Componentes Android

Alexis De la Hoz Manotas

adelahoz6@cuc.edu.co

2021-1

UNIVERSIDAD DE LA COSTA
CUC
1970

Evaluación Corte 1 2021-1

Sem.	Fecha Est.	Actividad	Descripción
03	Febrero 17	Proyecto Nº 1 (33%-Nota Talleres)	Informe de Diseño. Tiempo Estimado: 1 Semana.
04	Febrero 24	Artículo Nº 1 (33%-Nota Talleres)	Investigación en bases de datos. Tema: Indicado en la actividad. Tiempo estimado: 1 días.
05	Marzo 03	Cuestionario Nº 1 (33%-Nota Talleres)	Cuestionario de desarrollo online. Tema: Unidad 1. Preguntas: 10 Tiempo Estimado: 1 hora.

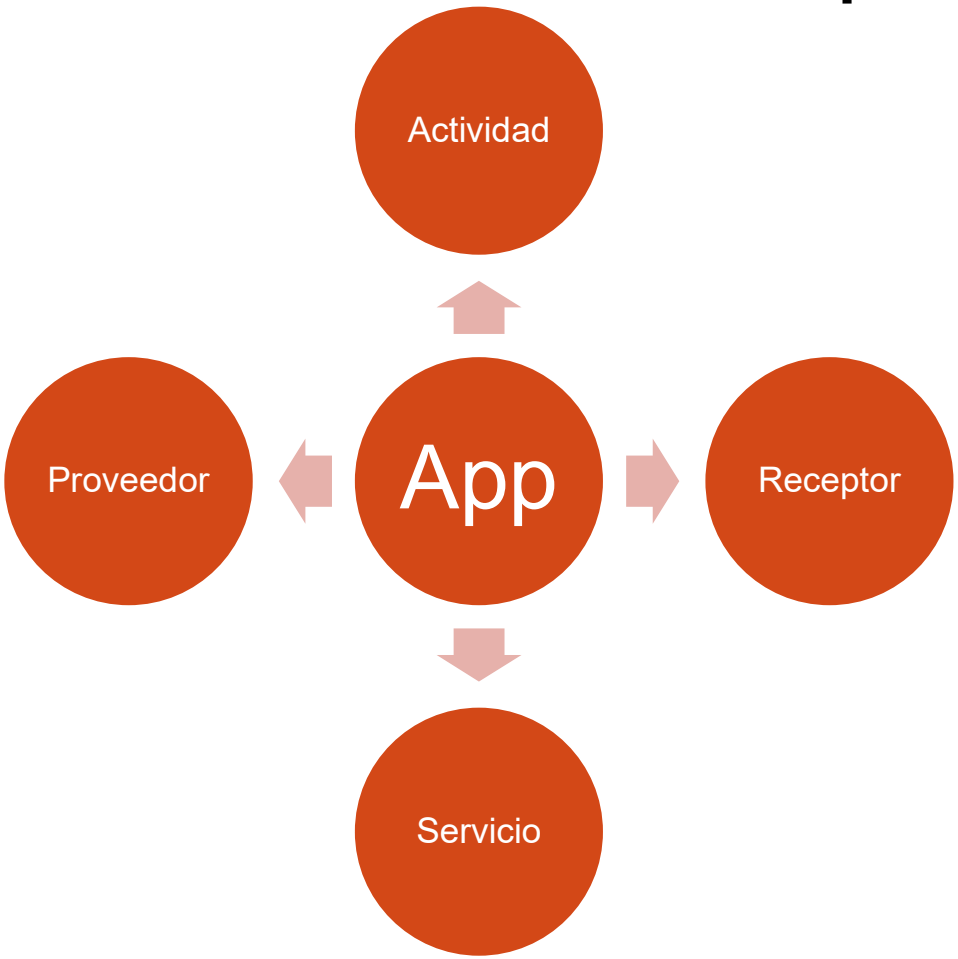
Componentes Android

- Componentes de una aplicación
- Estructura de un proyecto Android
- AVD (Dispositivos Virtuales)
- Android Studio
- Preguntas

Componentes de una aplicación

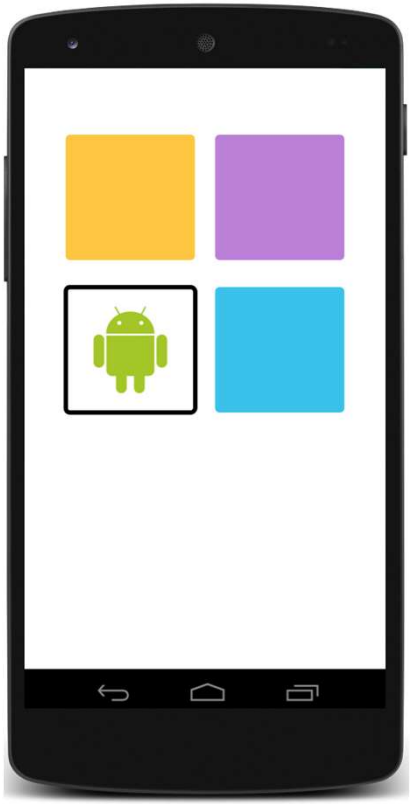
- Paquete de Android (APK)
  - Código Fuente (Java, Kotlin, C++)
  - Archivos de recursos
  - Datos
- Cada apk ejecuta un proceso (VM)
- Permisos deben ser concedidos explícitamente

Componentes de una aplicación



Actividades (Activity)

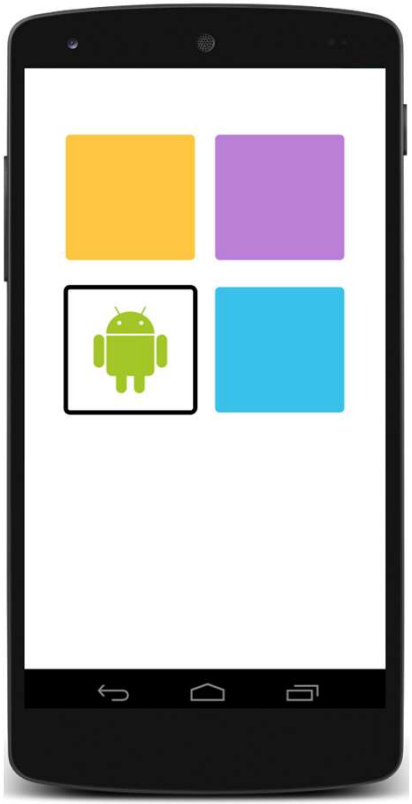
- Es el punto de entrada de interacción con el usuario.
- Representa una pantalla individual con una interfaz de usuario.
- Cada actividad es independiente.
- Entre sus alcances se encuentra:
  - Realizar un seguimiento de lo que realmente le interesa al usuario (lo que está en pantalla) para garantizar que el sistema siga ejecutando el proceso que aloja la actividad.
  - Saber que los procesos usados con anterioridad contienen elementos a los que el usuario puede regresar (actividades detenidas) y, en consecuencia, priorizar más esos procesos que otros.
  - Ayudar a la aplicación a controlar la finalización de su proceso para que el usuario pueda regresar a las actividades con el estado anterior restaurado.
  - Permitir que las aplicaciones implementen flujos de usuarios entre sí y que el sistema los coordine (el ejemplo más común es compartir).
- Las actividades se implementan como subclases de la clase Activity



Actividades (Activity)

- Una app usualmente contiene varias actividades.
- Generalmente implementa una pantalla en la app.
- Una actividad usualmente es identificada como principal.
- Están controladas por su ciclo de vida.
- Deben declararse en el manifiesto de la app:

```
<manifest ... >
  <application ... >
      <activity android:name=".ExampleActivity" />
      ...
  </application ... >
  ...
</manifest >
```



Actividades (Activity)

Ciclo de vida de una actividad

- onCreate():
  Se activa al crear la actividad.
  Inicializar los componentes esenciales (setContentView)
  Al finalizar, invoca onStart().

- onStart():
  Se activa cada vez que la actividad va a ser visible e interactiva.
- onResume():
  Se invoca antes de desplegar la actividad.
  Captura todo lo que el usuario ingresa.
  Representa la mayor parte de la funcionalidad de la app.
  onPause() siempre va después de onResume().

Actividades (Activity)

Ciclo de vida de una actividad

- onPause():
  Se activa cuando la actividad pierde el enfoque y se encuentra detenida.
  Generalmente se pasa a estado detenido o reanudado.
  La UI puede actualizarse sin inconvenientes aunque no tenga el enfoque.
  No debe usarse para almacenar datos de aplicación, llamadas de red o transacciones.
  La siguiente llamada será onStop() o onResume().

- onStop():
  Se activa cuando la actividad ya no es visible al usuario.
  La siguiente llamada puede ser onRestart() o onDestroy().

- onRestart():
  Se activa cuando la actividad pasa del de estado detenido a inicio.
  Restaura el estado de la actividad desde el punto de detención.
  La siguiente llamada es onStart().

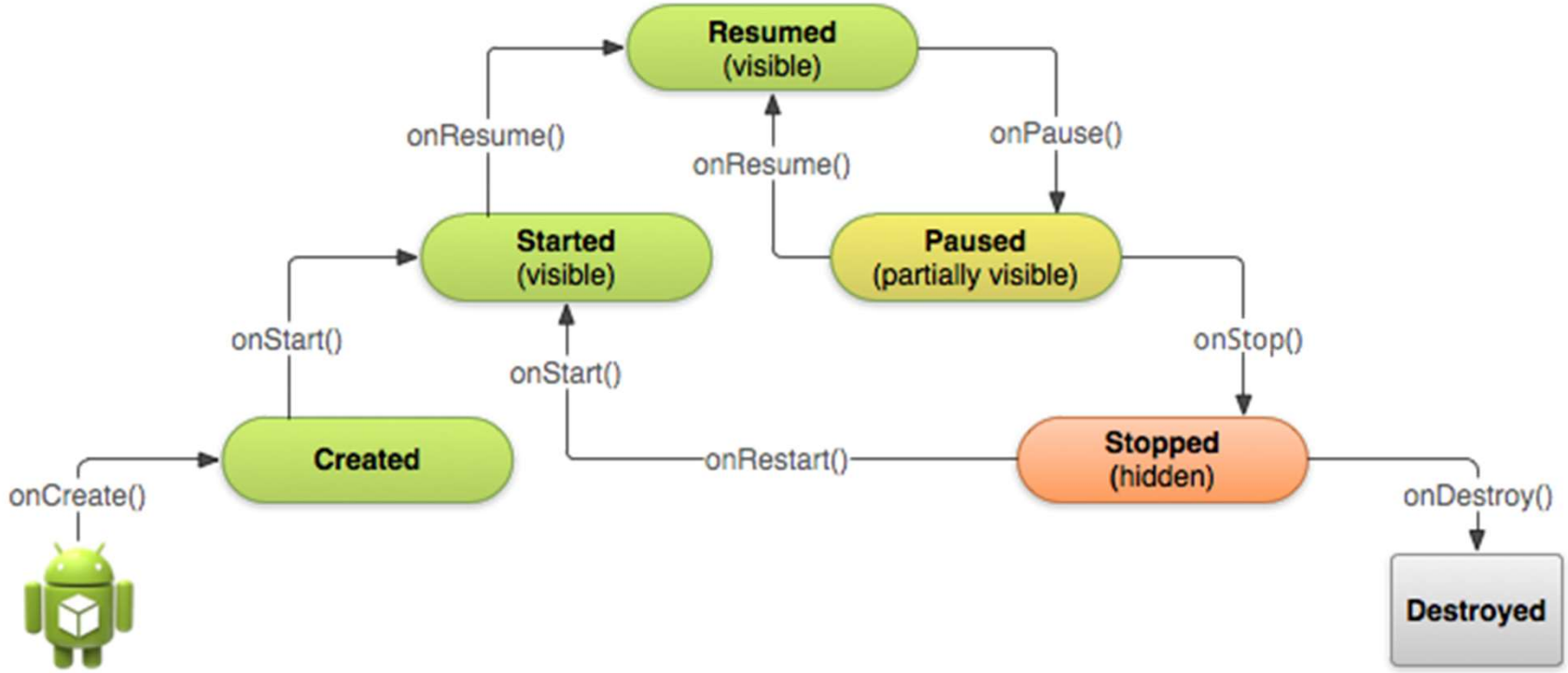
Actividades (Activity)

Ciclo de vida de una actividad

- onDestroy:
  - Se activa antes de destruir una actividad por completo.
  - Se deben liberar recursos y procesos de la actividad.

Actividades (Activity)

Ciclo de vida de una actividad



Servicios (Services)

- Punto de entrada general para una aplicación en segundo plano.
- Desarrolla operaciones de ejecución prolongada o procesos remotos.
- Se ejecutan de forma transparente para el usuario.
- Los servicios iniciados le indican al sistema que deben ejecutarse hasta su finalización.

Servicios (Services)

- Tienen más flexibilidad para administrarse y gestionarse por parte del sistema operativo.
- Ejemplos: Fondos de pantalla animados, protectores de pantalla, métodos de entrada, etc.
- Los servicios se implementan como subclases de Service.
- Declaración del servicio en el manifiesto:

```
<manifest ... >
  ...
  <application ... >
      <service android:name=".ExampleService" />
      ...
  </application>
</manifest>
```

Servicios (Services)

Tipos de servicios

- Primer Plano:
  Ejecuta una operación que el usuario detecta.
  Ejemplo: Una aplicación de audio reproduce una pista de audio.
  Deben mostrar una notificación.
  Se ejecutan aunque el usuario deje de interactuar con la aplicación.
- Segundo Plano:
  - Ejecuta una operación que el usuario no percibe directamente.
  - Ejemplo: Un servicio que comprima almacenamiento.
- Enlace:
  - Permite que un componente se vincule a él por el método bindService().
  - Ofrece una interfaz cliente-servidor para que los componentes interactúen con el servicio, gestione solicitudes, arroje resultados, y realice comunicación entre procsos (IPC).
  - Si ningún componente está lanzado a él, se destruye.

Servicios (Service)

Ciclo de vida de un servicio

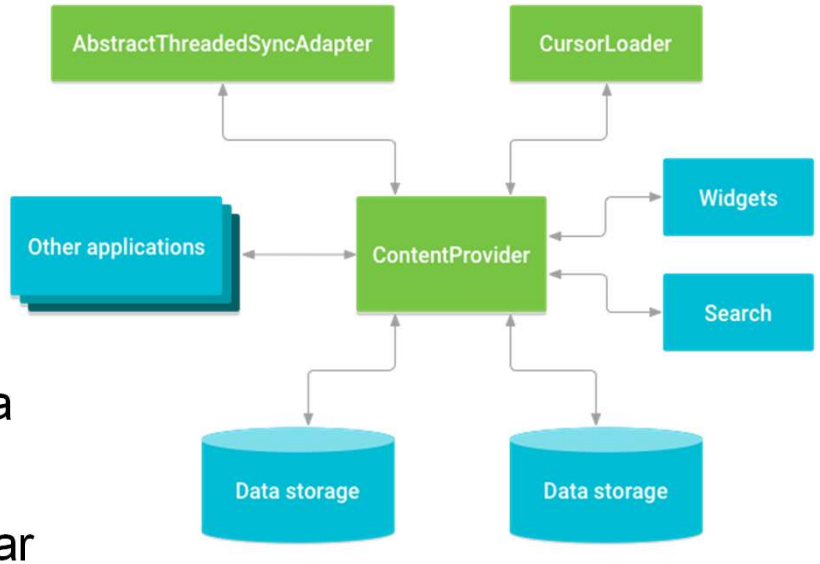
- onStartCommand():
  Se activa cuando se invoca startService() al iniciar el servicio.
  El servicio puede ejecutarse en segundo plano indefinidamente.
  Al finalizar, debe usarse stopSelf() o stopService() para terminarlo.
- onBind():
  Se activa cuando se invoca bindService() al enlazarse un componente al servicio.
  Debe retornar un Ibinder para los clientes del servicio.
  Retornar null si no se desean enlaces.
- onCreate():
  Se activa la primera vez que se crea el servicio (antes de onStartCommand() o onBind())
- onDestroy():
  Se activa cuando el servicio no se usa o va a ser destruido.
  Liberar recursos, subprocesos y receptores registrados o receptores.
  Es la última llamada recibida por el servicio.

Receptores (Broadcast Receivers)

- Posibilita que el sistema entregue eventos a la aplicación fuera de un flujo normal.
- Pueden entregarse mensajes a aplicaciones sin ejecutar.
- La mayoría de emisiones provienen del sistema.
- Ejemplo:
  Nivel de carga de batería, imagen capturada, pantalla apagada.
- Las aplicaciones también pueden realizar emisiones.
- No requieren una interfaz, pero pueden crear notificaciones en la barra de estado.
- Un receptor se implementa como subclase de BroadcastReceiver, y cada emisión se maneja como objetos Intent.

Proveedores (Content Providers)

- Administra un conjunto compartido de datos de la aplicación.
- Puede estar almacenado en el sistema de archivos, una base de datos Sqlite, en la Web o remotamente.
- Permite que otras aplicaciones pueden consultar y modificar los datos del proveedor.
- Representa un punto de entrada a una aplicación para publicar elementos de datos con nombre y se identifica mediante un esquema de URI.
- Las URI pueden sobrevivir al cierre de las aplicaciones.
- Pueden ser útiles para leer y escribir datos privados de la aplicación sin compartirlos.
- Un proveedor se implementa como subclase de ContentProvider y debe implementar un conjunto estándar de API que permita las transacciones con el mismo.



Proveedores del Sistema

Reloj Alarma

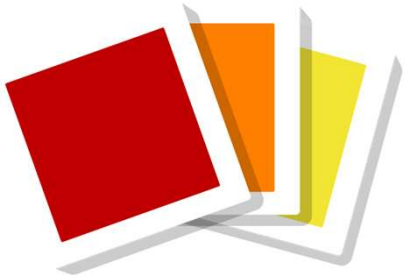
Navegador

Calendario

Contactos

Registro Llamadas

Documentos



Medios

Configuración

UNIVERSIDAD DE LA COSTA
CUC
1970

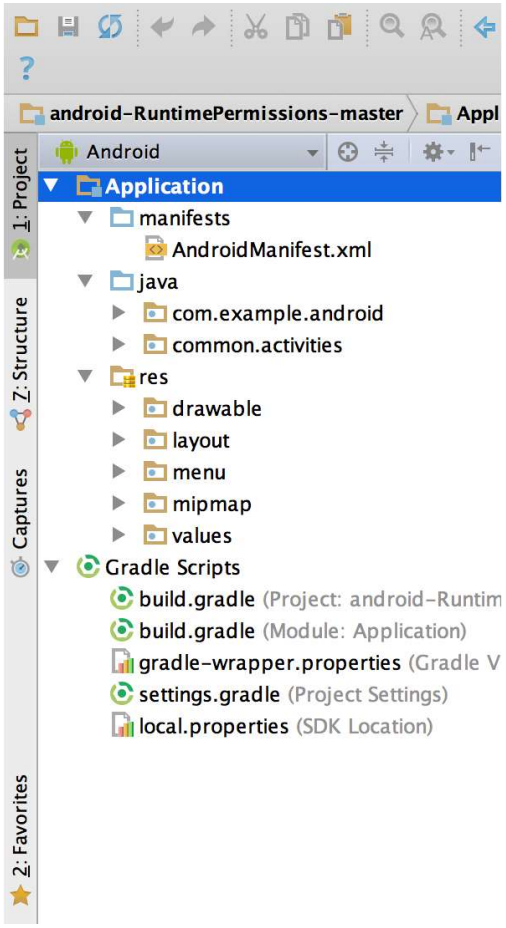
Manifiesto (Manifest)

- Archivo: AndroidManifest.xml
- Declara todos los componentes de la aplicación.
- Identifica los permisos de usuario requeridos por la aplicación.
- Declara el nivel de API mínimo requerido.
- Declara características de software y hardware requeridas por la aplicación.
- Declara bibliotecas de API requeridas por la aplicación (Vinculación).

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
    <application android:icon="@drawable/app_icon.png" ... >
        <activity android:name="com.example.project.ExampleActivity"
                  android:label="@string/example_label" ... >
        </activity>
        ...
    </application>
</manifest>
```

Estructura de un proyecto

- Manifiesto
- Código Fuente Java
- Recursos de la aplicación



Android Studio

- IDE Oficial desde 2014
- Reemplazo de Eclipse ADT
- Basado en Jetbrains Intellij IDEA
- Lenguajes soportados: Kotlin, Java, C++
- Kotlin oficialmente es lenguaje preferido desde 2019
- Disponible para Windows, MacOS, Linux
- Community Edition (Intellij IDEA cuenta con Ultimate Edition $499)

Android Studio

- Soporte Gradle
- Refactoring y atajos para Android
- Herramientas LINT para desempeño y usabilidad
- Integracion Proguard
- Layout Editor con soporte WYSWYG
- Soporte para Android Wear
- Soporte para Google Cloud Platform and Google App Engine
- Emulador con AVD (Android Virtual Device)

Android Studio

Windows

- Microsoft® Windows® 7/8/10 (64-bit)
- 4 GB RAM minimum, 8 GB RAM recommended
- 2 GB of available disk space minimum,
  4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
- 1280 x 800 minimum screen resolution

Mac

- Mac® OS X® 10.10 (Yosemite) or higher, up to 10.14 (macOS Mojave)
- 4 GB RAM minimum, 8 GB RAM recommended
- 2 GB of available disk space minimum,
  4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
- 1280 x 800 minimum screen resolution

Linux

- GNOME or KDE desktop
  Tested on gLinux based on Debian.
- 64-bit distribution capable of running 32-bit applications
- GNU C Library (glibc) 2.19 or later
- 4 GB RAM minimum, 8 GB RAM recommended
- 2 GB of available disk space minimum,
  4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
- 1280 x 800 minimum screen resolution

Preguntas

UNIVERSIDAD DE LA COSTA
CUC
1970