

# COROUTINES

# O QUE SÃO COROUTINES?

- Operações que são executadas sem travar uma thread.
- Uma forma de escrever código Assíncrono
- JÁ estão disponíveis desde a versão 1.1 do Kotlin



# ENTENDENDO

## ANTES DE CRIAR PRECISAMOS ENTENDER

- Suspend Function
- Context e Dispatchers
- Scopes e Jobs
- Tipos de Execução

# SUSPEND FUNCTIONS

- Qualquer função que pode ser pausada e retomada
- Exemplo usando Room:

```
@Dao
interface ListaDAO {

    @Query("SELECT * FROM gitHubRepo")
    suspend fun getRepositoriosCache(): MutableList<GitRepositorioEntity>

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertRepositorio(repositorio: GitRepositorioEntity)

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertAllRepositorio(repositorio: MutableList<GitRepositorioEntity>)

}
```

# SUSPEND FUNCTIONS

- Exemplo Usando RETROFIT

```
interface Service {  
    @GET("search/repositories")  
    suspend fun getHubRepos(  
        @Query(QUERY, encoded = true) query: String = KOTLIN_LANGUAGE,  
        @Query(SORT) sort: String = STARS,  
        @Query(PAGE) page: Int = INITIAL_PAGE  
    ): GitListaRepositoryResponse  
}
```

# CONTEXT & DISPATCHERS

- Dispatcher.UI
- Dispatcher.IO
- Dispatcher.Unconfined
- Dispatcher.Default

Um Dispatcher pode ser declarado no momento da chamada da coroutine, ou no momento em que você declara a suspend function, se não declarar o Dispatcher ele ira executar no Default

# DISPATCHERS EXEMPLO



# SCOPES

- Coroutine Scope
- Global Scope
- ViewModel Scope
- LifeCycle Scope
- Main Scope

# COROUTINE SCOPE

- É uma interface que somente possui o Coroutine Context
- É recomendado a utilização deste escopo

```
fun testaCoroutines(dispatcher: CoroutineDispatcher, tipolog: String) {  
    runBlocking { this: CoroutineScope  
        CoroutineScope(dispatcher).launch { this: CoroutineScope  
            println(tipolog + "1 - Thread ao iniciar coroutine ${Thread.currentThread()} executou.")  
        }  
    }  
}
```

# GLOBAL SCOPE

- É usado para iniciar coroutines de nível superior
- Não é Recomendado
- Recebe Empty no CoroutineContext

## MAIN SCOPE, VIEWMODEL SCOPE E LIFECYCLE SCOPE

- São derivações do Coroutine Scope
- É recomendado a utilização deste escopo



# ALGUMAS FORMAS DE EXECUÇÕES

- RunBlocking
- Launch
- Async

Existem várias maneiras possíveis de criar coroutines com base em diferentes requisitos. Nesta seção, vamos dar uma olhada em alguns deles

# RUNBLOCKING

- Executa uma nova rotina e bloqueia a thread atual até a conclusão da execução

```
@Test
fun `quando chamar a Api para a proxima pagina retornar sucesso e retornar os dados da api`() {
    val repository = ListaRepository(service, database)
    runBlocking {
        val lista = repository.listaRepositoriosProximaPagina(1)
        Assert.assertEquals(2, lista.size)
        Assert.assertFalse(repository.dadosCache)
    }
}
```

# LAUNCH

Inicia uma nova corotina sem bloquear a thread atual e retorna uma referência à corotina como um trabalho.

```
private fun proximaPaginaRepositorio() {
    viewModelScope.launch {
        try {
            if (useCase.aListaEstaLocal()) {
                event.value = ListaGitHubEvent.ExibeInformacaoCache(View.VISIBLE)
            } else {
                state.value = ListaGitHubStates.ListaGitHubSucesso(useCase.listarProximaPaginaRepositoriosGitHub())
            }
        } catch (exception: Exception) {
            exception.printStackTrace()
            Log.e("Viewmodel", exception.message)
        }
    }
}
```

# ENTENDENDO

## ANTES DO ASYNC - DEFERRED

- Objeto Futuro
- Promessa
- Uso com o. Async



# ASYNC

- Executa uma nova rotina e bloqueia a thread atual até a conclusão da execução

## EXEMPLO



# JOB

- É uma atividade cancelável
- Podem ser divididos em Hierarquias
- Um escopo sempre é associado a um Job

# SUPERVISOR JOB

- Os filhos deste job podem falhar de forma independente
- É o mesmo job que o viewmodelscope e o lifecycle usam

# COROUTINES EXEMPLO

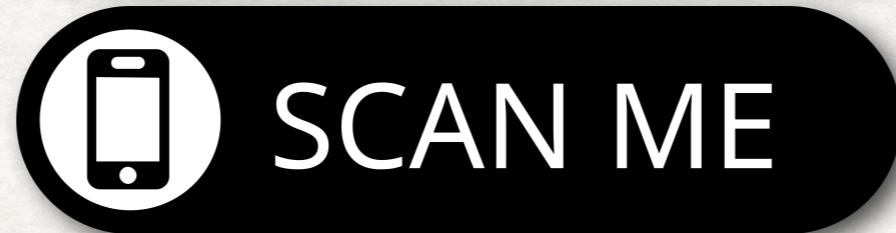


# E TEM MAIS

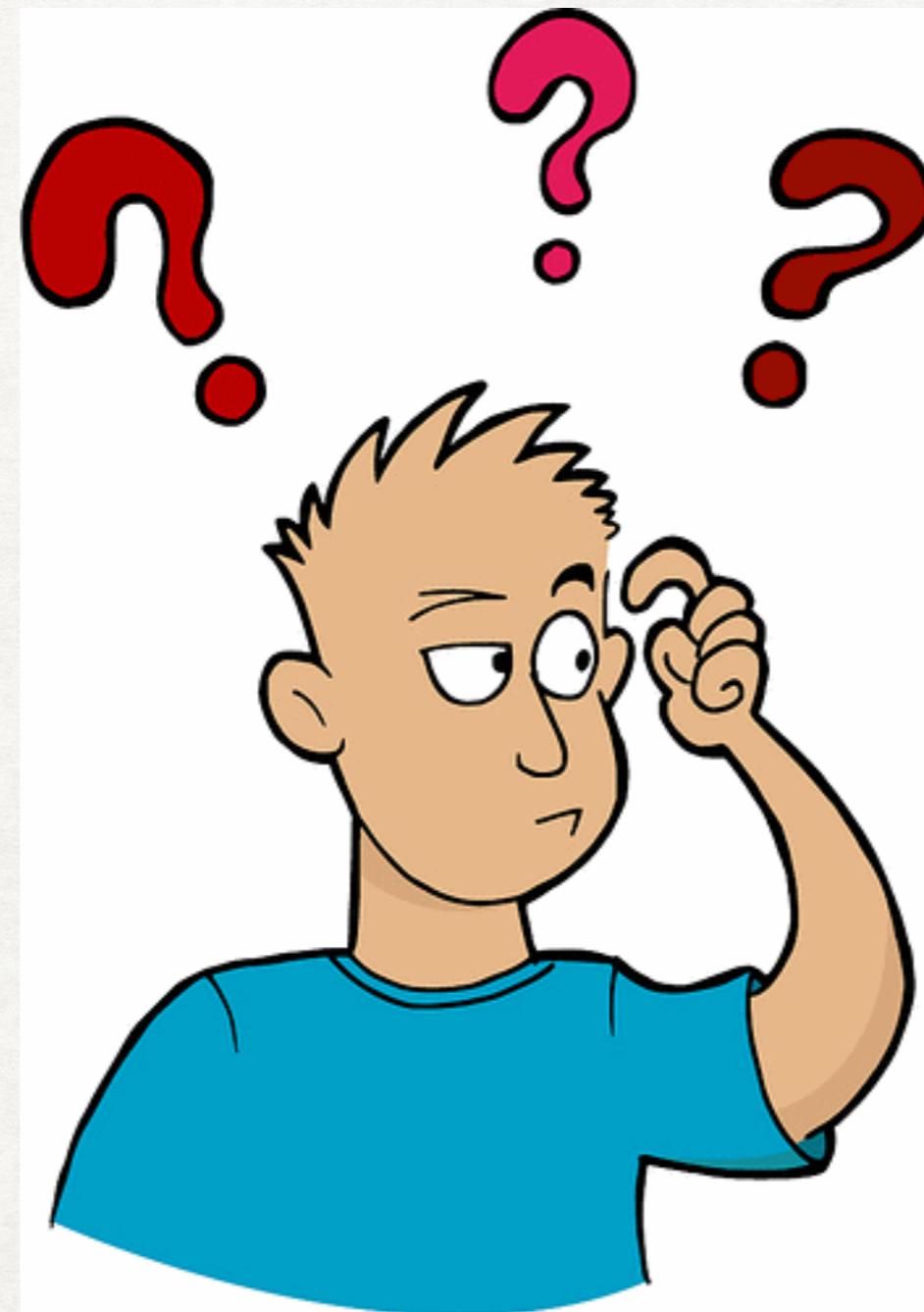
- Channel
- Flow
- ...

# LINKS UTEIS

## PARA MAIS CONHECIMENTO DE COROUTINES



# DÚVIDAS



# CONTATOS

## RENATO RIBEIRO DOS SANTOS

- github: <https://github.com/rribesa>
- Linkedin: <https://www.linkedin.com/in/renato-santos-9a25b221/>

