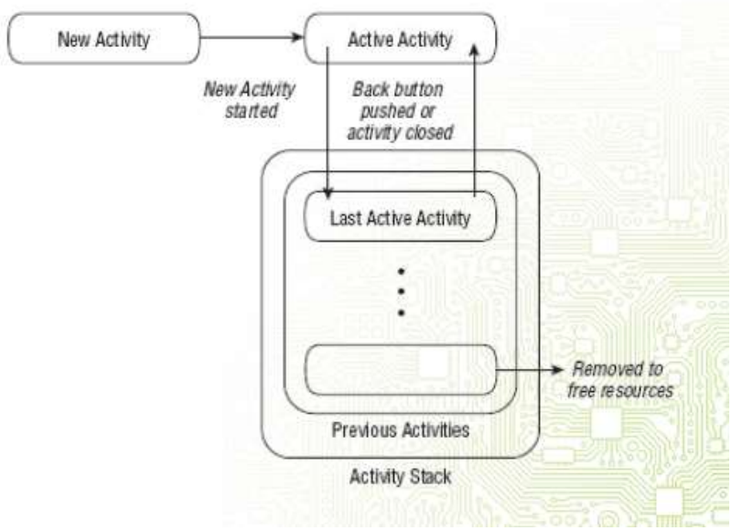
4. ACTIVIDADES E INTENCIONES

Xavier Lara

INDICE

1. APLICACIONES

- La aplicaciones en Android solo tienen un primer plano que ocupa toda la pantalla
- Las aplicaciones están formadas por **actividades**
- En un momento dado una actividad pasa al primer plano y se coloca por encima de otra formado **una pila de actividades**
- El botón back cierra la actividad y recupera la anterior de la pila



1. APLICACIONES

- La aplicaciones en Android no tienen control de su ciclo de vida
- Deben estar preparadas para su terminación en cualquier momento
- Cada aplicación se ejecuta en su propio proceso
- El runtime de Android gestiona el proceso de cada aplicación y por extensión de cada Actividad que contenga.

2. ACTIVIDADES

Actividad

- Representa una cosa concreta que puede hacer el usuario de la interfaz de usuario
- Corresponden con una pantalla
- Muestra los controles de la interfaz de usuario y reacciona ante las interacciones del mismo
- Es una clases derivada de la clase Activity
- Toda actividad se declara en el archivo AndroidManifest.xml

2. ACTIVIDADES

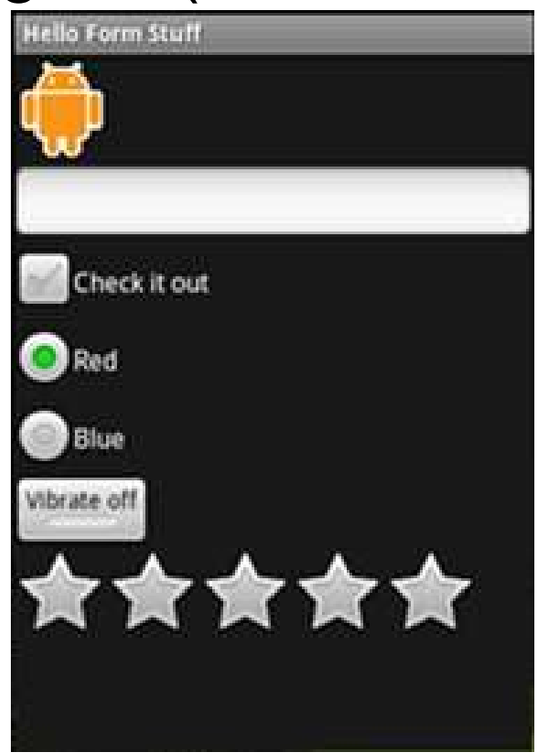
Normalmente una aplicación consta de varias actividades

- Cada pantalla se implementa como una actividad
- Moverse a la siguiente actividad supone llamar al método
  - **startActivity()**
  - **startActivityForResult()**
- Una aplicación puede reusar actividades de android o de otras aplicaciones

2. ACTIVIDADES

View

- Una actividad se compone de todo tipo de controles o widgets llamados **View** en Android.
- La clase **View** es la clase base de todos los widgets. (Button, EditText, TextView…)
- La clase **ViewGroup** es la clase base de los layouts y de otras vistas compuestas



2. ACTIVIDADES

Creando una actividad

```
public class HelloWorld extends Activity {
@Override
    public void onCreate(Bundle saveInstanceState) {
        super.onCreate(saveInstanceState);
        setContentView(R.layout.main);
    }
        :
}
```

onCreate: Se llama a este método cuando se crea la actividad

setContentView: Asigna a la vista el contenido del recurso layout

R.layout.main: Recurso de layout de la aplicación

2. ACTIVIDADES

Moverse a la siguiente actividad

Lanza una nueva actividad sin recibir el resultado

```
startActivity(intent);
```

Lanza una nueva actividad y espera el resultado

```
startActivityForResult(intent, requestCode);
```

Cuando retorna la actividad llamada, se invoca al método **onActivityResult** pasándole el requestCode con el que se lanzó desde la actividad

onActivityResult(int requestCode, int resultCode, Intent result)

3. CICLO DE VIDA DE UNA ACTIVIDAD

- Durante la vida de una actividad esta pasa por una serie de estados

- La clase Activity existen métodos para ser redefinidos (override) en sus clases derivadas que incluyen el código a ejecutar en las transiciones entre estados

- Los métodos redefinidos siempre deben llamar al método de la superclase

```
public class HelloWorld extends Activity {
@Override
 public void onCreate(Bundle saveInstanceState) {
   super.onCreate(saveInstanceState);
 }
}
```

3. CICLO DE VIDA DE UNA ACTIVIDAD

Estados de una actividad

- **Activo (Running):** La actividad está encima de la pila, es visible, tiene el foco.

- **Pausado (Paused):** La actividad es visible pero no tiene el foco. Se alcanza este estado cuando pasa a activa otra actividad transparente o que no ocupa toda la pantalla. Cuando una Actividad es tapada por completo pasa a estar parada.

- **Parado (Stopped):** Cuando la actividad no es visible. Se recomienda guardar el estado de la ui, preferencias, etc

- **Destruido (Destroyed):** Cuando la Actividad termina, o es matada por el runtime de Android. Sale de la Pila de Actividades.

3. CICLO DE VIDA DE UNA ACTIVIDAD

Métodos de transición de estados

- **onCreate(Bundle)**
  - Se invoca cuando la Actividad se arranca por primera vez.
  - Se utiliza para tareas de inicialización como crear la interfaz de usuario de la Actividad.
  - Su parámetro es null o información de estado guardada previamente por onSaveInstanceState()

- **onStart()**
  - Se invoca cuando la Actividad va a ser mostrada al usuario

- **onResume()**
  - Se invoca cuando la actividad va a empezar a interactuar con el usuario

3. CICLO DE VIDA DE UNA ACTIVIDAD

Métodos de transición de estados

- **onPause()**
  - Se invoca cuando la actividad va a pasar al fondo porque otra actividad ha sido lanzada para ponerse delante.
  - Se utiliza para guardar el estado de la Actividad

- **onStop()**
  - Se invoca cuando la actividad va a dejar de ser visible y no se necesitará durante un tiempo.
  - Si hay escasez de recursos en el sistema, este método podría no llegar a ser invocado y la actividad ser destruida directamente

3. CICLO DE VIDA DE UNA ACTIVIDAD

Métodos de transición de estados

- **onRestart()**
  - Se invoca cuando una actividad parada pasa a estar activa

- **onDestory()**
  - Se invoca cuando la Actividad va a ser destruida.
  - Si hay escasez de recursos en el sistema, este método podría no llegar a ser invocado y la actividad ser destruida directamente.

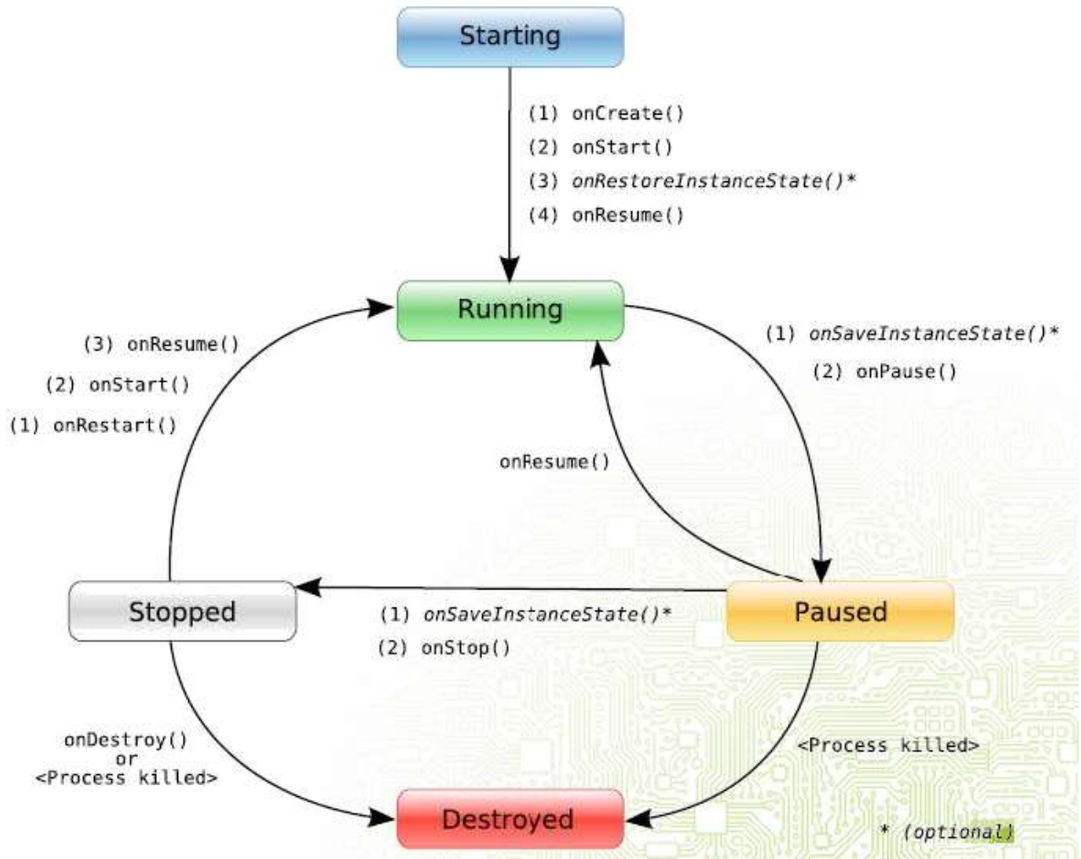
3. CICLO DE VIDA DE UNA ACTIVIDAD

Métodos de transición de estados

- **onSaveInstanceState(Bundle)**
  - Se invoca para permitir a la actividad guardar su estado de la ui
  - Normalmente no necesita ser redefinido

- **onRestoreInstanceState(Bundle)**
  - Se invoca para recuperar el estado guardado por onSaveInstanceState().
  - Normalmente no necesita ser redefinido

3. CICLO DE VIDA DE UNA ACTIVIDAD



4. INTENCIONES

- **Representan la “intención” o solicitud de que alguno de los componentes lleve a cabo una tarea**

- Las intenciones ofrecen un servicio de paso de mensajes que permite interconectar componentes de la misma o de distintas aplicaciones

- Las intenciones se utilizan para:
  - Arrancar actividades
  - Enviar eventos a múltiples destinatarios

4. INTENCIONES

Una intención queda descrita por:

- Acción que se quiere lanzar (MAIN - EDIT- PICK, …)
- Dato sobre el que actúa la acción (URI)
- Extras (int, String, Serializable,…)
- Component (org.osl.curso.HelloActivity)

```
Intent intent = new Intent(Intent.ACTION_EDIT)
Intent.setData(Uri.parse(“content://contacts/people/1”);

Intent intent = new Intent(Intent.ACTION_EDIT)
Intent.setData(Uri.parse(“http://www.google.es”);
```

4. INTENCIONES

Hay dos formas de invocar a una intención: **explícita, implícita**

- **Invocación explícita**
  Se especifica explícitamente en el código que componente es el encargado de manejar la intención.

- **Invocación implícita**
  Es la plataforma la que determina, a través de un **proceso de resolución de intenciones**, que componente es el más apropiado para manejar la intención. Un componente declara su capacidad para atender a una intención mediante el tag <intent-filter> en el archivo AndroidManifest.xml

4. INTENCIONES

- Los **filtros de intenciones** registran cuales son los componentes capaces de llevar una acción sobre un conjunto de datos.

  Los componentes se registran usando el tab <intent-filter> del manifiesto especificando una acción, una categoría y unos datos

```
<activity android:name=".Hello“
                        android:label="@string/app_name">
 <intent-filter>
  <action android:name="android.intent.action.MAIN" />
  <category android:name="android.intent.category.LAUNCHER"/>
 </intent-filter>
</activity>
```

- El emisor de la intención puede pertenecer a cualquier aplicación

4. INTENCIONES

- Las intenciones se pueden usar para moverse entre actividades

- La intención puede ser explícita indicando la actividad destino o implícita especificando una acción y unos datos y dejando que se encuentre la actividad adecuada en tiempo de ejecución

Lanzar una actividad de manera explícita

```
Intent intent = new Intent(Context, Activity.class);
startActivity(intent);
```

Lanzar una actividad de manera implícita

```
Intent intent = new Intent(Intent.ACTION_DIAL,
                        URI.parse(tel:928-76-34-26));
startActivity(intent);
```

4. INTENCIONES

- Una actividad también se puede lanzar para que devuelva un resultado mediante el método **startActivityForResult**

- Cuando termina esta subactividad se llama al método onActivityResult de la actividad padre desde la que fue arrancada

4. INTENCIONES

- Las intenciones implícitas pueden utilizarse para que futuros componentes proporcionen acciones que pueda añadirse al menú, sin necesidad de recompilar en el futuro.

- Muchas aplicaciones nativas emplean este mecanismo para extender su funcionalidad a medida que nuevas actividades van implementando nuevas acciones previstas

- Permiten extender las aplicaciones de la misma manera que lo hacen los plugins

4. INTENCIONES

- Las intenciones se pueden enviar a muchos receptores mediante el método sendBroadcast
- Las intenciones las reciben los broadcast receivers
- Las intenciones de broadcast se utilizan para notificar sobre eventos del sistema o de otras aplicaciones
- Por ejemplo Android utiliza las intenciones de broadcast para enviar eventos del sistema como cambios en la carga de la batería, conexiones de red, llamadas entrantes, sms entrantes,...