

Activities, ciclo de vida y recursos básicos



KEEPCODING

Tech School



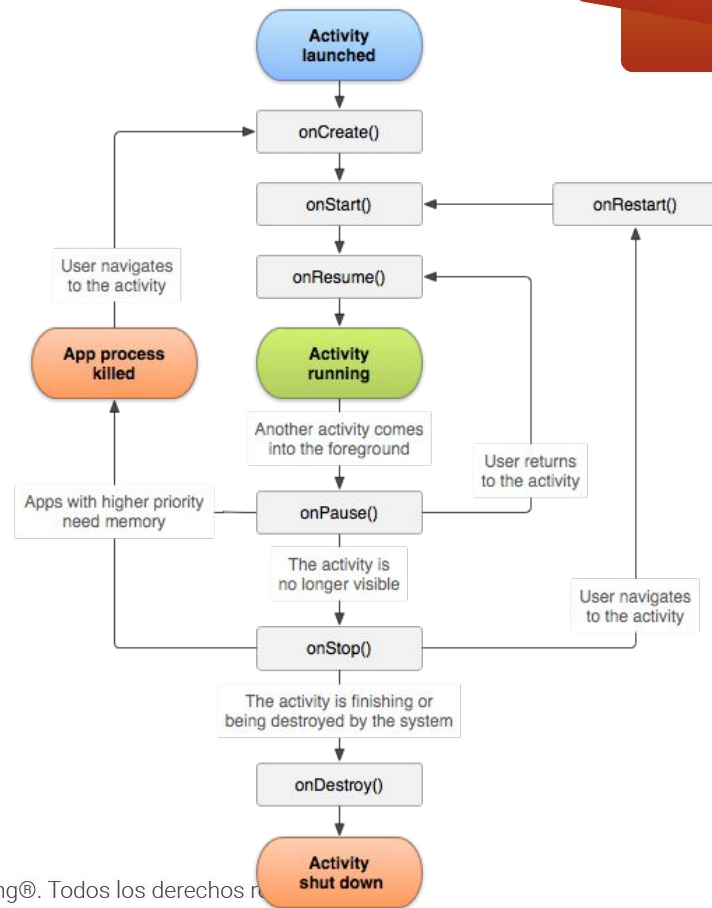
Activities

El ciclo de Vida I

onCreate: Se ejecuta cuando se crea una Activity por primera vez.

onStart: Se ejecuta después del onCreate cuando se crea por primera vez o cuando una Activity ha sido sacada de memoria y quiere volverse a mostrar.

onResume: Se ejecuta después del onStart o cuando una Activity que estaba en segundo plano vuelve a estar en primer plano.

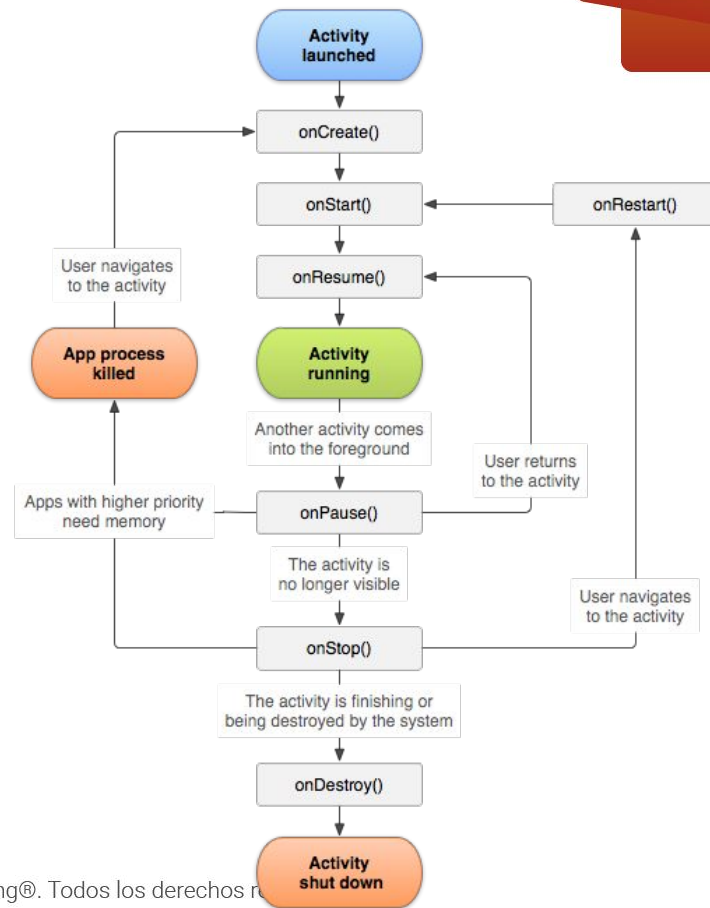


El ciclo de vida II

onPause: Se ejecuta cuando el usuario manda una Activity al segundo plano.

onStop: Se ejecuta cuando la Activity deja de ser visible.

onDestroy: Se ejecuta cuando el usuario elimina la Activity del segundo plano o Android requiere memoria. No siempre se ejecuta



Ver más: [Documentación Ciclo de Vida](#)

Proyecto Resumen

<https://github.com/Openbank-2023-Mobile-Android/fundamentosAndroid/commit/da47dc7c975acb1d9366171f0c78ec77b4f7aacf>



GitHub



Investigación sobre activities

¿Qué pasa cuando...

... rotas el dispositivo?

... cierras la aplicación y la vuelves a abrir?

... y si lo haces varias veces?

... y si eliminas a la app de la lista de app en segundo plano?



Almacenamiento Básico

onSaveInstanceState I

Cuando una activity va a destruirse con perspectivas de se recreada, entonces se llama a método `onSaveInstanceState(outState: Bundle)`

En `outState` es posible guardar valores.

```
outState.putInt(TAG_NUM, num)
```

Siendo `TAG_NUM` un String y `num` una variable de tipo Integer

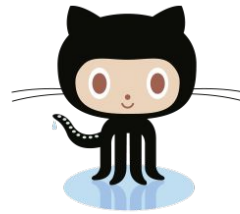
onSaveInstanceState II

Cuando se vuelva a ejecutar el método `onCreate(savedInstanceState: Bundle?)` se recibirán los valores guardados en el `outState` dentro de `savedInstanceState`.

Para recuperar los valores guardados anteriormente, utilizamos el siguiente código.

```
savedInstanceState?.run {  
  
    getString(TAG_NUM)?.let {  
  
        twText.setText(it)  
  
    }  
  
}
```

Proyecto Resumen



GitHub

SharedPreferences I

La forma más sencilla de almacenar datos en forma persistente es utilizando la API `SharedPreferences`. Se puede acceder a ella a través de un `Context` y permite almacenar duplas de clave valor (similar a lo que se ha realizado en `onSaveInstanceState`).

Para acceder a las `SharedPreferences` se debe ejecutar el siguiente código:

```
val sharedPref = getPreferences(Context.MODE_PRIVATE)
```

SharedPreferences II

Para guardar un valor en las las sharedPreferences debes:

```
with (sharedPref.edit()) {
```

```
    putString(TAG_USUARIO, string)
```

```
    commit() }
```

Para cargar un valor:

```
return sharedPref.getString(TAG_USUARIO, "")
```

SharedPreferences III

Es posible guardar un objeto de clases más allá de las clases básicas (String, Int, etc).

Para ello necesitaremos que:

- a) Nuestra clase implemente la anotación Serializable
- b) Transforma nuestra clase en un Json y guardar ese Json como String

SharedPreferences IV

Las sharedPreferences son:

- Un fichero de texto
- Fácil de usar
- No cifrado

Apto a para datos poco importantes

Proyecto Resumen

<https://github.com/Openbank-2023-Mobile-Android/fundamentosAndroid/commit/d47dc7c975acb1d9366171f0c78ec77b4f7aacf>



GitHub

Ejercicio Destacado



- Guárdate en las sharedPreferences el nombre de usuario y la contraseña que ha puesto el usuario.





Toast

Toast


Una forma sencilla de dar feedback al usuario es mediante los Toast.

Un toast es una ventana emergente de corta duración capaz de mostrar textos.

```
Toast.makeText(this, "Escribe algo primero", Toast.LENGTH_LONG).show()
```




Context



| ¿Qué es?

Interfaz abstracta que permite acceder a los **recursos del sistema Android** e **interactuar con sus componentes**.



| ¿Interfaz abstracta?

Interfaz abstracta:

Clases que la implementan. Ejemplo: Activity y Application

Permiten realizar ciertas acciones comunes en estos elementos

Ya lo habéis visto, ¡Repaso!



| ¿Recursos del sistema?

¡Recursos del sistema!

Strings, Dimens, Colors y Styles

Se verá en detalle a continuación

| ¿Interactuar con sus componentes?

¡Activities!

Mediante **intents** nos permite iniciar nuevas activities, servicios o enviar/solicitar información a otros sistemas en Android

Se verá en detalle en temas posteriores



| ¿Intents?

¡Intents!

Describen una acción que quieres **encargar** realizar

¿Quién realiza la acción? **ANDROID**

Además permite configurarlo, enviar información, solicitar información de vuelta

¿Qué puede hacer?

Abrir nuevas actividades (internas o externas), iniciar un servicio, enviar información, etc.



Recursos Básicos

|String

Acceso:

```
context.getString(R.string.success)
```

También accesible desde activities y fragments.



Dimens

Las dimensiones (dp) en Android se deben añadir en:
res/values/dimens.xml

Acceso:

```
context.resources.getDimension(R.dimen.icon_small)
```

Drawable

Los drawables en Android se deben añadir en:
res/values/drawable

Acceso:

```
ContextCompat.getDrawable(context, R.drawable.checkbox)
```

Colors

Los colores en Android se deben añadir en:
res/values/colors.xml

Acceso:

```
ContextCompat.getColor(context, R.color.gray)
```



Styles

Un estilo es un conjunto de atributos que se aplican a las vistas que, en Android, se deben añadir en:
`res/values/styles.xml`

Se verá en detalle en temas posteriores

Ejercicio Destacado



- Crea un estilo para tus editText y ponlos a tu gusto (utiliza la fuente que has añadido anteriormente, cambia el tamaño de la letra, etc.)
- Extrae todos los textos que introducidos a fuego por Strings





Arquitectura



| ¿Dónde estamos?

Dentro de MVWM:

- Modelo
- **Vista (Activity + Xml)**
- ViewModel



| ¿Objetivo?

Lograr una Activity lo más simple posible

Sus tareas son:

- Gestionar la UI
- Gestionar los recursos que dependen de Android
 - Obtener los Strings, Colores, Tamaños...
 - Llamadas al sistema...



Testing

| ¿Cómo se prueban las Activities?

Activities -> instrumentationTest

- Inconvenientes que conllevan:
 - Lentitud
 - Complejidad

Fin

