TECNOLÓGICO NACIONAL DE MEXICO

INSTITUTO TECNOLÓGICO DE CIUDAD MADERO

Ingeniería en Sistemas Computacionales.

Tarea 12_ Unidad 7 WorkManager

Asignatura: Programación Nativa para Moviles

Docente: Jorge Peralta Escobar

Hora: 14:00 – 15:00 pm

Integrantes:

Del Angel Del Angel Erika Yaneth #20071926

Villaseñor Grimaldo Alejandro #19071548

Semestre: Enero – Junio 2025.

DEJANDO HUELLA

GITHUB -[Workspace/Programacion_movil at master · Alejandro19071548/Workspace]

RUTA DE APRENDIZAJE 1 – COMO PROGRAMAR TAREAS CON WORKMANAGER

VIDEO 1 – EUTA DE APRENDIZAJE 1

WorkManager es ideal para tareas que deben ejecutarse sí o sí, pero que no requieren respuesta inmediata. Es muy útil en aplicaciones que necesitan programar recordatorios, hacer sincronizaciones en segundo plano o realizar procesos largos sin afectar la experiencia del usuario.

WorkManager es útil para tareas que:

- Deben completarse aunque la app se cierre o el dispositivo se reinicie.
- No necesitan ejecutarse inmediatamente, como:
    - Enviar recordatorios.
    - Subir fotos o archivos.
    - Sincronizar datos.
    - Realizar copias de seguridad.
- Requieren condiciones específicas (por ejemplo: estar conectado a WiFi o tener suficiente batería).

Componentes básicos

- Worker: clase donde defines la tarea que quieres ejecutar.
- WorkRequest: solicitud que indica cuándo y cómo ejecutar una tarea (OneTimeWorkRequest o PeriodicWorkRequest).
- Constraints: condiciones que deben cumplirse para ejecutar la tarea (ej. batería cargando, red WiFi).
- WorkManager: clase que gestiona y programa las tareas.

CUESTIONARIO RUTA 1

1. ¿Qué herramienta te permite visualizar, supervisar y depurar los workers de tu app?

- ○ Profiler
- ◉ Inspector de tareas en segundo plano — ¡Correcto!
- ○ Logcat
- ○ Administrador de dispositivos

2. ¿Cuáles de las siguientes opciones son estados de trabajo terminal válidos?

Selecciona todas las respuestas que consideres correctas.

Parcialmente correcto.

- ☑ CANCELADO — ¡Correcto!
- ☐ ELIMINADO
- ☐ ERROR
- ☐ COMPLETADO

3. ¿Cuáles de las siguientes opciones son tipos válidos de solicitudes de trabajo?

Selecciona todas las respuestas que consideres correctas.

- ☑ `OneTimeWorkRequest` — ¡Correcto!
- ☐ `SingleWorkRequest`

4. La creación y puesta en cola de varias tareas dependientes y el orden en el que se deben ejecutar se denomina vinculación.

- ○ Verdadero
- ◉ Falso — ¡Correcto!

5. ¿En cuál de las siguientes situaciones son útiles las restricciones de trabajo?

- ◉ Verificar que se guarde una forma de pago válida en el dispositivo del usuario antes de que se ejecute el trabajo — Incorrecto.
- ○ Comprobar la hora antes de que se ejecute el trabajo
- ○ Verificar que el dispositivo esté conectado a una red Wi-Fi antes de descargar una gran cantidad de datos de la app
- ○ Comprobar que la app se haya abierto una cantidad determinada de veces antes de que se ejecute el trabajo

6. ¿Cuál de las siguientes opciones es una manera de pasar datos de entrada a un worker?

- ○ Pasa los datos como un argumento cuando llames a la función `doWork()`.
- ◉ Usar un objeto de datos para pasar pares clave-valor — ¡Correcto!
- ○ Pasar datos como una cadena, pero debe tener menos de 140 caracteres
- ○ Asígnalo a la variable `worker.inputData`.

7. Después de poner el trabajo en cola, puedes comprobar su estado antes de ___.

Selecciona todas las respuestas que consideres correctas.

- [x] Nombre ¡Correcto!
- [x] ID ¡Correcto!
- [x] Etiqueta ¡Correcto!
- [ ] Tipo de trabajo

8. El Inspector de tareas en segundo plano te permite detener a los trabajadores durante su ejecución.

- ( ) Verdadero
- (•) Falso ¡Correcto!

9. ¿Qué compilador de Worker se recomienda para probar elementos `CoroutineWorker`?

- ( ) `OneTimeWorkRequestBuilder`
- ( ) `PeriodicWorkRequestBuilder`
- ( ) `TestWorkerBuilder`
- (•) `TestListenableWorkerBuilder` ¡Correcto!

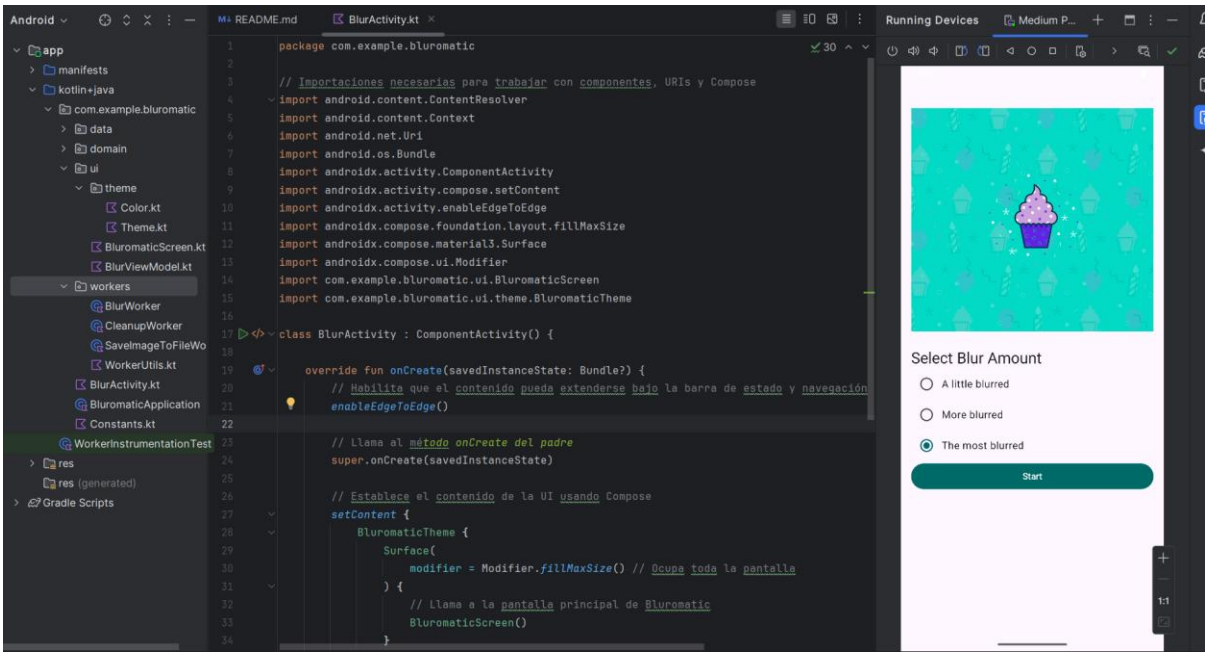
10. Cuando pruebas las implementaciones de Worker, puedes llamar a los trabajadores directamente con `doWork()` en lugar de ponerlos en cola.

- (•) Verdadero ¡Correcto!
- ( ) Falso

Results

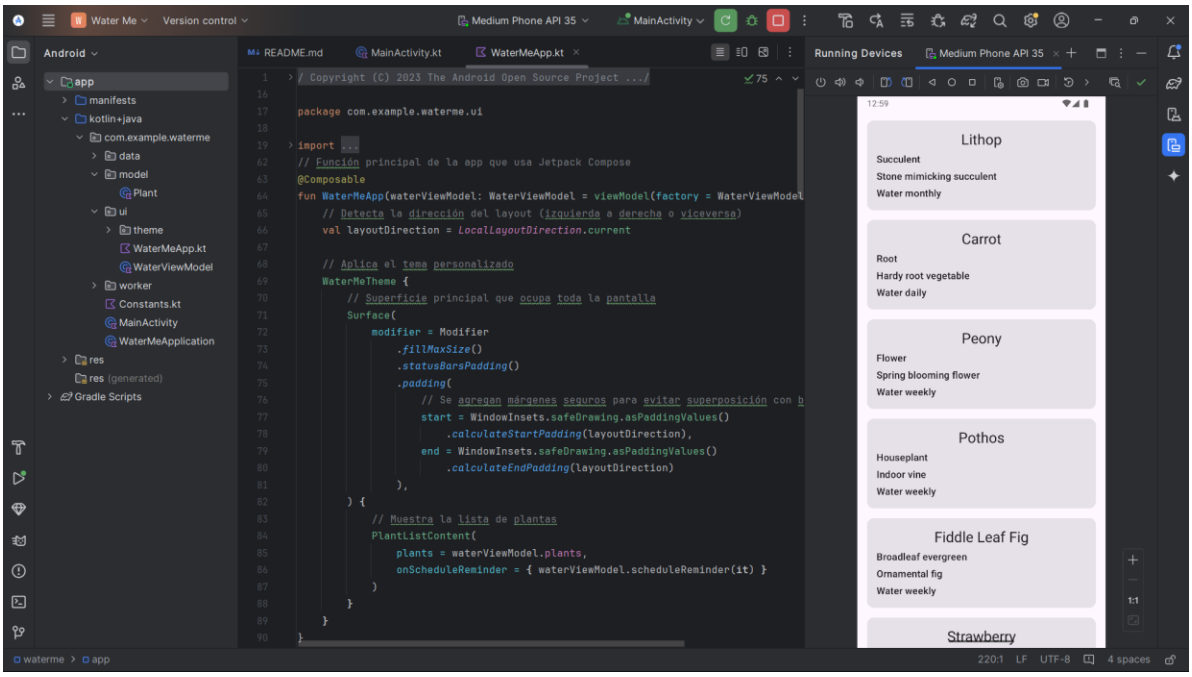
Tu puntuación es 8 de 10. ¡Felicitaciones! Aprobaste el cuestionario.

WORKMANAGER



Este código define una actividad (BlurActivity) que utiliza Jetpack Compose para mostrar la interfaz gráfica. Usa un tema personalizado (BluromaticTheme) y una superficie que cubre toda la pantalla. Dentro de esa superficie se llama a BluromaticScreen, que representa la pantalla principal. Además, hay una función getImageUri() que genera una URI válida para acceder a una imagen (android_cupcake) desde los recursos internos (drawable) del proyecto.

WATERME



- Muestra una lista de plantas en forma de tarjetas.
- Al tocar una planta, se abre un diálogo para programar un recordatorio (5 segundos, 1 día, 1 semana o 1 mes).
- Usa Jetpack Compose con una arquitectura moderna basada en ViewModel.