**lifecycleScope (RECOMENDADO em Activity/Fragment)**

lifecycleScope é **um escopo de coroutine integrado ao ciclo de vida** da Activity ou Fragment.

**Vantagens:**

* **Cancela automaticamente** a coroutine quando o Fragment ou Activity for destruído (evita vazamentos de memória).
* Mais seguro para tarefas relacionadas à **UI**.
* Ideal para chamadas de rede, Firebase, banco local etc.

kotlin

CopiarEditar

lifecycleScope.launch {

// Esta coroutine será cancelada automaticamente se o Fragment for destruído

}

**⚠️ CoroutineScope(Dispatchers.IO).launch**

Essa é uma **coroutine manual**, criada em um escopo que **não está atrelado ao ciclo de vida** da Activity ou Fragment.

**Problemas:**

* A coroutine **continua executando mesmo se o Fragment for destruído**.
* Pode causar **crashes**, vazamento de memória ou atualizações de UI em um componente que já não existe mais.

kotlin

CopiarEditar

CoroutineScope(Dispatchers.IO).launch {

// Pode continuar rodando mesmo se o usuário sair da tela!

}

**✅ Quando usar cada um?**

| **Situação** | **Use lifecycleScope?** | **Use CoroutineScope manual?** |
| --- | --- | --- |
| Atualizar a UI após buscar dados | ✅ Sim | ❌ Não recomendado |
| Operação depende da tela visível | ✅ Sim | ❌ Pode dar erro |
| Código global (ex: serviço) | ❌ Não | ✅ Pode usar |
| Dentro de ViewModel | ❌ Use viewModelScope | ✅ Somente se necessário |