Máster Universitario en Ingeniería Informática

Gestión de Información en Dispositivos Móviles

Actividades

Javier Abad (abad@decsai.ugr.es)
Dpto. de Ciencias de la Computación e Inteligencia Artificial
http://decsai.ugr.es

Recordemos...

- Componentes fundamentales de una aplicación Android:
  - Actividades (Activities)
  - Servicios (Services)
  - Receptores de anuncios (Broadcast Receivers)
  - Proveedores de contenido (Content Providers)
- **Contenidos:**
  - La clase Activity
  - La pila de actividades (Task Backstack)
  - El ciclo de vida de las actividades
  - Iniciación programática de actividades
  - Gestión de cambios en el dispositivo

Actividades

- Proporciona una interfaz visual que permite la interacción del usuario.
- Modulares: cada actividad debe soportar una tarea concreta que pueda realizar el usuario (leer un correo, mostrar una pantalla de login, etc.).
- Aplicaciones compuestas por varias actividades.
- Android ayuda a la navegación entre actividades:
  - Tareas
  - Pila de tareas
  - Suspensión y reanudación de actividades

Tareas

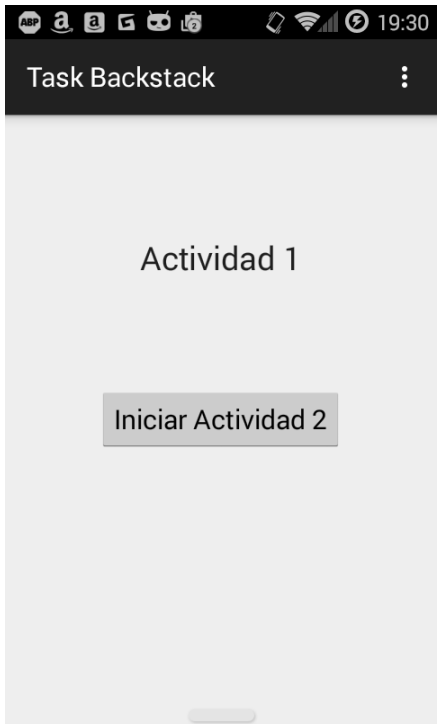
- Una tarea es un conjunto de actividades relacionadas.
- Estas actividades pueden pertenecer a una misma aplicación, pero no tienen por qué.
- La mayoría de las tareas se inician desde la pantalla de inicio.

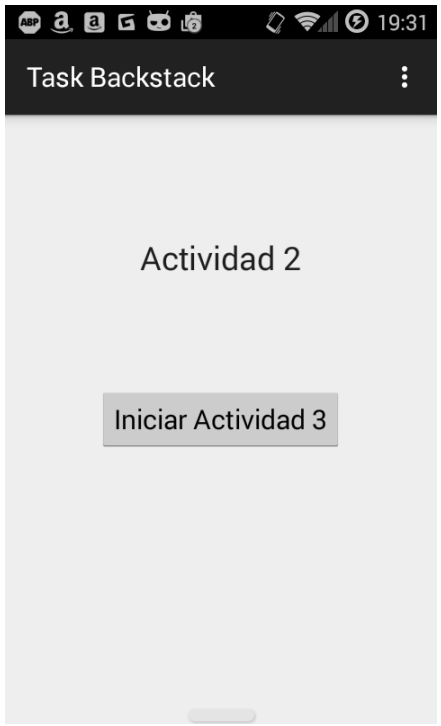
http://developer.android.com/guide/components/tasks-and-back-stack.html

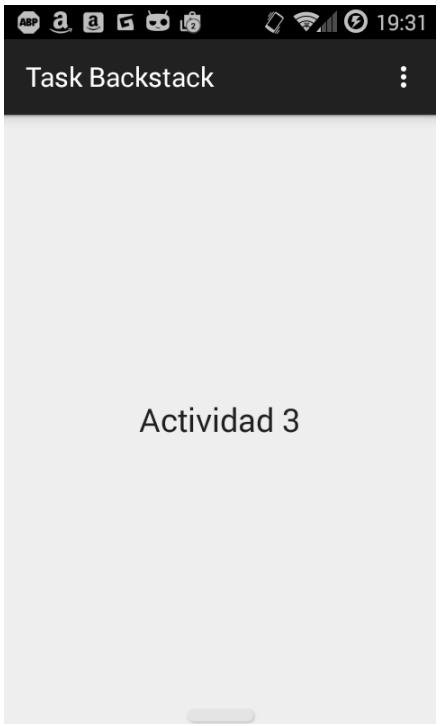
La pila de tareas

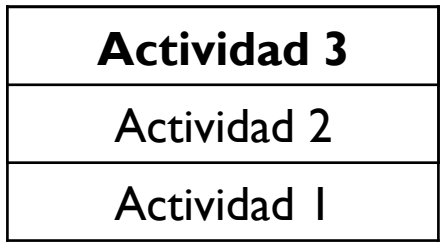
- Cuando se lanza una actividad, se inserta en el tope de la pila de tareas (se apila - push).
- Cuando la actividad se destruye, se saca de la pila de tareas (se desapila - pop).

La pila de tareas









El ciclo de vida de las actividades

- Cuando ejecutamos una aplicación, las actividades se crean, se suspenden, se reanudan y se terminan conforme resulta preciso.
- Algunos de estos cambios de estado dependen de la actuación del usuario.
  - P.ej., al pulsar el botón Back o el botón Home.
- Otros dependen de Android.
  - P.ej., puede matar actividades cuando necesite sus recursos.

Estados del ciclo de vida de las actividades

- Reanudada/En ejecución (Resumed, Running)
  - Visible, el usuario interactúa con la actividad
- Pausada (Paused)
  - Visible, el usuario no interactúa
- Detenida (Stopped)
  - No visible, puede terminarse

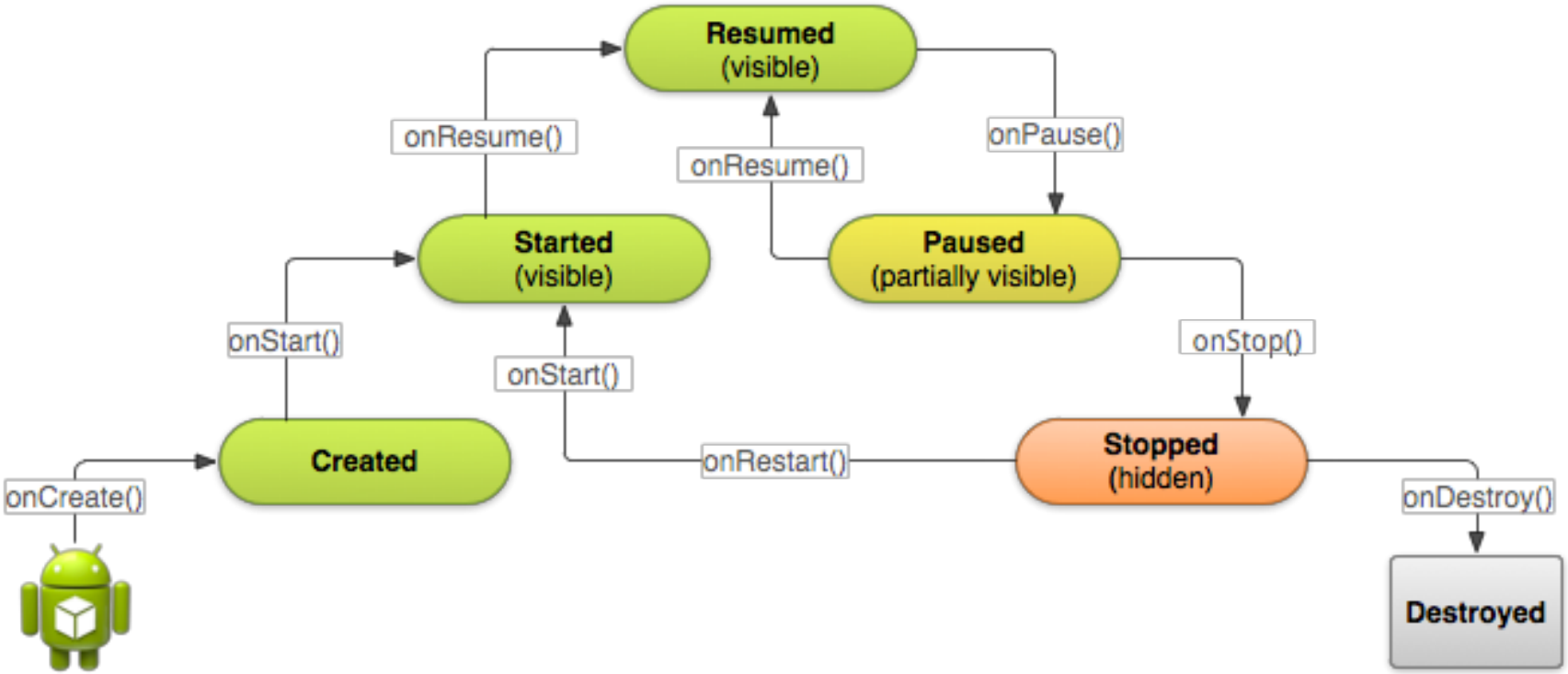
Métodos del ciclo de vida de las actividades

- Android anuncia a la actividad los cambios en el ciclo de vida mediante la llamada a métodos específicos.

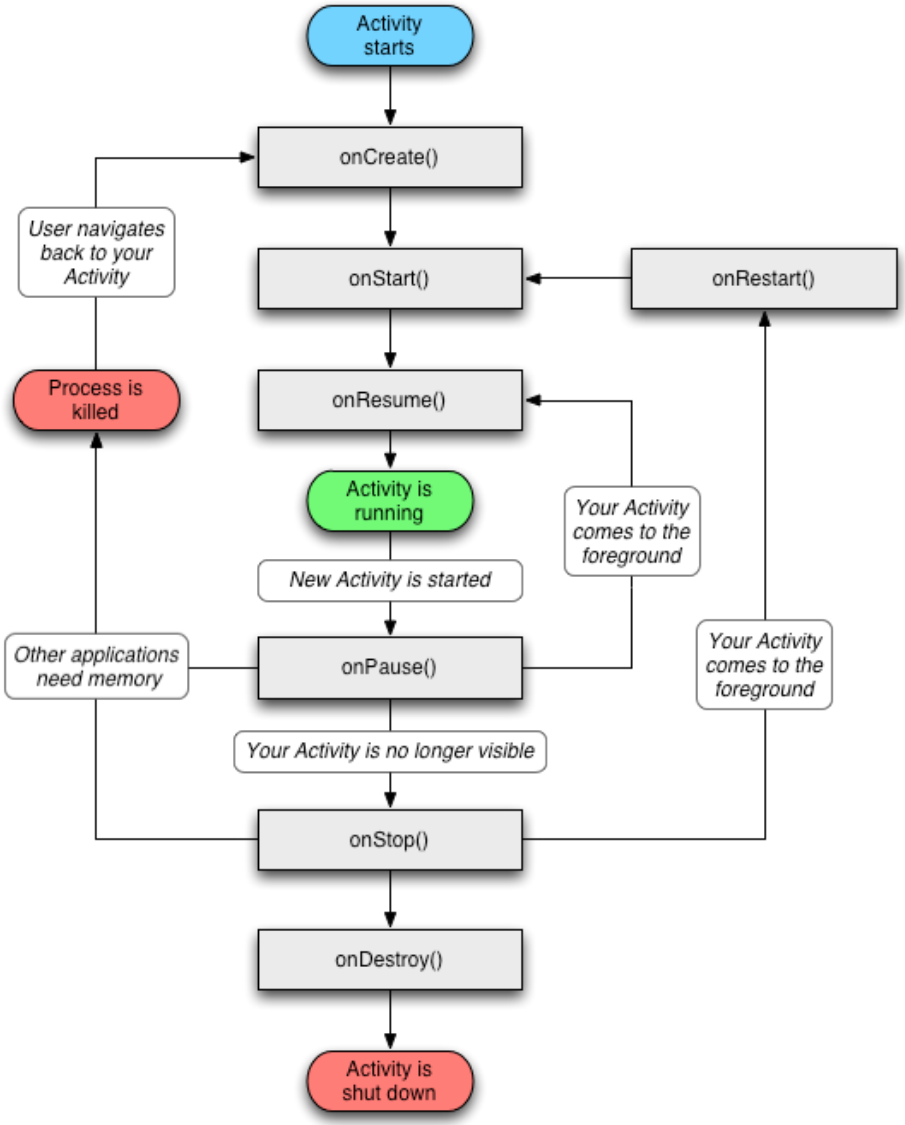
```
protected void onCreate(Bundle savedInstanceState)
protected void onStart()
protected void onResume()
protected void onPause()
protected void onRestart()
protected void onStop()
protected void onDestroy()
```

- Tendremos que redefinir (Override) uno o varios de estos métodos en nuestra actividad.

El ciclo de vida de las actividades

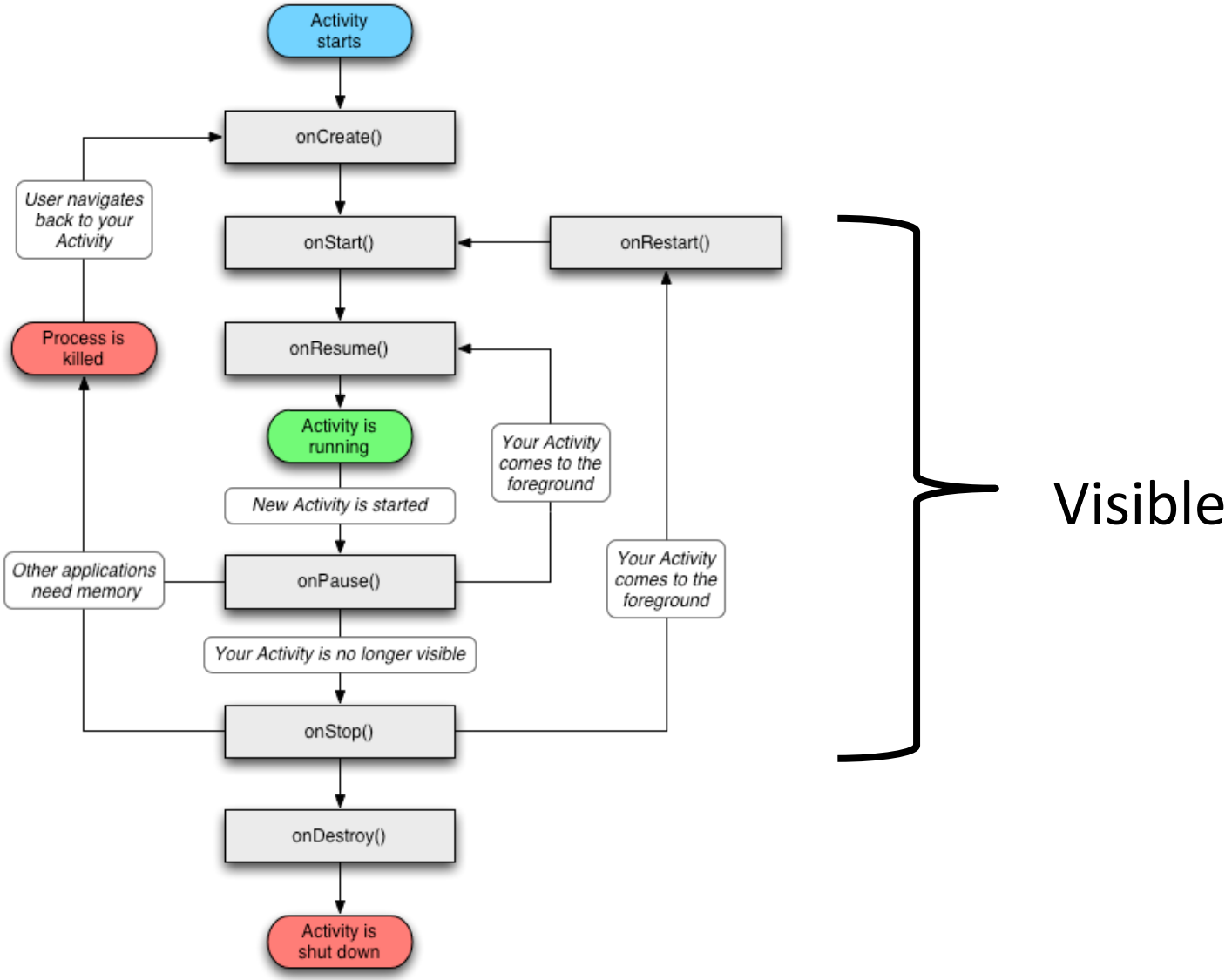


El ciclo de vida de las actividades

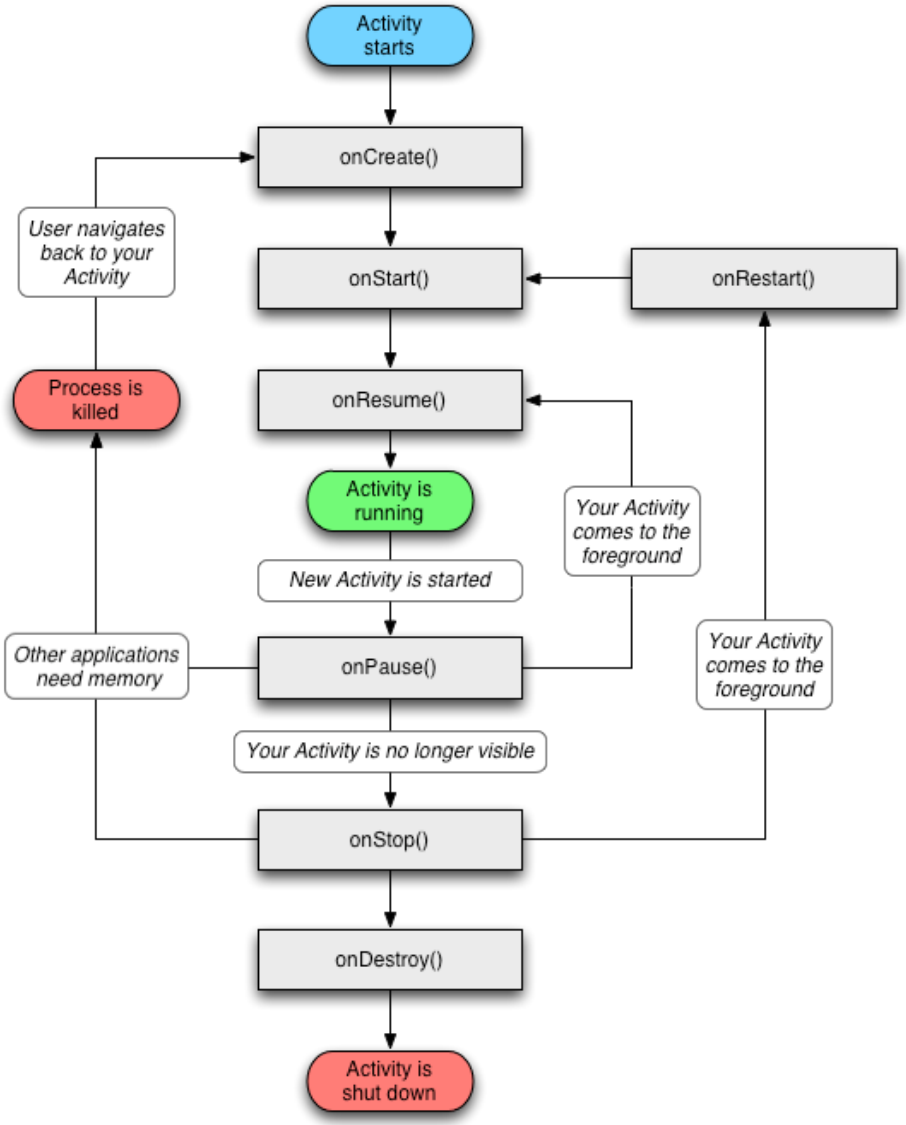


Vida completa

El ciclo de vida de las actividades

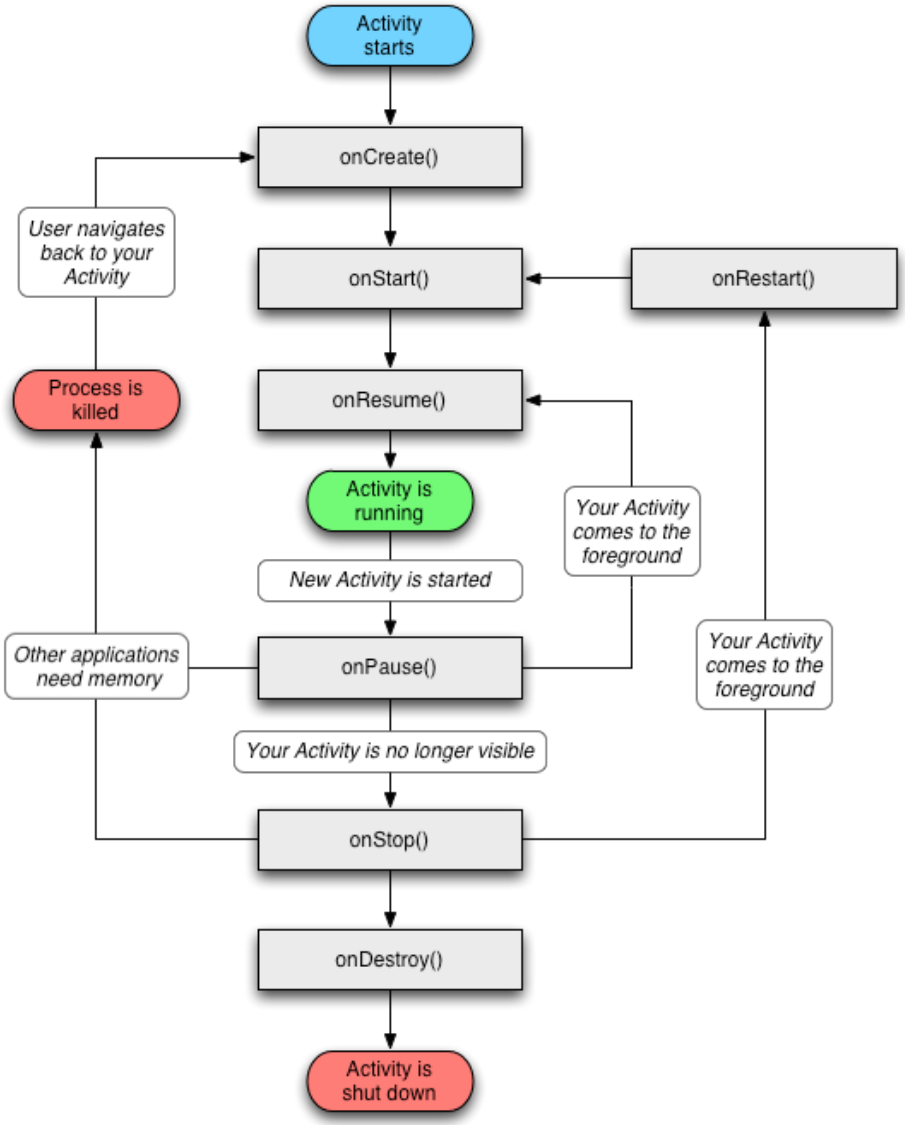


El ciclo de vida de las actividades



Visible y
en foreground

El ciclo de vida de las actividades



onCreate()

- Se invoca cuando se crea la actividad
- Establece el estado inicial de la actividad
  - Llama a super.onCreate()
  - Establece la interfaz con setContentView()
  - Obtiene la referencia a views o elementos de la interfaz de usuario
  - Configura las views según sea preciso

onCreate()

```
public class MainActivity extends Activity {

    private final String TAG = "UbicacionenMapa";

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

        Toast.makeText(this, R.string.string_onCreate, Toast.LENGTH_SHORT).show();

        final EditText texto = (EditText)findViewById(R.id.location);
        final Button boton = (Button) findViewById(R.id.map_button);

        boton.setOnClickListener(new Button.OnClickListener() {
            @Override
            public void onClick(View v) {
                try {
                    String direccion = texto.getText().toString();
                    direccion = direccion.replace(' ', '+');
                    Intent geoIntent = new Intent(android.content.Intent.ACTION_VIEW,
                            Uri.parse("geo:0,0?q=" + direccion));
                    startActivity(geoIntent);
                } catch (Exception e) {
                    Log.e(TAG, e.toString());
                }
            }
        });
    }
```

onRestart()

- Se invoca si la actividad se ha detenido, pero se va a iniciar de nuevo.
- Acciones típicas:
  - Cualquier procesamiento especial necesario sólo después de haberse detenido la actividad

onStart()

- Se invoca cuando la actividad va a hacerse visible
- Acciones típicas:
  - Iniciar procedimientos propios sólo del estado visible
  - Cargar o resetear el estado de una aplicación persistente

onResume()

- Se invoca cuando la actividad es visible y va a interactuar con el usuario
- Acciones típicas:
  - Procedimientos propios del primer plano

onPause()

- Se invoca cuando la actividad va a perder el primer plano, porque otra actividad pasa al foreground
- Acciones típicas:
  - Detención de procedimientos propios del foreground
  - Almacenar estado persistente

onStop()

- Se invoca cuando la actividad ya no está visible para el usuario
  - (Puede reiniciarse más adelante)
- Acciones típicas:
  - Guardar el estado de la actividad

- **Importante:** puede que no se invoque si Android mata la aplicación

onDestroy()

- Se invoca cuando se va a destruir la actividad
- Acciones típicas
  - Liberar recursos de la actividad

- **Importante:** puede que no se invoque si Android mata la aplicación

Métodos del ciclo de vida de las actividades

```
@Override
protected void onStart(){
    super.onStart();
    Log.i(TAG, (String) getText(R.string.string_onStart));
    Toast.makeText(this,R.string.string_onStart,Toast.LENGTH_SHORT).show();
}

@Override
protected void onRestart(){
    super.onRestart();
    Log.i(TAG, (String) getText(R.string.string_onRestart));
    Toast.makeText(this,R.string.string_onRestart,Toast.LENGTH_SHORT).show();
}

@Override
protected void onResume(){
    super.onResume();
    Log.i(TAG, (String) getText(R.string.string_onResume));
    Toast.makeText(this,R.string.string_onResume, Toast.LENGTH_SHORT).show();
}
```

Métodos del ciclo de vida de las actividades

```
@Override
protected void onPause(){
    super.onPause();
    Log.i(TAG, (String) getText(R.string.string_onPause));
    Toast.makeText(this,R.string.string_onPause,Toast.LENGTH_SHORT).show();
}

@Override
protected void onStop(){
    super.onStop();
    Log.i(TAG, (String) getText(R.string.string_onStop));
    Toast.makeText(this,R.string.string_onStop,Toast.LENGTH_SHORT).show();
}

@Override
protected void onDestroy(){
    super.onDestroy();
    Log.i(TAG, (String) getText(R.string.string_onDestroy));
    Toast.makeText(this,R.string.string_onDestroy,Toast.LENGTH_SHORT).show();
}
```

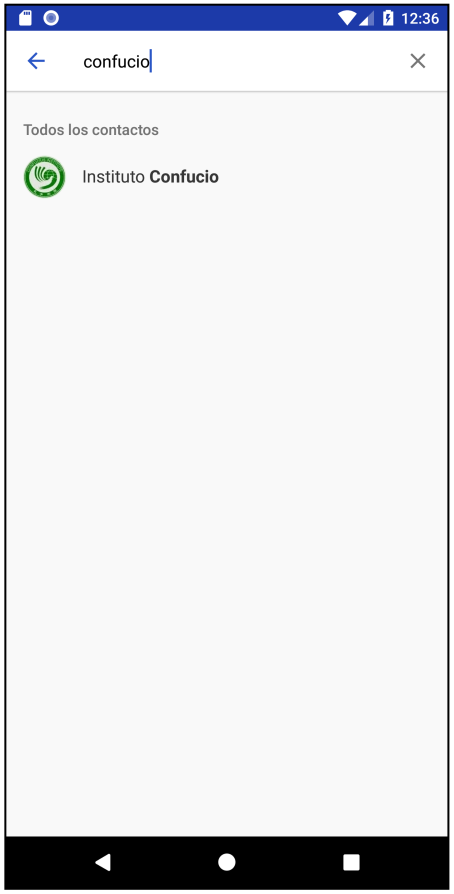
Iniciación programática de actividades

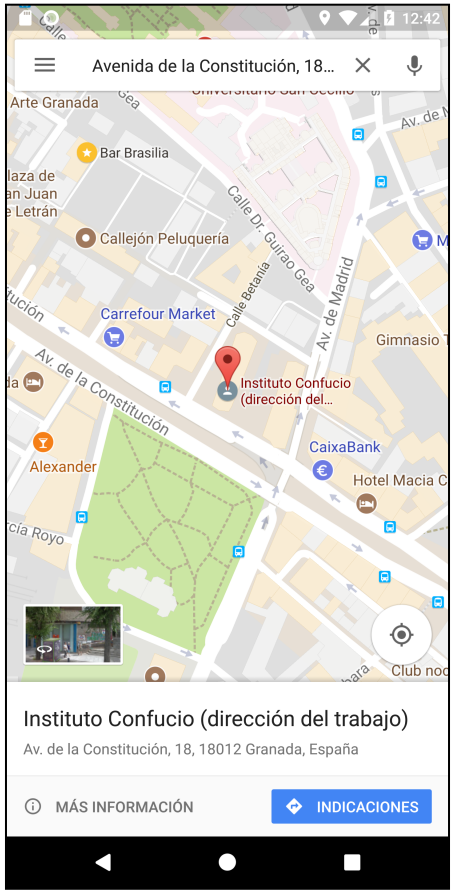
- Crear un objeto Intent que especifique la actividad a iniciar
- Pasar el objeto Intent que hemos creado a métodos como:
  - startActivity()
  - StartActivityForResult()
    - La actividad invocada llama a un método de devolución cuando termina para devolver un resultado

Iniciación de actividades

```
boton.setOnClickListener(new Button.OnClickListener() {
    @Override
    public void onClick(View v) {
        try {
            String direccion = texto.getText().toString();
            direccion = direccion.replace(' ', '+');
            Intent geoIntent = new Intent(android.content.Intent.ACTION_VIEW,
                    Uri.parse("geo:0,0?q=" + direccion));
            startActivity(geoIntent);
        } catch (Exception e) {
            Log.e(TAG, e.toString());
        }
    }
});
```

ContactosenMapa





ContactosenMapa

```
public class MainActivity extends AppCompatActivity {

    private static final String DATA_MIMETYPE = ContactsContract.Data.MIMETYPE;
    private static final Uri DATA_CONTENT_URI = ContactsContract.Data.CONTENT_URI;
    private static final String DATA_CONTACT_ID = ContactsContract.Data.CONTACT_ID;

    private static final String CONTACTS_ID = ContactsContract.Contacts._ID;
    private static final Uri CONTACTS_CONTENT_URI = ContactsContract.Contacts.CONTENT_URI;

    private static final String STRUCTURED_POSTAL_CONTENT_ITEM_TYPE =
            ContactsContract.CommonDataKinds.StructuredPostal.CONTENT_ITEM_TYPE;
    private static final String STRUCTURED_POSTAL_FORMATTED_ADDRESS =
            ContactsContract.CommonDataKinds.StructuredPostal.FORMATTED_ADDRESS;

    private static final int PICK_CONTACT_REQUEST = 0;
    static String TAG = "Contactos en Mapa";

…
```

ContactosenMapa

```
…
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.i(TAG, (String) getText(R.string.string_onCreate));
        Toast.makeText(this,R.string.string_onCreate, Toast.LENGTH_SHORT).show();

        final Button boton = (Button) findViewById(R.id.boton);

        boton.setOnClickListener(new Button.OnClickListener() {
            @Override
            public void onClick(View v) {
                try {
                    Intent intent = new Intent(Intent.ACTION_PICK, CONTACTS_CONTENT_URI);
                    startActivityForResult(intent, PICK_CONTACT_REQUEST);
                } catch (Exception e) {
                    //No hacer nada
                }
            }
        });
    }
```

Activity.setResult()

- La actividad iniciada puede establecer el resultado a devolver invocando el método Activity.setResult()
  - public final void setResult(int resultCode)
  - public final void setResult(int resultCode, Intent data)
- resultCode (valor entero)
  - RESULT_CANCELED
  - RESULT_OK
  - RESULT_FIRST_USER
  - (podemos añadir códigos personalizados)

onActivityResult

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
  if (resultCode == Activity.RESULT_OK && requestCode == PICK_CONTACT_REQUEST) {
    ContentResolver cr = getContentResolver();
    Cursor cursor = cr.query(data.getData(), null, null, null, null);

    if (null != cursor && cursor.moveToFirst()) {
      String id = cursor.getString(cursor.getColumnIndex(CONTACTS_ID));
      String where = DATA_CONTACT_ID + " = ? AND " + DATA_MIMETYPE + " = ?";
      String[] whereParameters=new String[] {id, STRUCTURED_POSTAL_CONTENT_ITEM_TYPE};
      Cursor addrCur = cr.query(DATA_CONTENT_URI, null, where, whereParameters, null);
```

onActivityResult

```
if (null != addrCur && addrCur.moveToFirst()) {
        String formattedAddress = addrCur.getString(addrCur
                  .getColumnIndex(STRUCTURED_POSTAL_FORMATTED_ADDRESS));
        if (null != formattedAddress) {
          formattedAddress = formattedAddress.replace(' ', '+');
          Intent geoIntent = new Intent(android.content.Intent.ACTION_VIEW,
              Uri.parse("geo:0,0?q=" + formattedAddress));
          startActivity(geoIntent);
        }
      }
      if (null != addrCur)
        addrCur.close();
    }
    if (null != cursor)
      cursor.close();
  }
}
```

Cambios en la configuración

- La configuración del dispositivo puede cambiar durante la ejecución de la aplicación
  - Teclado
  - Idioma
  - Orientación
  - Etc.
- Cuando se producen cambios de configuración, Android suele matar la actividad actual y reiniciarla

Cambios en la configuración

- El reinicio de la actividad debe ser suficientemente rápido
- Si es necesario podemos:
  - Conservar un objeto que contenga información de estado importante durante el cambio de configuración
  - Gestionar manualmente el cambio de configuración

Conservar un objeto Java

- Aquellos datos cuyo (re)cálculo sea costoso se pueden almacenar en un objeto para acelerar los cambios de configuración.
- Redefinir el método onRetainNonConfigurationInstance() para que construya y devuelva el objeto que contiene la configuración.
  - Se llama entre onStop() y onDestroy()
- Llamar a getLastNonConfigurationInstance() en onCreate() para recuperar el objeto guardado

IMPORTANTE: Métodos "deprecated" en favor de otros de la clase Fragment (la estudiaremos)

Reconfiguración manual

- Podemos evitar que el sistema reinicie la actividad
- Declararemos los cambios de configuración que la actividad puede gestionar en el fichero AndroidManifest.xml

```
<activity android:name="MiActividad"
 android:configChanges="orientation|screensize|keyboardHidden"
 ...>
```

- Cuando se produce el cambio en la configuración
  - Se llama al método onConfigurationChanged()
  - Se pasa como parámetro un objeto que especifica la nueva configuración del dispositivo.