

Universidad Politécnica de Chiapas

Ingeniería en Tecnologías de la Información e Innovación Digital

[Programacion Para moviles]

[C2 – A4 - Coroutines - actividad asincrona]

[Nomenclatura del nombre de archivo C2 – A4 - Coroutines - actividad asincrona-223216-Daniel Peregrino Perez.pdf]

[Alumno -Daniel Peregrino Perez] - [223216]

Docente: [José Alonso Macias Montoya]

Fecha de entrega: [17/06/2025]

1. DESCRIPCIÓN DE LA ACTIVIDAD

1.1. Enunciado del problema

Desarrollar una aplicación móvil nativa para Android utilizando Android Studio y el lenguaje Kotlin. El objetivo principal es implementar una base de datos local utilizando Room e insertar datos en ella de manera asíncrona mediante Coroutines. La aplicación deberá mostrar el estado del proceso al usuario, incluyendo un delay previo a la inserción para simular una tarea de larga duración.

1.2. Objetivos de aprendizaje

- Comprender el funcionamiento básico de Room y su integración con Android.
- Implementar corrutinas (coroutines) para realizar operaciones asincrónicas en la UI.
- Ejecutar inserciones en la base de datos sin bloquear el hilo principal.
- Mejorar la experiencia del usuario con mensajes de estado durante tareas largas.
- Aplicar buenas prácticas al manejar el ciclo de vida con lifecycleScope.

2. FUNDAMENTOS TEÓRICOS

Room es una biblioteca de persistencia de datos de Android que proporciona una capa de abstracción sobre SQLite para permitir un acceso robusto a la base de datos.

Las Coroutines en Kotlin permiten escribir código asíncrono de manera secuencial, facilitando la ejecución de operaciones largas como el acceso a la base de datos o delays sin bloquear el hilo principal.

Componentes clave:

- @Entity: Marca una clase como una tabla en la base de datos.
- @Dao: Interfaz que define métodos de acceso a la base de datos.
- AppDatabase: Clase abstracta que representa la base de datos y conecta las entidades con los DAOs.
- lifecycleScope.launch: Lanza una coroutine que se cancela automáticamente cuando se

destruye el Activity o Fragment.

3. DESARROLLO DE LA ACTIVIDAD

3.1. Desarrollo

- 1. **Configuración del Proyecto:**
 - Se añadió Room al archivo build.gradle.kts.
 - Se configuró la clase AppDatabase con una tabla llamada tabla_de_tareas.
- 2. **Diseño de Interfaz:**
 - Se utilizó ViewBinding para controlar los elementos de UI.
 - Se implementó un botón y un TextView para mostrar el estado del proceso.
- 3. **Implementación de Código Kotlin:**
 - Se creó la entidad Tarea y su correspondiente DAO.
 - Se programó el botón para que, al ser presionado, inicie una coroutine con un delay de 3 segundos antes de insertar la tarea.
 - Se genera una tarea con una marca de tiempo (fechaActual) para distinguir cada entrada.
 - Se actualiza la UI en cada etapa: antes del delay, después del delay y tras insertar la tarea.
- 4. **Pruebas y Depuración:**
 - Se verificó que las tareas se insertan correctamente y que la UI no se congela durante la espera.
 - Se observaron los cambios utilizando el Database Inspector de Android Studio.
 -

```
1 // Archivo: app/src/main/java/com/example/coroutinesroomactivity/Tarea.kt
2
3 package com.example.coroutinesroomactivity // <-- ¡Asegúrate de que este sea tu paquete!
4
5 import androidx.room.Entity
6 import androidx.room.PrimaryKey
7
8 /**
9  * Esta clase representa la tabla 'tabla_de_tareas' en la base de datos.
10  */
11 @Entity(tableName = "tabla_de_tareas")
12 data class Tarea(
13     @PrimaryKey(autoGenerate = true)
14     val id: Int = 0,
15     val descripcion: String
16 )
```

```
MainActivity.kt x build.gradle.kts (:app) AndroidManifest.xml TareaDao.kt AppDatabase.kt Tarea.kt
1 // Archivo: app/src/main/java/com/example/coroutinesroomactivity/MainActivity.kt
2
3 package com.example.coroutinesroomactivity // <-- ¡Asegúrate de que este sea tu paquete!
4
5 import androidx.appcompat.app.AppCompatActivity
6 import android.os.Bundle
7 import androidx.lifecycle.LifecycleScope
8 import com.example.coroutinesroomactivity.databinding.ActivityMainBinding
9 import kotlinx.coroutines.delay
10 import kotlinx.coroutines.launch
11 import java.text.SimpleDateFormat
12 import java.util.Date
13 import java.util.Locale
14
15 class MainActivity : AppCompatActivity() {
16
17     private lateinit var binding: ActivityMainBinding
18     private lateinit var database: AppDatabase
19
20     override fun onCreate(savedInstanceState: Bundle?) {
21         super.onCreate(savedInstanceState)
22         binding = ActivityMainBinding.inflate(layoutInflater)
23         setContentView(binding.root)
24
25         // Inicializar la base de datos
26         database = AppDatabase.getDatabase(context = this)
27
28         // Configurar el listener del botón
29         binding.btnInsertar.setOnClickListener {
30             iniciarProcesoDeInsercion()
31         }
32
33         binding.tvEstado.text = "Presiona el botón para empezar"
```

4. RESULTADOS

4.1. Resultados obtenidos

- La aplicación inserta correctamente tareas en la base de datos local.
- La inserción se realiza después de un delay simulado de 3 segundos.
- La UI se mantiene responsiva durante todo el proceso.
- Se utilizaron coroutines para manejar el delay y la inserción sin bloquear el hilo principal.
- Se confirmó que las tareas se almacenan correctamente en la base de datos de Room.

tabla_de_tareas x

Live updates ↗

50

	id	descripcion
1	1	Tarea creada el 20/6/2025 22:50:25
2	2	Tarea creada el 20/6/2025 22:50:30
3	3	Tarea creada el 20/6/2025 22:50:54

> src > main > java > com > example > coroutinesroomactivity > MainActivity 63:2 LF UTF-8 4 spaces

5. CONCLUSIONES

El proyecto permitió aprender de forma práctica el uso de Room y Kotlin Coroutines para gestionar operaciones asíncronas de manera eficiente. También se reforzó el uso del ciclo de vida con lifecycleScope, así como el diseño de interfaces reactivas y amigables al usuario.

Se consolidaron buenas prácticas como el uso de ViewBinding y la separación de lógica de base de datos mediante DAOs.

7. REFERENCIAS

- **Android Developers. (s.f.). Room Persistence Library.**
<https://developer.android.com/training/data-storage/room>
- **Kotlin Coroutines Guide.**
<https://kotlinlang.org/docs/coroutines-overview.html>
- **ViewBinding en Android.**
<https://developer.android.com/topic/libraries/view-binding>