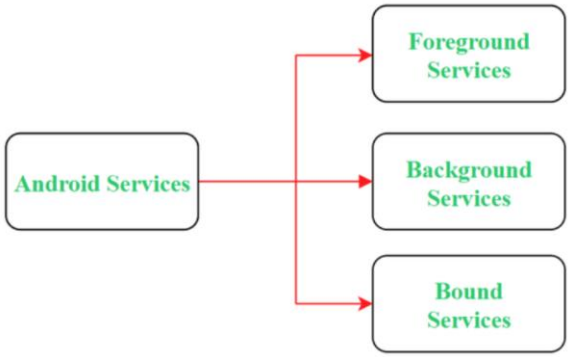
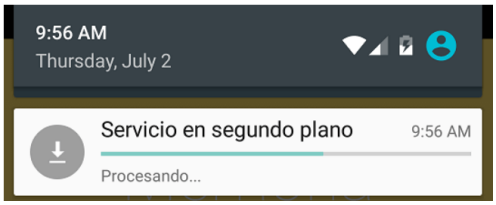
Crear servicios en Android

Un servicio es una entidad que ejecuta instrucciones en segundo plano sin que el usuario lo note en la interfaz. Son muy utilizados para realizar acciones de larga duración mientras las actividades muestran otro tipo de información. Por ejemplo, guardar la información en la base de datos, escuchar música mientras se ejecuta la aplicación, administrar conexiones de red, etc.

Hay tres tipos de servicios, y en todos ellos, **para crear un servicio siempre hay que extender de la clase Service**:



- **Servicios de Primer Plano (Foreground Services)**: Un servicio en primer plano tiene una prioridad crítica en Android, tanto así, que el usuario no puede terminarlo manualmente hasta que se termine. Para que esto sea posible, el servicio en primer plano debe proveer una notificación en la barra de estado que indique su existencia. La única forma de quitarlo del primer plano es terminándolo programáticamente o cuando este finalice su trabajo. El traslado a primer plano se lleva cabo con el método startForeground() dentro del servicio. Si deseas pasarlo a segundo plano de nuevo, entonces usas stopForeground(). Por ejemplo, aplicaciones de navegación GPS que necesitan estar visibles y accesibles constantemente.



- **Servicios de Fondo (Background Services)**: Son ideales para tareas que necesitan ejecutarse sin interacción directa con el usuario, como descargar datos o reproducir música. Estos servicios pueden seguir funcionando, aunque el usuario cambie de aplicación. Realiza una operación determinada y se cierra al momento de finalizarla.

- **Servicios Vinculados (Bound Services)**: Permiten la interacción entre una aplicación y el servicio, proporcionándoles una interfaz cliente-servidor. Estos servicios se ejecutan solo mientras una aplicación está conectada a ellos, lo que los hace perfectos para tareas que requieren comunicación continua, como la reproducción de música en aplicaciones de streaming. Un bound service puede comunicarse directamente con el componente al que está ligado, esto significa compartir datos estructurados, realizar operaciones en conjunto, etc.

Al crear un servicio, deberemos tener en cuenta lo siguiente:

- onCreate(): Se ejecuta cuando el servicio está creado en memoria. Si el servicio ya está activo, entonces se evita de nuevo su llamada.
- onStartCommand():Método donde se implementan las instrucciones del servicio. Se llama solo si el servicio se inició con startService(), aunque no es obligatorio. Puede retornar:
  - **START_NOT_STICKY**. Este campo indica que el servicio no debe recrearse al ser destruido sin importar que haya quedado un trabajo pendiente
  - **START_STICKY**: Crea de nuevo el servicio después de haber sido destruido por el sistema. En este caso llamará a onStartCommand() referenciando un intent nulo.
  - **START_REDELIVER_INTENT**: Crea de nuevo el servicio si el sistema lo destruyó. A diferencia de START_STICKY, esta vez sí se retoma el último intent que recibió el servicio.

- onBind(): Solo se ejecuta si el servicio fue ligado con bindService() por un componente. Retorna una interfaz de comunicación del tipo IBinder. Este método siempre debe llamarse, incluso dentro de los otros tipos de servicios, los cuales retornan null.
- onDestroy(): Se llama cuando el servicio está siendo destruido. Importantísimo que dentro de este método detengas los hilos iniciados.

Tus servicios no se instanciarán si no los declaras en el Android Manifest como un componente de la aplicación. Así que solo incluye la etiqueta <service> dentro del nodo <application>:

```
<application
... >
...
    <service
        android:name=".DownloadService"
        android:enabled="true"
        android:exported="true" >
    </service>

<application>
```

La etiqueta <service> contiene tres parámetros: name se refiere al nombre de la clase de implementación del servicio; enabled indica si es posible crear instancias del servicio; y exported establece si los componentes de otras apps pueden iniciar o interactuar con el servicio.

Ejemplo: Vamos a ver un ejemplo de un servicio de tipo background, que detecte si es noche o de día y en función de eso establezca el tema al modo oscuro o no.

Primero, necesitarás crear un BroadcastReceiver para detectar cambios en la hora o simplemente verificar la hora cada cierto intervalo. Después, configuras tu Service en segundo plano.

El BroadcastReceiver actúa como una especie de "escucha" que se activa cuando ocurre un evento específico para el que ha sido registrado. Estos eventos pueden incluir acciones como cambios en la conectividad de red, notificaciones del sistema, cambios en la configuración de la hora, entre otros.

1. **Crea el** BroadcastReceiver **para detectar el cambio en la hora, y en función de la hora pone el modo noche o no**:

```
import android.content.BroadcastReceiver
import android.content.Context
import android.content.Intent
import android.icu.util.Calendar
import android.util.Log
import androidx.appcompat.app.AppCompatDelegate

class TimeChangeReceiver : BroadcastReceiver() {

    override fun onReceive(context: Context, intent: Intent) {
        // Verifica si es de noche y cambia el tema
        if (isNightTime()) {
            // Cambia el tema al modo oscuro
            setDarkTheme(context)
        } else {
            // Cambia el tema al modo claro
            setLightTheme(context)
        }
    }

    private fun isNightTime(): Boolean {
        val currentTime =
Calendar.getInstance().get(Calendar.HOUR_OF_DAY)
        return currentTime >= 18 || currentTime < 6
    }

    private fun setDarkTheme(context: Context?) {
        // Cambia el tema de la app al modo oscuro

AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_YES
)
        Log.i("TimeChangeReceiver","setDarkTheme ON")
    }

    private fun setLightTheme(context: Context?) {
        // Cambia el tema de la app al modo claro

AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_NO)
        Log.i("TimeChangeReceiver","setLightTheme ON")
    }

}
```

2. **Registra el** BroadcastReceiver **en tu** Service **en segundo plano, e indica que reciba actualizaciones cada minuto**:

```
import android.app.Service
import android.content.Intent
import android.content.IntentFilter
import android.os.IBinder

class NightModeService : Service() {
    private lateinit var timeChangeReceiver: TimeChangeReceiver
    override fun onCreate() {
        super.onCreate()
        timeChangeReceiver = TimeChangeReceiver()
    }

    override fun onStartCommand(intent: Intent?, flags: Int, startId:
Int): Int {
        // Para recibir actualizaciones de tiempo cada minuto
        val filter = IntentFilter(Intent.ACTION_TIME_TICK)
        registerReceiver(timeChangeReceiver, filter)

        return START_REDELIVER_INTENT
    }

    override fun onBind(intent: Intent): IBinder? {
        return null
    }

    override fun onDestroy() {
        super.onDestroy()
        unregisterReceiver(timeChangeReceiver)
    }
}
```

3. **Declara el** Servicio y el BroadcastReceiver **en tu archivo** AndroidManifest.xml:

```
<service
    android:name="services.NightModeService"
    android:enabled="true"
    android:exported="false"></service>

<receiver
    android:name="services.TimeChangeReceiver"
    android:enabled="true"
    android:exported="false" />
```

4. **Inicia el** Service **en el** onCreate **de tu** MainActivity

```
// Inicia el servicio en segundo plano
val serviceIntent = Intent(this, NightModeService::class.java)
startService(serviceIntent)
```