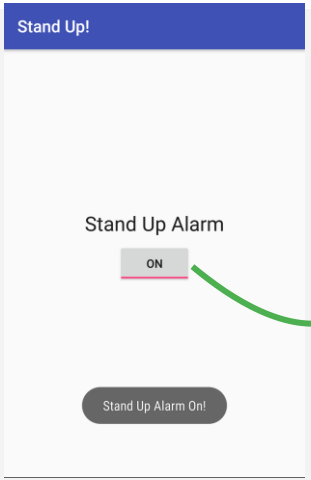
Desarrollo de apps con Android

10. Alarmas

¿Qué es una alarma en Android?

- Una alarma en Android no representa la alarma típica de un reloj.
- Se planifica algo para que se realice en un momento determinado.
- Inicia `Intents` en instantes de tiempo determinados o en intervalos de tiempo.
- Se pueden dar una única vez o ser recurrentes.
- Se pueden basar en horas de reloj o bien en tiempo transcurrido (ej., cada hora).
- No es necesario que la app se esté ejecutando para que las alarmas se activen.

Cómo funciona una alarma

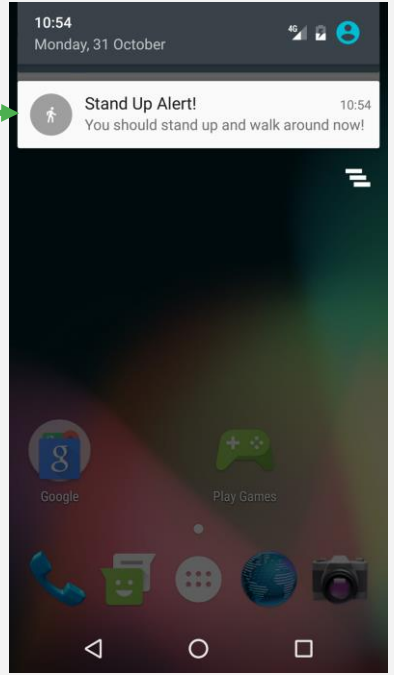


`Activity` crea una notificación y establece una alarma.

Alarm se activa y envía un `Intent`.

La app se puede destruir por lo que...

`BroadcastReceiver` despierta la app y entrega la notificación (aunque no siempre tiene que entregar una notificación).



Ventajas de utilizar alarmas

- Permite programar operaciones fuera del ciclo de vida de la app.
- No es necesario que la app se esté ejecutando para que se activen las alarmas.
- El dispositivo puede estar en modo suspensión.
- Permite programar sin depender de temporizadores ni de servicios que estén ejecutándose de manera continua.
- Se pueden utilizar conjuntamente con receptores de Broadcast para planificar trabajos o WorkRequests.

Unidades temporales

- Tiempo real transcurrido: tiempo desde que se inició el sistema.
  - Independiente de la zona horaria y local.
  - Hacer uso de intervalos y tiempos relativos.
  - Utilizar siempre que sea posible.
  - El tiempo transcurrido incluye el tiempo que el dispositivo ha estado en modo suspensión.
- Real Time Clock (RTC): hora UTC.

Comportamiento en activación

- Despierta la CPU del dispositivo si la pantalla estaba apagada.
  - Utilizar solamente para operaciones críticas en el tiempo.
  - Pueden agotar la batería.
- No “despierta” el dispositivo.
  - Activar la alarma cuando detecte que el dispositivo ya no está en modo suspensión.
  - Menos invasivo.

Tipos de alarma

	Elapsed Real Time (ERT): desde que se inicia el sistema	**Real Time Clock (RTC): hora del día**
No despierta dispositivo	ELAPSED_REALTIME	RTC
Despierta dispositivo	ELAPSED_REALTIME_WAKEUP	RTC_WAKEUP

Buenas prácticas en alarmas

Si todo el mundo se sincroniza al mismo tiempo...

Pensar en una app con millones de usuarios:

- Operación de sincronización del servidor basada en la hora del reloj.
- Cada instancia de la app sincroniza a las 11:00 p.m.

La carga en el servidor se podría traducir en una latencia muy alta o, en el peor de los casos, una denegación del servicio (*"denial of service"*).

Buenas prácticas en alarmas

- Añadir aleatoriedad a las peticiones de red que hagan las alarmas.
- Minimizar la frecuencia de las alarmas.
- Siempre que sea posible utilizar `ELAPSED_REALTIME`, en vez de la hora del reloj.

Batería

- Evitar despertar el dispositivo en la medida de lo posible.
- Hacer uso de las alarmas imprecisas.
  - Android sincroniza diferentes alarmas repetitivas declaradas como inexactas y las activa todas al mismo tiempo.
  - Reduce el consumo de batería.
  - Utilizar `setInexactRepeating()` en vez de `setRepeating()`.

No se debe utilizar una alarma

- Para tics, temporizadores, y mientras la app se está ejecutando utilizar `Handler`.
- Para sincronización del servidor, utilizar `SyncAdapter` con el servicio de mensajería en la nube.
- Para eficiencia de recursos y sin exactitud en el tiempo, utilizar `JobScheduler`.

AlarmManager

¿Qué es AlarmManager?

- `AlarmManager` proporciona el acceso a los servicios de alarma del sistema.
- Planifica una operación futura.
- Cuando la alarma expira, se activa y se hace la transmisión *(broadcast)* sobre el `Intent` registrado.
- Las alarmas son retenidas mientras el dispositivo está en modo suspensión.
- Activar alarmas puede despertar el dispositivo.

Creación del objeto AlarmManager

```
AlarmManager alarmManager =
    (AlarmManager) getSystemService(ALARM_SERVICE);
```

Programación de alarmas

Requisitos para programar alarmas

Programación de alarmas:

1. Tipo de alarma.
2. Hora de la activación.
3. Intervalo para las alarmas repetitivas.
4. Indicar `PendingIntent` para entregar a la hora especificada (de la misma manera que con las notificaciones).

Programar una alarma

- `set()`: única, alarma inexacta.
- `setWindow()`: única, alarma inexacta en una ventana de tiempo.
- `setExact()`: única.
  - Anterior al nivel de API 19, crea una alarma única exacta.
  - A partir del nivel de API 19, misma funcionalidad que `set()`.

Programar una alarma repetitiva

- `setInexactRepeating()` : repetitiva, alarma inexacta.
- `setRepeating()` :
  - Anterior al nivel de API 19, crea una alarma exacta repetitiva.
  - A partir del nivel de API 19, misma funcionalidad que `setInexactRepeating()`.

setInexactRepeating()

```
setInexactRepeating(
    int alarmType,
    long triggerAtMillis,
    long intervalMillis,
    PendingIntent operation)
```

Crear una alarma inexacta

```
alarmManager.setInexactRepeating(
    AlarmManager.ELAPSED_REALTIME_WAKEUP,
        SystemClock.elapsedRealtime()
            + AlarmManager.INTERVAL_FIFTEEN_MINUTES,
        AlarmManager.INTERVAL_FIFTEEN_MINUTES,
        alarmPendingIntent);
```

Comprobar una alarma existente

```
boolean alarmExists =
 (PendingIntent.getBroadcast(this,
       0, alarmPendingIntent,
       PendingIntent.FLAG_NO_CREATE) !=
null);
```

Cancelar una alarma

- Invocar al método `cancel()` de `AlarmManager`: Pasar el `PendingIntent`.

```
alarmManager.cancel(alarmPendingIntent);
```