ **LÍNEAS 23-30**:
  - Si giramos la pantalla o si el sistema tiene que destruir la actividad se va a llamar a método: **onSaveInstanceState(Bundle).**
  - En este caso sobrescribimos el método de la clase View. En este caso a objeto Bundle (estado) le vamos añadiendo parejas de **CLAVE-VALOR** (**Línea 29**):**estado.putString("NOMBREE_CONSTANTE","Texto que asignamos a NOMBRE_CONSTANTE")**.
  - Al objeto Bundle podemos añadirle los pares que deseamos y del tipo de datos que precisemos. Por ejemplo para enteros: **estado.putInt("CANCION",n_cancion)**.

- Finalmente llamamos al método onSaveInstanceState() de la superclase, para que guarde el estado del resto de los componentes de la actividad. Ojo que no se llama al principio!!!, pues al final le pasamos el objeto **estado**, con pares CLAVE-VALOR.

✓ **LÍNEAS 32-38**:

- AL **Recrear** La Actividad se lanza automáticamente **onRestoreInstanceState(Bundle)**.
- En este caso sobrescribimos el método de la superclase.
- Comenzamos llamando al método del padre para que establezca el estado de los elementos de la actividad.
- En la **línea 39** se recupera el valor de uno de los pares almacenado en el Bundle.

✓ **LÍNEAS 20-24**:

- En cuanto a sobrescribir el método **onRestoreInstanceState(Bundle)** podemos recuperar los datos del Bundle en el método onCreate().
- Pero el método onCreate() se ejecuta cuando se lanza la aplicación desde cero o cuando se "Recrea". En el primer caso el Bundle vale null.
- Por eso tenemos que controlar si el objeto Bundle tiene valores (línea 20).