¿QUÉ SON LAS COROUTINES?

¿QUÉ SON LAS COROUTINES?

- Utilidad de Kotlin que facilita la creación de tareas en segundo plano.
- Evitar bloquear el hilo principal creando hilos secundarios para tareas asíncronas.
- Escribir código asíncrono de forma secuencial.
- **Funciones de suspensión**.

¿QUÉ SON LAS COROUTINES?

Funciones de suspensión

- Pueden detener la ejecución de la coroutine en cualquier punto y devolver el resultado al hilo original.
- Pueden ejecutarse en el hilo principal o en un hilo diferente.
- Solo pueden ejecutarse dentro de una coroutine o dentro de otra función de suspensión.

¿QUÉ SON LAS COROUTINES?

Funciones de suspensión

```
suspend fun suspendFunction() : Int {
    // Async task
    return  0
}
```

¿QUÉ SON LAS COROUTINES?

Contexto de la coroutine

- Configuración que indica cómo se ejecutará la coroutine.
- Mediante los **Dispatchers**, a través de **withContext** se puede configurar el hilo de ejecución.
- Utilizando Dispatchers.IO se está usando el hilo en background para realizar la operación.

```
suspend fun suspendFunction() =
    withContext(Dispatchers.IO)  {
        return api.doOperation()
    }
}
```

¿QUÉ SON LAS COROUTINES?

Contexto de la coroutine (Dispatchers)

- **Main**: Ejecuta acciones en el hilo de la UI.
- **Default**: Dispatcher por defecto. Tareas de uso intensivo de la CPU.
- **IO**: Operaciones que bloquean el hilo principal mientras esperan la respuesta de otro sistema: Acceso a base de datos, respuesta de un API, etc.

¿QUÉ SON LAS COROUTINES?

Builders de Coroutines

- Se utilizan para iniciar las coroutines.
- El más usado es **lauch.**
- **launch** necesita un Scope para indicar bajo qué condiciones se ejecutará la coroutine.

```
GlobalScope.launch(Dispatchers.Main)  {
    ...
}
```

¿QUÉ SON LAS COROUTINES?

Scope de Coroutines

- **GlobalScope** se usa para cualquier coroutine que continúe con la ejecución mientras la aplicación se esté ejecutando.
- **GlobalScope** no debe vincularse a ningún componente que pueda ser destruido (Activity, Fragment, etc).
- **lifecycle-aware scopes:** Existen scopes para componentes con ciclo de vida como **ViewModelScope** que harán que la coroutine se cancele si el componente no está activo.