Software para Dispositivos Móviles

Miguel Sánchez Santillán
sanchezsmiguel@uniovi.es

Corrutinas – ¿Qué son?

- Simplifican la escritura de código a ejecutar de forma asíncrona
- Las **funciones** se **suspend**en. **El hilo no queda bloqueado**
- El Hilo pasará a procesar otra corrutina

Lanzar X corrutinas no es lo mismo que lanzar X hilos

Corrutinas - Ámbito

- Especificamos un límite a su tiempo de vida:
  - GlobalScope: Mientras la app esté funcionando.
  - **lifeCycleScope**: Mientras la Activity/Fragment estén vivos.
  - viewModelScope: Hasta que se destruya el viewModel (otro día).

- Cuando la corrutina muere/cancela, mueren sus derivadas.
  - Evita fugas de memoria

Corrutinas - Dispatchers

- Se componen de un conjunto de workers / hilos

- Optimizados para resolver diferentes tipos de operaciones:
  - **Dispatchers.IO**. Lectura/Escritura en: bd, peticiones red, ficheros
  - **Dispatchers.Main**. Para interactuar con elementos de la UI
  - **Dispatchers.Default**. Operaciones de uso intensivo de la CPU.

Corrutinas - Ejemplo

- Leer números de **fichero**
  - Dispatchers.**IO**

- Luego pasa a **Main ¿Por qué?**

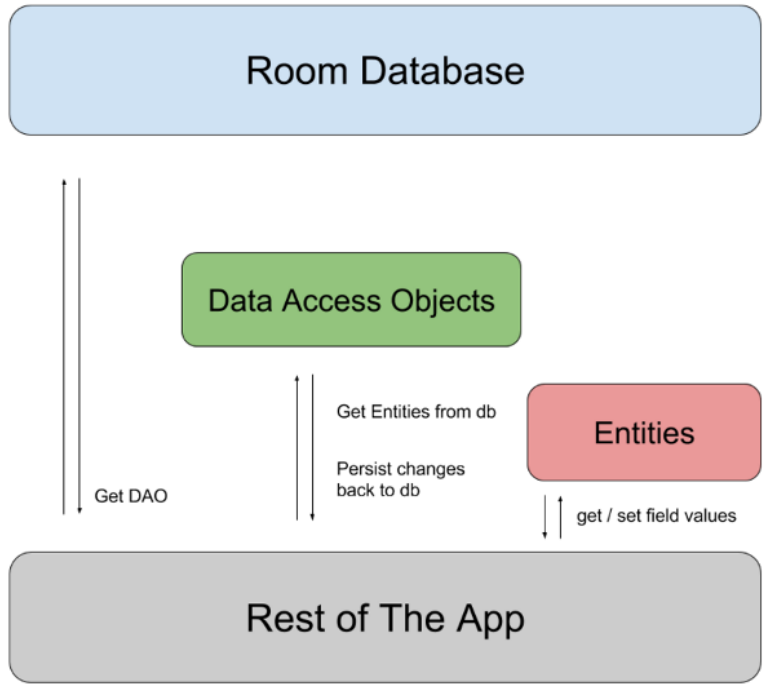
```
lifecycleScope.launch(Dispatchers.IO) {
    val lista = LeerFicheroDeNumeros()
    //Más operaciones
    //...

    withContext(Dispatchers.Main) {
        tvTotalNumeros.text = lista.size.toString()
    }
}
```

Recuerda: launch() no está bloqueando el código que está debajo

- También existe async()

Room - Base de datos local

- Capa de **abstracción** sobre SQLITE
- **Verificación** SQL en compilación
- Uso de **anotaciones**
- Data Access Object (**DAO**)
  - https://developer.android.com/training/data-storage/room/accessing-data



https://developer.android.com/training/data-storage/room

Room - Generación de código

- Kotlin Symbol Processing (**KSP**) para generar código

  https://kotlinlang.org/docs/ksp-overview.html

- En **build.gradle.kts (carpeta app)** añade a **plugins**

  id("com.google.devtools.ksp") version **"1.9.24-1.0.20"**

- Existe una **relación** entre la **versión del plugin y la de Kotlin**
  - Según el contexto, quizás deberías ajustar la versión del plugin

Room - Dependencias

- La propia **implementación** de Room

- Gestión de la **generación** de código con **anotaciones**
  - annotationProcessor + **ksp**

- Habilitar el uso de **corrutinas** y extensores
  - ktx

```
val room_version = "2.6.1"
implementation("androidx.room:room-runtime:$room_version")
annotationProcessor("androidx.room:room-compiler:$room_version")
ksp("androidx.room:room-compiler:$room_version")
implementation("androidx.room:room-ktx:$room_version")
```

Room – Post dependencias

Sincroniza + Botón Build

Room - Anotaciones de Entidades

- Para mapear entidades como tablas, se utiliza **@Entity**

  @Entity(tablename = “nombre_tabla”)

- **@PrimaryKey** sobre un atributo indica es clave primaria

  @PrimaryKey(autogenerate = true)

- **@ColumnInfo**(name = “nombre_columna”) fuerza el nombre de una columna en la tabla

- Más: https://developer.android.com/training/data-storage/room/defining-data

Ejercicio I – Anotación de Entidades

- Anota la clase **Contacto**

- Añade el atributo id (Int y var) con valor por defecto 0.
  - Será la clave **primaria** y **autogenerada**.

- La **columna** del atributo imagenURL será **imagen_url**

Room – Anotaciones DAO

- **Interfaces** con la anotación **@Dao**

- Dentro se declaran mensajes para:
  - Insertar → @Insert // @Upsert → Inserta, pero si existe, lo actualiza
  - Actualizar → @Update
  - Eliminar → @Delete
  - Consultar → @Query(“código SQL”)

- Suelen marcarse con suspend para el uso con corrutinas

- https://developer.android.com/training/data-storage/room/accessing-data

Ejercicio II – Dao para contactos

- Crea una **interface ContactoDao** y anótala con **@Dao**

- Este mensaje sirve para recuperar un contacto concreto:

```
@Query("SELECT * FROM contactos WHERE id = (:contactoId)")
suspend fun findById(contactoId: Int): Contacto
```

- **Añade el resto** de los mensajes y anotaciones que permitan:
  - Actualizar un contacto.
  - Eliminar un contacto.
  - Seleccionar TODOS los contactos.
  - Insertar un contacto.

Room – Clase que modela la BD

- La clase que representa la base de datos incluye la información previa: Daos, entidades…

- Esta clase te la proporciono **parcialmente** **completa**.

- **Complétala.**

Ejercicio III – Aplicación contactos

- Haz que sea posible en la aplicación:
  - Cargar la lista de contactos desde la BD y enviársela al RecyclerView
  - ContactoNuevoFragment → Crear el contacto en la BD
  - ContactoDetallesFragment → Eliminar el contacto en la BD
  - ContactoEditarFragment → Actualizar el contacto en la BD

- ContactoDetallesFragment y ContactoEditarFragment
  - Actualmente reciben un objeto contacto mediante la navegación
  - Ahora **deberían recibir la id** del contacto

- **¡Necesitarás utilizar corrutinas! ¡Ojo con tocar la interfaz!**

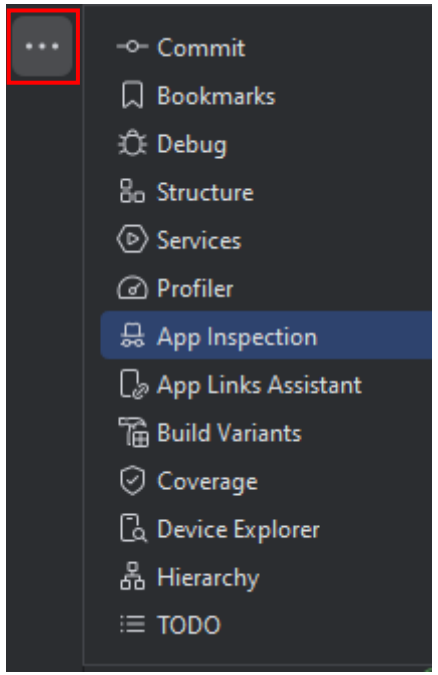
Base de datos inicial – Método X de N

- Puedes generar tu propia Base de Datos
  - Herramientas externas: https://sqlitebrowser.org/

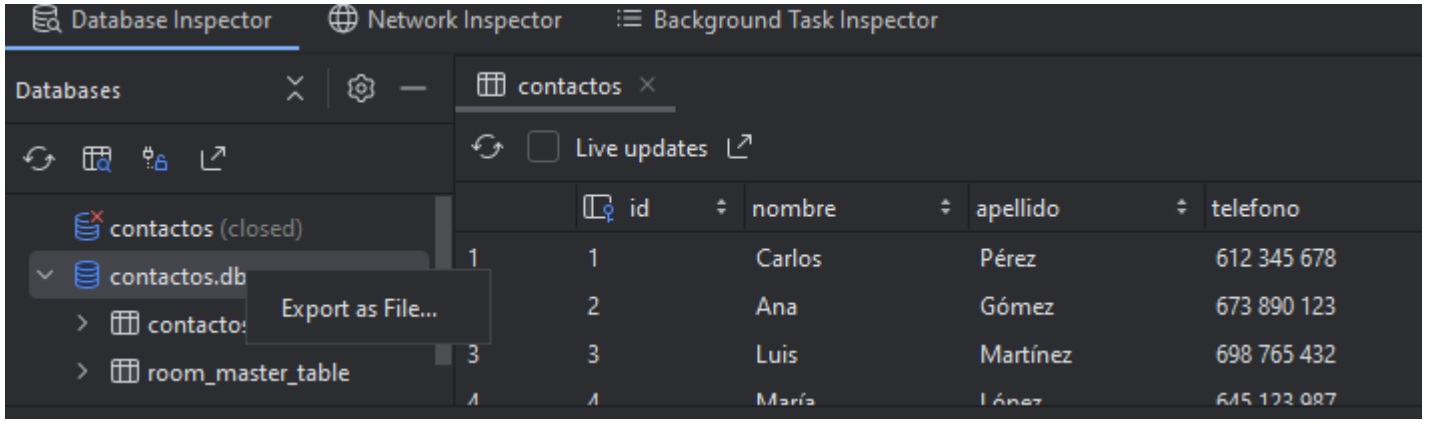
- El esquema de la BD generada, debe coincidir con las anotaciones de las entidades.

Base de datos inicial – Método Y de N

- Exportándolo desde la APP, mediante **App Inspection**



DataBase Inspector → clic derecho en la bd → Exportar **como db**



Room – Dos Enfoques para las relaciones

- **Opción a**: Uso de **clase(s) intermedia(s).** Ejemplo 1-N:
  - Clase ContactoMensajes
    - Atributos: contacto. Lista de los mensajes pertenecientes a ese usuario.

- **Opción b**: Uso de **mapas**. Ejemplo 1-N:
  - Map<Contacto, List<Mensaje> >
    - Para cada clave Contacto, está asociado como valor una lista de los mensajes.

- https://developer.android.com/training/data-storage/room/accessing-data
  - El enlace anterior es una lectura/ojeada muy recomendada.

Ejercicio IV – Mensajes de contactos

- En la vista de detalle de un contacto → Botón para añadir un mensaje a ese usuario.
  - Obviamente, con su correspondiente navegación a fragment y demás.

- En la misma vista detalle, muestra todos los mensajes para ese usuario.
  - Puede ser un TextView con todos ellos concatenados en String.

- Completa la navegación, con un RecyclerView para mostrar TODOS los mensajes.